Extended R-Tree Indexing Structure for Ensemble Stream Data Classification

P. Sravanthi
M.Tech Student, Department of CSE
KMM Institute of Technology and Sciences
Tirupati, India

K. Hema
Assistant Professor, Department of CSE
KMM Institute of Technology and Sciences
Tirupati, India

***Abstract*– Ensemble learning is one of the most important techniques in machine learning. Ensemble based solutions are more accurate to the problem of large scale data classification. Ensemble learning method handles both large volumes of stream data and concept drifting feature. All the ensemble methods to date are mainly concentrated in constructing more accurate ensemble models only but not on prediction of the class label of the incoming record. For a large set of ensemble classifiers, linear scan of all classifiers in the ensemble is a time taking process and it may not be suitable for many real life applications in prediction. Finding highly rated websites, business services, document flows, trends in share marketing etc are potential candidate applications for ensemble learning.**

Currently the Linear time complexity of dynamic data stream classification/prediction using ensemble learning technique is costly in terms of response time. Ensemble learning techniques with linear time complexity are not suitable for many real world and time critical dynamic data stream applications such as spam detection, web traffic stream monitoring and intrusion detection. Newly proposed Extended Ensemble tree (EE-tree) indexing structure technique reduces the time complexity from linear to sub linear in predicting a randomly selected stream record. E-trees represent all the ensembles in a spatial database. E-tree are height balanced tree structures with automatically updated features in addition to the continuously inserting new classifiers and deleting old classifiers and self-adapting to new trends, features, patterns in connection of changes in the dynamic data streams. Experimental results show that the performance of the proposed approach with sub-linear time complexity is superior than the current approach with linear time complexity O(n).

Keywords–EE-tree, spatial data, ensemble method, stream data classification

I. INTRODUCTION

Classification is an important technique in data mining. Dynamic data stream classification is also an important technique in dynamic data stream mining. Stream data mining has been popularly used in real-time intrusion detection, spam filtering, and malicious website monitoring [1]. In the applications, data arrive continuously in a stream fashion, timely predictions in identifying malicious records are of essential importance [1]. Dynamic data stream mining is used in many real life and time critical application such as intrusion detection, spam filtering and web traffic stream monitoring. In such applications data arrives continuously in a stream fashion. Predicting the class label of desired stream record or identifying out-linear stream records are very important dynamic data stream functions.

Dynamic data stream classification approach suffers with many problems such as continuously fast increasing of dynamic data stream volumes, concept drift (the changes in the distributions of stream data) features, patterns, properties in dynamic stream data. To overcome many of the existing problems in dynamic data stream classification, many ensemble methods such as average classifiers ensembles, incremental classifier ensembles, combined classifier and cluster ensembles, optimal ensembles, have been proposed.

To date, existing works on ensemble learning in data streams mainly focus on building accurate ensemble models. Prediction efficiency has not been concerned mainly because (1) prediction typically takes linear time, which is sufficient for general applications with undemanding prediction efficiency. (2) Existing works only consider combining a small number of base classifiers, e.g., no more than 30 [1].

Ensemble learning design technique is popular. Many state-of-the-art ensemble learning techniques in data stream classification consider only the construction of accurate ensemble learning models with least priority given to prediction efficiently. For many real world applications, linear time complexities of existing ensemble learning models are sufficient because there is no demand for prediction efficiency. Existing ensemble learning techniques generally combine only a limited (finite) number of base classifiers. But in many real life applications, a large number of ensemble base classifiers are needed in-order to capture many desired patterns of dynamic stream data. In other words, we need ensemble learning models with sub-linear (logarithmic) prediction time complexity. Generally, the linear relationship between prediction time and ensemble size is not at all suitable for many real world applications.

Sub-linear (logarithmic) time complexity of prediction is possible by considering only shared patterns and features among all base classifiers. Decision trees are reliable ensemble classifiers and each base classifier is composed of a batch of decision rules. Each decision rule is represented as a spatial object in the decision space. Entire ensemble model is represented as a spatial database. Popular indexing structures for spatial data are R-tree, R*-tree, R+-tree and M-tree. These type of trees have indexing structures with richer information - classifier weights, probability distributions and class labels can also be used as ensemble models are mainly used for fast prediction of incoming dynamic stream data records. In ensemble models a decision region is associated with different class labels.

