Comparative Study of Various Decision Tree Methods for Data Stream Mining

Vaishali Mehta and Vishakha Sanghavi

Abstract Nowadays, many physical world appliances found data streams like telecommunication system, multimedia data, medical data streams. Traditional data stream mining allows storage of data and multiple scan of dataset. But it is next to impossible to save or scan it more than one or two times, because of its mountainous size. It is essential to develop the processing systems which scans once and examines the methods. Because of this, data stream mining becomes an emerging topic for research in knowledge discovery. Effective classification of such data streams finds many stream mining provocations like immeasurable length, increment learning, concept drift. So, we have to either update existing mining classifiers or generate a new technique for data stream classification. In this paper, we point out three different classification methods of decision tree called Hoeffding tree, VFDT, and CVFDT, which focuses on these classification problems.

Keywords Data stream mining · Classification · Preprocessing techniques Hoeffding tree · VFDT and CVFDT

1 Introduction

As technology develops, a number of appliances generate mountainous size of dataset refereed as data stream, which arrives continuously and rapidly grows over the time [1]. Examples of data stream include supermarket dataset, multimedia data, medical data streams and live video streams [2–4]. There is a huge requisition of scrutinize and mine streaming data as they contain expensive knowledge [2].

The traditional way of data mining algorithms which says store the entire data first and then scan it multiple times which is not applicable for streaming dataset [2]. This

V. Mehta (✉) · V. Sanghavi (✉)
Vyavasayi Vidya Pratishthan Engineering College, Rajkot, Gujrat, India
e-mail: mehtavaishali27@gmail.com

V. Sanghavi
e-mail: vishakha.vvpit@gmail.com

© Springer Nature Singapore Pte Ltd. 2019

way is not reliable, and consistent storage of data stream is not possible because of its tremendous volume of continuous data which is possibly infinite and may rapidly change over the time. So, data stream mining requires single scan algorithm [5].

There are number of techniques available for data stream mining like classification, clustering, frequent pattern mining, change detection, load shedding [6]. Data stream classification is the process of class label prediction from continuous data streams which becomes an emerging area of research because of its number of physical world applications such as fraud detection, malicious Web page, detection of medical abnormal disease arriving at hospitals.

There are lots of techniques available which may be used to solve these problems like decision tree, naïve Bayes, nearest neighbor [6]. In this paper, we compare some decision tree methods like Hoeffding tree, very fast decision tree (VFDT), and concept-adapting very fast decision tree (CVFDT). Traditional data stream classification techniques like Hoeffding tree and VFDT mainly focus on these two problems: infinite length and increment learning. At present, in most of the applications, data classes are defined temporarily even if features of data classes are not changed or somewhat changed. This change of the temporal class can be defined as concept drift or class drift [7]. This problem is not solved by Hoeffding tree and VFDT. So, the extended version of VFDT referred as CVFDT is developed. It provides same advantages as VFDT in terms of speed and accuracy. Also, it can find and react to the changes if any arises during example generation process.

The formulation of this paper is: Sect. 2 briefly studies and summarizes preprocessing techniques for data stream mining. In Sect. 3, we discuss some classification algorithms and how they work. Section 4 concludes our paper and presents the future work.

2 Preprocessing Techniques

Process of data stream mining contains many problems and challenges. For minimizing this issue, we have to process the data stream so that mining of data stream will become easier. There are number of preprocessing techniques available for data stream mining which can be split into two parts: (1) data-based techniques and (2) task-based techniques [8]. Data-based techniques either choose the subset of the incoming stream or summarize the whole dataset. While task-based techniques, either modify the existing techniques or generate new ones.

2.1 Data-Based Techniques

Sampling. Sampling is the process of finding the data item to be processed or not by using probabilistic choice. The problem in this is we cannot use sampling for data stream analysis because of its unknown dataset size. Also, sampling is not able to

solve the problem of fluctuating data rates. The issue with the sampling is how to investigate the connection between data rate, sampling rate, and error bounds [8, 9].
Sketching. Process of randomly selecting the subset of features is called sketching. It vertically samples the stream. The biggest disadvantage of sketching is reduced accuracy [8].
Load Shedding. Here they use dropping mechanism to manage the load of data stream. This process is very successful for querying data streams. The problem is that it drops number of data streams which might be useful for structuring the generated models or for selecting the pattern of interest [8].
Synopsis Data Structures. It summarizes the data stream for further analysis. The drawback of this system is that it generates approximate answer which may not specify appropriate dataset [8].
Aggregation. It organizes input in summarized form. We can use these aggregate data in data mining algorithms. But the problem is that it decreases the efficiency due to highly fluctuating data distributions [8].

