Advances in Computational Sciences and Technology
ISSN 0973-6107 Volume 10, Number 5 (2017) pp. 1531-1542
© Research India Publications
http://www.ripublication.com

Ensemble Technique for Efficient Stream Data Processing

S. Aquter Babu

*Assistant Professor, Department of Computer Science,
Dravidian University, Kuppam, Chittoor District, Andhra Pradesh, India.*

Abstract

Ensemble learning technique is a suitable method for stream data classification. In data mining literature, ensemble learning techniques can handle and control very large volumes of stream data and concept drifting features. Stream data classification is one of the most important data analysis techniques in stream data management. Existing ensemble learning techniques mainly concentrates only on constructing accurate stream data classifiers but not fully concentrated on prediction efficiency of the stream data classifiers. Ensemble learning uses a set of pre-specified base classifiers for prediction. Time complexity for referencing base classifiers is O(n). Stream data classification is costly when the ensemble size is very large. Linear time complexity of ensembles learning is inadequate for many real time stream data classification applications. A new method is proposed for node splitting of state-of-the-art ensemble indexing tree structure creation.

Keywords: Stream data classification, streams, stream data features, stream data applications, drifting features

1. INTRODUCTION

There exist many real time stream data classification applications such as e-mail, mobile data, satellite data, and video conferencing data and so on. Data stream classification represents one of the most important tasks in data stream mining [1], which has been popularly used in real-time intrusion detection, spam filtering, and malicious website monitoring. Compared to traditional classification, data stream classification is facing two extra challenges: large/increasing data volumes and drifting/evolving concepts [1].

Identifying correct or outlier stream record within the specified time interval is of great important requirement in stream data classification. It is a well known fact that stream data classification is inherently associated with two potential problems - concept drifting and very large sizes of stream data volumes. In the data mining literature, numerous new ensemble learning models have been introduced. While these models vary from one to another, they share very most important striking similarity feature in their design: using divide-and-conquer techniques to handle large volumes of stream data with concept drifting [1]. Many ensemble-based models have been proposed recently including weighted classifier ensembles [2].

List of important ensemble based models are

1. Incremental classifier ensemble models
2. Weighted classifier ensemble models for stream data classification
3. Classifier and cluster ensemble stream data models
4. Fuzzy ensemble based steam data models

Ensemble tree indexing data structure plays an important role in stream data classification. The time complexity of insertion, deletion, updating and search operations in ensemble tree is O(log n). In particular time complexity of any non linear tree data structure is O(log n). Tree indexing including ensemble tree indexing, follows divide and conquer strategy. Ensemble indexing tree data structure is efficient and effective indexing structure to handle potentially infinite stream data with inherently associated drifting features. Existing works only consider combining a small number of base classifiers [3].

Ensemble learning techniques divide the continuous flow of stream data into discrete parts of stream data records. Each part is of finite collection of stream data records. Ensemble model constructs ‘n’ number of base decision tree classifiers corresponding to the ‘n’ number of parts of continuous data. For classifier prediction these ‘n’ number of individual base decision tree classifiers are combined for determining collective prediction.

Nowadays numbers of online applications are increasing rapidly and online applications are the potential candidate applications for applying many of the stream related measurement determinations. Webpage stream data monitoring is one best example for stream data related applications and it is used for finding outlier detections. Many spatial database systems are practically in use and for efficient representation of spatial database objects in the spatial database definitely state-of-the-art indexing data structures are required. These indexing data structures must be robust, scalable, accurate, and highly predictable.

Ensemble based learning has many advantages such as

1. Ensemble models are easily scalable
2. Ensemble models are easily portable
3. They learn and respond very quickly in managing new concepts, features and techniques
4. They produce low variance errors
5. Suitable for easy parallelization of stream data processing
6. Suitable for many real-time applications

In the literature of stream data classification, generally, the stream data handling classifiers are limited in number say, for example 40 to 50 only. For many real-time applications this limited size of ensemble is not suitable for prediction efficiency. The prediction efficiency increases when ensemble size increases. Also, the ensemble must handle variety of stream data features in order to reflect very close to the real-life applications. Ensemble based stream data handling technique must manage, control, capture, reflect, use, modify, read, write and learn and respond dynamically with fast, correct, efficient and quality ensemble results.

2. RELATED WORK

Stream data classification methods are divided into two categories – ensemble learning models and online incremental models. Stream data classification is nowadays slowly gaining its popularity in many applications and lot of demand is there in various fields. Many researchers are entering into this field continuously. Ensemble with large model size is more use full than ensemble with smaller sizes. Also one must realize the importance of ensemble learning models that are scalable, parallelizable, robust, dynamic feature derivable, adaptable very quickly to new trends, techniques, changes, and produce very low variance errors.

