International Journal of Scientific Engineering and Research (IJSER)
www.ijser.in
Issn (Online): 2347-3878
Volume 2 Issue 4, April 2014

Mining Infrequent Patterns across Multiple Streams of Data

Saleem[1], Vanaja Ranjan[2]

[1]M.Tech Student, College of Engineering Guindy, Anna University, Chennai-25, Tamil Nadu, India
[2]Professor, EEE, College Of Engineering Guindy, Anna University, Chennai-25, Tamil Nadu, India

Abstract: ***The problem of extracting infrequent patterns from streams and building relations between them is gradually significant to-day as many events of interest such as network attacks or unusual stories in news data occur rarely. The complexity of the problem is multipart when a system is required to deal with data from several streams. To discourse these problems we present an approach that combines pyramidal trees with association rule mining to discover infrequent patterns in data streams. This maintains a summary of the data without requiring increasing amounts of memory resources.***

Keywords: Association, Datasets, Extraction, Infrequent, Mutually Dependent, Patterns, Pruning

1. Introduction

Infrequent pattern mining is concerned with extracting "uncommon" or "scarce" patterns from streams of data. In the past, frequent pattern mining has been investigated in detail with little research being done in infrequent pattern mining. However, infrequent patterns are often more useful than frequent patterns as they provide information about events of interest (such as in network intrusion detection).

We address two major issues in infrequent pattern mining scalability in terms of memory requirements and pattern selection over time horizons that vary in span. We have developed a framework for identifying infrequent patterns in stream data which allows the discovery of the patterns at different time resolutions. The framework selects the infrequent patterns and stores them into a data structure such that only the unique patterns across the stream are stored at the top of the structure. An important aspect of our framework is that it can handle multiple data streams and thus allows the discovery of patterns that are infrequent across multiple data streams. At the core of our approach is the use of an online hierarchical structure that allows the concise description of the infrequent patterns from the data stream

2. Infrequent Pattern Mining Algorithm

A. Infrequent Pattern Processing Challenges

The problem of mining infrequent patterns and building associations from these patterns extracted from a stream poses two challenges: (1) the problem of extracting the patterns and storing them efficiently and (2) the problem that the infrequent patterns and associations discovered may over time become frequent. We are only interested in the patterns that remain infrequent over the entire stream (or part of) processed.

B. Preliminaries

There are many techniques that can be used to extract infrequent patterns but typically, these techniques produce a large amount of infrequent items which require increasing amounts of resources both in terms of memory and computations. The typical approach [10] to infrequent pattern mining is to first identify the frequent patterns and then prune these patterns from the dataset. The remaining patterns are considered to be infrequent. The main problem is that a large number of infrequent items are typically generated by the extraction process and hence as more data is observed in the stream, more patterns need to be stored.

Let S be the data stream [(d1; i1; t1), (d2; i2; t2). . . (dn; in; tn)] Where (dk; ik; tk) represents the data instance, its class label or item id and its time stamp in the stream respectively. Let I represent the set of all class labels and thus ik 2 I. Since we are proposing a hierarchical data structure, let h denote the height of the hierarchy, where the root node is at the level l=h and the leaf node is at the level l=0. The data instances are processed at the leaf node level and the summary statistics are maintained at higher levels in the hierarchy. Since the stream is processed by a moving window of length L, let wli denote the ith window at level l and its time span be denoted by tl[si; ei], where tl[si; ei] = [tsi : tei] and tsi and tei are the start and end time span of wl. Let x(tl[si; ei]) represent the set of items, xk(tl[si; ei]) represent the item with id k, xks(tl[si; ei]) represent the support of the item with id k and xa(tl[si; ei]) denote the set of mutually dependent infrequent items in wli over time span tl[si; ei] respectively. If item ik occurs nik times in tl[si; ei]), then the support threshold (_ik) = nik=n. A single processor board (XM2110) can be configured to run your sensor application/processing and the network/radio communications stack simultaneously. The IRIS 51-pin expansion connector supports Analog Inputs, Digital I/O, I2C, SPI and UART interfaces. These interfaces make it easy to connect to a wide variety of external peripherals.

