Discovering Null and Outliers by K-means Cluster and Attribute Value Frequency Algorithms

Final Report

Yulin Shen
New York University NYC
ys2542@nyu.edu

Yitao Liu
New York University Shanghai
yl3438@nyu.edu

Yiyang Sun
New York University Shanghai
ys2380@nyu.edu

ABSTRACT

In the project, we have 50 different datasets with various numbers and types of attributes. Also, the number of instances in the datasets are different, and some of them can be treated as the large-scale datasets. Our main goal is to use Spark RDD to find outliers in a large-scale dataset environment, because RDD is an efficient data structure in python. Also, we have finished implementing K-means cluster algorithm and Attribute Value Frequency algorithm both with some new improvements to find outliers in an efficient way. The results of them can be compared for accuracy checking.

1 INTRODUCTION

As we known, a large-scale dataset must have some abnormal rows of data. If we directly train the dataset or use the dataset for analysis without anomaly detection and data clean, it may give us very bad performance in the machine training project with bad prediction accuracy, and it may affect the analysis result heavily to make a wrong decision. So, it is very important to do anomaly detection to help for data clean before we utilize the dataset.

Discovering null and outliers is the key part to do anomaly detection. It is trying to find instances, which do not conform to other items in a dataset, and null instances with a lot meaningless attributes. The detection sometimes can help for some kinds of applications such as credit card fraud detection, medical errors, and so on.

There are a lot popular techniques to detect outliers in a dataset, such as density-based algorithm like KNN, tree-based algorithm like isolation forest, cluster-based algorithm like K-means, and so on. However, the size of dataset becomes larger and larger in the modern era today. Many algorithms can do find outliers with a very long time. Our goal is to find the accurate outliers with a higher speed.

First algorithm we thought is based on K-means, the vanilla K-means algorithm uses Euclidean distance, which only deals with numerical data, and it has a slow time complexity O(n^2). However, K-means package in the spark with RDD structure is implemented based on kmeans||, a parallelized variant of the k-means++, which should improve the efficiency in Spark environment than using vanilla K-means algorithm. Another algorithm we used is called attribute value frequency, this algorithm has a linear time complexity, and is used to deal with some text data like categorical data.

The two algorithms deal with only one kind of data respectively, so we have improved both of them in the project. In a word, we decide to make sure we have an efficient algorithm, which can successfully detect outliers with an accurate prediction performance at first. Then we try to adjust both two algorithms to deal with both categorical and numerical data. We can compare their results to verify their accuracies, also their time-consuming to compare their efficiencies.

2 PROBLEM FORMULATION

At first, we are assigned 50 different datasets with different rows, columns, and data types. We need to design a system to detect outliers of all of the datasets automatically. So, beside outliers discover algorithms, we still need to design an data kind detection to make sure each column which is numerical or categorical.

At second, we also need to detect null attribute in the datasets. The easiest way is to discover all rows of data having null attributes. However, it is definitely not a good idea to clean the dataset. Some null information in the attributes are also meaningful. Also, some rows of data may only have one to two null information. It do provides information for training and analyzing the dataset. We could set a threshold number to judge whether to keep the row, because this kinds of data are not outliers.

At third, almost of numerical data columns have a lot of typos. At this time, the code could not successfully do mathematical calculation for it. We need to design a dirty data detection to filter them out.

At fourth, almost 50 datasets both have numerical attributes and categorical attributes. K-means cluster algorithm and AVF algorithm are both based on one kind of data columns. How to deal with both two data columns is a big problem. Our idea is to translate categorical data into numerical data in K-means clustering and numerical data into categorical data in AVF. So we need to come up with a standard idea to do translation.

At fifth, we need to use K-means cluster algorithm to find outliers in the project now. However, there is not a general K value for all datasets to perform the best discovery with highest accuracy. We may do a brute-force search for K value to check each K value to have a best performance.

