LITERATURE REVIEW: GPU-powered outlier detection on stream data

Kangqing YU
School of Computer Science
Carleton University
Ottawa, Canada K1S 5B6
kangqingyu@cmail.carleton.ca

October 7, 2017

1 Introduction

In recent years, parallel computing has drawn a tremendous amount attentions and it is becoming a main stream for many applications including data science, biochemistry, medical etc. Among all different categories of parallel computing, one which is really easy to achieve from the hardware perspective is GPU programming, CUDA to be specific, which takes advantages of Graphical Processor Unit to accelerate the performance for many tasks which are considered computational expensive. However, due to the nature of SIMD[1] in GPU programming, programming in frameworks such as CUDA can be very challenging and therefore may incur a deep learning curve.

In this project, I want to use GPU programming power to solve a specific kind of data mining task, called outlier detection(or anomaly detection) to address the increasing challenge in solving this problem in the streaming environment. An outlier in a dataset is a data point that is considerably different from the rest of the data as if it is generated by a different mechanism[11]. Applications of outlier detections vary in numerous fields, including fraud detection, network intrusion detection, medical image screening, environment monitoring etc. A stream environment is where data could come constantly at a high volume and may change over time. This can impose a very high requirement of computation power as decisions need to be made in real time within limited time period among all data. Specifically, I use **incremental kernel density estimation approach**, where an outlier decision is made not only based the data points in the current data stream, but also taking consideration of historical data without the necessity to store the entire dataset in the secondary memory. The accuracy of the result is compared with the traditional approach where data is analyzed in a static environment to further illustrate that GPU can definitely be used in a streaming environment to facilitate outlier detection at a high rate of data volume where static approaches can not achieve otherwise.

[1]Single Instruction Multiple Data

2 Literature Review

A lot of techniques have been introduced in last decades to solve the outlier detection problem. And these techniques can be briefly summarized into three different categories:

1. Supervised approaches
2. Semi-supervised approaches
3. Unsupervised approaches

Supervised approaches use techniques from machine learning and typically require building a prediction model for rare events based on manually labelled data(the training set), and use it to classify new event based on the learnt model[12, 13]. In other words, the outlier problem in this case becomes a classification problem where we are only interested in the minority class whose data deviate largely from the rest. This can somehow be a overkill because minority can often be detected based on the data distributions and patterns rather than manually labelled data. The main drawback of using supervised methods is that it fails to capture the new treading of the outliers, and also assuming the underlying distribution of the data is known. However, supervised approaches may give guaranteed performance when the data distribution is static. A lot of supervised learning methods exists that could work quite well, including Support Vector Machines, Neural Network, K-Mean and KNN etc.

Another problem of supervised approaches is that labelled data is really hard to generate, and in most cases, it is impractical for outlier data. Semi-supervised learning[5, 20] combines labeled and unlabeled data during training to improve performance. In semi-supervised clustering, some labeled data are used with unlabeled data to obtain better clustering. It is known that applying semi-supervised learning to anomaly detection can improve the detection accuracy[21]. One approach introduced by Jing Gao[10] takes advantage of K-mean clustering in unsupervised learning by adding penalties to the objective function for mislabelled data points and optimize the overall objective function.

Although some of those techniques may generate very promising results, they work well only in static data and typically don't fit into the context of dynamic streaming environment. In other words, both supervised and unsupervised methods will assume that it will have **random access** to the underlying data while this is not possible for streaming data when you can only have portion of it at one time. The definition of data stream is as follows:

Definition 1 ***Data Stream:*** *A data stream is a possible infinite series of data points ..., $o_{n?2}$, $o_{n?1}$, o_n , ..., where data point o_n is received at time $o_n.t$.*

Based on the nature of data stream, when building an application that process data stream, these three key characteristics of data stream need to be taken into consideration: uncertainty, transiency, and incompleteness[18]. Uncertainty means the data distribution of the model may change over the time as new data comming in in a unpredictible way. This term is sometimes known as **concept drift** in some literatures. Transiency means it is not possible to store the entire dataset from data stream in the memory. The data can only be accessed in one pass and when it is processed, it should be deleted from memory. Completeness just assume that the data will come indefinitely and they will never stop. This will create a really huge challenge for outlier detection as suppose you have data stream

that keep comming it. At one time t_i, you identify object o_k as being outlierness in the current window. And after some time W at time $ti + W$ when the whole recent history is considered, object o_k may become an inliner. And vice versa. This can be illustrate in Figure 1

