IJETST- Vol.||03||Issue||02||Pages 3559-3564||February||ISSN 2348-9480 2016

Open access Journal International Journal of Emerging Trends in Science and Technology

Impact Factor: 2.838 **DOI:** http://dx.doi.org/10.18535/ijetst/v3i02.11

An Efficient Indexing Structure for Group Models On Data Streams

Authors

G. Sugantha, Dr Akila

Vels University

Abstract

Group learning is a common tool for data stream classification, mainly because of its inherent advantages of handling huge volume of stream data and concept drifting. Have been mainly focused on building accurate group models from stream data. a linear scan of a huge number of base classifiers in the group during prediction incurs significant costs in response time, preventing group learning from being practical for many real world time critical data stream applications, such as web traffic monitoring spam detection, intrusion detection In these applications, data streams usually arrive at a speed of Giga byte per Seconds, and it is necessary to classify each stream record in a timely manner we propose a novel Ensemble tree indexing structure to organize all base classifiers in an grouped for fast prediction On Ensemble trees treat group as spatial databases and employee an Random tree like height balanced structure to minimize the expected prediction time of from linear to sub linear complexity. On the other hand, Ensemble trees can be automatically updated by continuously integrating new classifiers and other discarding outdated ones, well adapting to new trends and patterns underneath data streams.

Keywords: *Stream data mining, classification, ensemble learning, spatial indexing, and concept drifting.*

I. INTRODUCTION

Data Stream Mining is the process of extracting knowledge structures from continuous, rapid data records. A data stream is an ordered sequence of instances that in many applications of data stream mining can be read only once or a small number of times using limited computing and storage capabilities. Examples of data streams include computer network traffic, phone conversations, ATM transactions, web searches, and sensor data. Data stream mining can be considered a subfield of data mining, machine learning, and knowledge discovery.

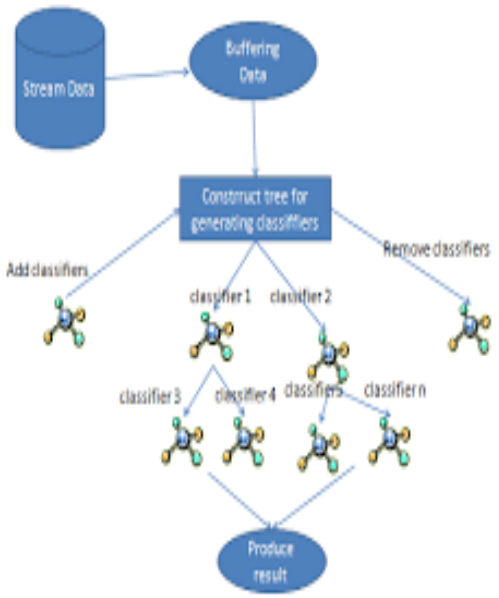
In many data stream mining applications, the goal is to predict the class or value of new instances in the data stream given some knowledge about the class membership or values of previous instances in the data stream. Machine learning techniques can be used to learn this prediction task from labeled examples in an automated fashion. Often, concepts from the field of incremental learning, a generalization of Incremental heuristic search are applied to cope with structural changes, on-line learning and real-time demands. In many applications, especially operating within non-stationary environments, the distribution underlying the instances or the rules underlying their labeling may change over time, i.e.The goal of the prediction, the class to be predicted or the target value to be predicted may change over time. This problem is referred to as concept drift.

Classification and clustering are examples of the more general problem of pattern recognition, which is the assignment of some sort of output value to a given input value. Other examples are regression, which assigns a real-valued output to each input; sequence labeling, which assigns a class to each member of a sequence of values (for example, part of speech tagging, which assigns a part of speech to each word in an input sentence); parsing, which

assigns a parse tree to an input sentence, describing the syntactic structure of the sentence; etc.

A common subclass of classification is probabilistic classification. Algorithms of this nature use statistical inference to find the best class for a given instance. Unlike other algorithms, which simply output a "best" class, probabilistic algorithms output a probability of the instance being a member of each of the possible classes. The best class is normally then selected as the one with the highest probability. However, such an algorithm has numerous advantages over non-probabilistic classifiers

In ensemble learning, an agent takes a number of learning algorithms and combines their output to make a prediction. The algorithms being combined are called base-level algorithms.The simplest case of ensemble learning is to train the base-level algorithms on random subsets of the data and either let these vote for the most popular classification (for definitive predictions) or average the predictions of the base-level algorithm. For example, one could train a number of decision trees, each on random samples of, say, 50% of the training data, and then either vote for the most popular classification or average the numerical predictions. The outputs of the decision trees could even be inputs to a linear classifier, and the weights of this classifier could be learned.