Dynamic webpage stream monitoring through online is one best example for ensemble learning techniques that use divide and conquer technique to manage huge volumes of dynamic stream data with concept drifting. Ensemble learning models are scalable, low error rate, parallelizable, quickly adoptable to new features. Ensemble technique divides the stream data into partitions and builds one or more base classifiers for each partition and these base classifiers are combined for predicting the unknown stream record. The working principle of all the ensemble learning models is stated as the principle design that uses – divide and conquer techniques to handle large volumes of stream data with concept drift or concept change.

Paper is organized as follows. In section II actual problem of ensemble stream data classification is defined. In section III ensemble E-tree structure is given. In section IV algorithms are given. In section V stream data classification methods are given. In section VI conclusions are given.

II. PROBLEM DEFINITION

Dynamic stream records are represented as $S=\{(x_i, y_i)\}$, where x_i represents a set of k attributes and y_i is a class label with two values-yes or no. For simplicity a two class problem is considered. Stream data is divided into n partitions and n decision tree base classifiers c_1, c_2, c_3…….c_n are constructed all n base classifiers are combined into a single ensemble classifier E. Each base classifier c_i consists of x decision rules represented by if-then clauses. Total number of decision rules are n*x = nx. All these nx decision rules are represented in spatial database as nx spatial objects. Main goal is to construct best ensemble model to predict incoming stream record accurately within logarithm (sub-linear) time complexity O(log n). To achieve this goal each base decision tree classifier C_i is converted into a batch of spatial objects (SOs). This batch of spatial objects is called spatial database, SD. Entire ensemble model E is converted to a spatial database (SD) containing all spatial objects. With this idea given original problem is changed to classifying each incoming dynamic stream record r by systematically searching over the spatial database (SD).

EXAMPLE 1

Generally maps, graphs, cities, universities, and places are represented in R-tree indexing structure. In this paper ensemble decision tree classifiers are represented as a spatial database. Each decision tree classifier is represented as a set of rules. Each decision rule is represented as a spatial object in the spatial database.

Assume S is a web monitoring dynamic stream with two classes and each stream record has two attributes A_1 and B_1. Values of attribute A_1 are a_1, a_2, a_3…a_m and values of attribute B_1 are b_1, b_2, b_3…b_m where $0 <= a_i <= 6$ and $0 <= b_i <= 6$. For simplicity only latest six classifiers c_1, c_2, c_3, c_4, c_5 and c_6 are considered. An ensemble E is created based on these six classifiers. Generally each base classifier represents a set of if-then rules. Again for simplicity each base classifier is represented using only one if-then condition. Ensemble E is used to predict the class label of an incoming dynamic stream record r. More generally a classifier is represented by anding/oring many if-then expressions.

An ensemble model E is converted into a spatial database model (Spatial $_{\text{database}}$) and the mapping procedure of converting ensemble model into $\text{Spatial}_{\text{database}}$ model is shown in the fig.3. Entire spatial database is represented by a two-dimensional rectangle A = (0, 0, 6, 6). Each base classifier c_i is represented by a small black rectangle. The $(i+1)^{th}$ black rectangle drifts to the right side of the i^{th} rectangle from bottom to top of the decision are of spatial database because of concept drifting. The complete ensemble classification model is represented by using a set of spatial objects, black rectangles. Collection of spatial objects (black rectangles) represents the spatial database (Spatial $_{\text{database}}$). Ensemble stream model is such that for a given small circle x = (5, 5) as an incoming record, we would like to classify x as both accurate and fast in logarithmic time complexity, O(log n).

III. ENSEMBLE TREE INDEXING STRUCTURE

Ensemble-tree (E-tree) data structure is a multi-way height balanced indexing R-tree like structure. It is an extended R-tree structure. Most important operations of E-tree are insertion, search, deletion and all these operations are performed similar to R-tree operations.

A. Structure of E-Trees

E-tree contains two main components:

1. R-tree like indexing structure that stores all decision rules of the ensemble.
2. A two-dimensional table structure that stores base classifier details such as IDs and weights of base classifiers.

Two components of E-tree are directly linked to each other such that base classifier in the table is linked to its corresponding decision rules in the R-tree. E-tree is height balanced multi-way indexing tree structure. E-tree stores all decision rules of the ensemble. An ensemble is a collection of rules from a group of classifiers. Each classifier is represented as a set of decision rules.

