MAT JOURNALS

Journal of Computer Science Engineering and Software Testing
Volume 4 Issue 2

Dynamic Real-time Classification of Data Streams

Mohini Patil, Dnyandev Bagul, Mayur Indrekar, Shraddha Pawar
Computer Engineering Department, K. K.W. I. E. E. R, Nashik, Maharashtra, India.
Email: mohini.patil1412@gmail.com**,** mayur.indrekar07@gmail.com

Abstract

To develop real time classification from high throughput of data stream (dynamic data) is one of the most challenging areas of big data analysis. In this proposed system we are using concept drift. (Changes of the pattern encoded in the stream over time). And imposes unique challenges in comparison with real time classification data mining from dynamic data. Several real-time classifications of data stream algorithms exist. The proposed system highlights the Fixed Width clustering, variable width calculation and global width clustering for data stream classifier. The result of these algorithms provides high accuracy, less time & high speed.

Keywords: *Data preprocessing, Global Width clustering, Fixed width clustering, Variable width clustering*

INTRODUCTION

Big data has 4 properties: Velocity (data generated at a quick rate), Volume (very giant and probably unknown information quantities), Veracity i.e truthfulness (uncertainty within the information) and Variety (different types of data like text, structured information etc.)Previous reference paper presents an algorithm that addresses the overlap of the Velocity and Volume aspects of Big Data Analytics. Here the classifier is trained in period on incoming tagged information instances in data stream classification. The classifier is required to accept the changes of concepts that can occur over time (known as concept drift [4]). Due to dynamic & infinite data streams, it becomes difficult to capture, store and process the data. Hence it has lead to the development and publication of data stream classifiers. An empirical analysis [15] shows that the serial implementation of MC-NN is fast and sturdy to noise and concept drifts. However, it's restricted by the outturn of one machine node. A measurability analysis is additionally dole out, distinctive insights, difficulties and solutions in implementing parallel period information stream classifiers. In the proposed system FWC, GWC and VWC algorithms are used which provide high accuracy, less time consumption and fast speed.

LITERATURE SURVEY

As data streams are infinite in length, iterative processes cannot be used. If these streams are left un-monitored, the algorithms would encode on the whole stream and neglect 'Concept Drifts'. 'Concepts' may be thought of as blocks of homogeneous/statistically similar information in a very linear timeframe. Because the length and variety of drifts is unknown, the information stream should be monitored in period of time. Previously Drift Detector Algorithms were used to handle these drifts. Examples of Concept Drift detectors are DDM [16], ECDD [17] and ELM [18]. Typically concept Drift detectors work concurrently from specific data processing algorithm or say, Data mining algorithm. Therefore the algorithm must be efficient. However, batch algorithms need many passes, it should be too slow if knowledge or data is incoming at a high speed. Various techniques exist to adapt data mining algorithms to streaming data, such as sliding window [2]

and reservoir sampling [22]. However, these techniques would work too slow for data incoming at high speed. Other techniques such as Hoeffding bound based techniques [23] and Micro-Clusters [24] have been used to create inherently adaptive data stream mining algorithms. Hoeffding based techniques aims to form and adapt the data processing models that applied math edge on the chance that the received attribute values deviates from its mean value or expected value**.** The Hoeffding bound has been successfully used to create various data stream mining algorithms known as Very Fast Machine Learning (VFDT [25]). Micro-Cluster primarily based techniques aim to make a applied math outline in terms of feature values, worth distribution and time-stamps of the information retrieved from the stream (CluStream [24], On Demand Classification of information Streams [26]). Samza can be thought of as a simplified 'pilot job' [30]. In the pilot terminology a Samza job is an 'early bound' container that will process an unknown future workload. The key distinction is that a pilot task is known as a private with its own work to method, and so has no interaction with different tasks. Samza containers they need access to all or any information more responsible the underlying stream till they're stopped outwardly. Different alternatives with similar practicality include: Apache Storm, Spark Streaming and Apache S4 — a comparison will be found in [29].