At sixth, different numerical data columns have various range. If we directly use all of them to calculate the distances, some of columns

may have large weights which have large ranges. So, we need to standardize all columns in a standard range before doing K-means.

Finally, not all text attributes are categorical data like people's address. So we still need pay more attention and time on selecting the useful categorical attributes.

3 RELATED WORKS AND REFERENCES

Kyung-A Yoon, Oh-Sung Kwon, and Doo-Hwan Bae[2] introduce and define internal and external outliers in a dataset at first. Then they discuss K-means cluster algorithm mathematical architecture and theory to detect outliers in a dataset. Also, it shows some procedures to test different database, and the way to validate the outliers. Finally, they summarize each performance and time complexity to show K-means works for discovering outliers.

A. Koufakou,E.G. Ortiz, M. Georgiopoulos, G.C. Anagnostopoulos, K.M. ReynoldsA[1] presented a frequency based linear scalable algorithm called Attribute Value Frequency (AVF). They say some outlier detection algorithms consume quadratic time with respect to the dataset size. They are not quick for large-scale datasets. Their AVF algorithm scales linearly with the number of data points and attributes. They compare the algorithm with the distance-based algorithm and cluster-based algorithm. Also, they introduce its mathematical architecture, and show the result of each algorithm.

4 METHODS, ARCHITECTURE AND DESIGN

4.1 Data kind judgment

Step 1: We assume categorical data columns have 50 classes at maximum. So, we find all distinct values in all columns to keep all columns, whose numbers of distinct values are less than 51, as categorical data columns.

Step 2: In the remaining columns, we check the kind of data value. If they are almost numerical data, we can set the column as a numerical data column. Or, it is a text column. In the project, we will throw away it, because it is not an useful categorical data column.

Step 3: Print the column number of each categorical and numerical data column.

4.2 Null data detection

Step 1: We use an if statement to find all null values in the column one by one in a linear searching.

Step 2: We count the number of null values in each row. And we can set a threshold number to throw away all rows whose number of null values is more than this threshold number. In our program, we set 50% as the threshold, that is, if more than 50% of the values in the row are null, we classify the row as a null row.

Step 3: Print all the row number when we clear the row out.

4.3 Dirty data clean

Step 1: After finding all numerical columns, we use try and except statement to convert all numerical value with string format into float format.

Step 2: Delete the typo row and print all the row numbers when we clear the row out.

4.4 K-means cluster

We use K-means clustering algorithm to train both categorical data and numerical data.

Step 1: Getting data after data preprocessing, receiving the number of clusters and percentage of the anomaly.

Step 2: Transfer the categorical data in the selected columns to numeric data, the detailed method is to change the categorical data to the frequency it appears in the column. Consider one column of data [a, a, b, a, c] where "a" appears 3 times, "b" and "c" appear one time, then the numeric expression of that column is [0.6, 0.6, 0.2, 0.6, 0.2].

Step 3: Format data to "int" or "float". Use Spark K-Means model from pyspark.mllib to train dataset, and get the result of clusters.

Step 4: Normalizing all numerical data column so that they all have same range.

Step 5: Add each line (tuple) to clusters and calculate each's distance to cluster's center. Distance for each line (i-dimension tuple) is defined as:

$$d = min(\sqrt{(x_0 - c_{0,0})^2 + ... + (x_i - c_{0,i})^2}, ..., \sqrt{(x_0 - c_{k,0})^2 + ... + (x_i - c_{k,i})^2})$$

where k is the cluster number, i is the column number, $(x_0, x_1, ..., x_i)$ represents the line (tuple), and $(c_{k,0}, c_{k,1}, ..., c_{k,i})$ represents the center of cluster k. Consider a 3-dimension tuple (1,0,1), and two clusters with centers (0,0,0) and (1,1,1), then the distance for tuple (1,0,1) is:

$$d = min(\sqrt{(1-0)^2 + (0-0)^2 + (1-0)^2}, \sqrt{(1-1)^2 + (0-1)^2 + (1-1)^2}) = min(\sqrt{2}, 1) = 1$$

Step 6: After calculating the distance for every rows (tuples), order the rows from the highest distance to the lowest distance. Then, select top n% rows as outliers, where n is defined by user as percentage.