E-tree creates and maintains two different types of nodes.
1. Pivot node 2. Leaf node

Each leaf node stores a set of decision rules represented by a heavily overlapping area in the spatial decision space. All decision rules are converted into spatial objects and each spatial object is represented in the form (Rectangle bounded by a decision rule, classifier-id, and memory address of next decision rule). Classifier-id represents the identification of the particular classifier that has been generated the decision rule.

(minimum-rectangle, classifier-id, sibling)

The entries of the pivot node in the E-tree are represented as

(minimum-rectangle, child-pointer)

Where minimum-rectangle is the smallest rectangle that covers all decision rules in its child nodes and child-pointer is a pointer (reference) pointing to its child nodes. A non-root node should contain entries between m and M as in the case of B+-tree, where M is the maximum number of entries of the non-root node and m is the minimum number of entries of the non-root node. The root should contain at least two entries.

For simplicity we have taken only two dimensional spaces. In general, in d-dimensional spatial space a decision rule covers a closed space $\Theta = (\Theta_0, \Theta_1, \Theta_2, \ldots\ldots\Theta_n)$ where each Θ_i represents a closed bounded interval along ith dimension. The TABLE structure is called E-table that stores all the base classifier details of the selected ensemble, E. Each entry of the E-table is represented as
(Classifier-identification, Classifier-weight, Pointer)

Where classifier identification represents identification of a particular classifier in the ensemble, E, classifier-weight represents the weight of the classifier, and pointer represents or stores address or reference of the first component decision rule of the classifier.

Here the following TABLE 1 stores six classifier rules. R-trees are popularly used as indexing structure to manage conventional spatial objects such as graphics objects, vehicle objects, maps, images, location objects, spatial objects, multimedia objects, mechanical objects, stars, satellites etc. E-tree indexing structures are particularly used to manage a new and different type of spatial data objects, decision rules belonging to the ensemble, E

ID	Decision rules
C1	If(0<=r1<=1)and (0<=r2<=1) then correct ; otherwise incorrect
C2	If(0.5<=r1<=2)and (0.5<=r2<=2) then correct ; otherwise incorrect
C3	If(1.5<=r1<=3)and (1.5<=r2<=3) then correct ; otherwise incorrect
C4	If(3<=r1<=4.5)and (3<=r2<=4.5) then correct ; otherwise incorrect
C5	If(4<=r1<=5.5)and (4<=r2<=5.5) then correct ; otherwise incorrect
C6	If(5<=r1<=6) and (5<=r2<=6) then correct ; otherwise incorrect

TABLE 1. Six base classifiers

Decision rule is converted to Rectangle
Base classifier is converted to spatial objects
Ensemble is converted to spatial database

TABLE 2 Mapping an ensemble model to a spatial database

1. E-tree stores decision rules that are constructed in a continuous attribute space and discreet attributes are converted into continuous attributes by applying suitable changes.

2. Each decision rule represented in the E-tree covers a closed spatial space. All partially closed spatial spaces must be converted into fully closed spatial spaces by changing lower or upper bounds appropriately. For example, consider the decision rule of the classifier C_1. This decision rule, (r_1<=1.5) and (r_2<=1.5), is half closed and it is converted into fully closed spatial space as (0<r_1<=1.5) and (0<=r_2<=1.5). In general, if a decision rule covers only a few dimensions (i) then it will be converted into remaining (d-i) dimensions also. Suppose, assume a decision rule that is defined in a partially closed spatial space R = (0<=r_1<=1.2) then R must be changed to a closed space as

R = (0<=r_1<=1.2) and (0<=r_2<=6)

1) All decision rules of the ensemble E are taken as “hard” decision rules only but not fuzzy or other rules. For example, an image is correctly represented or not. Another example is, an image is correctly represented for the country or not.

2) For simplicity purpose, only binary classification model is considered same procedure can be used to extend this E-tree indexing technique for representing multiclass problems in spatial space. In binary classification a simple technique of representing E-tree indexing decision rules is that stores only the decision rule of a minor class containing small set of decision rules.

