IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 16, Issue 5, Ver. VIII (Sep – Oct. 2014), PP 61-64
www.iosrjournals.org

Enhanced Decision Tree Algorithm for Data Streams using adaptation of Concept Drift

Bhavkaran Singh Walia[1], Navan Preet Singh[2]
[1](Department of Computer Science, Guru Tegh Bahadur Institute of Technology – Guru Gobind Singh Indraprastha University, India)
[2](Department of Computer Science, Guru Tegh Bahadur Institute of Technology – Guru Gobind Singh Indraprastha University, India)

Abstract: *Construction of a decision tree is a well researched problem in data mining. Mining of streaming data is a very useful and necessary application. Algorithms such as VFDT and CVFDT are used for decision tree construction, but as a lot of new examples are added, a new optimal model needs to be constructed. Here in this paper, we have provided an algorithm for decision tree construction which uses discriminant analysis, to select the cut point used for splitting tests, thus optimizing time complexity from O(nlogn) to O(n). We have also analyzed several learning strategies such as dynamic ensemble, contextual, forgetting and detection approaches. We have also discussed handling of concept drift which occurs due to gradual change in the data set using the naive Bayes classifer at each of the inner node.*

Keywords*: Bayes classifier, Adaptive learning strategies, Concept drift, Decision Tree, Data Streams, VFDT, Discriminant analysis.*

I. Introduction

Many applications involve the generation and analysis of a new kind of data called stream data. Where data flow in and out of an observation platform (or window) dynamically. Such data streams have following unique features: huge or possibly infinite volume, dynamically changing, flowing in and out in a fixed order, allowing only one or a small number of scans, and demanding fast response time. In real time applications the data flows at a high speed continuously. The cardinal issue in data mining on data streams is that only one pass is permitted over entire data. Also, there is another real-time constraint, the processing time is confined by the arrival rate of instances in the streaming of data, and memory available to save any summarised data may be limited.

In this paper we present a structure of an algorithm that will generate a decision tree for data streams. We also discuss several adaptive techniques to detect concept drift. The time complexity of the proposed algorithm is reduced by the use of discriminant analysis – choosing the segregation point for splitting function, using Bayes classifier at every node and to determine concept-drift and thus limiting the error rate.

II. Related Work

In the context of machine learning, many techniques have been presented to cater to time changing concepts. We are analyzing the related work with two dimensions. One is enhancing the splitting criterion function of the decision tree algorithm and the other is concept-drift. The reasons for a concept drift are the several types of changes visible in distribution of data. Patterns cannot be identified in case of a random drift, however if a drift is systematic in nature, identification of pattern is possible. Hence, an adaptive technique needs labelled, current data.

Several adaptive techniques can be applied, depending on the kind of drift. If labelled, current data is available or drift is gradual, incremental learning techniques can be applied. However if labelled, current data is not available adaptive techniques are still applicable. Such techniques need a specific underlying drift model to be assumed.

Domingos, Spencer and Hulten have addressed the issue of construction of decision tree on streaming data [8]

Their algorithm gives a probabilistic limit on accuracy of the constructed decision tree. In this paper, we review the issue of the construction of a decision tree on streaming data using discriminant analysis. Hence, a machine learning algorithm used for streaming data serves as a useful tool for creating scalable implementations for big datasets.

III. Problem Definition

Here, we use an algorithm to construct a decision tree on streaming data. The framework of the algorithm is given below, which will serve as the root for the next sections.

Decision tree construction algorithm
Stream Tree (Stream DI)
global Tree root, Queue Q1, Q2;
local Node node, Tuple t;
Q1←NULL; Q2←NULL:
Add (root, Q1);
While not (empty (PQ) and empty (AQ))
T← DI.get ();
Node←classify (root, t);
If node Q1
Add (node.sample, t)
If node.stop_condition_satisfied
Remove (node, Q1)
If node.enough_samples()
Use split_function to get the best split;
(node1, node2) ← node.create();
Remove (node, Q1);
Add ((node1, node2), Q2);
While required_memory_exists (Q1, Q2)
Get (node, Q1);
Add (node, Q2);

Two queues are used in this algorithm, Q1 and Q2. Q1 denotes the active queue and represents the set of current decision tree nodes on expanding. Q2 is the group of decision tree nodes that haven't been split yet and are passive (not currently being processed). This can be differentiated on the basis of the extra memory required by the queue that is being currently processed, i.e Q1. Depending on the free memory available, Q1 is built from Q2 by including maximum number of nodes possible. The process of constructing the decision tree is initiated by inserting the root node of the tree in the set.