2.2 Task-Based Techniques

Approximation Algorithms. Sometimes, it is next to impossible to generate exact result for some computationally hard problems. Approximation algorithms are the process of designing the algorithm for such computationally hard problems, which generates approximate results with some error bounds [8].
Sliding Window. When user needs to analyze most recent data streams, sliding window can be used. For more recent dataset, detailed analysis is performed and for old ones, summarized versions are used. But the major drawback of this system is that it is sensitive to its window size [8, 9].
Algorithm Output Granularity (AOG). It is the only approach introduced by AOG which depends upon resources as per storage capacity and speed. If there is very high fluctuating data rates, then AOG can easily copes up with it. AOG can be divided into three main stages. First, mining is done by adaptation of resources. Second, data stream rates, and third, merging the generated knowledge structures. We can use AOG for clustering, classification, and frequency counting [8].

3 Literature Survey

3.1 Hoeffding Tree

Domingos and Hulten introduced Hoeffding tree method for inducing incremental decision tree from data stream mining [10]. This algorithm is a starting point as it is effective and innovative for high-speed data stream classification. Still there are opportunities for improvement. Hoeffding tree is generalized method for stream mining [10]. It is a single scan algorithm as it inspects each and every example in the stream only once. There is no need to store this example in memory after they update the tree. It only stores the tree in main memory, which contains required data in order to build the tree and predicts the class label at any time even in between the processing of training example.

The first thing to do is to choose the root node from the given set of example. After root node is chosen, the succeeding examples are used to select the appropriate attributes by going through the corresponding leaves. This is the recursive process in data stream mining. The problem is to decide required number of examples for each node. To solve this problem, we can use Hoeffding bound (€) with use of statistical results. Equation of Hoeffding bound is as per given below [11].

$$€ = \sqrt{\frac{R^2 \ln\left(\frac{1}{\delta}\right)}{2n}} \tag{1}$$

Algorithm: Hoeffding tree.

```
1. Initially consider Hoeffding tree with single root node
2. Repeat for all incoming training example
    2.1 Sort each example into leaf node (l) using Hoeffding tree (HT),
    2.2 Update Hoeffding tree with leaf node l
    2.3 Increment number of examples seen at l
3. If the examples seen till now at leaf are not all of the same class, then
    3.1 Compute split evaluation function G () for each attribute xi and suppose Xa
        is attribute with highest G() and Xb is attribute with second highest G()
    3.2 Compute Hoeffding bound $€ = \sqrt{\frac{R^2 \ln\left(\frac{1}{\delta}\right)}{2n}}$
4. Check If G(Xa)-G(Xb) > ε and Xa is not a null attribute, then splits Xa with its
   internal node and replace leaf l.
5. For all branches of the split, repeat the following steps
    5.1 Add a new leaf
    5.2 Update set of attribute list
    5.3 Calculate the split evolution function for new leaf
6. Return HT.
```

Comparing it with traditional techniques, Hoeffding tree generates more accurate result for large amount of dataset [11]. It introduces Hoeffding bound, and the generalize method for splitting the attribute which is not available in traditional techniques. Traditional technique requires more time because they need multiple scan of data while Hoeffding tree requires less time because it allows single scan of continuous dataset. Hoeffding tree can classify the data while tree is growing because it allows incremental learning while traditional technique is not able to do this.

3.2 VFDT

VFDT is a one type of learning method occupied from decision tree, which is augmented from the Hoeffding tree [12]. It uses either information gain or Gini index as the measure of attribute evaluation. It provides many refinements to the Hoeffding tree algorithm [12].

Refinement Parameters.

Ties. When two or more attributes have very close G's, it requires to process many examples to determine the best one. For figure out splitting point based on the current best attribute and to find out effective tie, we can use user-supplied tie threshold (τ) provided by VFDT [11, 12].
G computation. Even if there is only one example is change or any new example is introduced, we have to recalculate the G, which is not required and it takes huge amount of time for recompilation. So solution is we can use a number (nmin) for new examples before the recompilation of G each and every time [11].
Memory and poor attribute. VFDT allows to deactivate the leaf and attribute which is not useful for long time and which leads to minimize the memory usage. Because of this, the memory can available for new leaf [11, 12].
Rescan. The examples which are already scanned, VFDT allows to rescan them if there is a requirement. This option can only be activated if data arrives slowly or dataset is small enough that it is feasible to be scanned multiple times [11].