In the case of stream data classification, existing data stream classification models can be roughly categorized into two groups: online/incremental models and ensemble learning [4]. Many spatial indexing structures exist that utilize shared patterns among spatial data to reduce query and update costs [5].

Ensemble model is a big model and it is represented as a collection of small classification models. Real time applications for online incremental model are very fast decision tree models and these models are most efficient incremental support vector machine (SVM) model. Many research people think that ensemble model is the newest model that is particularly suitable for stream data classification.

Stream data sizes are potentially infinite and for very large stream data sizes indexing is compulsory for efficient management of stream data. In the literature different people have proposed different types of stream data indexing structures. There exist many differences among these indexing structures. For example, dynamic indexing structures are needed for multimedia data, micro-clusters data, video conference data, e-mail data, satellite data, and phone data and so on. The 1D random interval and its properties in R-trees have been studied by Tao and Papadias [6].

Tree indexing data structures have sub-linear time complexity for all the operations like insert, delete and update. Various types of indexing tree data structures that are relevant and related in the literature are – R-tree, R*-tree R^+-tree, M-tree, k-d tree, s-tree, B-trees, B^+-trees, B^*-trees, Oct-trees, Hilbert trees, R-trees, ss-tree and L-tree and so on. Ensemble trees are extended version of R-trees. Sometimes ensemble pruning techniques may be very useful to reduce computational costs and memory costs.

Varieties of online decision trees have been proposed for many real time streams of classification techniques or regression techniques. Option trees are suitable for any time learning and prediction. Classifier indexing is useful for increasing the prediction accuracy in data rich applications. One classifier can be indexed or several classifiers can be indexed. Indexing several decision tree classifiers is particularly useful for many real time data intensive stream data classification applications.

3. PROBLEM DEFINITION

Assume that S represents stream data consisting of n tuples each of which has k number of predictor attributes and one class attribute. This problem is a supervised learning problem. For simplicity consider only two class problem. Class labels are 1 and 0. At present consider only two dimensional problems, later on it can be extended to multidimensional stream data management problem.

In the case of stream data classification the main desirable property is that the time complexity for predicting the stream data record must be logarithmic, log(n), time complexity only. Initially all the decision tree classifier models are constructed and then all the decision rules derived from the decision tree classfiers are represented in terms of if-then rules. All these if-then rules are converted into spatial objects and these spatial objects are stored in the ensemble indexing tree data structure for efficient processing.

Each decision rule is represented by one rectangle and n x k decision rules are represented by n.k rectangles and correspondingly n.k rectangles are converted into nk spatial objects and stored in the spatial database indexing data structure. Arriving new stream data record is predicted in logarithmic, (log N), time complexity by using state-of-the-art ensemble stream indexing data structure. For prediction, a search operation initiated in the indexing tree data structure and then based on the search results the incoming stream data record predicted in logarithmic time complexity, O(log(N)). This logarithmic time complexity opens the doors for applying many real time problems for prediction of stream data record.

For simplicity purpose consider only two dimensional spatial database objects. In reality there may exist many dimensional objects. That is entire spatial database is modeled by using two dimensional representation only. Complete spatial database is represented by a big rectangle called BIGRECTANGLE with dimensions

BIGRECTANGLE = (0, 0, 7, 7). Here big rectangle is specified by giving lower and upper corners of the special database rectangle.

Many of the existing spatial database indexing structures are designed and developed only for efficient management of applications involving images, graphics, maps, multimedia, videoconferencing, and satellite based details only. In this paper a new concept is introduced for representing decision tree classifier models in the spatial database in terms of classifiers rules to rectangles to spatial database objects.

All the decision rules induced by the decision tree classifiers are represented in the area of BIGRECTANGLE only. Main operations of ensemble indexing tree data structure are – insertion, search and delete operations. Consider one hypothetical example given in the following TABLE-1. Here, ten decision tree classifiers details are given. All the decision tree classifier models are represented in terms of if-then rules. This is only a sample and small spatial database only. In real time actual sizes of spatial databases are potentially very large.

Decision trees are abundantly used in both machine learning and data mining applications. Decision tree classifier models are selected for ease of explanation and implementation and also because of their ease of interpretability. Also decision tree classifiers are used as benchmark techniques in many real applications. Decisions trees are superior to many available tools for classification and clustering.

