2008 International Symposium on Information Science and Engieering

An Improved Algorithm of Decision Trees for Streaming Data Based on VFDT

Feixiong Li* Quan Liu
Provincial Key Laboratory for Computer Information Processing Technology, Soochow University, Suzhou, 215006, China.
*064227065028@suda.edu.cn, quanliu@suda.edu.cn

***Abstract*-Decision tree is a good model of Classification. Recently, there has been much interest in mining streaming data. Because streaming data is large and no limited, it is unpractical that passing the entire data over more than one time. A one pass online algorithm is necessary. One of the most successful algorithms for mining data streams is VFDT(Very Fast Decision Tree).we extend the VFDT system to EVFDT(Efficient-VFDT) in two directions: (1)We present Uneven Interval Numerical Pruning (shortly UINP) approach for efficiently processing numerical attributes. (2)We use naive Bayes classifiers associated with the node to process the samples to detect the outlying samples and reduce the scale of the trees. From the experimental comparison, the two techniques significantly improve the efficiency and the accuracy of decision tree construction on streaming data.**

Keywords-Streaming Data Mining; Decision Trees; Unequal Interval Numerical Pruning(UINP); Naive Bayes Classifiers

I. INTRODUCTION

Classification is a important issue in data mining, and decision tree is a good model of classification. Traditional algorithms construct a decision tree based on disk-resident datasets. Recently, there has been much interest in mining streaming data. We can scan the datasets many times when it is stored in the disk, but it's impossible to do same with streaming data. There is only one pass allowed over the streaming data because of the high-speed and the infinity of the data stream[1]. we must process it online and fast.

There are many algorithms proposed in streaming data mining [2-4]. These algorithms get much progress with traditional algorithms, but there are much space to improve. VFDT is a famous algorithm for streaming data mining. It uses Hoeffding bounds to guarantee that its output is asymptotically nearly identical to that of a conventional learner[5]. It can resolve the issue that there have too many examples in data streaming, but it can't handle the numerical attributes.

We present a new algorithm named EVFDT based on VFDT. This algorithm can handle numerical attributes efficiently used the Uneven Interval Numerical Pruning (shortly UINP) that we present in this paper, and we used the naive Bayes classifiers to reduce the scale of the trees. Experimental results show it can significantly improve the efficiency of decision tree construction on streaming data.

II. DECISION TREE CONSTRUCTION ON STREAMING DATA

A. *Decision Tree Classifier*

Decision tree is one of the widely used and practical methods for classification. A decision tree traverse the node from the root to leaves to classify the instances. A class label is associated with each leaf node. When the traveler arrives at a leaf node, a instance is classified. Our discussion will assume that there are only two distinct class labels, it will simplify the issues and not change the essentiality.

Building a decision tree classifier generally includes two stages, a growing stage and a pruning stage. Assume there are three data sets, an instances set $D=\{t_1,t_2,\ldots,t_n\}$, an attributes set $X=\{x_1,x_2,\ldots,x_m\}$, and a class label sets $C=\{c_1,c_2\}$. One attribute may be only numerical or discrete. Constructing a decision tree from the attributes set to classify the instances set. A good algorithm should have high accuracy and small scale.

B. *Streaming data mining and VFDT*

There are some difference between streaming data and the traditional datasets which stored in disk. It's impossible to read the whole data set to memory. the scale of the data will overflow the memory. we can only scan the data stream one time. If the instances of the stream satisfy some distribution, we can take it as a random sample set.

VFDT is a famous algorithm which was improved from the Hoeffding tree algorithm. It uses Hoeffding bounds to detect the number of the samples to split a node. Assume r is information gain, R is the value range of r, considered n values of r and computed their mean $\bar{r}$, The Hoeffding bound states that, with probability 1-δ, the true mean of the variable is at least $\bar{r} - \delta$, where

$$\varepsilon = \sqrt{\frac{R^2 \mathrm{In}(1/\delta)}{2N}}.$$

The Hoeffding bound is independent of the probability distribution generating the observations, this make the bound more conservative than distribution-dependent ones.

III. HANDLING NUMERICAL ATTRIBUTES

A key issue in processing the numerical attributes is the high cost of computing and storing. We present a method called Uneven Interval Numerical Pruning (UINP) to handle the numerical attributes. It uses some unequal-width intervals to split the numerical attributes then prune these intervals. The unequal-width intervals are more flexible than equal-width intervals. There is a algorithm using equal-width intervals handle the numerical attributes[6],we extend it to unequal-width intervals to get more efficiency.

An interval maybe pruned or unpruned. An unpruned interval is intact which we create it to split the numerical attribute. If it can't split a numerical attribute properly, we must prune it. We used two structures to maintain the information for each node that is being processed.

(1) Class Histograms. This is primarily comprised of class histograms for all categorical attributes and numerical attributes. It also records the number of occurrences of instances with each class label and the value of the categorical attributes and numerical attribute within that interval.

(2) Binary Tree. This is comprised of the information of intervals, e.g. the number and the width of each interval. We also stored the information of the nodes in it.

Table 1: UINP Algorithm for Numerical Attributes Handling

```
UINP(Node N, Stream D)
while not satisfy_stop_condition(N){
    Simple S ←(D.getStream());
   Update_ClassHist(S);
   Update_BinaryTree(S);
   g←Find_Best_Gain(ClassHist);
   if Statistically_Best_Gain(g)
       Split_Node(N);}
End UINP;
```

The pseudo-code for Uneven Interval Numerical Pruning (UINP) algorithm is presented in Table 1. Hereafter collecting some samples, we use class histograms from unpruned intervals and get an estimate of the best gain. This is denoted as $g^{'}$. Then, by using $g^{'}$, we unpruned intervals that look promising to contain the best gain, based upon the current sample set. The best gain g can come from $g^{'}$ or other newly unpruned intervals. Then, by performing a statistical test, we check if we can now split this node. If not, we need to collect more samples. Before that, however, we check if some other additional intervals can be pruned.