C. Algorithm Overview

We process the data in two stages: first we eliminate the frequent items and second we build the associations between the infrequent items. Overall, our framework has three stages. First, the initial set of infrequent patterns are extracted and stored in the pyramid. Second the rest of the stream is processed and the infrequent pattern set updated. In the final

International Journal of Scientific Engineering and Research (IJSER)
www.ijser.in
Issn (Online): 2347-3878
Volume 2 Issue 4, April 2014

stage the infrequent pattern set for the entire stream is finalized and compared with the sets extracted from other streams in order to determine the pattern set that is infrequent across streams.

3. Infrequent Pattern and Extraction

Once a candidate window has been identified, we extract all infrequent patterns y(tl[si; ei]) for which xks(wli) < _ from the window and store them in the infrequent set _infrequent. The processing is repeated for all windows in the candidate queue thus producing the infrequent sets for the individual windows. However, this process only ensures that the patterns are infrequent for each window rather than across multiple windows or across parts of the entire stream. To ensure that the patterns are infrequent across more than one window, it is necessary to compare the support of the infrequent items sets across windows. This requires that we update the support for each infrequent item as more windows are processed and involves the summary of two consecutive candidate windows. The summary of two windows consists of the items that are ”infrequent” in both windows. If there are infrequent items similar in both windows and the sum of their support exceeds the threshold limit _, then these items would not be included in the summary of the windows.

A. Storing The Sporadic Patterns

The next step in the processing involves building the Pyramidal data structure that stores the infrequent patterns discovered from the stream. The pyramidal data structure is built by merging the infrequent patterns sets extracted from increasingly larger ordered groupings of candidate windows.

Figure 1: Pyramidal Tree

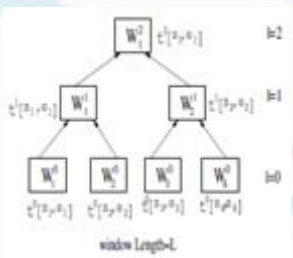


Figure 2: Infrequent Pattern Extraction

Consider the example in Figure 1. The bottom level contains all the infrequent item sets covered by a predefined time span (T) which in the case of Figure 1 has 8 candidate windows. The nodes at the next level in the pyramid contain the set of items that are infrequent for pairings of candidate windows. As the item sets are generated at the higher levels in the pyramid, the algorithm checks to ensure that items are indeed infrequent by checking the item against all the data points contained in the windows associated with that branch of the pyramid. Consider the following example. Let yk(t0[s1; e1]) and yk+4(t0[s1; e1]) be patterns that are infrequent at level 0 in window w01in the data in window w02 is processed, if the support for either yk(t0[s1; e1]) or yk+4(t0[s1; e1]) increases to a value that exceeds , then the pattern is no longer propagated to the node at level 1. Therefore, an infrequent item at Level2 in the pyramid that was extracted originally from window w01 would be checked against the data from windows w02, w03 and w04. Similarly, infrequent items extracted from the windows w03 and w04 at Level 1 would be checked against windows w01, w02.This process removes the dependency on the length of the window and for any value of L, we will always obtain the same infrequent items at the root of the pyramidal tree as outlined below.

Lemma 1: The infrequent patterns extracted from each window are independent of the length of the data window.

Proof: If the length of the moving window (L) = t0[s1; e4] (see. Figure 2), then the infrequent items over the time span t0[s1; e4] derived using equation 1 are given by y(t0[s1; e4]) = Summary(t0[s1; e4]). If we change the length of the window from L to L/2, then the time span would be covered by two equally sized windows L’ and L” where L’ = t0[s1; e2] and L”=t0[s3; e4]. Moreover, we can write that t0[s1; e4] = t0[s1; e2] + t0[s3; e4] = t1[s1; e1] + t1[s2; e2] = t2[s1; e1]. In addition, by using equation 1 we obtain y(t2[s1; e1]) from the union of y(t1[s1; e1]) and y(t1[s2; e2]) . Since, y(t2[s1; e1]) covers the all infrequent items over the time span t0[s1; e4], then y(t2[s1; e1]) = y(t0[s1; e4]). Similarly, if we change the length of the window to L/4, then using equation 1 we can mine all infrequent items over the same time span. Therefore, extracting the infrequent items over a time span is independent on the length of the moving window.

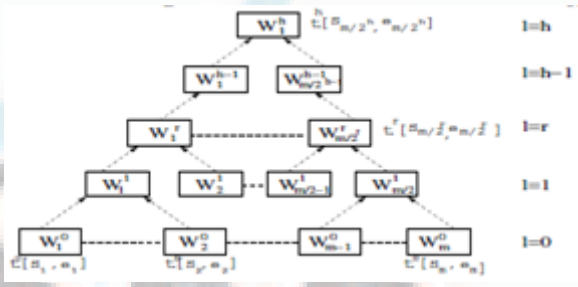


Figure 3: Building the Pyramidal Tree

Lemma 2: Pruning old data windows does not affect the infrequent patterns stored at the root of the pyramidal tree.