4.5 Attribute value frequency

Like K-means approach, we also use AVF algorithm to process both categorical data and numerical data.

Step 1: Getting data after data preprocessing, as described in section 4.1, 4.2, and 4.3, receiving the percentage of the anomaly.

Step 2: Transfer the numerical data to categorical data, and we use bucket approach to accomplish this transformation. We set 50 buckets, and each bucket has the interval of $\frac{(max-min)}{50}$, where max and min are maximum and minimum value in the column. Then, put the numerical into the buckets, and change the numbers in the column to the bucket number; for example: if numerical data 23 falls in bucket number 3, the number in the column will change to 3 from 23. With this process, all numerical data will change to 50 bucket numbers, which act as categorical data.

Step 3: Change data into frequency, with the same method shown in section 4.4 step 2.

Step 4: For the AVF algorithm, we assume the database has m rows and n columns. We represent the whole dataset as X where X = $[x_1, x_2, \ldots, x_m]$(1<= i <=m) and x_i itself consists of n values as its attributes.

The idea of the algorithm is that an outlier is much possible to be the more infrequent record. Therefore, the algorithm counts the number of times of an attribute occurs in a column. For all the columns, the algorithm sums up all the frequencies of records in the row. The AVF score of a record is the average of the sum of frequencies, lower score means more likely it is an outlier.

For each row, $AVFScore(x_i) = \frac{1}{m}\sum_{l=1}^{m} f(x_{il})$. Then k records with lowest AVF scores are chosen as outliers. The algorithm only requires $n \times m$ passes through the dataset. Therefore the time complexity is O($n \times m$), which means it is a linear computation algorithm.

Step 5: After calculating the AVF score for each rows, order the rows from the lowest AVF score to highest AVF score. Then, select top n% rows as outliers, where n is defined by user as percentage.

5 RESULTS

5.1 Result interpretation

In our programs, number of outliers is defined as a percentage of totals rows the dataset have. For most datasets, we set the percentage as 5% (or some number around 5%); for some extreme large datasets, we decrease the percentage to 1% or less than 1%.

Among the 50 datasets given, we have run most of them. Because of the space limitation of this report, we can only show some examples of results. Full results of the data sets we ran can be found on our `GitHub repository`.

For the results from K-means algorithm, we list numerical columns, categorical columns, null rows, and dirty rows in the front, and following are the outputs of outliers. Outputs are formatted in "A, B, C", where A is the row number (where the first line of original dataset is row no. 0), B is the cluster ID where that row is in, and C is the distance between the cluster's center and the point which represents that row.

Take the output of dataset "6bic-qvek.tsv" as an example, in the output, it lists numeric columns and categorical columns:

Num Cols: [0, 1, 3, 4, 5, 28, 33, 35, 42, 45]

Cat Cols: [2, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 29, 30, 31, 32, 34, 36, 37, 38, 39, 40, 41, 43, 44]

These results mean our program classified column no. 0, 1, 3, 4, 5, 28, 33, 35, 42, 45 as numerical columns; and column no. 2, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 29, 30, 31, 32, 34, 36, 37, 38, 39, 40, 41, 43, 44 as categorical columns.

Following lists show the null rows and dirty rows:

Null Rows: []

Dirty Rows: []

Empty lists of null and dirty rows means our program does not discover the "null row" and "dirty row" in this dataset.