ALGORITHMS

Algorithm 1. Fixed-width clustering

1. **Input**: U
/ * A list of objects,
*/
2 Inputs: w
/* The predefined radius of cluster */
3 C ←φ; C^r←φ; C^w←φ; /* A set of cluster and their centroid and width */
4 n=0; /* The number of credited clusters */
5 **foreach** j_i in U objects **do**
6 **if** n==0 **then**
7 n+ = 1;
8put j_i in C_n; put j_i in C^r_n; put C^w_n← 0;
9**else**
10 < id, dis>← Closest Cluster (j_i, C^r);
/* Find the ID and distance of the closest cluster to j_i */
11 **if** dis ≤ω **then**
12 put j_i in C_{id};
13**if** C^w_{id}< dis **then**
14C^w_{id}< dis;
15 **else**
16 n+ = 1;
17put j_i in C_n; put j_i in C^r_n; put C^w_n← 0;
18**return** [C, C^r,C^w];

The various-widths clustering part is presented, where a data set is partitioned off into variety of clusters whose sizes are affected by the user defined threshold. 3 processes are concerned during this operation: cluster-width learning, partitioning and merging. Then they are executed till the factors or conditions are met. That is, the processes of partitioning and merging are stopped when the size of the largest cluster is less than a user-defined threshold b, or when the number of clusters prior to and after the execution of the mentioned processes are equal to b. Algorithm 2 shows the steps of the mentioned processes.

Let D be a dataset to be clustered and NN (H) be the function of k nearest neighbors for the item H, clsWidth be another function computing the dimension (radius) of NN (H), wherever the dimension is distance between the item H and therefore the farthest object among its neighbors. The value of k is set to 50% * |D| to guarantee a large cluster. To find the appropriate global width, we randomly draw a few objects from D, H= {H1, H2…Hr} where r<<|D| and for each randomly selected object the radius of its k nearest neighbours is computed and the

average is used as a global width for D as follows:

Algorithm 2. Variable Width Clustering

```
1       Input : Data
2       Input :β
3       Outputs : Clusters
4       Clusters ← ǿ: add (clusters[data,zeros,0]);
5       Finished ←0;
6       While finished == 0 do
7       Cl8,Size ←Clusters.getSize/*
        The no of clusters*/
8       Partitioning(Clusters,β);
9       Merging(Clusters,β);
10      If |LargesstCluster(Cluster)|≤β or
11      Clusters.getSize==ClsSize then
12      Finished ← 1
13      Return [Clusters];
14      Procedure Partitioning (Clusters,β)
15          U ← LargestCluster(Clusters);
16          While |U.objects| > β do
17          ώ ← Using eq. (1);
18      if (w==0) then
19              U.nonpartitioned(1);
20      Update(Clusters,U);
21          Continue;
22          <tmpClusters> ← Algorithm(U,ώ);
23      If ClusterNum (tmpClusters) > 1 then
24      Remove(clusters,U);
25          Add (clusters,tmpClusters);
26          U ← LargestClusters(clusters);
27      Else
28      ώ←ώ-(ώ*0.1);
29      goto line 21
30       Procedure Merging(Clusters,β);
31          MergingList ← ǿ /* List of tuples < */
    /* Childclusterid, parentclusterid */

        Foreach U in clusters do
    Using eq.(2) and eq.(3);
/* ID of Cluster Contained */
32          If j≠0 then
33          Put<U.getID,j> in merginglist;
34          While merginglist ≠ ǿ do
35          Foreach tuple in merginglist do
36          <i,j> ← tuple;
37          If !isParent(merginglist,i) then
38          Mergeclus(clusters,I,j);
39              Remove tuple from merginglist;
```

PROPOSED SYSTEM

This section explains the proposed system in detail. The several modules of the system are-

(1) Data preprocessing
(2) Global width calculation
(3) Variable width clustering
(4) Fixed width clustering

The block diagram of proposed system is shown as below

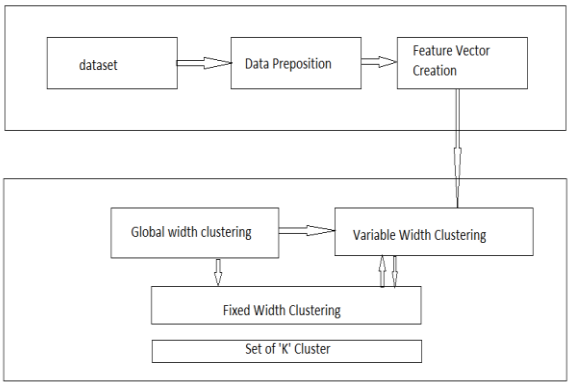


Fig 1. *Block diagram*

The brief description of each component of proposed system is described as follows:

A. Data Preprocessing

In data processing, the unwanted values, null values, stop words, noise, unwanted spaces are removed and data is cleaned. Then this cleaned data is sent to next block as input.