1.1 An Efficient Indexing Structure for group Models on Data Streams

In bagging, if there are *m* training examples, the base-level algorithms are trained on sets of *m* randomly drawn, with replacement, sets of the training examples. In each of these sets, some examples are not chosen, and some are duplicated. On average, each set contains about 63% of the original examples.

In boosting there is a sequence of classifiers in which each classifier uses a weighted set of examples. Those examples that the previous classifiers misclassified are weighted more. Weighting examples can either be incorporated into the base-level algorithms or can affect which examples are chosen as training examples for the future classifiers.

Another way to create base-level classifiers is to manipulate the input features. Different base-level classifiers can be trained on different features. Often the sets of features are hand-tuned.

Another way to get diverse base-level classifiers is to randomize the algorithm. For example, neural network algorithms that start at different parameter settings may find different local minima, which make different predictions. These different networks can be combined.

2. OBJECTIVE OF THE STUDY

We first define the key data structures and notations used in this paper. Geographical information of User. Each user u has a register location generated in various EBSNs or LBSNs.

User Activity. A user activity is a triple (u, v, r) that user u rates item v with score R. User activity historical data is given by $S \subseteq U \times V \times R$, where user activities are positive observations in the past. User Group. A user group can be a set of users by considering users' geographical information. Item group. Sets of item with similar preference from users. In our analysis, rating scores are strongly correlated with location-based user group preferences, where z l and zg represent two difference topic models. This enables rating r to be

mutually influenced during the both topic discovery processed.

3 LITRATURE REVIEW

3.1 PROBLEM DEFINITION

A two class data stream S consisting of an infinite number of records fxi, yi pg, where xi 2 Rd is a d - dimensional attribute vector, and yet 2 formal, abnormal is the class label, which is unobservable unless the sample is properly labeled. Suppose that we have built on base classifiers C1, C2 . . . ; Cn from historical stream data using a decision tree algorithm (such as C4.5). All the n base classifies are combined together as an ensemble classifier E. Each base classifier Ci 1 i nÞ is comprised of l decision rules

J prep resented by conjunction literals (i.e., rules are expressed as "if . . . then . . ."). Then, there are N 1⁄4 n Al decision rules in the ensemble E. The aim of this paper is to generate accurate prediction for an incoming stream record x, using the ensemble model E, with sub-linear time complexity O (log (N)).

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 However, there are increasingly more real world applications where stream data arrive intensively in large volumes. In addition, the hidden patterns underneath data streams may change continuously, which requires a large number of base classifiers to capture various patterns and form a quality ensemble. Such applications call for fast sub linear prediction solutions.

3.2 PROPOSED TECHNIQUE

Structure that organizes base classifiers in a height-balanced tree structure to achieve logarithmic time complexity for prediction. Technically, an E-tree has three key operations

1. Search: traverse an E-tree to classify an incoming stream record x;
2. Insertion: Integrate new classifiers into an E-tree;
3. Deletion: Remove outdates classifiers from a Tree.

As a result, the E-tree approach not only guarantees a logarithmic time complexity for prediction, but is also able to adapt to new trends and patterns in stream data. The rest of the paper is structured as follows. Section introduces the ensemble indexing problem. Section describes the main structure and key operations of E-trees.

4. ALGORITHEM USED

4.1 SEARCH ALGORITHM

Search Algorithm is an algorithm for finding an item with specified properties among a collection of items which are coded into a computer program, that look for clues to return what is wanted. The items may be stored individually as records in a database; or may be elements of a search space defined by a mathematical formula or procedure, such as the roots of an equation with integer variables; or a combination of the two, such as the Hamiltonian circuits of a graph.

Virtual Search Spaces

Algorithms for searching virtual spaces are used in constraint satisfaction problem, where the goal is to find a set of value assignments to certain variables that will satisfy specific mathematical equations and in equations. They are also used when the goal is to find a variable assignment that will maximize or minimize a certain function of those variables. Algorithms for these problems include the basic brute-force search (also called "naïve" or "uninformed" search), and a variety of heuristics that try to exploit partial knowledge about structure of the space, such as linear relaxation, 'constraint generation, and constraint propagation.

An important subclass are the local search methods, that view the elements of the search space as the vertices of a graph, with edges defined by a set of heuristics applicable to the case; and scan the space

by moving from item to item along the edges, for example according to the steepest descent or best-first criterion, or in a stochastic search. This category includes a great variety of general met heuristic methods, such as simulated annealing, tab search, A-teams, and genetic programming that combine arbitrary heuristics in specific ways.