DS denotes the stream of data instances, which is the input to the algorithm. A data instance i is obtained from the data stream DS and the current decision tree node is determined. If node is a part of Q1, then i will be added to the group of samples available for process node. It is then checked if the node meets the break condition. If the condition is met, node will be removed from Q1. If not, it is checked if now we have adequate information to split the node. If yes, the node will be split and removed from Q1, and its child nodes will be added to the set. When both Q1 and Q2 are empty, the algorithm will terminate.

The algorithm is sensitive to a number of problems in construction of decision tree of streaming data, one of which is computational efficiency of checking all possible splitting conditions of a node.
The next section will give the details of how this particular issue will be handled.

IV. Using Discriminant Analysis For Splitting Criteria

The algorithm will start with the construction of a leaf. When a splitting test is used at a leaf, the leaf will become a decision node, and will generate two child nodes. The splitting test will have two attainable outcomes i) True, which will be associated with one branch and ii) false which will be associated with the second branch.

The splitting tests are produced over a mathematical attribute and are of the type $attribute_i < value_i$ from all numerical attributes, the most favourable value is chosen. We use an amended version of the analytical method presented in [4] for the selection of the split point. The only adequate statistics required are the mean and the variance per class of each numerical attribute. This is a major advantage over other comprehensive methods in C4.5 [12] and in VFDTc [2]. Because all the essential statistics are computed dynamically. This is a sought after property for vast data streams as it assures constant time processing for each example. The analytical method uses an amended form of quadratic discriminant analysis to include distinct variances of the two classes and the distribution of values of an attribute which follows a normal distribution for each of the two classes.

Let,

$$\varphi(\bar{x}, \sigma) = 1/(\sigma\sqrt{2\pi})\ \exp\left(-(x-\bar{x})\text{^}2/(2\sigma\text{^}2\ )\right)$$

be the normal density function. Where σ_2 and $\bar{x}$ are the sample variance and mean of the class. The class variance and mean from the set of examples at the node are estimated for the normal density function. The x axis is split into three intervals by (r1, ∞), (r1,r2), (-∞,r1) by the discriminant and where r1and r2 are the two roots of the equation,

$p(-)\varphi\{ (\bar{x} - , \sigma -)\} = p(+)\varphi\{(\bar{x}+ , \sigma+)\}$,where p(i) denotes the expected probability that an example belongs to class **i**. Because a root is approximately equal to the means of both classes is preferred, we apply a heuristic method. A splitting test is performed based on the information learned. This method needs the standard deviation and mean for each class and for each of its attribute both of which can be easily maintained. Then the splitting test with maximum information gained is chosen. In the generated tree a naive Bayes classifier[1] is maintained at each inner node to detect concept drift. This Bayes classifier is trained with the examples that navigate the nodes. The following section will show the method of detection concept drift.

V. Concept-drift

In real time, concepts are often not static but are dynamic that change with time.

The fundamental data distribution may also change. The model which is built on old information will be updated necessarily. This issue is known as concept drift. This change in the concept drift or the data distribution can be of three types – Sudden, gradual or recurring.

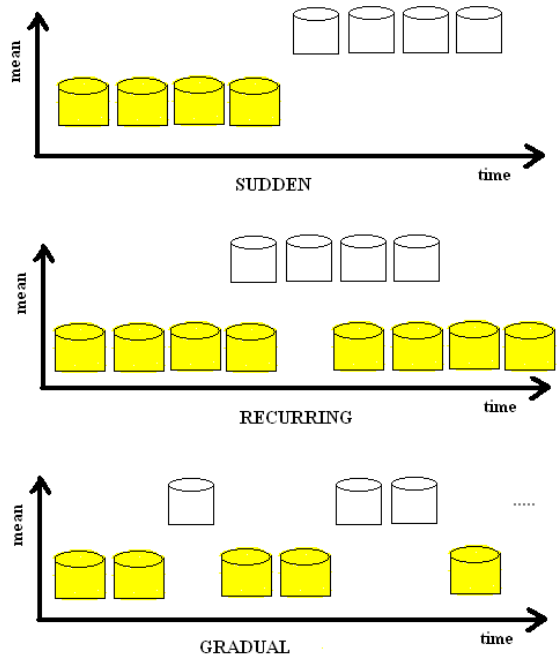

Fig.1. Some of the types of drifts occurred due to change in the distribution of data.