Proof: Given a sequence of m processed candidate windows,the pyramid derived the windows will at the root level (l = h) store the infrequent patterns defined by equation 1 (i.e.y(t0[sm=2;em=2]) = Summary(t0[s1; em])) that cover the time span t0[s1; em] (i.e. th[sm=2; em=2] = t0[s1; em] as shown in Figure 3). Because of the properties of the infrequent items extracted using equation 1 which tracks the time interval and the support for each item, for any node in the pyramid at level l, we can prune any of the l-1 or lower

International Journal of Scientific Engineering and Research (IJSER)
www.ijser.in
Issn (Online): 2347-3878
Volume 2 Issue 4, April 2014

nodes/branches from the pyramid without affecting the infrequent patterns stored in the nodes at level l.

4. Experimental Evaluation

A. Voltage flow Datasets

The Voltage intrusion dataset has around 10000 network connection records. Each connection record in the dataset is labeled as either normal or as a specific attack (24 types of attacks). We used sampling to divide the original dataset into 8 streams of roughly 500 records each. The streams covered only a subset of the attacks in the original dataset and therefore for each stream we recorded the label and the frequency of the attacks contained in the stream. In all experiments, we used an entropy value of less than 110 and greater than 320 to determine whether a data window is used in the infrequent pattern extraction stage in the algorithm.

B. Window Size Analysis

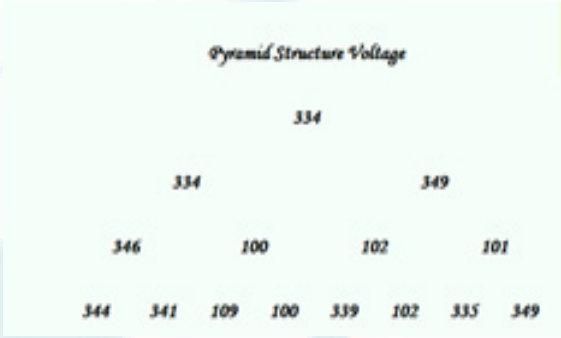
The aim of the first set of experiments is to determine whether the same set of infrequent patterns is propagated to the root of the pyramidal data structure that covers an entire stream, regardless of the window size used to process the data. Please note that not all attack patterns were infrequent - specifically, attacks 1 to 4 occurred frequently in most streams when compared with attacks 5 to 24. We processed the streams using a varying window size w that covered an interval of points ranging from as few as 2000 data points to the entire stream. Of significance are the columns showing the difference (_) between the ground truth and the infrequent patterns stored at the root of the pyramid. The results demonstrate that the changes made to the window size had no effect on the final set of infrequent items.

Data Sets	Voltage	Occurrence	Difference
1	344	08/04/2014 17:33:936	00:30:00
2	341	10/04/2014 09:33:936	00:40:00
3	334	12/04/2014 18:13:936	01:10:00
4	337	14/04/2014 03:33:936	01:10:00
5	339	16/04/2014 15:43:936	00:05:00
6	320	17/04/2014 11:18:936	00:10:00
7	335	19/04/2014 03:33:936	00:05:00
8	349	20/04/2014 12:33:936	00:35:00

C. Infrequent Pattern Frequency Analysis

The aim of the second type of experiments was to determine how the frequency of patterns affects the level to which an infrequent pattern is propagated up in the pyramidal tree. Items with support < _ for i = 1..4w would be expected to reach the midlevel nodes of the pyramidal tree, while items with support < _ for i = 1..nw would be found at the top level of the pyramid. We used a window size w = 2000 to process the data stream and we recorded the infrequent patterns extracted at all levels in the pyramid along with the window id.

Items with support < _ for i = 1..4w would be expected to reach the midlevel nodes of the pyramidal tree, while items with support < _ for i = 1..nw would be found at the top level of the pyramid. We used a window size w = 2000 to process the data stream and we recorded the infrequent patterns extracted at all levels in the pyramid along with the window id



D. Multiple Stream Infrequent Pattern and Association

Rule Mining: The aim of the last experiment was to derive temporal associations from the infrequent item sets discovered in the streams. At the root level streams had multiple infrequent items and furthermore, only three of these streams contained enough support to generate associations between the infrequent items. For example, the rule 5) indicates that whenever an attack of type 335 was observed, one can also expect an attack of type 329 within 2000 data points.