Table 1. Decision rules of 10 decision tree classifiers

Decision Tree Classifier No	**Classifier Decision Rules**
1 (R1)	if $(1 \le x \le 3)$ and $(1 \le y \le 2)$ then class 1 else class 0
2 (R2)	if $(2 \le x \le 4)$ and $(0.5 \le y \le 3)$ then class 1; else class 0
3 (R3)	if $(3.5 \le x \le 4.5)$ and $(1 \le y \le 2.5)$ then class1; else class 0
4 (R4)	if $(3 \le x \le 5)$ and $(2 \le y \le 4)$ then class 1; else class 0
5 (R5)	if $(5.5 \le x \le 7)$ and $(2 \le y \le 3)$ then class 1; else class 0
6 (R6)	if $(6 \le x \le 7)$ and $(4 \le y \le 5)$ then class 1; else class 0
7 (R7)	if $(8 \le x \le 9)$ and $(8 \le y \le 9)$ then class 1; else class 0
8 (R8)	if $(8 \le x \le 9)$ and $(5 \le y \le 7)$ then class 1; else class 0
9 (R9)	if $(8 \le x \le 10)$ and $(2 \le y \le 3)$ then class 1; else class 0
10 (R10)	if $(9 \le x \le 10)$ and $(4 \le y \le 9)$ then class 1; else class 0

Insertion into indexing tree data structure

A set of decision rules of the new base classifier is inserted into the ensemble indexing tree data structure. After insertion resultant indexing structure reflects all the changes, patterns, modifications, features, drift features and many other statistical details. Before insertion of a new rule into indexing tree data structure a search operation is initiated in the indexing structure for finding an appropriate leaf node that contains space for new entry.

If the insertion space is available after search operation the new entry is inserted in the appropriate leaf node. After successful insertion the tree structure is updated so that the tree satisfies all the rules of minimum bounding rectangle rules. During insertion the minimum and maximum bound entries of leaf node are taken into consideration. Indexing tree updating follows bottom to top up gradation (modification) approach.

When there is no space in the leaf node for inserting a new decision rule then the tree node split occurs. After node split the new entry will be inserted into the appropriate node with satisfying all rules. Of course, node splitting is very difficult during indexing tree construction. Proposed technique introduces an easy node splitting technique with time complexity, O(1), only. O(1) means constant time complexity.

ALGORITHM-1: Ensemble_Insert_Operation

Insertion into ensemble indexing tree data structure

INPUT:

Ensemble indexing tree, set of new classifier rules, minimumBound, and maximumBound of tree node

OUTPUT:

Modified indexing tree data structure

```
1.    p ← rootNode(Tree)
2.    for each new decision rule i of the new classifier do
3.        leaf ← searchLeafNode(i, p)
4.        if (leaf.size < minimumBound) then
5.            add rule i to leaf
6.            newRoot ← modifyParent (leaf)
7.        else
8.            (left, right)←call splitNodeMethod (leaf, i)
9.            newRoot←call update (Tree, left, right)
10.   Endfor
```

Variable	Explanation
Tree	is the root node pointer of the indexing tree data structure
minimumBound	Minimum number of entries that must be present in each node of the tree
maximumBound	Maximum number of entries that must be present in each node of the tree
rootNode	starting address of the indexing tree data structure

Leaf	leaf node of the indexing tree
newRoot	Modified address of the rootNode of the indexing tree data structure
Left	left pointer after split

After each insertion of the value into the indexing structure, the tree must be updated dynamically in order to reflect new operations.

ALGORITHM-2: Ensemble_Tree_Search

To apply search operation in indexing data structure

INPUT:

ensemble indexing tree, T, input record x

OUTPUT:

Predicted class label of input record x

```
1.      Create stack
2.      S = { }
3.      for each entry i of the indexing tree do
4.          push (stack, i)
5.      End of for each
6.      while (stack is not empty) do
7.        element ← pop(stack)
8.        if (element is present in the leaf) then
9.            S ← S U element
10.       else
11.           pointer ← getchild of e
12.           foreach entry k belongs to pointer do
13.               if (x ϵ k) then
14.                   push(stack, k)
15.               end-if
16.           end-for
17.       end-if
18.         y ← compute class label
19.     end-while
```

stack ---- stack data structure stores all references of nodes in the indexing tree

stack is a data structure used for storing node references for future processing with possible and required all of the stack operations.

S ---- This set stores all the entries of the leaf node

pointer ----- pointer is a reference which stores address of a node

----

PROPOSED ALGORITHM-3:

ENSEMBLE_TREE_NEW_NODESPIT

This algorithm finds place for inserting a new node into ensemble indexing tree data structure

INPUT:

Ensemble indexing tree, Tree, set of new classifier rules, minimum-Bound, and maximum-Bounds of the node in the indexing tree data structure.

