Outlier Detection in Data Streams
A Research Survey

Aditi Singla (2014CS50277)
Ankush Phulia (2014CS50279)
Vaibhav Bhagee (2014CS50297)

September 30, 2018

---

1 Introduction

In data mining, the task of Outlier Detection is to identify those data points in a dataset which differ significantly from the rest of the points in the dataset and occur in rarity. There are various examples of scenarios where detecting outlier data samples is of use for eg. fraud detection in the financial sector, separation of "noisy" samples from the normal ones to refine the dataset for further processing, detection of "anomalous" event samples in system logs, to prevent large scale system failures etc.

Traditional algorithms have looked at outlier detection in scenarios where the dataset is "static", i.e the data does not change over the period of time and the algorithm has the access to the entire dataset, in memory. However, as the size of the datasets grows large, it might not be always easy to get the entire dataset in memory, for processing. Moreover, there are many scenarios like prevention of system failure, where the data samples like logs, are generated temporally, in a continuous fashion. In such cases, the outlier detection algorithm can never have access to the complete dataset and the analysis for outliers needs to be performed over the "seen" data.

As a part of this survey, we present and discuss various algorithms and techniques, which have been proposed in context of detecting outliers in the setting of streaming data. In particular, we discuss the challenges which are posed when detecting outliers in data streams what approaches are followed to overcome these.

2 Motivation

Outlier detection on large datasets has been traditionally looked at alongside clustering and density estimation. Many of the earliest outlier detection algorithms have been the ones performing clustering and in process identifying the "noisy" samples. However, in high dimensional data[2], the neighbourhood and clustering based outlier detection algorithms, face issues due to sparsity, failing to escape the curse of dimensionality. In order to avoid the distance computation among the points, exact and approximate algorithms have been proposed, based on projection of data onto smaller dimensions. The basic intuition behind this set of algorithms is that the outlier behaviour of a point is more pronounced in the subspaces of a high dimensional vector space. Hence, these algorithms are known to give better results than the former set of algorithms.

A key assumption which these algorithms make is that the entire dataset is available for processing, in memory. For smaller datasets, this assumption is decent enough to make. However, data has been growing at an exponential rate and modern data problems related to knowledge discovery in databases work on very large datasets, which can be of giga scale or tera scale in size. These datasets cannot be stored in the main memory for processing and therefore need to be chunked and processed. This is an example of a stream which is of finite size.

In addition to that, consider the problem of failure detection in systems, which has been described above. In such a typical modern day distributed system, there are multiple micro-services which send their logs to a central process responsible for collection and analysis of logs. In such a case, detection and analysis of log lines corresponding to "anomalous" events, only has access to the logs which have been collected by the process up to that instance of time. Hence, the outlier detection algorithm is looking at a stream of data which is potentially infinite in size.

Recent work[3] looks at streams which can evolve over time in multiple ways:

- Feature values of existing data points can change over time
- New unseen features can emerge over a period of time

These type of feature-evolving streams are common in scenarios like data center monitoring, where the jobs, which are submitted over time, can have feature values like number of unsuccessful retries, number of threads, system calls etc. and can have features, which get added over time, like URLs of log files, repositories etc.

The existing algorithms and techniques for data streams fail to work for feature evolving streams as they make an assumption that the data streams cannot change dimensionality and the values of existing samples cannot be modified.

The authors propose an efficient algorithm for this case, which generalises for the static and the row streams as well.

3 Problem Formulation

The problem we are looking at essentially, is as follows: Given a data stream $D = \{x1, x2, ...\}$, we want to identify the outlier data points.

There have been various techniques[5][4][1] which have been proposed to deal with the streaming nature of the data. Among the most popular ones are the ensemble based techniques and subspace partitioning based techniques.

3.1 Ensemble based techniques

Ensemble based techniques[1] have gained popularity after their performance in supervised setting of classification in machine learning. The idea behind ensembles is to utilise the power of many. Multiple models can be trained over the same dataset or multiple subsets of the dataset and their outputs or scores can be suitably combined at to produce the output or the score for the given data point. The component of an ensemble can be trained in a way such that they could be dependent on each other, like sequential ensembles or they could be independent of each other, like in feature bagging.

3.2 Partitioning based techniques

Streams have outlier samples in rarity and can evolve over time, besides being on potentially infinite length. Partitioning based techniques[4] essentially construct an ensemble of trees which perform semi supervised, one-class detection of the rare outliers. These techniques attempt to detect outliers by capturing the density of points in multiple different partitions of the subspaces. The key idea to be able to incorporate for the evolving nature of the streams, is to maintain contiguous, latest and reference windows, to perform outlier detection on the unseen data while updating the reference model atomically with the latest one, after the latest window ends.

The feature-evolution setting can be described as follows: Given a data stream D containing updates of the form (id, f, δ) where f could be a feature which has been seen or unseen until that instant and id is the unique identifier corresponding to the data point. δ is the update to the feature value of the data point as described above.

Due to the feature evolving nature of the stream, the set of points need to be in memory all the time. Given the fact that new features can be added, the space allocation for the data points can't be bounded before hand and hence is

challenging. Also, the feature updates need to be fast and efficient.

The algorithm proposed by the authors is a constant space and constant time per update algorithm which combines ideas from subspace partitioning and ensembling while incorporating the feature evolving nature of the input stream. The details of the algorithm along with the advantages and disadvantages of ensembling and partitioning based techniques, will be discussed in the further sections.

References

[1] Aggarwal, C. C. Outlier ensembles: position paper. *ACM SIGKDD Explorations Newsletter 14*, 2 (2013), 49–58.

[2] Aggarwal, C. C., and Yu, P. S. Outlier detection for high dimensional data. In *ACM Sigmod Record* (2001), vol. 30, ACM, pp. 37–46.

[3] Manzoor, E., Lamba, H., and Akoglu, L. xstream: Outlier detection in feature-evolving data streams. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (New York, NY, USA, 2018), KDD '18, ACM, pp. 1963–1972.

[4] Tan, S. C., Ting, K. M., and Liu, T. F. Fast anomaly detection for streaming data. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence* (2011), vol. 22, p. 1511.

[5] Tran, L., Fan, L., and Shahabi, C. Distance-based outlier detection in data streams. *Proceedings of the VLDB Endowment 9*, 12 (2016), 1089–1100.