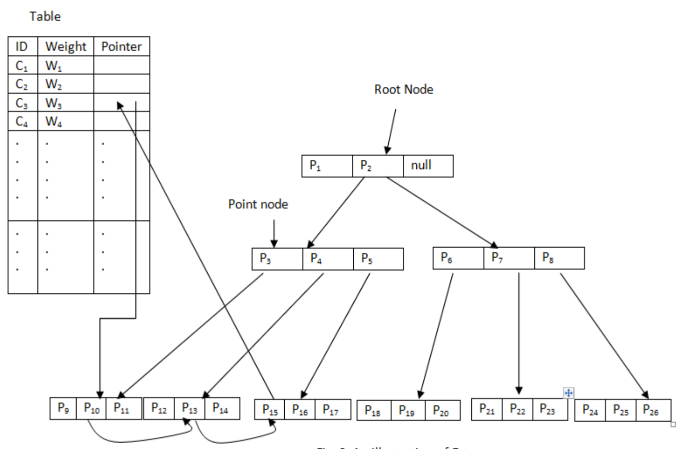

Fig. 2, An illustration of E-tree

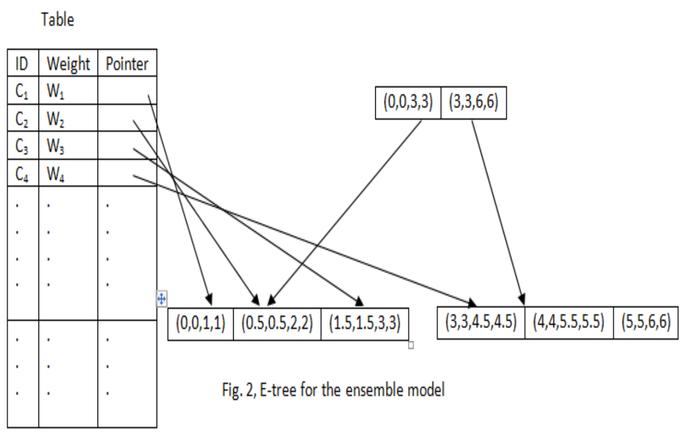

Fig. 2, E-tree for the ensemble model

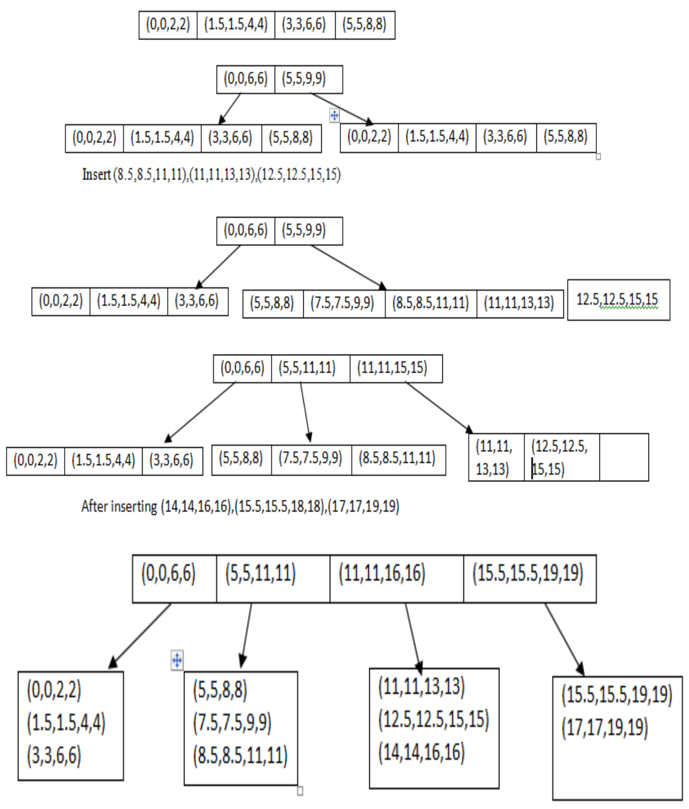

Fig. 3 Inserting decision rules into the E-Tree

IV. ALGORITHMS

A. E-Tree Insertion Algorithm

Algorithm 1: Inserting a node into E-tree

Input :
1. E- tree T
2. Classifier C (one or more rules)
3. m , minimum number of entries in a node
4. M, maximum number of entries in a node

