GPU-powered outlier detection on stream data

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

November 6, 2017

Abstract

Outlier detection has become an increasing challenging task in modern applications due to the fact that the data may come in the form of streams rather than statically as it was before. A lot of algorithms have been modified so that they can work in the streaming environment. Among all of them, a very popular technique called *sliding window* is used, which only keep a portion of streaming data in memory and detect outliers only among all of those data. A main drawback with this approach is that the decision of outliers is only based on the data in current window and historical data are simply dropped and not considered in decision making of outlier data in current window. Therefore, this method fails to address the nature of *concerpt drift* in data streams. Another challenge is that since the data may come at a very high rates and it is impossible to store all data in memory, the decision should be made in a timely manner with only one pass over data stream. This will pose a very harsh requirement in computational power and in most case, it is impossible for CPU programming to achieve. In this project, I purposed a novel solution to detect outliers in streaming environment, powered by GPU, based on a very efficient classical algorithm called LOF(Local Outlier Factor) to address the increasing challenge of outlier detection over continuous data streams. The proposed method, named LOF_GPU is able to address the *concerpt drift* in data streams, as well as detecting outliers in high-dimensional, high-rates data streams and produces timely results without compromising performances. Since the LOF algorithm is very computational expensive, very few works(almost none) have been conducted to extend the algorithm to work in streaming environment. As far as I know, the solution I purposed is the first algorithm that try to modify the LOF algorithm in the streaming environment and it is also the first parallel implementation of LOF algorithm in the streaming context.

1 Introduction

In recent years, parallel computing has drawn a tremendous amount of 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(Graphics Processing Units) programming, which takes advantages of Graphical Processor Unit to accelerate the performance for many tasks which are considered computational expensive in CPU(Central Processing Unit). There are two major frameworks(libraries) that are widely used for GPU program-

ming. One of them is OpenCL[1], which is a cross-platform open source frameworks for GPU programming maintained by khronos. The other is called CUDA(Compute Unified Device Architecture)[2] that are designed specifically based on the architecture of NVIDIA graphics cards. Throughout this project, I will use CUDA as the programming framework for GPU acceleration on outlier detection task.

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[13]. 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 come constantly at a high volume and may change over time. This can impose a very high requirement for computation power as decisions need to be made in real time within limited amount of time among all data. In addition to that, since the applications do not have random access to the underlying data in the streaming environment, 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[24]. Uncertainty means the data distribution of the model may change over the time as new data coming in a unpredictible way. This term is sometimes known as *concept drift* in some literatures. Dealing with concept drift is a main challenge in most streaming applications. 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 assume that the data will come indefinitely and they will never stop. These all make outlier detection in data streams extremely challenging from both algorithms and hardware perspective.

To be more specific, suppose you have a data stream that keeps coming indefinitely, 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

In this project, I developed an novel, modified version of *Local Outlier Factor* algorithm, LOF_GPU where an outlier decision is made not only based the data points in the current data window, but also taking consideration of historical data without the necessity storing the entire dataset in the secondary memory. In doing so, I kept a statistical binned summary of all the observed data to help making decisions on outlier for data points within current processing window, and gradually fade away the impact on the obsolete data when making decisions on current data. And thus the proposed algorithm should provide a more accurate results compared to the widely used sliding window approach.

Note that most density-based approaches in outlier detection, including LOF, despite being accurate, are notorious of being computational expensive. And therefore, it is almost impossible to detect outliers with high volume, high speed data streams without introducing parallelism. Other computational in-expensive algorithms exists but they either need to sacrifice on the accuracy or assuming fixed distribution over the underlying data, which fails to capture the nature of *concerpt drift* in data streams. The LOF method is purely based on calculating the density of data point compared with its neighbourhood, where density is measured based on the reachability within its K nearest neighbours. The GPU is used to accelerate the computation time in order to provide timely results to keep up with tine high input rate of streaming data.