Content followed by the null and dirty rows is the outlier our program discovered. For this dataset, we set 5% of the rows are outliers. After K-means clustering, we get following outliers:

Row No.	Cluster No.	Distance
68	2	14.528966
229	2	11.780747
73	2	11.669646
63	1	11.661056
232	2	11.293254
74	2	11.271663
222	2	10.451683
210	2	10.444890
218	2	10.301228
130	2	9.741315

As shown in the above table, outliers are in the descending order of each row's distance to the cluster center.

For the results from AVF algorithm, we also list numerical columns, categorical columns, null rows, and dirty rows in the front, as described in above. The following content is the outlier discovered by AVF algorithm, formatted in "A, B", where A is the row number, and B is the AVF score. For the dataset "6bic-qvek.tsv", we set 5% of the rows are outliers:

Row No.	AVF Score
68	11.275424
74	12.889831
125	14.394068
65	14.838983
119	14.855932
124	14.957627
135	15.097458
129	15.097458
130	15.699153
70	16.487288
58	16.525424

As shown in the above table, outliers are in the ascending order of each row's AVF score.

Compare K-means results with AVF results, we find there are some common outliers. For example, row No. 68 ranks first in both K-means and AVF results, thus this row has the highest probability as

an outlier. Row No. 74, 130 also appear in both results.

Other results from the datasets we have run are on our `GitHub repository`. Some of the results may contains no outlier, for example dataset "7ujc-hpzg.tsv" has 3 rows in original data, it is worthless to find an outlier out of 3 rows. For other datasets that contain only a few rows, one can increase the user defined outlier percentage, for example, from 5% to 50%, but that will also be worthless because outlier should be a small percentage of the dataset. Some results may contain the results that are meaningless, such as, for dataset "7dfh-3irt.tsv", our program finds no column is useful in finding outliers, that is, we filter out all columns in the procedure described in section 4.1; therefore, the results are all 0s, which has no meaning.

5.2 Time comparison

We have plotted the running time of the datasets with K-means and AVF algorithms with the size of the corresponding datasets, as shown in Figure 1 and Figure 2. Both algorithms have similar running time.

K-means algorithm performs well while the size of input dataset is in the range we run (from 1KB to around 1,800,000KB). This is mainly because of 2 reasons:

First is K-means used in Spark library is not vanilla K-means; it is kmeans|| instead, which is a parallelized modification of the k-means++. Therefore, it has really good performance in the size range mentioned above, which gives almost linearly time growth with the increase of size.

Second reason is the process of transferring data to frequency (section 4.4 step 2 and section 4.5 step 3) heavily rely on the dictionary in our codes. While K-means program only need to transfer the data in categorical columns; AVF program need to transfer data in both numerical and categorical columns, because we have already transferred the numerical data into categorical data by bucket approach in the previous steps. Therefore K-means program is actually dealing with the data with less dimensions than the AVF program, because the sum of the number categorical and numerical columns should always be larger or equal to merely numerical columns.

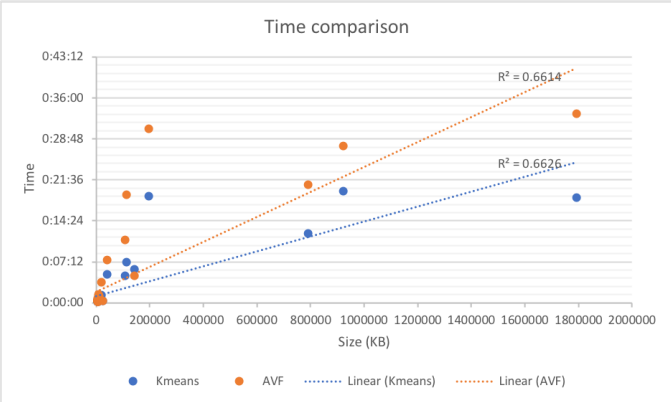

Figure 1: Relationship of dataset size and running time.

