Histogram-Based Outlier Score: A Statistical Approach to Anomaly Detection

McKenna Skoczen, Jayla Hall, Silvia Alfaro, Anakin Ligman, Derek Baron

*Faculty Advisor: Kathryn Dobeck, Lorain County Community College*

*Industrial Liaison: Josh Tang, Key Bank*

Industrial Problem

Anomaly detection is the act of identifying cases within datasets that deviate from expected behavior [1]. It is applicable across several sectors in industry, but here we demonstrate how it can be used to help optimize resources for computer system maintenance. We have a sample of computer metric data collected from a number of workstations, provided by our industrial partner, Key Bank. This sample of data consists of variables measuring CPU utilization, memory usage, hard drive, the system processes, and network utilization data collected over a time span of several hours in one day. The goal is to increase the efficiency of their information technology team, by prioritizing the computers most in need of attention, and our position with this project is to adapt and implement an algorithm that identifies the workstations exhibiting the most unusual behavior, indicating that they are in need of maintenance.

Executive Summary

Taking into consideration the nature of our data, the focus on global anomalies, and the goals of prioritizing computation time and minimizing false positives, we decide upon a statistical unsupervised setup for identifying the most anomalous workstations in our sample. The algorithm we select is the Histogram-Based Outlier Score (HBOS). While our dataset is multivariate, this algorithm constructs univariate dynamic-width histograms to estimate probability densities for each variable and takes the multiplicative inverse of these densities across all variables to form the final score. We execute this in Python with the use of the Pandas library to obtain the top ten most anomalous workstations in each of ten time ‘instances’, which we combine via two different ranking strategies to obtain two different versions of the top ten across the entire time span.

Problem Approach

There is no singular optimal strategy for determining the top ten offenders out of these thousands of workstations. From a perspective of machine learning, which is a method of data analysis that automates analytical model building, various approaches for making detections exist [2]. There are supervised, semi-supervised, and unsupervised setups. Different anomaly detection methods are better suited towards different setups. There are advantages and challenges with each setup. Supervised setups are used for labeled training sets. Training sets are used to train an algorithm to predict response variable values based on previously known values from a trained model. Semi-supervised setups also use training sets, but the training data consists of data without any known anomalies. With presorted data, it would be advised to consider an approach better suited for supervised or semi-supervised anomaly detection setups, for it relies on this feature to teach itself and learn from provided labeled examples. Our sample does not consist of labeled data so we fall under the unsupervised category. As such, we narrow our search to methods compatible with an unsupervised setup.

In narrowing down our list of techniques, we examine our problem and note specific characteristics we are looking for in the algorithm to be selected. The best algorithm for our case is an unsupervised, statistical approach that emphasizes global anomalies over local anomalies. Because global anomalies are anomalous data points that are considered outliers for the entirety of the dataset, we are more concerned with finding global anomalies as opposed to local anomalies. Our choice of algorithm must be able to work with time series, and, in our case, we want to try to minimize results that are falsely indicative of an outlier, or false positives. The top three overarching categories of methods we found compatible with unsupervised setups include nearest-neighbor based algorithms, clustering based algorithms, and processes involving statistical concepts. One example of an unsupervised, statistical algorithm is Hierarchical Temporal Memory (HTM). HTM is a biological neural network approach; it is based on the study of thought processing in the human brain and the idea of creating neuron models for artificial intelligence. HTM systems are able to learn the structure of streaming data, make predictions and detect anomalies. They learn continuously from unlabeled data [3]. Accuracy and minimal false positives are commonly known attributes of HTM systems, which is valuable for us in that historically, false positives have been an issue in identifying problematic workstations in the past. There is still much progress to be made, however, with the research and application of this technique due to its complexity. Because HTM is modeled after the brain, and there is much more to learn about the brain, it is hard to say whether this algorithm will ever be useful in practical application.

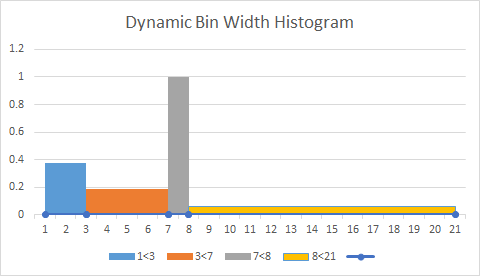
Of most importance to us is the histogram-based outlier scoring method (HBOS). This sort of statistical application is especially useful when quick computation is preferred or needed. On a smaller scale, the difference in computation time between the algorithms is negligible. In our case, though we are handling a small subset of data, we consider the larger problem (and much larger dataset) at hand. The larger the dataset, the more important minimizing computation time becomes. This is especially true when the results are meant to display as close to real time as possible. It follows that this is a promising method for handling sequential data. Having looked at the pros and cons of several techniques, we decide to use the HBOS algorithm.

