DS-GA 1004: Unsupervised Outlier Detection Using K-Modes and Greedy Weighted Entropy

Team Kalman Filter

Isaac Haberman
New York University
ijh216@nyu.edu

Joseph Sloan
New York University
jts550@nyu.edu

ABSTRACT

By adapting and combining previous work in clustering and outlier detection, we develop a clustering algorithm for the discovery of point and contextual outliers and have assessed its feasibility on 50 datasets of varying size and composition. We have also provided implementations in both Python 3 and Pyspark.

KEYWORDS

Outlier Detection, Big Data, Entropy, K-Modes, Clustering, Anomaly Detection

ACM Reference Format:
Isaac Haberman and Joseph Sloan. 2018. DS-GA 1004: Unsupervised Outlier Detection Using K-Modes and Greedy Weighted Entropy: Team Kalman Filter. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, Article 4, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION AND PROBLEM FORMULATION

Outlier Detection has become one of the most researched and studied fields across the STEM disciplines, with well-regarded work originating from the fields including statistics, electrical engineering and information theory. With the deluge of techniques available, we sought to focus on two types of outliers, as defined by Chandola, Banerjee and Kumar's survey of anomaly detection algorithms[1]. The Survey details three classes of outliers: point, contextual, and collective. Point outliers are the singular point anomalies, and can often be discovered through most supervised or unsupervised techniques, especially if numerical. Contextual outliers are points that appear anomalous conditional on other data or information, and collective outliers are groups or clusters of similar anomalous instances that appear together but differ heavily from the other data. Our work, seeks to find point and contextual outliers, focusing particularly on outliers within categorical data. We also chose to to treat missing data (NA, NaN, or blank fields) as point outliers that our method should be able to detect, with specific support for handling these instances, if the user wishes.

To achieve our goals, we have developed an unsupervised outlier detection algorithm in Python for use with data in Pyspark and Python. It combines an entropy-based method to detect point outliers, detailed in Section 3.2, with K-Modes, a derivative of the class K-Means algorithm, detailed in Section 3.3. Using these algorithms together, enables the user to discover both point and contextual outliers, using the clusters of K-Modes as context. To enable our algorithms to function in Spark, we also developed a suite of utility functions to read in and process the data as a Pyspark RDD.

2 RELATED WORKS

For our purposes, we have reviewed Chandola, Banerjee and Kumar's aforementioned survey of anomaly detection[1] and relied on their definitions for our work. For our work with entropy-based methods, we began with He, Xe, and Deng [3], using a derivative of their hash-table technique, to store our data features. Since their technique requires the user to specify the number of expected outliers, we merged their work with the work of Sagade and Thakur [4], an extension of Wu and Wang [5] which requires fewer iterations through the data, and does not need an expected number of outliers. Though, we did not include their work explicitly, we relied on the work of Liu, Li, Wu and Fu [6], to inform our decision making.

The entropy-based methods we investigated focus only on point outliers. We sought to extend this approach to contextual outliers by merging this scoring with a clustering technique like those used in Liu, Li, Wu and Fu [6]. To that end, we read and implemented Huang's original K-Modes paper as well as the algorithms subsequent extensions. Among the K-Modes extensions we used, were Cao, Liang and Bai's [11] technique for centroid initialization, and the advanced categorical dissimilarity technique from Huang, Li, Nh and He [10]. Our implementation uses a dissimilarity weighting based on missing data (to scale features with missing data to the same distance scale while discounting their missing dimensions), inspired by similar ideas from Jiang et al[13]. We also tested our implementation of K-modes against the implementation of github user nicodv [12], confirming that our implementation worked and had similar speed.

While we have not implemented their work, we have researched and reviewed the work of Ramaswamy, Rastogi, and Shim [8], who outline a distance-based method, Das and Schneider[9] who outline a probabilistic approach and Hua and Pei[2], whose work focuses on missing data.

3 METHODS

Our implementation of the aforementioned methods are detailed below.