[1]https://www.khronos.org/opencl/
[2]https://developer.nvidia.com/

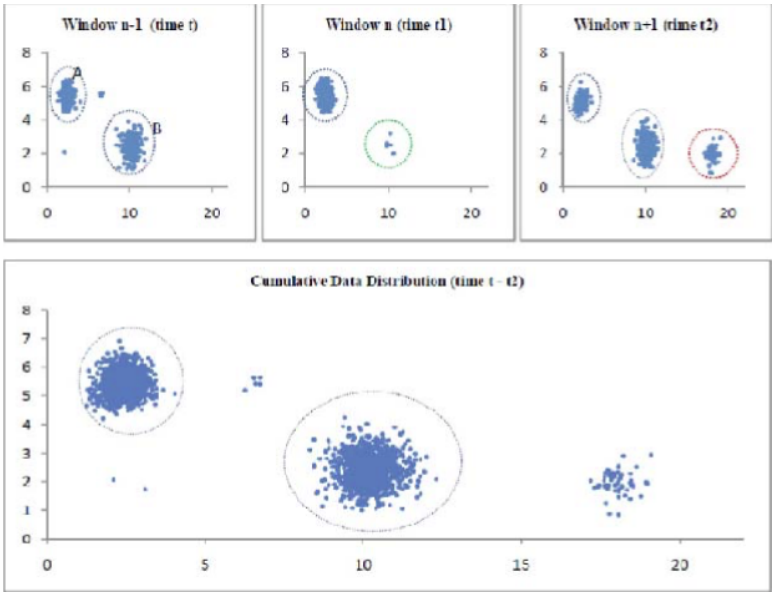

Figure 1: Evolving 2D data stream

The accuracy of the result in this method is compared with another GPU accelerated approach, SOD_GPU as proposed in [13], which is based on *kernel density estimator*. And the speedup on performance is compared with distance-based method based on sliding window, which runs in a multi-core CPU to further illustrate that LOF_GPU can practically be used in a streaming environment to detect outliers at a high rate of data volume where it is otherwise incapable to handle by CPU.

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 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[14, 15]. 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. The main problem with this approach is that in order to ensure accuracy, a large number of labelled data need to be generated which is unpractical in most cases Compared to supervised approaches, Semi-supervised approaches[7, 27] only require a small number of training data with some unlabeled data to obtain better predictions. One approach introduced by Jing Gao et all.[12] 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 semi-supervised methods will assume that they 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 and they also fails to address the problem of the potential change of data distribution.

In contrast, unsupervised learning methods don't require labelled input and typically don't assume a fixed data distribution as the model can be dynamically built based on variations of 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 was among the very first outlier detection method introduced by Knorr and Ng[16]. 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.

There are some variants of the static distance-based approaches, and their ideas are quite similar. For instance, Ramaswamy et all.[23] 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[6] 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 and it should be noted that it is different from the term KNN in supervised machine learning.

Density-based approach 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[8], which is what this proposed algorithm is based on, use the concept called *reachability* to coin the density of data point. Another popular density-based outlier detection method is called LOCI(Local Correlation Integral)[21]

Statistics approach 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[9].

Deviation is another way of statically outlier detection first introduced by Arning et al.[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[11].

Another technique introduced by Harkins et al.[2] takes 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 to detect outliers statically.

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 streaming 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.

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, *sliding window* technique was introduced which only keep a number of active objects in current window. When objects expire, they are deleted from memory as new object comes in. There are mainly two window models in data streams: count-based window and time-based window

Numerous algorithms 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 et al.[26], the MCOD algorithm introduced by M.Kontaki et al.[17] 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[16]. 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. Since the these methods only have access to only a portion of data, they all lack a global view of the entire dataset and sometimes, this may affect accuracy.

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**[11]. 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**[10] 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.

To address this problem of storage and users' input parameters, an ideal method need to find a efficient way to efficiently mine its historical data or gradually fade old data away without users' intervention. An technique inspired from *sensor network* is mentioned in[25], 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 data point p_i is estimated by the distribution function $f(p_i)$. In [22], 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 $O(N \log N)$. These two methods are both based on computing of the **densities of local neighbourhood**. However, there are still a lot of noise around these algorithms and many researchers argue that it is still computational expensive in practice. Also the LOS approach proposed by D. Pokrajac et al will need to store the entire dataset, which is not applicable in the streaming context.

Even though there are many studies in the last decade that focusing on detecting outlier online in data stream, only a few tried to solve this problem using a parallel implementation. In [13] C. HewaNadungodage et al.implemented a so-called SOD_GPU[3] algorithm which is based on kernel density estimator powered by GPU to effectively detect outlier in continuous data streams. The results seems very promising as it is 20X faster compared to a multicore CPU implementation and even a higher accuracy rate compared to the sliding window approach. And this is the only work, I found by the time of writing, which tried to solve the outlier detection problem in streaming environment using parallel computing approach. Other parallel implementations for outlier detection exist but they only work in a static fashion, in which case, it does not seem to be quite necessary. In [5], Angiulli et al. proposed a distance-based KNN algorithm powered by GPU and similarly, Matsumoto and Hung[20] introduced a GPU-accelerated approach to detect the outliers in uncertain data. Another GPU-accelerated approach to detect outlier in static data was purposed in [4], where the LOF(Local outlier factor) algorithm is used. Anna Koufakou et all.[19] developed a parallel outlier detection strategy based on **Attribute Value Frequency(AVF)**[18][4] algorithm using MapReduce programming paradigm. Other incremental parallel methods exist for online outlier detection but they either require storing the entire history of data stream or giving a set of user-defined parameters, which is hard to define in most cases.