Output : updated or modified E-tree T^1

```
1.  P←T .tree.root        // get root of E-tree
2.  Foreach decision rule R∈C do
3.      L←searchLeaf(R, P)
4.      If( L.size < m )  then
5.          $L \leftarrow L \cup R$
6.          $T' \leftarrow$ UpdateParentnode (L)
7.      Else
8.          <$P_L$,$P_R$> ← splitNode(L, R)
9.          $T' \leftarrow$ adjustTree(T, $P_L$, $P_R$)
10.     Endif
11. Endfor
12. Insert the classifier, C into table structure
13. Output the tree, $T'$, after inserting the new classifier,
    C
```

Explanation of E-Tree insertion operation

Whenever new trends and patterns are found during dynamic stream data classification, the E-tree insertion algorithm insert new base classifiers into the ensemble model and this model is automatically converted into the spatial space data object and inserted into the E-tree indexing structure. Whenever a new classifier, C, is created, a new entry associated with the newly created classifier is inserted into the E-table structure and similarly all the decision rules, R, belonging to the newly created classifier are inserted into the E-tree structure one after the other, and linked all the rules together by the respective pointer entries.

Inserting decision rules of a new classifier into E-tree is very similar to the insertion operation of R-tree indexing structure starting from the root search proceeds downwards in order to find a leaf node that covers the rules of the new classifier. Once a leaf node is found, it is checked to find whether space for insertion is available or not. If the leaf node contains less than the maximum number of entries, M, then new set of rules are inserted into the leaf node. Parent node is updated appropriately. After searching, if it is found that there is no space for new insertion, then the current leaf node is split into two nodes and then new set of rules are inserted. Node splitting in E-tree is a difficult step.

Search operation of E-tree

A search operation is initiated to find the class label of a newly incoming stream record, x. Search algorithm of E-tree first starts from the root to downwards of the E-tree and finds all the decision rules in the leaf node or leaf nodes covering the newly arrived dynamic stream record x. Class label of the newly arrived stream record is calculated by combining all the decision rules. After the completion of the search operation class label of the newly arrived dynamic stream record is calculated using the following formula

$$y_x = sgn(\sum_{i=1}^{u} w_i \, P(C_{index} \mid x), \gamma),\text{------------ (4)}$$

where y_x is the class label of newly arrived dynamic stream record, u is the total number of retrieved decision rules covering new record x, sgn(temp, γ) is a threshold function that decides class label of the new record x, by comparing temp and γ, and C_{index} is the indexing class (minor class) in the E-tree. E-trees are particularly useful when the minor class that has fewer decision rules is indexed in E-trees. In the given example1, C_{index} represents the target class (abnormal class). $P(C_{index}/x)$ is a "hard" posterior probability of either 0 or 1. Hard means the value is taken as either 100% correct or 100% incorrect.

The search algorithm of E-tree performs a depth-first search (DFS) technique for searching. In-order to find class label of a newly arriving dynamic stream record x, the E-tree search algorithm first traverses along the E-tree branches whose rectangles cover x and then class label of x computed by using the equation 4. Classifier-id and Wait of each classifier is given in the table. Search algorithm finds class label of the incoming stream record by using a technique known as overlapped patterns or shared patterns. More number of ensemble classifiers is required in order to reflect all the patterns, features, modifications, and other dynamic changes in the stream data classification. All the algorithms must be fast enough to maintain and update all the details of ensemble management.

B. E-Tree Search Algorithm

Algorithm 2: E_Tree Search

Input:
1. E-tree T
2. Stream record, x
3. Parameter γ

Output: Class label of x (Here y_x is class label of x)

```
1.  Initialize(stack)   // initialize a stack U←Φ
                        // U stores all records covering
x
2.  P←T.tree.root       // obtain the root of tree, T
3.  Foreach entry R∈ T do
4.        Push(stack, R)
5.  Endfor
6.  While (stack ≠ Φ) do
7.        e ←pop(stack)
8.        if (e is an entry of a leaf ) then
9.              $U \leftarrow U \cup e$
10.       else
11.             P ← e.child
12.             foreach entry e ∈ P do
13.                   If ( x∈ e) then
14.                         push (stack, e)
15.                   endif
16.             endfor
17.       endif
18. endwhile
19. foreach entry e ∈ U do
20.    Find its weights in the table structure
21. Endfor
22. $y_x = sgn(\sum_{i=1}^{u} w_i \, P(C_{index} \mid x), \gamma)$
23. Output $y_x$
```