OUTPUT:

Modified indexing tree data structure

```
11.    p ← rootNode(Tree)
12.    for each new decision rule i of the new classifier do
13.        leaf ← searchLeafNode(i, p)
14.        findInsertionPlaceUsingBinarySearch
15.        if (leaf.size < minimumBound) then
16.            add rule i to leaf
17.            newRoot ← modifyParent (leaf)
18.        else
19.            (left, right)←call splitNodeMethod (leaf, i)
20.            newRoot←call update (Tree, left, right)
21.    Endfor
```

A new technique is proposed for splitting a node whenever node split occurs during insertion of the ensemble indexing tree creation. During insertion operation a new entry will be added to the node corresponding node. The process of entry insertion is divided into two cases. The first case is called without splitting and the second case s called with splitting. I the first case new entry will be inserted into the corresponding leaf node. During this entry insertion time a search is performed in the leaf node by using binary search technique and then insert the new entry in the appropriate order place of the node so that all entry values are in the sorted order within the node. This procedure is very easy and it requires logarithmic time complexity, log(n), for insertion.

In the second case of node splitting insertion place is determined first by applying binary search technique and the new entry is inserted in that place. Once the new entry is inserted into the appropriate place of the node then divide the node values into two parts simply by dividing all the node values into two halves with the time complexity of O(1). Total time required for insertion is = O(log n) + O(1) = O(log n). This is very much efficient than applying the sorting technique for sorting all the node values. That is on the fly insert new entry in sorted order using binary search to reduce the time complexity of node splitting from O(n log(n)) to O(n). When the data size in the node is very large, the newly proposed node splitting process makes the bigger difference in insertion time.

A sample, small and hypothetical spatial database example is shown in the FIGURE-1. Each rectangle represents one spatial database object. Set of rectangle objects constitutes a new spatial database. The goal of general spatial database indexing technique is to speed up the insert, retrieval and updating of spatial database objects where as the goal of ensemble indexing technique is for fast prediction of incoming stream data record with multiple and multi facet dynamically changing features such as drift changes, feature changes, speed changes, and quantity changes.

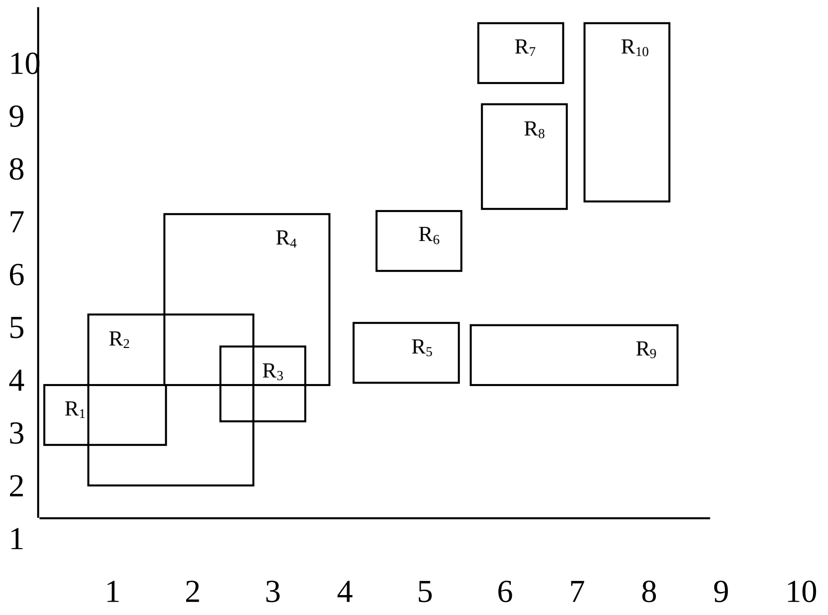

Figure 1: Spatial database representation using rectangle objects

Spatial database objects are represented in FIGURE-1 and the corresponding spatial database objects representation in the ensemble indexing tree data structure are shown in the FIGURE-2. Every time during insertion of a new object, the process starts from the root and continues towards finding the leaf node of the indexing tree. Once the leaf is found then the leaf is checked whether space is available or not. If the space is available then spatial database object is inserted successfully. If the space is not available for insertion then node is split into two nodes with appropriate new insertion.