Algorithm 1: Search

Input : tree T, stream record x, parameter γ
Output: x's class label y_x

```
Initialize(stack) ; // initialize a stack U ← ∅ ; // U
records all rules covering x P ← T.tree.root ;
// get the root of the tree
foreach entry R ∈ T do
    push(stack, R) ;
while stack ≠ ∅ do
    e ← pop(stack) ;
    if e is an entry of a leaf then
        U ← U ∪ e ;
    else
        P ← e.child ;
        foreach entry e ∈ P do
            if x ∈ e then
                push (stack, e) ;
foreach entry e ∈ U do
    find its weights in the table structure ;
y_x ← Call Eq. (4) to calculate x's class label ;
Output y_x ;
```

This class also includes various tree search algorithms that view the elements as vertices of a tree, and traverse that tree in some special order. Examples of the latter include the exhaustive methods such as depth-first search and breadth-first search, as well as various heuristic-based search trees pruning methods such as backtracking and branch and bound. Unlike general met heuristics, which at best work only in a probabilistic sense, many of these tree-search methods are guaranteed to find the exact or optimal solution, if given enough time. This is called "completeness".

Another important sub-class consists of algorithms for exploring the game tree of multiple-player games, such as chess or backgammon, whose nodes consist of all possible game situations that could result from the current situation. The goal in these problems is to find the move that provides the best chance of a win, taking into account all possible moves of the opponent(s). Similar problems occur when humans or machines have to make successive decisions whose outcomes are not entirely under one's control, such as in robot guidance or in marketing, financial, or military strategy planning. This kind of problem — combinatorial search — has been extensively studied in the context of artificial intelligence. Examples of algorithms for this class are the mini max algorithm, alpha–beta pruning, and the A* algorithm.

Sub-Structures of a Given Structure

The name combinatorial search is generally used for algorithms that look for a specific sub-structure of a given discrete structure, such as a graph, a string, a finite group, and so on. The term combinatorial optimization is typically used when the goal is to find a sub-structure with a maximum (or minimum) value of some parameter. (Since the sub-structure is usually represented in the computer by a set of integer variables with constraints, these problems can be viewed as special cases of constraint satisfaction or discrete optimization; but they are usually formulated and solved in a more abstract setting where the internal representation is not explicitly mentioned.)

An important and extensively studied subclass are the graph algorithms, in particular graph traversal algorithms, for finding specific sub-structures in a given graph — such as sub graphs, paths, circuits, and so on. Examples include Dijkstra's algorithm, Kruskal's algorithm, the nearest neighbour algorithm, and Prim's algorithm.

Another important subclass of this category is the string searching algorithms that search for patterns within strings. Two famous examples are the Boyer–Moore and Knuth–Morris–Pratt algorithms,

and several algorithms based on the suffix tree data structure.

Search for the Maximum of a Function

In 1953, American statistician Jack Kiefer devised Fibonacci search which can be used to find the maximum of a unit modal function and has many other applications in computer science.

5. THEORETICAL ANALYSIS

TREE TRAVERSAL ANALAYSIS:

Depth-first search is Trees can be traversed in pre-order, in-order, or post-order. These searches are referred to as depth-first search (DFS), as the search tree is deepened as much as possible on each child before going to the next sibling. For a binary tree, they are defined as display operations recursively at each node, starting with the root.

Breadth-first search Trees can also be traversed in level-order, where we visit every node on a level before going to a lower level. This search is referred to as breadth-first search (BFS), as the search tree is broadened as much as possible on each depth before going to the next depth.

6. RELATED WORK

Stream classification. Existing data stream classification models can be roughly categorized into two groups: online/ incremental models and ensemble learning [5], [6], [7], [8], [9], [10], [11]. The former aims to build a single sophisticated model that can be continuously updated. Examples include the Very Fast Decision Tree (VFDT) model and the incremental SVM model.

Ensemble pruning. For the purpose of reducing computational and memory costs, ensemble pruning [15], [16] searches for a good subset of all ensemble members that perform as good as the original ensemble.

Classifier indexing Grouping images into (semantically) meaningful categories using low-level visual features is a challenging and important problem in content-based image retrieval [17] [18] [10] [11]. Using binary Bayesian classifiers, we attempt to capture high-level concepts from low-level image features under the constraint that the test image does belong to one of the classes. Specifically, we consider the hierarchical classification of vacation images; at the highest level, images are classified as indoor or outdoor; outdoor images are further classified as city or landscape; finally, a subset of landscape images is classified into sunset, forest, and mountain classes.