Constructing the Histograms

While traditionally, histograms are created with fixed bin-widths, we elect to use histograms with varying bin-widths determined by ending each bin after a fixed number of values are included in each bin. This yields more accurate density estimations over the fixed-bin histograms in situations where there are gaps and long-tail distributions present in data [4]. Below we have an example of how the histograms are created. We take a single variable and sort the cases in ascending order by the values of that variable. The cases are then grouped into bins containing a fixed number of N/k cases, where N is the number of cases and k, in our case, is the square root of the number of cases. It should be noted that the bins must be extended to encompass more than N/k cases when there are a large number of repeated values in a single bin leading to the bin being composed of one value. This ensures there are always at least two unique values in a single bin. This eliminates the possibility of having a bin of width zero. It is common to have a final bin containing less than N/k values, where there have been extended bins and/or when the number of cases is not a perfect square [4]. These cases are both illustrated in the example below, where the cases have already been sorted in ascending order by the variable used to create the histogram. From the histogram we calculate the width, height, and the normalized height of each of the bins, where the height here represents the estimated probability density. The heights of the histograms are normalized so that each variable is weighted equally in the final score. This can be modified manually to reflect more nuanced relationships between variables. The width is calculated by taking the value (for the variable at hand) corresponding to the last case in a bin and subtracting the value corresponding to last case in the previous bin. For the first bin we take the value corresponding to the last case in the first bin and subtracting the value from the first case in the bin. The area of each bin is the number of variables in a bin and the height is determined by dividing the area by the width. The normalized height, which is our probability density estimation, is calculated by dividing the height of a bin by the maximum height of the bins [4].

**Table 1.**  Example Data for Histogram

|  |  |  |
| --- | --- | --- |
| Cases | Variable 1 | Bin Number |
| 1 | 1 | 1 |
| 2 | 2 | 1 |
| 3 | 3 | 1 |
| 4 | 5 | 2 |
| 5 | 7 | 2 |
| 6 | 7 | 2 |
| 7 | 7 | 3 |
| 8 | 7 | 3 |
| 9 | 7 | 3 |
| 10 | 8 | 3 |
| 11 | 9 | 4 |
| 12 | 9 | 4 |
| 13 | 21 | 4 |

**  
Figure 1.** Example Dynamic Bin Width Histogram

(1)

Implementing the HBOS Algorithm

Implementing the HBOS algorithm in Python is remarkably uncomplicated, due to the simplicity of the calculations required to be performed.[[5]](https://github.com/mcskoczen/PROG_Grahamcrackers) In order for us to use the HBOS algorithm with the data we were provided, we first have to clean and reformat the data to get it into a structure upon which math can be easily performed. We reformat the data, removing all non-number characters and removing extraneous identification data. Due to the limitations of the dynamic bin width approach to the HBOS algorithm, we also have to remove several columns of data which would result in only a single bin. Once that has been done, we can import the file into a DataFrame. A DataFrame is a form of array structure in the Pandas library, which allows for easy organizing and manipulating of the data once it has been imported. After that we determine the dimensions of the data we are using, and from there we define the parameters N and k. After that we create new columns in the DataFrame for the tracking of the bin parameters normalized height, width, and bin number, for the math we will perform as well as the running total for the HBOS score which is the only column that will remain in the output. We then begin a variable for loop which starts at the first piece of data and ends once it has finished the final variable. At the beginning of each variable, we sort the DataFrame by the values of that variable in ascending order, define the initial parameters of the first bin, accounting for the fact that arrays always count from zero, redefine k to account for any adjustments that might have occurred in the previous variable, and then begin a second for loop which will go through each workstation. In that for loop, each workstation (assuming it does not hold a value of zero for that variable) is assigned to a specific bin, and the height for that bin is determined using the endpoints of that bin and that height is assigned to that workstation. Whenever we reach a workstation that would go into the next bin, we determine the endpoints of the new bin, and continue until we reach the last workstation. Once we have reached that last workstation, we determine the maximum height of any of those bins before beginning a new for loop to go through the workstations. For each workstation in this new loop, we normalize the height of its bin so that the largest bin has a value of 1 and the height of every other bin is defined relative to the largest bin. We then take the base 10 logarithm of the inverse of that normalized height and add the result of that to the running total of the individual HBOS scores. After we have done that for every variable, we then sort the DataFrame in descending order by the values of the overall HBOS scores and output the modified DataFrame to a comma separated value sheet which can be viewed in MicrosoftⓇ Excel.

