Journal of Engineering and Applied Sciences 12 (23): 7389-7392, 2017
ISSN: 1816-949X

Frequency-Based Fast Algorithm for Anomaly Detection in Big Data

Adeel S. Hashmi and Tanvir Ahmad
Department of Computer Engineering, Faculty of Engineering,
Jamia Millia Islamia, New Delhi, India

Abstract: Anomaly/outlier detection is an important area of machine learning which finds its application in intrusion-detection, fraud-detection, etc. In recent times, the focus of data analytics has shifted to big data analytics, i.e., analytics on large-scale data and fast-moving data streams. The traditional data processing tools and algorithms are not able to handle big data, so, there is a need of algorithms to be implemented in a parallel model like MapReduce to solve this problem. In this study, the researchers implement frequency-based algorithm on Spark MapReduce as a scalable and accurate solution for anomaly detection on large-scale as well as streaming datasets.

Key words: Data mining, distributed computing, parallel processing, predictive models, machine learning, tools

INTRODUCTION

Anomaly detection or outlier detection is the process of identifying abnormal data in the given dataset. A simple example of anomaly detection is intrusion detection where some unusual activity is identified to detect the intruder. There are various unsupervised and supervised learning techniques for anomaly detection (Agrawal and Agrawal, 2015) like cluster analysis based outlier detection, local outlier factor, KNN, replicator neural networks, support vector machines, ensemble learning, sub-space and correlation based outlier detection, etc. ELKI (Achtert *et al*., 2008) is an open-source java-based data mining toolkit which provides several anomaly detection algorithms like EM-outlier, LOF, OPTICS-OF, DB-outlier (distance-based), LDOF (Local Distance-based Outlier Factor), LOCI (Local Correlation Integral).

With the rise of Big data and IoT (Internet of Things), there is a need for analytics on large-scale data as well as real-time analytics and one of the most important areas is outlier detection which could help us detect anomalous data points and take necessary action. In this study, the researchers aim to develop a big data solution to perform outlier detection on large datasets as well as on fast-moving data streams. The aim of the solution is scalability along with accuracy as the available algorithms are not able to scale-up to the growing datasets. The algorithm that has been selected to develop the solution is frequency-based which has been parallelized in MapReduce, so that, it can be implemented in a distributed environment to reduce the processing time.

Big data: The term, "big data" is used for those datasets which can't be handled by traditional data management and processing tools due to its large volume, the rate at which it is generated or the variety of data in it. These are called as 3V's of big data: volume, velocity and variety. However, some authors have introduced more dimensions like veracity, value, etc. The traditional DBMS (Oracle, MySQL, DB2, etc.) and data processing/mining tools (like MS-Excel, Weka, SPSS, etc.) have a certain limit on the volume and dimensions of data which they can handle and are neither able to scale-up to growing data, nor they are able to process fast moving data streams. So, these tools are clearly not able to satisfy the needs of a big data tool, i.e., ability to scale-up to data, perform fast analysis (preferably parallel/distributed), ability to handle data streams and ability to handle different variety of data. Moreover, traditional algorithms (data mining) are also not able to fulfill the needs of big data as they are unable to provide fast results and perform real-time analytics.

To handle such datasets (large, streaming, growing, multi-source, multi-variety), we need different tools and algorithms (Hashmi and Ahmad, 2016a). One of the most popular tools for handling big data is Hadoop/YARN which provides a distributed storage (HDFS-Hadoop Distributed File System) and a parallel programming model MapReduce which helps a programmer to write parallel/distributed applications that can run on a Hadoop cluster to handle large files. The major limitation of Hadoop is that it can't handle data streams but it supports other data stream processing engines like Apache Storm. Apache Spark is tool which can be used as an alternative

Corresponding Author: Adeel S. Hashmi, Department of Computer Engineering, Faculty of Engineering, Jamia Millia Islamia, New Delhi, India

to Hadoop or also can be used with Hadoop. The major advantage of Spark is that, it can handle data streams and writing MapReduce programs is simpler as well.