3 Project Report

Present the results of your project. Add subsections as appropriate...

3.1 Subsection 1

...

3.2 Subsection 2

...

3.3 Subsection 3

...

You can also have figures in your paper. Figure 2 is a typical example of an experimental evaluation result. Such graphs are ususally created with GnuPlot. Figure 3 is an example of a drawing created with *mdraw* or *epsfig*.

[3]Stream Outlier Detector-GPU

[4]Note that the AVF algorithm can only detect outlier in categorical data.

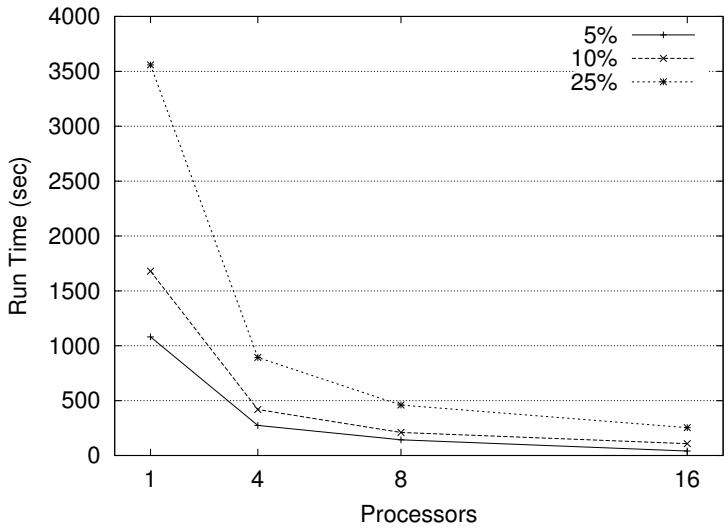

Figure 2: Measured Running Times Of Some Unknown Algorithm Implementation

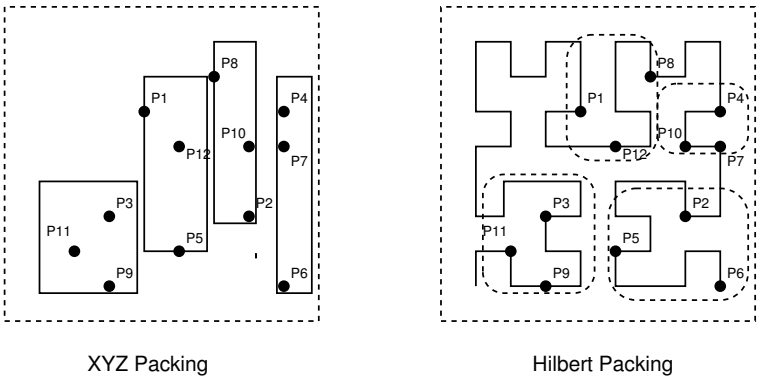

Figure 3: XYZ and Hilbert Packings

4 Conclusion

The "moral of the story": What have we learned? What did we achieve? What did we not achieve? What would we do better next time? Possibilities for future research...

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] Malak Alshawabkeh, Byunghyun Jang, and David Kaeli. Accelerating the local outlier factor algorithm on a gpu for intrusion detection systems. In *Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units*, GPGPU-3, pages 104–110, New York, NY, USA, 2010. ACM.

[5] F. Angiulli, S. Basta, S. Lodi, and C. Sartori. Fast outlier detection using a gpu. In *2013 International Conference on High Performance Computing Simulation (HPCS)*, pages 143–150, July 2013.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

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

[16] 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.

[17] 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.

[18] A. Koufakou, E. G. Ortiz, M. Georgiopoulos, G. C. Anagnostopoulos, and K. M. Reynolds. A scalable and efficient outlier detection strategy for categorical data. In *19th IEEE International Conference on Tools with Artificial Intelligence(ICTAI 2007)*, volume 2, pages 210–217, Oct 2007.

[19] A. Koufakou, J. Secretan, J. Reeder, K. Cardona, and M. Georgiopoulos. Fast parallel outlier detection for categorical datasets using mapreduce. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 3298–3304, June 2008.

[20] Takazumi Matsumoto and Edward Hung. *Accelerating Outlier Detection with Uncertain Data Using Graphics Processors*, pages 169–180. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[21] S. Papadimitriou, H. Kitagawa, P. B. Gibbons, and C. Faloutsos. Loci: fast outlier detection using the local correlation integral. In *Proceedings 19th International Conference on Data Engineering (Cat. No.03CH37405)*, pages 315–326, March 2003.

[22] 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.

[23] 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.

[24] 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.

[25] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos. Online outlier detection in sensor data using non-parametric models. In *Proceedings of the 32Nd International Conference on Very Large Data Bases*, VLDB '06, pages 187–198. VLDB Endowment, 2006.

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

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