Algorithm: VFDT.

```
1. Initially consider VFDT with single root node
2. Read next example
    2.1 Pass the example starting from the root till the leaf node (l)
    2.2 Update tree with leaf node (l)
    2.3 Increment number of examples seen at l
3. If the examples seen till now at leaf are not all of the same class or leaf (Number
   of examples) > Nmin, then
    3.1 Compute split evaluation function G() for each attribute xi and suppose Xa
        is attribute with highest G() and Xb is attribute with second highest G()
    3.2 Compute Hoeffding bound $\epsilon = \sqrt{\frac{R^2 \ln(\frac{1}{\delta})}{2n}}$
4. Consider $\Delta G =$ G (Xa)-G (Xb)
    4.1 Check If (($\Delta$G > $\epsilon$) or ($\Delta$G <= $\epsilon$ < $\tau$)) where $\tau$ is user-supplied tie threshold
    4.2 If Xa is not a null attribute, then splits Xa with its internal node and replace
        leaf (l).
5. For all branches of the split, repeat the following steps
    5.1 Add a new leaf
    5.2 Update set of attribute list
6. Return HT.
```

In the paper [11], comparison of VFDT and C4.5 is given to verify whether VFDT can generate more accurate trees then conventional system or not. To test and compare how C4.5 and VFDT react when number of examples are increases, they use some parameters like accuracy, number of nodes, noise.

Accuracy. C4.5 generates more accuracy for less number of examples normally up to 25 k examples. After that in the range of 25–100 k both has almost same accuracy. But when number of example is more than 100 k VFDT, generate more accurate result.
Number of nodes. In order to achieve good accuracy, C4.5 generates more number of nodes with small number of examples, while VFDT requires very less number of nodes for generate greater accuracy.
Noise. In C4.5 as noise is increased to 50%, accuracy is decreased within the 100 k of examples. While in VFDT after 20 millions of examples, accuracy comes at 50%.

This indicates that VFDT generates more accurate result when huge amount of data is there.

3.3 CVFDT

CVFDT provides same speed and accuracy advantages as VFDT. As VFDT is not able to detect and respond any change which introduces an existing attribute in data stream mining [12]. It generates the need for extend the VFDT algorithm. An extended version of VFDT algorithm is called CVFDT. Whenever any new data arrives, most of the systems need to learn a new model. But CVFDT supervises the aspect of unknown data continuously and replace old ones which are no longer useful. For that, CVFDT uses sliding window for various dataset [13]. CVFDT increment the counter for unknown data and decrement the counter for known data in the window. In order to asset valuable attribute at root, CVFDT creates alternative sub-tree and replaces old sub-tree with new best tree which is more accurate on new data [13].

Algorithm: CVFDT.

```
1. Initially consider tree with single root node
2. Alternate tree for each node of HT is start as empty
3. For each example repeat
    3.1 Read next example of string
    3.2 Using HT pass example down to a set of leaves
    3.3 Also pass through all alternate trees of the attributes
4. Add new example to the sliding window
```

```
5. If sliding window overflow, then
    5.1 Remove oldest example and forget its effect
6. CVFDTGrow
7. If from last checking of alternate tree for examples has been seen, then
    7.1 Check split validity
8. Return HT.
```

Algorithm: CVFDTGrow.

```
1. Process the example through the leaf using HT.
2. For each nodes within set of nodes traverse the attribute. Read next example
    2.1 Increment number of examples seen
    2.2 Consider CVFDTGrow for each alternate tree
3. Count number of example seen so far and count majority class visited among
   examples seen so far.
4. If the examples seen till now are not all of the same class or leaf (number of
   examples) > Nmin, then
    4.1 Compute split evaluation function G() for each attribute xi and suppose Xa
        is attribute with highest G() and Xb is attribute with second highest G()
    4.2 Compute Hoeffding bound $\epsilon = \sqrt{\frac{R^2 \ln\left(\frac{1}{\delta}\right)}{2n}}$
5. Consider $\Delta G = G(Xa) - G(Xb)$
    5.1 Check If (($\Delta G > \epsilon$) or ($\Delta G <= \epsilon < \tau$)) and Xa is not a null attribute where $\tau$
        is user-supplied tie threshold
    5.2 Splits Xa with its internal node and replace leaf l.
6. For all branches of the split, repeat the following steps
    6.1 Add a new leaf
    6.2 Update set of attribute list
    6.3 Generate the empty alternate tree
7. Return HT.
```

In paper [12], comparison of VFDT and CVFDT is given. It tests the ability of both algorithms in order to deal with large concept drifting dataset. Some of the parameters which are used for this comparison are error rate, number of attributes, number of examples, tree size, concept drift, etc.