3.1 Missing Data

We provide three different ways to handle missing values when running our algorithm on a dataset. First, users can choose to impute missing values with either the mode or median of the feature, for categorical and numerical features respectively. Second, a user can drop rows that contain NA values. This effectively treats them as another set of outliers, and then our method can cluster and classify further outliers on the remaining data. Finally, our K-modes clustering implementation can actually handle the missing data as is, treating it as categorical data. The *prepRDD* function in *utils.py* handles this functionality for Pyspark.

3.2 Greedy Weighted Entropy

For standalone entropy-based detection we convert each column in the data to our Features class and iterates through the rows, removing each row and recomputing the holo-entropy of the data-set for each iteration. The resulting holo-entropies are subtracted from the original holo-entropy, called outlier factors, of the data-set, with positive values indicating outliers and negative values indicating normal data. In other words, if the removal of a row increased the holo-entropy of the data-set, the row is not an outlier, while if the holo-entropy increased, the row is an outlier. The magnitude of the outlier factors determines the strength of outlier belief of a given row. When integrated with K-Modes, Greedy Weighted Entropy is repeated for each centroid and returns suspected outliers on a per centroid basis.

3.3 K-modes

The core of this method is the computation of weighted entropy scores for data, based on the features of the data-set. To handle this, we developed a *Feature* class in python to hold additional data and perform specific functions on each column. Each individual feature can calculate a centroid center, using a mask for value membership, returning either the mean or mode value, depending on the feature type. The feature can also return the dissimilarity of the feature values and a given centroid center, returning euclidean distance or either of Huang's dissimilarity techniques for categorical features, weighting the dissimilarities by amount of missing data per row. Both of those functions, call an internal missing data function, that fills the arrays with missing values to maintain the proper size of the array. Lastly, the feature can calculate holo-entropy given masked value membership.

Using the aforementioned *Feature* class, we also implemented both local and Pyspark versions of K-Modes, an extension of K-Means which handles both numerical and categorical data. Both implementations take data, additional information, specified number of centroids and specific dissimilarity measure and output the data membership and suspected outliers based on the Greedy Weighted Entropy score mentioned above. Our K-Modes implementation uses the Cao method of centroid initialization [1], a deterministic method that chooses rows as centroids based on their densities and subsequent relationships between those densities and subsequent dissimilarity to previous centroids. Our implementation will continue fitting until one of the following stopping criteria: the maximum number of iterations has been reached, there were no membership changes between iterations, or the sum dissimilarity between all rows has grown between iterations. All dissimilarities are weighted by the quantity of missing values per row, with rows with more missing values, weighted less than other rows, as they had fewer features to find dissimilarity with.

4 RESULTS

We tested our combined K-modes and weighted entropy score on 50 provided datasets of varying size using Pyspark. In addition, we locally tested the algorithm to generate comparisons to the simpler Greedy Weighted Entropy approach. These comparisons, which are show in Figure 1 and Figure 2, show that fusion of the approaches significantly improves the ability of Greedy Weighted Entropy in distinguishing contextual and point outliers. Standalone Greedy Weighted Entropy often has almost all outlier factors close to 0, which demonstrates little confidence in classifications. Whereas, clustered Greedy Weighted Entropy often displays larger outlier factors, with more confidence in outliers and non-outlier rows. Note that in the middle figure, where we show both on the same plot, the standard GWE scores are of a much smaller magnitude than the clustered GWE, showing how much less variability there is in distinguishing points in the data.

In addition to testing for outliers, we investigated the optimal K for a few of the smaller data-sets using the Elbow technique. Those results are shown in 3

With the Pyspark version of our algorithm, the algorithm ran smoothly and produced meaningful results for data-sets with 10^5 rows or less, and generally converged within 10 minutes. The results are also comparable to the local version. However, for larger data-sets either in rows or in features, the algorithm failed to scale properly and took hours to converge. We theorize that much of the added time is primarily due to the initialization process for centroids, which scales with both rows and features.

5 FURTHER WORK

Our primary area of possible improvement is performance runtime. Currently, data-sets on the order of 10^6 bytes generally converge within 10 minutes, whereas larger data-sets take exponentially more amount of time. The current Pyspark implementation, while functional, could do more to utilize the parallelization capabilities of Spark than it does currently, specifically to recompute cluster distances after centroids have been selected and for data preprocessing. We specifically were not able to leverage the K-Modes algorithm in spark-mllib, but inspecting it more closely may prove useful for future optimization ideas. We need also to investigate the convergence rate of K-Modes, which slows during the processing of certain test files.