As far as mining of big data is concerned, to perform data analytics on such datasets we can perform sampling, incremental learning or distributed processing (Hashmi and Ahmad, 2016b). The most popular approach being distributed processing obviously as it gives more accurate results while consuming less time, however, if we are processing a data stream then sampling or incremental learning may be used. The machine learning algorithms also need to be modified (parallelized) for distributed learning and tools like Apache Mahout, H_2O and Spark MlLib provide libraries of such algorithms which can be used to handle large datasets. However, writing our own distributed machine learning algorithm in MapReduce is quite challenging.

Outlier detection: An outlier detection algorithm for big data must be able to cope with volume (dimensions/scale), velocity and complexity of data. Big data is often high dimensional which causes data points to become sparse which makes concepts like Euclidean-distance and nearest-neighbor less applicable (Ertoz *et al*., 2003). So, the algorithm must be able to handle high dimensional and sparse data. These algorithms must also be able to minimize false negatives/positives considering the cost of analyzing each anomaly.

Koufakou *et al*. (2008) proposed Attribute Value Frequency (AVF) algorithm for fast outlier detection in a categorical datasets. The AVF method is based on a simple concept that the outliers are those points which have low frequencies. So, in AVF method an AVF score is assigned to each data point which is the sum of the frequencies of attribute values.

Liu *et al*. (2013) proposed a fast statistics based model for outlier detection in big data. The researchers proposed an Entropy-based Fast Detection (EFD) algorithm which utilizes the concept of entropy for outlier detection. The 'k' data points which increase the entropy of the system the maximum are considered to be the 'k' outliers. The purpose of the algorithm is to find this subset of 'k' points from the given dataset.

For detecting outliers in large and very large datasets, a statistical method was proposed by Das and Mandal (2004). They made use of Tukey's bi-weight function to obtain location and scale estimates of the data. Mahalanobis distances were calculated using these estimates for all data points. Next the probability density curve of the Mahalanobis distances by Parzen window was utilized. Those points whose Mahalanobis distances have very low probability density are the outliers.

RAD (Rapid Anomaly Detection) algorithm is used by NetFlix for outlier detection. RAD is able to handle high cardinality dimensions, non-normalized datasets, seasonality and minimizes false positives. RAD is based on Robust principal component analysis (Candes *et al*., 2011). RPCA repeatedly calculates SVD (Singular Value Decomposition) and applies thresholds to singular values in each iteration.

Yan *et al*. (2015) proposed a distributed outlier detection algorithm which employs compressive sensing for sampling high-dimensional data. A vector $x = [x_1, ..., x_n]^T \in R^N$ is called majority-dominated if there exists $c \in R$ such that the set $O = \{i: x_i \neq c\}$ has $|O| > N/2$. The koutlier problem is defined as finding a set $Ok \subseteq O$ with min $(k, |O|)$ elements that are furthest away from. Formally, for any $i \in O_k$ and $j \notin 0_k$, we have $|x_i - c| = x_j - c|$.

A popular approach of outlier detection in big data is through clustering as all big data mining libraries like Apache Mahout, Spark MlLib, H_2O, etc. provide a parallel implementation of popular k-means algorithm. For example by Souza and Amazonas (2015), the researchers utilized canopy k-means clustering algorithm available in Apache Mahout library.

Bay's parallel algorithms for distance-based and density-based outliers (Lozano and Acufia, 2005) are suitable candidates for outlier detection in big data. The main idea is to keep track of closest neighbors for each instance in the dataset and assign a score (sum of distances of k neighbors) to the instance. If the score is lower than a cut-off value then the instance is removed indicating that it can't be an outlier.

MATERIALS AND METHODS

AVF (Attribute Value Frequency) is one of the simplest and fastest algorithms for outlier detection. The algorithm is intuitive in the sense that the outliers are those data points which are less frequent in the dataset. For a multi-featured data points, the frequency is the sum of the frequency of its attribute. An outlier can be a point which has extremely high or extremely low frequency sum compared to other points in the dataset.