C. E-Tree Deletion Algorithm

E-Tree deletion operation

We assume that there is a pre-specified maximum capacity of classifiers in the E-tree. Deletion operation in E-tree deletes all the outdated and un-useful classifiers after the maximum capacity of E-tree classifiers is reached. Assume that current maximum capacity ensembles in the E-tree is 20 and c_1, c_2, c_3.....c_{20} are in E-tree. Whenever maximum capacity of E-tree is reached after the insertion of only new classifier, the old and un-useful classifiers are deleted automatically by E-tree deletion algorithm. That is when C_{21} enters, and then C_1 will be deleted.

The important deletion methods in E-tree are:
1) E-tree deletion, which is similar to the deletion in B-tree.
2) E-tree deletion which is similar to the deletion in R-tree

In the first method under-full nodes are merged into one of the siblings such that result will in the least area increase. In the second delete-then-insert operation is applied. Under-full node is deleted first and then remaining entries are inserted into the E-tree using the E-tree insertion operation. The second method is very easy to implement and it is also very advantageous. Also it is very to refine the existing spatial structure of the E-tree after re-insertion. First E-tree deletion method which is similar to B-tree is inefficient and it results many node splits. Second method is the best method for deleting a node in E-tree.

Algorithm 3 E-Tree Deletion

```
Input:
    1. E-tree, T
    2. Classifier, C
    3. Parameters, m, M

Output:
    updated E–tree $T'$

    1.  P ← T.tree.root     // obtain the root of the tree
    2.  A← T.Table.ref  // get the reference from the table
    3.  R← SearchClassifier(C,A)
    4.  P← R.pointer
    5.  While (P ≠ Φ)
    6.      L ← P.node()
    7.      Q ← P.sibling
    8.      $L'$ ← deleteEntry (L, P)
    9.      If ($L'$.size < m) then
    10.         $T'$← deleteNode(T, $L'$)
    11.         foreach entry e ∈ $L'$ do
    12.             $T'$ ← insertRule(e,T)
    13.         endfor
    14.     endif
    15.     P←q
    16. endwhile
    17. Output $T'$
```

D. Proposed Algorithm 4 for E-Tree Deletion

A new method is proposed in the E-Tree deletion operation to delete a selected classifier and its associated decision rules from the ensemble .We use a specific procedure. In the normal E-tree a classifier is deleted by first in first out principle. At any time E-tree ensemble size is constant. Whenever there is need to delete a classifier from the ensemble we will delete the oldest classifier first.

We propose a new method for deleting a classifier from the E-Tree ensemble. Fist we will calculate accuracy of each classifier, C_i, in the E-Tree ensemble on the incoming stream test data, and then delete a classifier whose accuracy is smallest. In other words, test each classifier on the test stream data and then discard the classifier whose misclassification error is maximum. Also there are other methods for deleting a classifier from the ensemble. Some examples are – false positive rate, false negative rate, mean, moving average, weighted moving average, some sampling techniques, and other methods that can detect concept drift or concept change.

Algorithm 4 Proposed Algorithm for E-Tree Deletion

```
Input:
    1. A set of base classifiers constructed and represented in
       the E-Tree.
    2.test stream data

Output:
1. a set of base classifiers after deleting the least important base
   classifier.
2. get test data from the incoming input stream data.
3. classifier_id ← 1.
3. newaccurancy ← 0.
4. foreach base classifier, from the ensemble,  E do
5.      $accurancy_i$← compute accuracy of classifier $C_i$ using
          selected test stream data.
6.      if($accurancy_i$ > newaccurancy) then
7.          newaccurancy← $accurancy_i$
8.          classifier-id ← i
9.      endif.
10. end for.
11. E_Tree_deletion( E-Tree T, classifier  $C_i$ )
```

A decision classifier C4.5 is used to generate decision rules from the dynamic stream data. All decision rules are “hard” decision rules only. That is decision rules are not fuzzy. Various measures used for online query evaluation are:

1.Time cost- computational cost of E-trees is much lower than the computational cost of traditional ensemble models because E-trees are height balanced indexing tree structures that are used to index all classifiers in the ensemble.