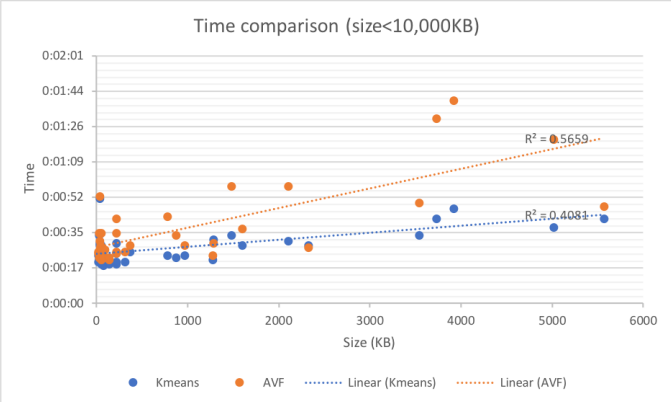

Figure 2: Relationship of dataset size and running time (size < 10000KB).

6 TECHNICAL DEPTH AND INNOVATION

K-means cluster algorithm and attribute value frequency algorithm are two widely-used outlier-detection algorithms. But they focus on different aspects of the datasets to detect outliers. The former is mainly fit for numerical dataset, and the latter is for categorical dataset. However, a dataset always has both kinds of data. We have improved both of them to deal with both kinds of data columns. We make an innovation to detect the type of data columns automatically. This innovation could classify the data columns into numerical, categorical, and useless text data columns.

After classifying numerical data columns and categorical data columns, we can use the concept of attribute value frequency algorithm to prepare the data for K-means clustering. We can also calculate the frequency of categorical data and translate it to the numerical data. Then we can use Euclidean distance calculation with same weights due to normalization for each column. That is our innovation for K-means cluster algorithm based on the idea of attribute value frequency algorithm.

In addition to the improved K-means, we also improve AVF algorithm to fit for both numerical and categorical data columns. We use the concept of buckets to divide numerical data column into 50 buckets to set 50 different classes. Then we can treat this data column as categorical data column. Now, AVF algorithm could use all of data columns. So, we have 2 algorithms to deal with any kinds of dataset, and one of them is efficient.

Also, not all null value in the column is meaningless. We need to judge which null value is useful. We develop a threshold number based on the number of columns as the standard to filter useless null row out. That is our innovation to deal with some null attributes, and we can generalize it to all the datasets.

7 CODE REPOSITORY, CORRECTNESS, AND READABILITY

We have used a Github repository to hold our project code. The `link` of the repository is https://github.com/ys2542/DS-GA-1004 .

8 CONCLUSIONS

In the project, we clean the data at first to filter out null and dirty rows. Also, we have successfully implemented both K-means cluster algorithm and AVF algorithm to detect outliers. Both of them are improved to use for any kinds of datasets with different data kinds. Their results have some overlaps and look reasonable.

A HEADINGS IN APPENDICES

A.1 Introduction

A.2 Problem formulation

A.3 Related works and references

A.4 Methods, architecture and design

A.4.1 Data kind judgment.

A.4.2 Null data detection.

A.4.3 Dirty data clean.

A.4.4 K-means cluster.

A.4.5 Attribute value frequency.

A.5 Results

A.5.1 Result interpretation.

A.5.2 Time comparison.

A.6 Technical depth and innovation

A.7 Code repository, correctness, and readability

A.8 Conclusions

A.9 References

REFERENCES

[1] Anna Koufakou, Enrique G Ortiz, Michael Georgiopoulos, Georgios C Anagnostopoulos, and Kenneth M Reynolds. 2007. A scalable and efficient outlier detection strategy for categorical data. In *Tools with Artificial Intelligence, 2007. ICTAI 2007. 19th IEEE International Conference on*, Vol. 2. IEEE, 210–217.

[2] Kyung-A Yoon, Oh-Sung Kwon, and Doo-Hwan Bae. 2007. An approach to outlier detection of software measurement data using the k-means clustering method. In *Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on*. IEEE, 443–445.