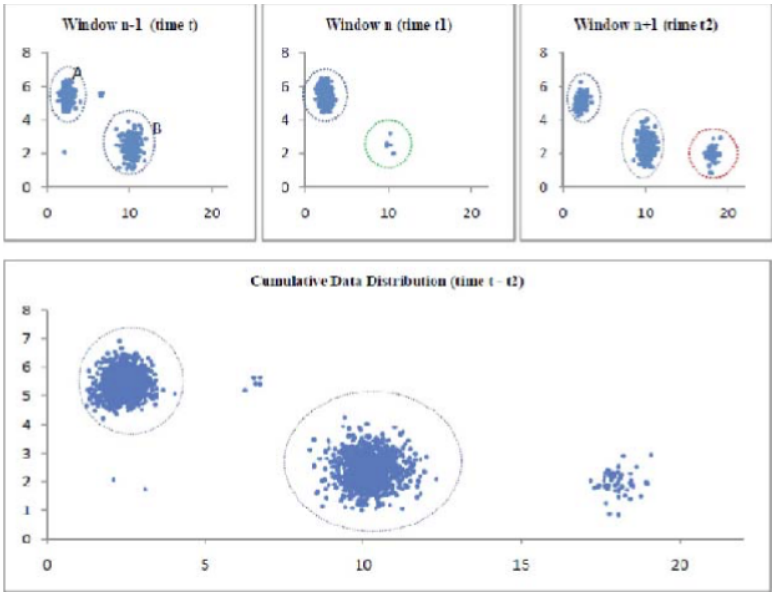

Figure 1: Evolving 2D data stream.

In contrast, unsupervised learning methods don't require labelled input and typically don't assume a fixed data distribution as the model l can be dynamically based on different data. Many best-known techniques of outlier detection fall into this category and based on the context, they can mostly be classified into two categories: **Unsupervised outlier detection on static data** and **Unsupervised outlier detection on streaming data**.

Distance-based outlier detection on static data was first introduced by Knorr and Ng[14]. It calculates the pair-wise Euclidian Distance between all data and if one data point has less than k neighbors within distance R, it is considered an outlier. The formal definition of it consists of the following:

Definition 2 ***Neighbor:*** *Given a distance threshold $R(R > 0)$, a data point p_i is a neighbor of datapoint p_j if the distance between p_i and p_j is not greater than R. A data point is not considered a neighbor of itself.*

Definition 3 ***Distant-based outlier:*** *Given a dataset D, a count threshold k $(k > 0)$ and a distance thresh- old R $(R > 0)$, a distance-based outlier in D is a data point that has less than k neighbors in D*

There are some variants of the static distance-based approaches, and their ideas are quite similar. For instance, Ramaswamy[17] purposed a method where an outlier is defined by considering the total number of objects whose distance to its k^{th} nearest neighbor is smaller than itself. Angiulli and Pizzuti[4] introduced a method where an outlier is defined by taking into account of the sum of the distances from 1^st up to the k^{th} nearest neighbors.

These methods are sometimes referred as KNN in some papers. Note it is different from the term KNN in supervised machine learning.

Density-based approach on static data is another way to detect outlier on static data. The basic idea is to assign a degree of being outlier(a score) based on the density of local neighbourhood, given some predefined restrictions. A popular example of this approach is Local Outlier Factor(LOF) algorithm[6], which is defined as follows: Let **k-distance(q)** denote as distance to the k_{th} nearest neighbor of q, then

$$\textbf{reach-distk(q,p)} = max(d(q,p), k - distance(p)) \tag{1}$$

where d(q,p) is Euclidean distance from q to p

$$\textbf{lrd(q)} = \frac{1}{\sum\limits_{k \in knn(q)} reach - distk(q,p)/k} \tag{2}$$

where **lrd(q)** is the local reachability density of data point q
And finally **LOF** of data point q is defined as:

$$\textbf{LOF(q)} = \frac{\frac{1}{k}\sum_{k \in knn(q)} lrd(q)}{lrd(q)} \tag{3}$$

Data records (points) with high LOF have local densities smaller than their neighborhood and typically represent stronger outliers, unlike data points belonging to uniform clusters that usually tend to have lower LOF values[16]

Both distance-based and density-based outlier detection techniques have a very wide range of applications, but one of their most notorious drawbacks is that it is usually computational expensive, especially in the scenario when the input data is high dimensional and a metric function is used to calculate the pair-wise distances.