2.Memory cost- Through E-trees consumes larger memory during stream data record classification, the memory size is in affordable range only.

3.Accuracy- prediction accuracy of E-trees is high and it is equal to the accuracy of original ensemble models.

E-Tree ensemble learning

Architecture of ensemble models on dynamic data streams using E-tree indexing structure is shown in the fig 4. Training module maintains and controls all the insertion and deletion operations of E-trees. Training module is responsible to monitor E-tree operations for constant updating of E-tree. Prediction module is responsible to predict the class label of newly arriving dynamic stream record x by using synchronized copy of E-tree received from training module. E-tree search operation is applied to make online predictions. We assume that stream is coming with un-labeled data. Initially the buffer in the training module is filled with incoming stream records. Records in the buffer are labeled by human experts or intelligent labeling machines. This labeling process is very slow, costly and time consuming and only a small percentage of incoming records at regular intervals regularly in order to provide uniform labeling. As soon as buffer is full, automatically a new classifier is constructed and then it is inserted in the ensemble. New classifiers of E-tree are constructed regularly. Once the maximum capacity of the E-tree is reached automatically old or outdated or un-useful classifiers are deleted from the E-tree by executing E-tree deletion operation. Updated and latest E-tree will be synchronized and E-tree copy is passed to the prediction module.

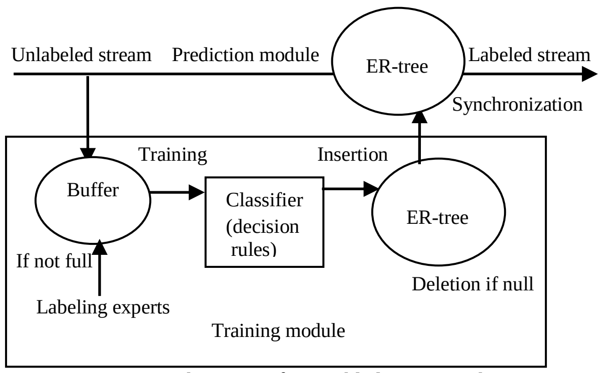

Fig 4. Architecture of ensemble learning with ER-tree

V. STREAM DATA CLASSIFICATION METHODS

Dynamic data stream classification methods are roughly divided into two categories (groups):

1. Online or incremental modes and
2. Ensemble learning modes

A. Online incremental models

Main goal of online incremental models is to build a single robust, efficient, effective and sophisticated model that can be continuously updated in order to reflect the latest changes in trends and patterns of dynamic stream data. Very fast decision tree(VFDT) model and the incremental SVM model are best examples for online incremental models.

B. Ensemble learning models

Ensemble learning model is average or weighted or group based approach that uses a divide-and-conquer strategy. Decision tree uses non-parametric approach and ensemble model pre-dominantly uses decision tree classifiers. Ensemble learning model in the first step splits continuous dynamic stream data into small groups of tuples and then constructs light-weight base classifiers (decision trees) for these small groups. Ensemble E is constructed by combining all these light weight base classifiers in a systematic way. Ensemble learning models scale well to very large volumes of dynamic stream data, very easy to parallelized this model, it adapts very quickly to new and dynamic changes, patterns, graphics, and it is susceptible to lower variance errors, and very useful for spatial data maintenance. Ensemble stream data model also reasonably good in managing concept drift features.

VI. CONCLUSIONS

In this paper we proposed a new algorithm for E-Tree deletion. Previously time complexity of n base classifiers in the ensemble of stream data mining is O(n). Multi-way E-Trees are developed to decrease the time complexity from linear, O(n), to sub-linear or logarithmic, O(log n). When classifiers are deleted in first in first out fashion, some of the classifiers with reasonably good accuracy might have deleted. Hence, we propose a new deletion algorithm in ensemble tree (E-tree deletion) deletion. New E-tree deletion algorithm deletes only the classifier which gives less accurate results on the new test data selected randomly from the dynamic stream data.

REFERENCES

[1] Peng Zhang, Chuan Zhou, Peng Wang, Byron J. Gao, Xingquan Zhu, and Li Guo “IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 27, NO. 2, FEBRUARY 2015”
[2] J. Han, M. Kamber, and J. Pei, Data Mining: Concepts and Techniques, third ed. Morgan Kaufmann, 2011.
[3] J. Quinlan, C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, 1993.