Error rate. Before VFDT is reacting to the change, its error rate is increased drastically. While CVFDT can quickly respond to change, error rate is constant.

Tree size. Size of CVFDT is small than VFDT, and the reason is that it uses only 100000 most relevant examples to build the tree while VFDT uses millions of outdated examples so it is comparatively large.

Drop the attribute. CVFDT allows to drop oldest attribute as they are no longer use.

4 Conclusion and Future Work

In this paper, we considered some preprocessing techniques of data stream mining and further we discussed some theoretical aspects of classification algorithms for data stream mining like Hoeffding tree, VFDT, and CVFDT. Hoeffding tree requires very small period of time for learning. There is no similarity between Hoeffding tree and batch trees. VFDT is augmented from the Hoeffding tree. As compared to Hoeffding tree, VFDT can generate more accurate result with less amount of time and memory. Its accuracy is greatly improved after 100 k examples, and it works better with large amount of continuous dataset. But VFDT and Hoeffding tree cannot work with concept drift. To solve this, problem CVFDT is introduced, which is the extended version of VFDT. Compare to VFDT, CVFDT can generate more accurate result with high-speed data stream.

Biggest advantage of CVFDT is that it solves concept drift problems. CVFDT uses sliding window concept for mining the data stream. The problem is that there is no optimal window size, if size is small then accuracy will decrease and if size is large, then we have to work with more number of examples. So, in future we can study the algorithm which can solve this sliding window problems. Another problem is CVFDT discards old subtrees which can be useful in future or in some other areas. We can study these situations and try to resolve this.

References

1. Han J, Kamber M, Pei J (2011) Data mining: concepts and techniques, 3rd edn. Morgan kaufmann publishers, Elsevier
2. Aggarwal C (2006) Data streams models and algorithms, advances in database systems. Springer Verlag
3. Kotecha R, Garg S (2015) Data streams and privacy: two emerging issues in data classification. In: 5th Nirma university international conference on engineering (NUiCONE) 2015. IEEE conference publication, pp 1–6
4. Abdulsalam H, Skillicorn B, Martin P (2011) Classification using streaming random forests. IEEE Trans Knowl Data Eng 23:22–36 (IEEE journals and magazines)
5. Mao G, Yang Y (2011) A micro-cluster based ensemble approach for classifying distributed data streams. In: 23rd IEEE International conference on tools with artificial intelligence 2011. IEEE conference publication, pp 753–759
6. Kirkby R (2007) Improving Hoeffding Trees. Ph.D. thesis, Department of Computer Science, University of Waikato
7. Masud M, Gao J, Khan L, Han J, Thuraisingham B (2011) Classification and novel class detection in concept-drifting data streams under time constraints. IEEE Trans Knowl Data Eng 23:859–874 (IEEE journals and magazines)

8. Kholghi M, Keyvanpour M (2011) An analytical framework for data stream mining techniques based on challenges and requirements. Int J Eng Sci Technol (IJEST) 3:1–7
9. Aggarwal C, Han J, Wang J, Yu P (2006) A framework for on-demand classification of evolving data streams. IEEE Trans Knowl Data Eng 18:577–589 (IEEE journals and magazines)
10. Shukla M, Rathod K (2013) Stream data mining and comparative study of classification algorithms. Int J Eng Res Appl 3(1):163–168
11. Domingos P, Hulten G (2001) Mining high-speed data streams. In: International conference on knowledge discovery and data mining
12. Domingos P, Spencer L, Hulten G (2001) Mining time-changing data streams. In: 7th ACM SIGKDD international conference on knowledge discovery and data mining
13. Raahemi1 B, Zhong W, Liu J (2008) Peer-to-Peer traffic identification by mining IP layer data streams using concept-adapting very fast decision tree. In: 20th IEEE International conference on tools with artificial intelligence 2008, vol 1. IEEE conference publication, pp 525–532
14. Thakong M, Phimoltares S, Jaiyen S, Lursinsap C (2017) Fast learning and testing for imbalanced multi-class changes in streaming data by dynamic multi-stratum network. IEEE Access 5:10633–10648 (IEEE journals and magazines)