Statistics approach on the other hand, is another way to perform outlier detection on data with random access without requiring expensive computational resources. It is based on the probability theory and normally models the underlying data using a stochastic distribution(e.g. Gaussian distribution). One of the most popular one used is **auto-regression** model or sometimes being referred as Gaussian mixture model[7]. It is very popular for time series outlier detection and its definition is given by

$$x(t) = a_1(t) \times x(t-1) + ... + a_n(t) \times x(t-n) + \xi(t)$$

where $\xi(t)$ is the noise and is almost always assumed to be a Gaussian white noise. Based on this formula, we can estimate the coefficient parameters $a_i(t)$ based on the given time series of $x(t), ...x(t-n)$. The model can then be used to predict future time series by defining a threshold, called cut-off limit and the data point is identified as an outlier if it is beyond this threshold.

Deviation is another way of statically outlier detection first introduced by Arning[1], where an outlier is detected if feature space of one data point deviates largely from other data points. Aggarwal and YU[3] proposed a technique for outlier detection. The basic idea in their definition is, a point is an outlier, if in some lower dimensional projection it is present in a local region of abnormally low density. This method is also an efficient method for high dimensional data set[**?**]. Another technique introduced by Harkins[2] is to take advantage of replicator neutral network(RNN) to detect outliers. There might be other

techniques used for unsupervised outlier detection but due to the limitation of this paper, I can not list all of them. The ones mentioned above are those best-known so far.

As modern applications have an increasing demands to process streaming data in real-time, a lot of these static methods mentioned before have been extended to work in the dynamic environments. The all based on the same ideas in static approaches but algorithms have been modified in an incremental fashion to address the **concept drift** of the data stream properties. Throughout this paper, these techniques are what I would be interested to apply in the world of parallel computation.

Distance-based outlier detection approach was among the first which start to apply the method in the streaming context. In the last decade, there are several studies which focus on **distance-based outlier detection in data streams(DODDS)**. Due to the fact that the distance-based require random access on the data and this is not possible with stream data, a concept called *sliding window* was introduced which only keep a number of active objects. When objects expire, they are deleted from memory as new object comes in.

Definition 4 ***Count-based window:*** *Given data point o_n and a fixed window size W, the count-based window D_n is the set of W data points: o_{n-W+1},o_{n-W+2},..., o_n*

Definition 5 ***Time-based window:*** *Given data point o_n and a time period T , the time-based window $D(n,T)$ is the set of W_n data points: $o_{n'}$,$o_{n'+1}$,...,o_n with $W_n = n - n' + 1$ and $o_n.t - o_{n'}.t = T$.*

There are numerous algorithms that have been invented to process stream data using sliding window on outlier detection. And based on the benchmark among all DODDS algorithms given by Luan Tran[19], the MCOD algorithm introduced by M.Kontaki[15] seems to have the most satisfying performance. Its basic idea is to pre-compute the *safe inliers* that have more than k neighbors which arrived after p_i by using an event queue, which can reduce greatly on space complexity. Because the neighbors which arrives before p_i may expire, by declaring the neighbors which arrived after p_i to be larger than k, we can safely mark p_i as inlier. The time complexity of this algorithm is guaranteed to be $O(n \log k)$ while maintaining the space complexity to be $O(nk)$

Most of DODDS algorithms are based on the original definition of distance-based technique given by Knorr and Ng[14]. The other distance-based techniques in outlier detection such as KNN remain unsolved in the streaming context. It would be interested to see if those methods can be extended to work in the data streaming context. And since they are based on *sliding window*, only active objects in current window are considered. And therefore, these methods lack a global view of the entire dataset and sometimes, this may affect accuracy.

To address this problem without storing the entire dataset, a method need to find a efficient way to **mine** its historical data to impact the current data or gradually fade it away. An technique inspired from **sensor network** is mentioned in[**?**], where it use a **kernel density estimator** to model the distribution of the sensor data. When used for outlier detection, the number of neighbors of a given datapoint p_i is estimated by the distribution function $f(p_i)$. In [16], D. Pokrajac et al illustrated that the LOF algorithm can be applied incrementally and the insertion of new data as well as deletion of obsolete data does not depend on the total number of N in dataset and therefore the time complexity of incremental LOF algorithm can theoretically be $O(N \log N)$. However, there are still a lot of noise around this algorithm and many researchers complain that it is still computational

expensive in practice. These two methods are both based on computing of the **densities of local neighborhood**.