Another possible area of improvement, is a deeper exploration into the relationship between outlier sensitivity and number of clusters. While we have examined the total sum dissimilarity as a function of cluster number, we have done little to investigate the relationship between the number of outliers and the number of clusters.

Lastly, while the K-Modes clustering process is fairly well tested in previous literature, there is room for additional exploration of clustering methods. With our current K-Modes implementation, we can explore different initialization methods and entropy-based score

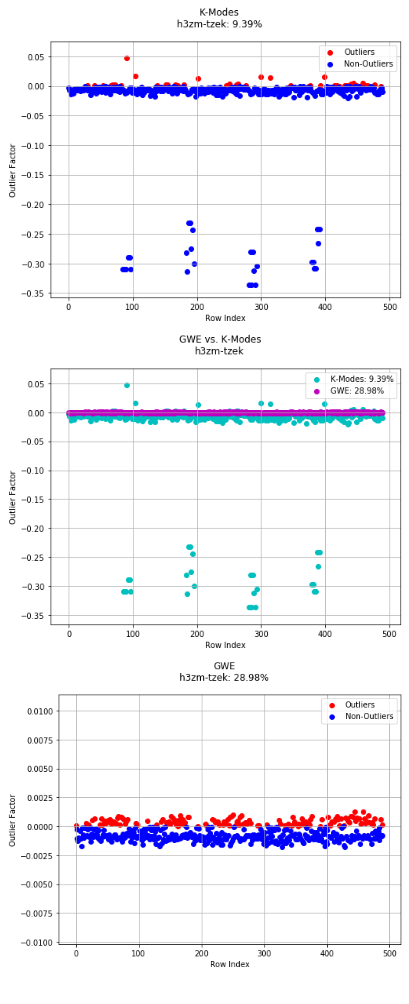

Figure 1: Results for h3zm-ta5h.tsv

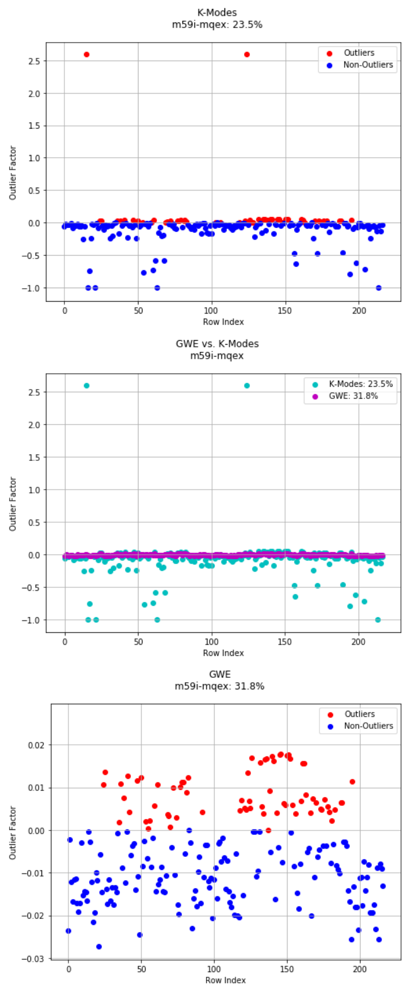

Figure 2: Results for m59i-mqex.tsv

functions. We have already implemented a second initialization method that creates an entropy-based outlier ranking based on the approach from Jiang, Liu, Du, and Sui [13]. This could likely be adapted into an alternative entropy score as well. In addition, we can explore further extensions of our clustering technique, such as fuzzy K-Modes (which assigns probabilities for cluster membership) and methods that adjust the optimal K during fitting, like X-Modes or Silhouette.

6 CONCLUSION

We have taken existing approaches to outlier detection, specifically variants of K-Means clustering and weighted entropy scoring, and combined them to create an unsupervised outlier detection algorithm algorithm. On our test datasets, we have shown that combining K-Modes with Greedy Weighted Entropy significantly improves the ability of Greedy Weighted Entropy to detect and classify contextual outliers and further the confidence in point outliers. While there is work to be done regarding performance on large datasets for the K-Modes portion of our algorithm, the method developed is a unique and useful approach to unsupervised outlier detection without specified outlier count.