B. Global width calculation

Various clusters are formed based on different centroids. Every cluster then calculates its n neighbors and again a new cluster is formed.

C. Variable width clustering

The various-widths clustering part is presented, where a data set is partitioned into a number of clusters whose sizes are constrained by the user-defined threshold. Three processes are concerned during this operation: cluster-width learning, partitioning and merging. Then they are executed till the factors or conditions are met. That is, the processes of partitioning and merging are stopped once the dimensions of the most important cluster is a smaller amount than a user-defined threshold B, or once the previous quantity of clusters and when the execution of the mentioned processes are equal to B.

D. Fixed width clustering

In this algorithm, the various centroids are calculated after clusters are formed. Then these centroids are updated as new clusters get created.

GRAPHICAL USER INTERFACE

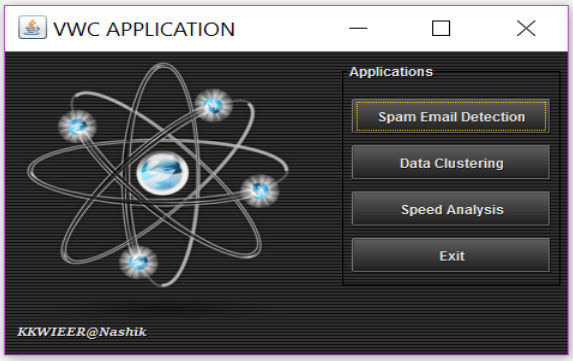
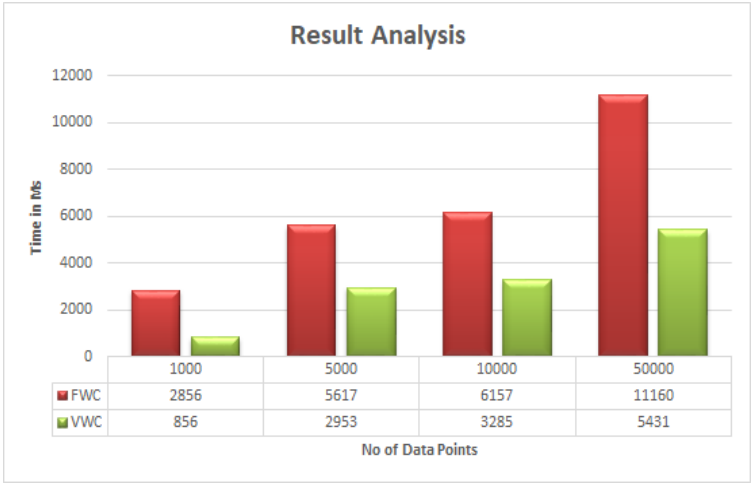


Fig 2. *GUI of Proposed system*

RESULT ANALYSIS



CHALLENGES

Fixed width clustering (FWC) algorithm takes more time in searching the points and clustering them. This reduces the performance and results in high time complexity. VWC algorithm calculates different cluster width on dense clusters created.As the points are dense in nature, searching time becomes fast. Hence overall time is reduced.

.

CONCLUSION

As the previous algorithms like FWC took more time in searching the points, time complexity was high and hence affected the performance. To overcome this, new algorithm is implemented i.e. VWC – Variable Width Clustering.