Clustering is another technique to outlier detection on stream data. Since clustering is a technique in unsupervised machine learning, it inspired the ideas of outlier detection in streaming environment. Two main algorithms exists for clustering-based approaches. One of them is called **K-Mean clustering**[9]. It divides the stream into chunks and cluster chunks using k-mean clustering into fixed number of k clusters. The **mean** of each clusters is calculated by metrics information of all data points in a cluster. If a data point is too far from the mean of its data point by a threshold, it will be considered as a candidate outlier. Both the *mean* and the *candidate outliers* detected in this chunk are carried over to next chunk in stream to further compare with data in other chunks. Other data in this chunk is simply deleted from memory. If the candidate outlier passed a given number of chunks, it is then identified as *real outlier*. Compared to K-Mean clustering, **K-Median clustering**[8] clusters each chunk of data into variable number of clusters(from k to $klog(n)$), and different from K-mean clustering, it passes the **weighted medians** found in current chunk into next chunk for testing outlierness rather than the mean and candidate outliers. Both of these two approaches will require users' input of value k but K-Median clustering theoretically is better since the number of clusters is not fixed.

References

[1] A linear method for deviation detection in large databases. In *In: Proc of KDD96*, pages 164–169, 1996.

[2] Outlier detection using replicator neural networks. In *Proc of DaWaK02*, pages 170–180, 2002.

[3] Charu C. Aggarwal and Philip S. Yu. Outlier detection for high dimensional data. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, SIGMOD '01, pages 37–46, New York, NY, USA, 2001. ACM.

[4] F. Angiulli and C. Pizzuti. Outlier mining in large high-dimensional data sets. *IEEE Transactions on Knowledge and Data Engineering*, 17(2):203–215, Feb 2005.

[5] Sugato Basu, Mikhail Bilenko, and Raymond J. Mooney. A probabilistic framework for semi-supervised clustering. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 59–68, New York, NY, USA, 2004. ACM.

[6] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. Lof: Identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, pages 93–104, New York, NY, USA, 2000. ACM.

[7] D. I. Curiac, O. Banias, F. Dragan, C. Volosencu, and O. Dranga. Malicious node detection in wireless sensor networks using an autoregression technique. In *Networking and Services, 2007. ICNS. Third International Conference on*, pages 83–83, June 2007.

[8] Parneeta Dhaliwal, M. P. S. Bhatia, and Priti Bansal. A cluster-based approach for outlier detection in dynamic data streams (KORM: k-median outlier miner). *CoRR*, abs/1002.4003, 2010.

[9] M. Elahi, K. Li, W. Nisar, X. Lv, and H. Wang. Efficient clustering-based outlier detection algorithm for dynamic data stream. In *2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery*, volume 5, pages 298–304, Oct 2008.

[10] Jing Gao, Haibin Cheng, and Pang-Ning Tan. Semi-supervised outlier detection. In *Proceedings of the 2006 ACM Symposium on Applied Computing*, SAC '06, pages 635–636, New York, NY, USA, 2006. ACM.

[11] C. HewaNadungodage, Y. Xia, and J. J. Lee. Gpu-accelerated outlier detection for continuous data streams. In *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 1133–1142, May 2016.

[12] Mahesh V. Joshi, Ramesh C. Agarwal, and Vipin Kumar. Mining needle in a haystack: Classifying rare classes via two-phase rule induction. *SIGMOD Rec.*, 30(2):91–102, May 2001.

[13] N.Chawla A.Lazarevic L.Hall K.Bowyer. Smoteboost:improving the prediction of minority class in boosting. 2003.

[14] E. Knorr and R. Ng. Algorithms for mining distance-based outliers in large datasets. In *Int. Conf. on Very Large Databases (VLDB98)*, pages 392–403, 1998.

[15] M. Kontaki, A. Gounaris, A. N. Papadopoulos, K. Tsichlas, and Y. Manolopoulos. Continuous monitoring of distance-based outliers over data streams. In *2011 IEEE 27th International Conference on Data Engineering*, pages 135–146, April 2011.

[16] D. Pokrajac, A. Lazarevic, and L. J. Latecki. Incremental local outlier detection for data streams. In *2007 IEEE Symposium on Computational Intelligence and Data Mining*, pages 504–515, March 2007.

[17] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, pages 427–438, New York, NY, USA, 2000. ACM.

[18] Shiblee Sadik and Le Gruenwald. Online outlier detection for data streams. In *Proceedings of the 15th Symposium on International Database Engineering & Applications*, IDEAS '11, pages 88–96, New York, NY, USA, 2011. ACM.

[19] Luan Tran, Liyue Fan, and Cyrus Shahabi. Distance-based outlier detection in data streams. *Proc. VLDB Endow.*, 9(12):1089–1100, August 2016.

[20] S. Schroedl S Wagstaff K.Cardie, C. Rogers. Constrained k-means clustering with background knowledge. page 577 584, 2001.

[21] Yan Yu, Shanqing Guo, Shaohua Lan, and Tao Ban. *Anomaly Intrusion Detection for Evolving Data Stream Based on Semi-supervised Learning*, pages 571–578. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.