Let's assume there are n instances/points in the dataset D and each instance has m attributes. Each instance is labeled as D_i and $D_i = \{x_{i1}, x_{i2}, \ldots, x_{ij}, \ldots, x_{im}\}$ where, x_{ij} is the value of the j^{th} attribute of D_i. The AVF score of each data point can be calculated as:

$$AVF(D_i) = \sum_{j=1}^{j=m} f\left(D_{ij}\right) \quad (1)$$

where, D_{ij} is the number of times the corresponding value appears in the j^{th} column. The AVF algorithm is a simple algorithm and can be easily parallelized as shown by Koufakou *et al*. (2008) where, it was implemented in Hadoop MapReduce showing quite promising results. However, the researchers of this study chose to parallelize it using Apache Spark (PySpark) as the code is much compact in Spark as compared to Hadoop and Spark is also known to give faster results compared to Hadoop (Algorithm 1).

AVF; Algorithm:

```
    Input: Dataset D (n points, m attributes)
    Output: k points (outliers)
    Calculate frequency of each attribute
value, f(Dij);
 foreach point Di (i = 1, ..., n)
    foreach attribute j (j = 1, ..., m)
        AVF(Di) += f(Dij)
    end
end
Return top k outliers with minimum/maximum AVF score
```

The steps in Spark implementation of AVF are: select an attribute, count values column-wise and produce "columnar" frequencies, broadcast the attribute frequencies (i.e., keep a read-only cached copy on each machine), assign the appropriate frequency to each record's attribute from the broadcast and finally add the frequencies in each row to get the sum of the frequencies in the corresponding row. This sum of frequencies in a row is the AVF score of the row (Algorithm 2).

Algorithem; PySpark Implementation of AVF:

```
for i in range(0,record_length):
  attribute = records.mapValues(lambda
x:x[i])
 attribute_frequencies =
attribute.values().countByValue()
 bi =
sc.broadcast(attribute_frequencies)
  attribute_freq =
attribute.mapValues(lambda x:
bi.value[x])
   if I:
      tsum =
t.join(attribute_freq).mapValues(sum)
      else:
         tsum = attribute_freq
t = tsum.sortBy(lambda x:x[1])
```

This algorithm can be easily extended to identify anomalies in the data streams. To do so, we need to divide the process into training and testing phases. In the training phase, we generate the frequency tables and generate frequency sum of each row. For the testing phase, we mark the frequency sum thresholds. In testing phase, we calculate the frequency sum of each instance using the frequency table of training phase and the instance which breaches the threshold is marked as an outlier.

RESULTS AND DISCUSSION

The performance of AVF (on Spark) was compared with k-means based outlier detection (on Spark), local outlier factor algorithm (on GraphLab) and replicator neural network (on H_2O). The datasets used for the experiments are the covertype/forest cover, kddcup99 and HIGGS. The covertype dataset has 581012 instances with 54 attributes. The kddcup99 dataset has 4000000 instances with 42 attributes. The HIGGS dataset has 11000000 instances with 28 attributes. The experiments were conducted on a cluster of i5 processors (dual-core) with 4GB of RAM each and the cluster consisted of 6 nodes.

Table 1 shows the execution time taken (in minutes) by each of the selected algorithms for finding anomalies in the chosen "big" datasets.

As far as accuracy of the algorithm is concerned, experiments were conducted on the small datasets like Wisconsin Breast cancer, lymphography, post-operative patient data and page-blocks. Table 2 shows the performance analysis of the AVF algorithm evaluated using metrics like sensitivity, specificity and accuracy.

$$\text{Sensitivity} = \frac{TP}{TP+N} \tag{2}$$

$$\text{Specificity} = \frac{TN}{TN+FP} \tag{3}$$

$$\text{Accuracy} = \frac{TN+TP}{TN+TP+FN+FP} \tag{4}$$

The accuracy of the AVF algorithm was found to be at par with the rest of the algorithms. From the results, it can be inferred that AVF is the fastest algorithm for anomaly detection in big data while being nearly as accurate as the state-of-art algorithms for anomaly detection.