A. Adaptive learning techniques

There are several techniques for handling concept drift, some of which are forgetting approach – it forgets the old data and then trains it again at a fixed rate, detector approach – detect any change in the data and cut old data, contextual approach – this approach builds many models and switches models according to the incoming information and dynamic ensemble approach – in this approach we build several models and then dynamically mix them. The table lists the types of drifts handled:

Adaptive Learning Techniques For Dealing With Various Types Of Drifts

S No.	Adaptive Learning Technique	Type of drift it works on
1	Forgetting	Sudden
2	Detector	Sudden
3	Dynamic ensemble	Gradual
4	Contextual	Recurring

B. Handling Concept-Drift due to gradual change in data distribution

The pith of the drift detection method is to limit the online error rate where in the sequence of examples of data stream, there is a change in the context from one example to another. There is a decrease in the error rate of naive Bayes if the distribution of examples that traverse a node is static. There is a increase in the error rate of

naive Bayes if there is a change on the distribution of examples. [5] When the method detects an increment in the naive Bayes error in a given node, which is an indication of change in the distribution of examples, it suggests that splitting test which had been installed at this node is no longer suitable and the whole sub tree rooted at that node is pared.

When a training example is available, it will cross the corresponding binary decision trees from the root node till a leaf. The naive Bayes classifies each node. The probability of random variables that represent the number of errors in a sample of n instances, is represented by the binomial distribution. We use the following function for the for classifying errors, $P_i = (error_{i\ /}\ i)$. Where, i is the total number of instances or examples and $error_i$ is the number of instances or examples that have been misclassified, both of which are measured in the actual context. The standard deviation for a binomial function is given by $s\_i = \sqrt{(p\_i * (1 – p\_i))/i}$. Where, i is the number of instances or examples observed. For large number of instances, the binomial distribution is approximated by a normal distribution function with the same variance and mean.

The drift detection method consists of two registers for the training of the learning algorithm, R_{min} and Q_{min}. When a new example is processed these values are updated when $R_i + Q_i$ is equal or lower than $R_{min} + Q_{min}$.

We use a warning indicator to define the optimum size of context window. The context window will contain old examples which belong to the new context and a nominal number of examples which are on the old context.

Suppose in the sequence of the examples that traverse a node, there exists an example i with correspondent R_i and Q_i. The warning indicator is reached if $R_i + Q_i \geq R_{min} + 1.5*R_{min}$. drift level is reached if $Q_i + R_i \geq Q_{min} + 3*R_{min}$.

With this technique of learning and forgetting, a model is continuously adapted with respect to the present context. This technique does not require any additional computational resources and uses the information available to the algorithm.

VI. Conclusion

Here in this paper we give an incremental learning algorithm optimal for processing high speed data streams with the ability to adapt concept drift. The analytical method which is used, is restricted to two class problems. The scope can be extended to more than classes. In[2] an extension to VFDT, which deals with continuous attributes has used a B Tree to store continuous attributes. The discriminant analysis method which we have used here has a complexity of O(n) and hence, efficiency of the algorithm has been improved. We also discussed a succinct summary of several learning strategies.

References

[1] R.O. Duda, Peter E. Hart, and David Stork. Pattern Classification. New York, Willey and Sons,1998.

[2] R. Jing and Gagan Agarwal. Efficient Decision Tree Construction on Streaming Data http://web.cse.ohio-state.edu/~agrawal/p/sigkdd03.pdf

[3] S. Babu and J. Wisdom. Continuous Queries over Data Stream http://ilpubs.stanford.edu:8090/527/1/2001-9.pdf

[4] F. Thomas Leighton. Introduction to parallel algorithms and architectures: array, trees, hypercubes. San Francisco, Morgan Kaufmann Publishers

[5] Mehryar Mohri, Afshin Rostamizadeh, Ameet Talwalkar. Foundations of Machine Learning.

[6] Paul Van Tilburg. Space-efficient Algoritms for Data Streams and Histograms http://paul.luon.net/papers/AA-Space-Efficient-Alg.pdf

[7] Y. Liu and B. Plale. Multi-model Based Optimization for Stream Query Processing. http://www.cs.indiana.edu/dde/papers/LiuSEKE06.pdf

[8] Kuncheva L.I. Classifier ensembles for changing environments, Proceedings 5th International Workshop on Multiple Classifier Systems, MCS2004, Cagliari, Italy in F.Roli, J. Kittler and T. Windeatt. Lecture notes in computer science, vol 3077,2004,1-15.

[9] G. Hulten and P. Domingos. Mining high speed data streams. (SIGKDD), 2000.

[10] Tutorial: Handling Concept Drift: Importance, Challenges and Solutions http://www.cs.waikato.ac.nz/~abifet/PAKDD2011/

[11] S. Acid, M. de Campos, J. Luna, S. Rodriguez, J. Rodriguez, J. Salcedo. Artificial Intelligence in Medicine, 2004.

[12] Zliobaite, I., Learning under Concept Drift: an Overview. Technical Report. 2009, Faculty of Mathematics and Informatics, Vilnius University, Lithuania