REFERENCES

1. M. Ebbers, A. Abdel-Gayed, V. Budhi, F. Dolot, Addressing Data Volume, Velocity, and Variety with IBM InfoSphere Streams V3.0, 2013.
2. B. Babcock, S. Babu, M. Datar, R. Motwani, J. Widom, Models and issues in data stream systems, in: PODS, 2002, pp. 1–16.
3. M.M. Gaber, A. Zaslavsky, S. Krishnaswamy, Mining data streams: a review, ACM SIGMOD Rec. 34 (2005) 18–26.
4. M. Gaber, A. Zaslavsky, S. Krishnaswamy, A survey of classification methods in data streams, Data Streams (2007) 39–59.
5. J. Gama, Knowledge Discovery from Data Streams, Chapman and Hall / CRC, 2010.
6. T. Bujlow, T. Riaz, J. Pedersen, A method for classification of network traffic based on c5.0 machine learning algorithm, in: 2012 International Conference on Computing, Networking and Communications, ICNC, 2012, pp. 237–241.
7. A. Jadhav, A. Jadhav, P. Jadhav, P. Kulkarni, A novel approach for the design of network intrusion detection system (NIDS), in: 2013 International Conference on Sensor Network Security Technology and Privacy Communication System, SNS PCS, 2013, pp. 22–27.
8. A. Salazar, G. Safont, A. Soriano, L. Vergara, Automatic credit card fraud detection based on non-linear signal processing, in: 2012 IEEE International Carnahan Conference on Security Technology, ICCST, 2012, pp. 207–212.
9. P. Domingos, G. Hulten, Mining high-speed data streams, in: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'00, ACM, New York, NY, USA, 2000, pp. 71–80.
10. T. Le, F. Stahl, J.B. Gomes, M.M. Gaber, G.D. Fatta, Computationally efficient rule-based classification for continuous streaming data, in: Research and Development in Intelligent Systems XXXI, Springer International Publishing, 2014, pp. 21–34. M. Tennant et al. / Future Generation Computer Systems 75 (2017) 187–199 199
11. J.a. Gama, P. Kosina, Learning decision rules from data streams, in: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two, IJCAI'11, AAAI Press, 2011, pp. 1255–1260.
12. Y. Ben-Haim, E. Tom-Tov, A streaming parallel decision tree algorithm, J. Mach. Learn. Res. 11 (2010) 849–872.
13. G.D.F. Morales, A. Bifet, Samoa: Scalable advanced massive online analysis, J. Mach. Learn. Res. 16 (2015) 149–153.
14. A. Andrzejak, J. Gomes, Parallel concept drift detection with online map reduce, in: 2012 IEEE 12th International Conference on Data

Mining Workshops, ICDMW, 2012, 402–407. http://dx.doi.org/10.1109/ICDMW.2012.102.
15. M. Tennant, F. Stahl, J. Gomes, Fast adaptive real-time classification for data streams with concept drift, in: Internet and Distributed Computing Systems, in: Lecture Notes in Computer Science, vol. 9258, Springer International Publishing, 2015, pp. 265–272.
16. J. Gama, P. Medas, G. Castillo, P. Rodrigues, Learning with drift detection, in: Advances in Artificial Intelligence–SBIA 2004, Springer Berlin, Heidelberg, 2004, pp. 286–295.
17. G.J. Ross, N.M. Adams, D.K. Tasoulis, D.J. Hand, Exponentially weighted moving average charts for detecting concept drift, Pattern Recognit. Lett. 33 (2) (2012) 191–198. http://dx.doi.org/10.1016/j.patrec.2011.08.019.
18. R. Cavalcante, A. Oliveira, An approach to handle concept drift in financial time series based on extreme learning machines and explicit drift detection, in: 2015 International Joint Conference on Neural Networks, IJCNN, 2015, pp. 1–8. http://dx.doi.org/10.1109/IJCNN.2015.7280721.
19. R.J. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, 1993.
20. C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn. 20 (3) (1995) 273–297. http://dx.doi.org/10.1023/A:1022627411411.
21. M.A. Bramer, Automatic induction of classification rules from examples using N-Prism, in: Research and Development in Intelligent Systems XVI, Springer- Verlag, Cambridge, 2000, pp. 99–121.
22. J. Han, M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufmann, 2001.
23. P. Domingos, G. Hulten, A general framework for mining massive data streams, J. Comput. Graph. Stat. 12 (2008).
24. C. Aggarwal, J. Han, J. Wang, P. Yu, A framework for clustering evolving data streams, in: Proceedings of the 29th VLDB Conference, Berlin Germany, 2003.
25. P. Domingos, G. Hulten, Mining high-speed data streams, in: KDD, 2000, pp. 71–80.
26. C.C. Aggarwal, J. Han, J. Wang, P.S. Yu, A framework for on-demand classification of evolving data streams, IEEE Trans. Knowl. Data Eng. 18 (5) (2006) 577–589.
27. Esper (http://www.espertech.com/esper/seen November 2015).
28. Samza (https://samza.apache.org/seen November 2015).
29. R. Ranjan, Streaming big data processing in datacenter clouds, IEEE Cloud Comput. 1 (1) (2014) 78–83. http://dx.doi.org/10.1109/MCC.2014.22.
30. M. Turilli, M. Santcroos, S. Jha, A comprehensive perspective on the pilot-job abstraction, CoRR abs/1508.04180. URL http://arxiv.org/abs/1508.04180.