7. CONCLUSION

Security E-tree indexing structure for sub linear time complexity for classifying high speed stream records. The main contributions of this study are threefold: (1) we formulate and address the prediction efficiency problem for ensemble models on data streams, which is a legitimate research problem well motivated by increasing real-time applications. (2) Our solution converts ensemble models into spatial databases and applies spatial indexing techniques to achieve sub-linear prediction. This novel technique can be extended to other data stream classification models besides ensemble learning, or general classification models that require timely prediction. (3) The proposed E-tree evaluation method can be extended to spatial/temporal data analysis where data nodes have arbitrary extents.

REFERENCES

1. C. Agawam, Data Streams: Models and Algorithms. Springer, 2006.
2. P. Zhang, J. Li, P. Wang, B. Gao, X. Zhu, and L. Guo, “Enabling Fast Prediction for Ensemble Models on Data Streams,” Proc. 17th ACM SIGKDD Int’l Conf. Knowledge Discovery and Data Mining (KDD), 2011.
3. M. Masud, J. Gao, L. Khan, J. Han, and B. Thuraisingham, “Classification and Novel Class Detection in Concept-Drifting Data Streams under Time Constraints,” IEEE Trans. Knowledge and Data Eng., vol. 23, no. 6, pp. 859-874, June 2011.
4. J. Gao, R. Sebastiao, and P. Rodrigues, “Issues in Evaluation ofStream Learning Algorithms,” Proc. 15th ACM SIGKDD

Int'l Conf.Knowledge Discovery and Data Mining (KDD), 2009.

5. 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.
6. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavalda, "New Ensemble Methods for Evolving Data Streams," Proc. 15th CM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD), 2009.
7. X. Zhu, P. Zhang, X. Lin, and Y. Shi, "Active Learning from Stream Data Using Optimal Weight Classifier Ensemble," IEEE Trans. System, Man, Cybernetics, Part B: Cybernetics, vol. 40, no. 4, pp. 1-15, Dec. 2010.
8. T. Sellis, N. Roussopoulos, and C. Faloutsos, "The Rþ-tree: ADynamic Index for Multi-Dimensional Objects," Proc. 13th Int'lConf. Very Large Data Bases (VLDB), 1987.
9. D. Balzarotti, M. Monga, and S. Sicari, "Assessing the Risk of Using Vulnerable Components," Proc. ACM Second Workshop Quality of Protection (QoP '05), pp. 65-78, 2005.
10. P. Ciaccia, M. Patella, and P. Zezula, "M-Tree: An Efficient AccessMethod for Similarity Search in Metric Spaces," Proc. 23rd Int'lConf. Very Large Data Bases (VLDB), 1997.
11. M. McQueen, T. McQueen, W. Boyer, and M. Chaffin, "Empirical Estimates and Observations of 0Day Vulnerabilities," Proc. Hawaii Int'l Conf. System Sciences, pp. 1-12, 2009.
12. Y. Zhang, S. Burer, and W. Street, "Ensemble Pruning via Semi-Definite Programming," J. Machine Learning Research, vol. 7, no. 2006, pp. 1315-1338, 2006.
13. Y. Theodoridis, E. Stefanakis, and T. Sellis, "Efficient Cost Models for Spatial Queries Using R-Trees," IEEE Trans. Knowledge and Data Eng., vol. 12, no. 1, pp. 19-32, Jan./Feb. 2000.
14. S. Babu, R. Motwani, K. Munagala, I. Nishizawa, and J. Widom, "Adaptive Ordering of Pipelined Stream Filters," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD), 2004..
15. Machanavajjhala, E. Vee, M. Garofalakis, and J. Shanmugasundaram, "Scalable Ranked Publish/Subscribe," Proc. VLDB Endowment, vol. 1, pp. 451-462, 2008.
16. E. Ikonomovska, J. Gama, B. Zenko, and S. Dzeroski, "Speeding- Up Hoeffding-Based Regression Trees with Options," Proc. 28thInt'l Conf. Machine Learning (ICML), 2011.
17. H. Yu, I. Ko, Y. Kim, S. Hwang, and W. Han, "Exact Indexing for Support Vector Machines," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD), 2011.
18. N. Panda and E. Chang, "Exploiting Geometry for Support Vector Machine Indexing," Proc. SIAMInt'l Conf. DataMining (SDM), 2005.
19. C. Chang and C. Lin, "LIBSVM: A Library for Support Vector Machines," http://www.csie.ntu.edu.tw/cjlin/libsvm, 2001.
20. Asuncion and D. Newman, "UCI Machine Learning Repository," http://mlearn.ics.uci.edu/databases/, 2007.