7 CODE REPOSITORY

All the code used in this project is freely available at our github: https://github.com/isaachaberman/KalmanFilter-1004

REFERENCES

[1] Varun Chandola, Arindam Banerjee, Vipin Kumar. Anomaly Detection: A Survey. 2008. http://www.dtc.umn.edu/publications/reports/2008_16.pdf
[2] Ming Hua, Jian Pei. Cleaning Disguised Missing Data: A Heuristic Approach. 2007. https://www.cs.sfu.ca/~jpei/publications/dmv-kdd07.pdf
[3] Zengyou He, Xiaofei Xu, Shengchun Deng A Fast Greedy ALgorithm for Outlier Mining https://arxiv.org/ftp/cs/papers/0507/0507065.pdf
[4] Ashwini G. Sagade, Ritesh Thakur. Excess Entropy Based Outlier Detection In Categorical Data Set. http://www.iraj.in/journal/journal_file/journal_pdf/3-72-140688430856-61.pdf
[5] Shu Wu and Shengrui Wang https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6109256
[6] Liu, Li, Wu, Fu https://arxiv.org/pdf/1801.01899.pdf
[7] Tomas Ollson, Anders, Horst A Probabilistic Approach to Aggregating Anomalies for Unsupervised Anomaly Detection with Industrial Applications. https://pdfs.semanticscholar.org/e74e/37e6fc1c5ad030ad1a553193034ff3afbd8f.pdf

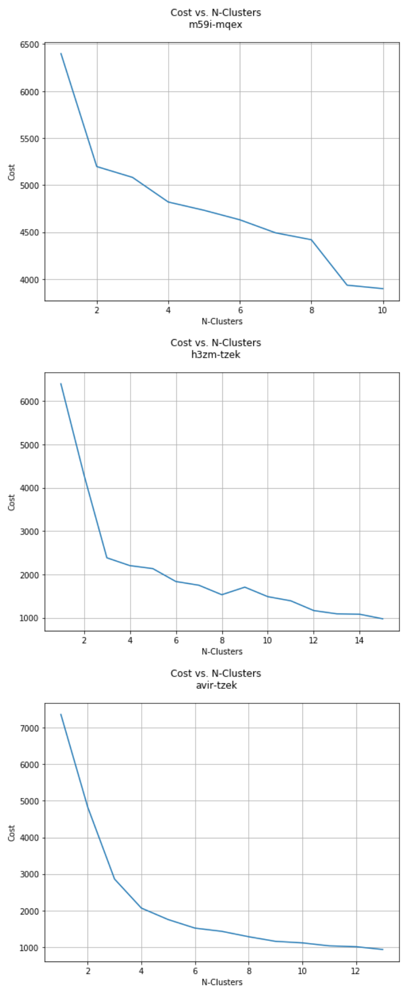

Figure 3: Elbow plots for a number of datasets, including the two w/results shown

[8] Efficient Algorithms for Mining Outliers from Large Data Sets Sridhar Ramaswamy, Rajeev Rastogi, Kyuseok Shim https://webdocs.cs.ualberta.ca/~zaiane/pub/check/ramaswamy.pdf

[9] Kaustav Das, Jeff Schneider. Detecting Anomalous Records in Categorical Datasets. https://www.cs.cmu.edu/~schneide/KaustavDas07.pdf

[10] Michael K. Ng, Mark Junjie Li, Joshua Zhexue Huang, Zengyou He https://pdfs.semanticscholar.org/73b4/739bbc6c8e95e5fc7853a706a527c54a6dc9.pdf

[11] Fuyuan Cao, Jiye Liang, Liang Bai https://pdfs.semanticscholar.org/1955/c6801bca5e95a44e70ce14180f00fd3e55b8.pdf

[12] https://github.com/nicodv/kmodes

[13] Feng Jiang, Guozhu Liu, Junwei Du, Yuefei Sui https://pdfs.semanticscholar.org/a7fe/e603669bc5347fe857cdba39b67927e5846b.pdf