Mutually Dependent Data Set with Temporal Association

Data Sets	Voltage	Occurrence	Difference
1	335	08/04/2014 10:23:936	00:10:00
1	346	08/04/2014 10:33:936	00:50:00
1	329	08/04/2014 15:53:936	00:05:00
1	105	08/04/2014 08:58:936	00:05:00
2	332	10/04/2014 04:03:936	00:45:00
2	320	10/04/2014 08:03:936	00:10:00
2	346	10/04/2014 03:33:936	00:30:00
2	323	10/04/2014 10:13:936	00:05:00
3	109	12/04/2014 12:23:936	00:25:00
3	334	12/04/2014 18:13:936	01:10:00
3	100	12/04/2014 17:18:936	00:55:00
3	333	11/04/2014 20:23:936	00:05:00
4	337	14/04/2014 03:33:936	01:10:00
4	332	14/04/2014 07:08:936	00:55:00
4	322	14/04/2014 17:08:936	00:10:00
4	103	13/04/2014 22:28:936	00:10:00

5. Conclusion and Future Work

The framework proposed in this paper can be used effectively to extract infrequent items from stream data and generate temporally ordered associations (mutually dependent items) from these items. The results obtained have demonstrated that the algorithm can handle vastly different types of data. The major advantages of our work are that it allows incremental processing of data streams with limited memory resources and it can be applied to multiple streams.

Acknowledgment

Saleem Azeeskhan thanks the staff members for their full support and the continuous encouragement in completing the proposed work and for extending in near future.

References

[1] D. Xin, X. Shen, Q. Mei, and J. Han. Discovering interesting patterns through user's interactive feedback. In KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, pages
[2] X. Zhou, S. Y. W. Su, M. P. Papazoglou, M. E. Orlowska,and K. G. Jeffery. Discovering minimal infrequent structures from xml documents. In WISE 2004, Proceedings 5th International Conference onWeb Information Systems Engineering,Brisbane, Australia, volume 3306. Springer, 2004.
[3] A. Manjhi, V. Shkapenyuk, K. Dhamdhere, and C. Olston.Finding (recently) frequent items in distributed data streams.In ICDE '05: Proceedings of the 21st International Conferenceon Data Engineering (ICDE'05), pages 767–778,Washington, DC, USA, 2005. IEEE Computer Society.
[4] C. Aggarwal, J. Han, J. Wang, and P. Yu. A framework forclustering evolving data streams. In Proceedings of the 29th VLDB conference, 2003
[5] Q. G. Zhao, M. Ogihara, H. Wang, and J. J. Xu. Finding global icebergs over distributed data sets. In PODS '06: Proceedings of the twenty-fifth ACM SIGMOD-SIGACTSIGARTsymposium on Principles of database systems,pages 298–307, New York, NY, USA, 2006. ACM Press.
[6] S. Muthukrishnan. Data streams: algorithms and applications.2003
[7] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom.Models and issues in data stream systems. In PODS '02: Proceedings of the twenty-first ACM SIGMOD-SIGACTSIGARTsymposium on Principles of database systems,pages 1–16, New York, NY, USA, 2002. ACM Press.
[8] C. Gianella, J. Han, J. Pei, X. Yan, and P. Yu. Mining frequent patterns in data streams at multiple time granularities.In Proceedings of the NFS Workshop on Next Generation Data Mining, 2002.
[9] B. Babcock and C. Olston. Distributed top-k monitoring. In Proceedings of the ACM SIGMOD International Conferenceon Management of Data, San Diego, June 2003.
[10] S. Ma and J. L. Hellerstein. Mining mutually dependent patterns. In ICDM '01: Proceedings of the 2001 IEEE International Conference on Data Mining, pages 409–416, Washington, DC, USA, 2001. IEEE Computer Society

Author Profile

Saleem Azeeskhan is currently pursuing master's degree program in embedded system technologies in College Of Engineering Guindy, Anna University, Chennai-25, Tamil Nadu, India

Vanaja Ranjan is currently working as Professor in electrical and electronics engineering department, College Of Engineering Guindy, Anna University, Chennai-25, Tamil Nadu, India