Table 1: Execution Time (in minutes)

	Algorithm			
Data-set	k-means	LOF	RNN	AVF
Forest cover	1.58	1.37	8.34	1.12
kdd-cup99	2.56	2.09	14.68	1.56
HIGGS	6.24	6.18	47.21	5.44

Table 2: Performance of the solution

Dataset	Sensitivity	Specificity	Accuracy
Breast cancer	98.42	82.05	97.10
Lympho-Graphy	98.59	66.66	97.29
Post-Operative	73.43	34.61	62.22
Page-blocks	98.37	42.85	96.83

CONCLUSION

Anomaly detection is a challenging task and it becomes even more challenging if it is to be done on large or streaming datasets. This study gives an easy yet powerful solution to anomaly detection in big data. The research done in this study has wide research implications as it can be used for anomaly detection on large volumes of data in areas like fraud-detection, intrusion-detection, identification of cancerous cells, etc. The search is significant in the sense there are no limitations on the data size which can be processed and the results are fast and accurate. The algorithm implemented in Spark is scalable as one just needs to increase the cluster size to handle larger dataset and to decrease the turn-around time.

REFERENCES

Achtert, E., H.P. Kriegel and A. Zimek, 2008. ELKI: A Software System for Evaluation of Subspace Clustering Algorithms. In: Scientific and Statistical Database Management, B. Ludascher and N. Mamoulis (Eds.). Springer, Berlin, Germany, ISBN:978-3-540-69476-2, pp: 580-585.

Agrawal, S. and J. Agrawal, 2015. Survey on anomaly detection using data mining techniques. Procedia Comput. Sci., 60: 708-713.

Candes, E.J., X. Li, Y. Ma and J. Wright, 2011. Robust principal component analysis? J. ACM, Vol. 58. 10.1145/1970392.1970395

Das, P. and D. Mandal, 2004. Statistical outlier detection in large multivariate datasets. Master Thesis, Netaji Subhash Engineering College, Kolkata, India.

Ertoz, L., M. Steinbach and V. Kumar, 2003. Finding clusters of different sizes, shapes and densities in noisy, high dimensional data. Proceedings of the 3rd SIAM International Conference on Data Mining, May 1-3, 2003, San Francisco, pp: 47-58.

Hashmi, A.S. and T. Ahmad, 2016b. Big data mining techniques. Indian J. Sci. Technol., Vol. 9, 10.17485/ijst/2016/v9i37/85826

Hashmi, A.S. and T. Ahmad, 2016a. Big data mining: Tools and algorithms. Intl. J. Recent Contrib. Eng. Sci. IT., 4: 36-40.

Koufakou, A., J. Secretan, J. Reeder, K. Cardona and M. Georgiopoulos, 2008. Fast parallel outlier detection for categorical datasets using MapReduce. Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN'08) and IEEE World Congress on Computational Intelligence, June 1-8, 2008, IEEE, Hong Kong, China, ISBN:978-1-4244-1820-6, pp: 3298-3304.

Liu, B., W. Fan and T. Xiao, 2013. A Fast Outlier Detection Method for Big Data. In: Communications in Computer and Information Science, Tan, G., G.K. Yeo, S.J. Turner and Y.M. Teo (Eds.). Springer, Berlin, Germany, ISBN:978-3-642-45036-5, pp: 379-384.

Lozano, E. and E. Acufia, 2005. Parallel algorithms for distance-based and density-based outliers. Proceedings of the 5th IEEE International Conference on Data Mining, November 27-30, 2005, IEEE, Houston, Texas, ISBN:0-7695-2278-5, pp: 1-4.

Souza, A.M. and J.R. Amazonas, 2015. An outlier detect algorithm using big data processing and internet of things architecture. Procedia Comput. Sci., 52: 1010-1015.

Yan, Y., J. Zhang, B. Huang, X. Sun and J. Mu *et al.*, 2015. Distributed outlier detection using compressive sensing. Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, May 31-June 04, 2015, ACM, Melbourne, Victoria, ISBN:978-1-4503-2758-9, pp: 3-16.