Objects are specified in the two dimensional scale and whenever it is required these measurement values more appropriate measures, then all these measures can be normalized for uniform processing by using any of the available standard uniform scale measurements in the appropriate system. Consider the following example of stream data records (decision tree rules):

1, 1, 3, 2 ------ 1

2, 0.5, 4, 3 ----- 1

3.5, 1, 4.5, 2.5 ----- 1

3, 2, 5, 4 ------ 1

5.5, 2, 7, 3 ------ 1

6, 4, 7, 5 ------1

8, 8, 9, 9 ----- 1

8, 5, 9, 7 ------ 1

8, 2, 10, 3 ----- 1

9, 4, 10, 9 ------ 1

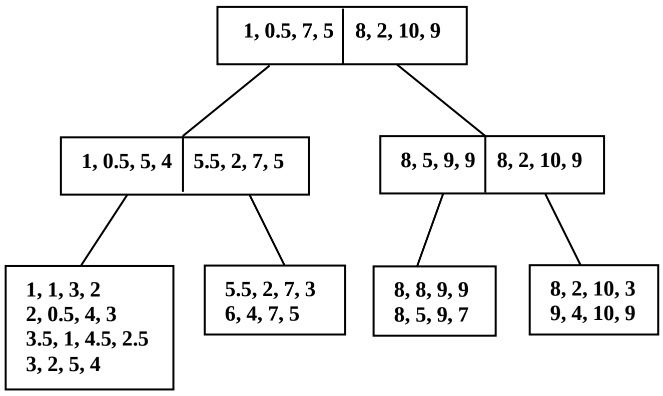

Figure 2: Ensemble indexing tree data structure for the data shown in Table 1.

Ensemble indexing tree search operation can be performed either by using Breadth First Search (BFS) or Depth First Search (DFS). Present study is based on DFS framework procedure through the entire indexing tree data structure for searching operation. Search operation is used for both prediction and insertion of a stream data record.

Depth first search (DFS) is more suitable to increase the speed of search tree operation and the insertion operation and in particularly DFS is useful for online search. That is depth first search is the potentially more suitable search procedure for spatial database searches for speeding up all operations such search, insert, and delete operations

Most important points in ensemble stream data management

In general, ensemble classifiers are more accurate and more efficient and they produce lower expected error rates when compared with randomly selected single classifiers. In the case of stream data classifications most important points to be considered are - dynamic features such as drift changes, feature changes, stream data speed changes, requirements changes, query changes and change in the other statistical measures.

CONCLUSION

Ensemble indexing tree data structure is an efficient indexing structure for efficient management of spatial databases with time complexity of logarithmic value. It is particularly useful for stream data classification. Now-a-days there are many applications that require stream data classification. Existing methods are not suitable for handling stream data classification. Present study has considered only two problems. In the future, ensemble indexing stream data management technique can be extended to multiclass label prediction also. Also there is a possibility to apply and use many diffusion based models for efficient, effective, scalable and dynamic management of many stream data related applications.

In the future, work will be extended to apply many new techniques for deleting a classifier in the ensemble tree indexing structure. One technique is deleting the classifier whose weight is the minimum. Another technique is deleting the classifier whose age is very high. Another technique is delete the classifier whose strength is very low in determining the dynamic features such drifting, statistical measures, feature extraction capability, and feature prediction

REFERENCES

[1] Peng Zhang, Chuan Zhou, Peng Wang, Byron J. Gao, Xingquan Zhu, and Li Guo, "E-Tree: An Efficient Indexing Structure for Ensemble Models on Data Streams,", IEEE TRANS. ON KNOWLEDGE AND DATA ENGINEERING, VOL. 27, NO. 2, FEBRUARY 2015 pages 461-474

[2] H. Wang, W. Fan, P. Yu, and J. Han, "Mining Concept-Drifting Data Streams Using Ensemble Classifiers," Proc. Ninth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD), 2003.

[3] P. Zhang, X. Zhu, J. Tan, and L. Guo, "Classifier and Cluster Ensembles for Mining Concept Drifting Data Streams," Proc. IEEE 10th Int'l Conf. Data Mining (ICDM), 2010.

[4] P. Zhang, X. Zhu, Y. Shi, L. Guo, and X. Wu, "Robust Ensembler Learning for Mining Noisy Data Streams," Decision Support Systems, vol. 50, no. 2, pp. 469-479, 2011.

[5] R. Guting, "An Introduction to Spatial Database Systems," VLDB J., vol. 3, no. 4, pp. 357-399, 1994.

[6] Y. Tao and D. Papadias, "Performance Analysis of R_-Trees with Arbitrary Node Extents," IEEE Trans. Knowledge and Data Eng., vol.16, no. 6, pp. 653-668, June 2014.