Detailed Results

Our sample included data across a time interval of several hours. It consisted of five subgroups of variables: CPU, diskio, memory, network, and process. Of the five subgroups of variables, the only subgroups whose timestamps were in line were CPU, diskio, and memory. To model instantaneous anomaly detection, we split the data up into ‘time intervals’ which contained at most one instance of each case, taken within approximately five minutes of each other. As such, we had to discard the data from the variables under the categories network and process. That said, those subgroups of variables could easily be incorporated where real time data is concerned. Of the remaining subgroups, there were 3 variables whose values were not included in the HBOS calculations due to having less than 3 unique values, resulting in a single bin. These were diskio write bytes, system memory total, and system memory swap total. Of the remaining variables, our algorithm produced ten sheets of results, each containing the ranked HBOS scores in descending order for a single time interval. Presented here are the top ten most anomalous workstations across the entire time period, as determined by two different ranking methods. In the first method, we take the average HBOS score across all time periods and sort in descending order (Table 2). In the second method, we list the workstations with the top ten scores of all time, omitting workstations that occur more than once until we have ten unique cases (Table 3). Common to both is the workstation 136 listed at the top, belonging to branch 17.

**Table 2.**

|  |  |  |
| --- | --- | --- |
| Workstation | Branch | Average HBOS Score |
| 136 | 17 | 25.04669 |
| 41 | 5 | 23.7045 |
| 28 | 3 | 21.90568 |
| 86 | 10 | 21.87967 |
| 80 | 9 | 20.80803 |
| 115 | 14 | 20.1487 |
| 76 | 8 | 19.32865 |
| 186 | 21 | 19.24384 |
| 172 | 22 | 17.9755 |
| 179 | 23 | 16.60495 |

**Table 3.**

|  |  |  |
| --- | --- | --- |
| Workstation | Branch | Specific HBOS Score |
| 136 | 17 | 31.90138322 |
| 28 | 3 | 29.49864432 |
| 41 | 5 | 29.49864432 |
| 109 | 13 | 28.34775852 |
| 174 | 22 | 27.82383271 |
| 80 | 9 | 27.67268209 |
| 115 | 14 | 27.03488703 |
| 152 | 19 | 25.64242374 |
| 76 | 8 | 25.58306956 |
| 153 | 19 | 25.16530248 |

Conclusions and Further Research

The goal of our research was to identify the ten most anomalous workstations using computer metric data. HBOS was used to give each workstation an outlier score and the workstations with the highest scores were considered the most anomalous. Our results provided us with two different strategies to identify the ten most anomalous workstations. Our first strategy was to average the individual HBOS scores from each time interval, providing us with the ten most anomalous workstations on average (see Appendix A). The second strategy was to identify the workstations with the highest HBOS scores overall; we did this until there were ten unique workstations (see Appendix B).

Based on our results, we recommend using HBOS as an appropriate strategy to identify the most unusual performance within computer metric data. In order to optimize HBOS on an industrial scale, we suggest manually adjusting the weight of each variable. Currently, each variable is weighted equally, causing the variables to have the same amount of influence on the overall outlier score. In industry, we suggest that variables that are more likely to be indicative of a problem be weighted more heavily than variables with minimal importance. Alternatively, scores can be calculated for subdivisions of the total variables. The workstations presented in Table 2 and Table 3 are the top ten most anomalous across the duration of time we were given. To show the most consistently anomalous workstations over time where data is coming in continuously, we recommend using data visualization tools to visualize HBOS scores and specific data points.

Acknowledgments

We would like to give thanks to Professor Dobeck for her role as a mentor and providing guidance on our project. We would also like to give thanks to Key Bank for providing both the problem and data required for this project. Special thanks to Kris Redden for his guidance on computer programing. Finally, we would like to mention that this project was sponsored by PIC Math, a program of the Mathematical Association of America (MAA) and the Society for Industrial and Applied Mathematics(SIAM). Support is provided by the National Science Foundation (NSF grant DMS-1722275).

Works Cited

[1] Heidmann L. Why Anomaly Detection Matters to (Almost) Everyone. Blog. [accessed 2019 Apr 19]. https://blog.dataiku.com/why-anomaly-detection-matters-to-almost-everyone

[2] Machine Learning: What it is and why it matters. SAS. [accessed 2019 Apr 19]. https://www.sas.com/en\_us/insights/analytics/machine-learning.html

[3] Numenta. What is Machine Intelligence vs. Machine Learning vs. Deep Learning vs. Artificial Intelligence (AI)? Numenta. [accessed 2019 Apr 19]. https://numenta.com/blog/2016/01/11/machine-intelligence-machine-learning-deep-learning-artificial-intelligence/

[4] Goldstein M, Dengel A. Histogram-based Outlier Score (HBOS): A fast Unsupervised Anomaly Detection Algorithm. 2012 [accessed 2019 Apr 18]. <https://pdfs.semanticscholar.org/5cf8/81d1db19834f123fcfc79ad32097aeafe17f.pdf>

[5] Alfaro S, Ligman A, Skoczen M, Hall J, Baron D. PROG-Grahamcrackers. 2019. Github Repository.

https://github.com/mcskoczen/PROG\_Grahamcrackers

Appendix A

A close up of a map

Description automatically generated

Appendix B

A close up of a map

Description automatically generated