Assuming that we are trying to prune the space of potential split points from a numerical attribute, and g is the best gain possible from any splitting condition associated with a node of the decision tree. Suppose $[X_i,X_i+1]$ is the ith interval on the numerical attribute X. Supposing the class distribution of the interval I be $H_i=(h_{i1},h_{i2},\ldots,h_{ic})$, where $1\leq j\leq c$ is the number of training records with the class label j that fall into the interval I, and c is the total number of class labels in the training data. We want to determine if any point within this interval can provide a higher gain than g.

Let $\bar{g}$ be the best gain computed using the current sample set. Further, let $g^{'}$ be the best gain noted after using class histograms from the intact intervals. This is the value first computed by our algorithm. Since of $g^{'} \leq \bar{g}$, this follows simply from the fact that $g^{'}$ is computed from among a subset of the split points that $\bar{g}$ would be computed from.

We can estimate the upper bound on the gain possible from a split point within a given interval i using the values of gain at interval boundaries. We denote this value as u_i, and is an estimate of the value u_i, that could be computed using the entire dataset. if we have computed g and u_i, we can prune an interval i using the entire dataset when $g > u_i$.

To spilt a point efficiently with numerical attribute is a hard job, which the computational and memory cost is very high. One key issue is to develop good structures to stored information for processing. UINP is an efficient algorithm to do this job as the experiment results shows.

IV. Naive Bayes Classifier for Data Stream

There are strong advantages in the performance of resulting decision models. Naive Bayes Classifier was first used in the algorithm named naive Bayes Tree(shortly NBTree). NBTree is a hybrid algorithm that generates a regular univariate decision tree, but the leaves contain a naive Bayes classifier built from the examples that fall at this node. Inner nodes and leaves contain naive Bayes classifiers playing different roles during the induction process. Naive Bayes in leaves are used to classify test examples, and naive Bayes in inner nodes can be used as splitting-tests if chosen by the splitting criteria, and used to concise the sample sets from the data stream[7]. We used naive Bayes in the inner node and the leaves to prompt the efficiency of the construction algorithm and concise the scale of the final trees. The approach retains the interpretability of naive Bayes and decision trees, while resulting in classifiers that frequently outperform both constituents, especially in large datasets.

To classify a test example, the classifier traverses the tree from the root to a leaf. The example is classified with the most representative class of the training examples that fall at that leaf[8]. EVFDT used naive Bayes classifier processing samples of the data stream to concise the range of the classes. The Bayes classifier not only considered class distribution, but also considered the conditional probability. Increased the rate of the information using, thus improved the rate of accuracy hopefully.

Consider a sample e=(x_1,…, x_i), used the Bayes' theorem $P(c_k|e) \propto \frac{P(c_k)}{P(e)} \prod P(x_j|c_k)$. These are the necessary statistics to compute the conditional probabilities of $P(x_j|c_k)$. To work out $P(x_j|c_k)$, we must distinguish the conditional attributes with the useless attributes, and get the information gain. Assuming all of attributes are normal distribution. We stored the value of the attributes and the information of the nodes in a binary tree. All of the numerical attributes will be discrete with the method we present formerly.

When a example was detected from the data stream, we used naive Bayes classifier to estimate whether it is a valid example related with the objective conception. If the value of $P(x_j|c_k)$ in the range of we set in former, we think this example is a objective sample and classify it, or else the example will be discarded. This step reduces the scale of the dataset and prompts the efficiency of the classification.

V. Performance Evaluation

We compare EVFDT to VFDT and C4.5, All our experiments were performed on a Pentium(R) D with 2.8GHz processor running Windows XP SP2 with 512MB of main memory.

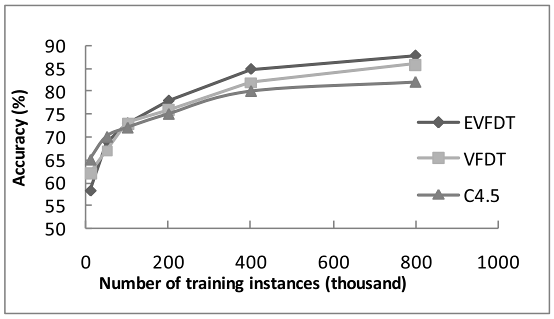

fig.1. Comparation of the accuracy of there algorithms

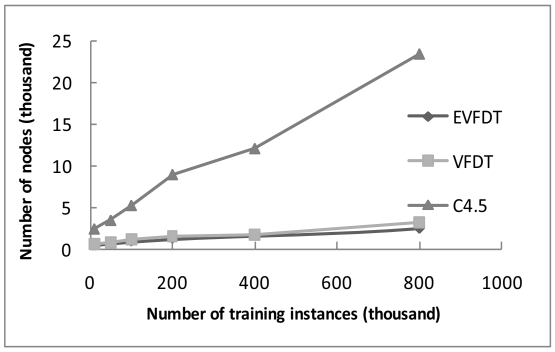

fig.2. Comparation of the scale of there algorithms

We used these algorithms mining the data stream from the first affiliated hospital of Soochow University. We get these figures based the experiment.

Experiment shows there are no much difference between these three algorithms when the data set is small. When the number of the samples increased by time, the accuracy of the C4.5 is not match the EVFDT and VFDT, and the number of nodes increase quickly. It shows the traditional algorithms can't adapt to mine data stream. Figure 1 and 2 show that the accuracy of the EVFDT's exceeds the VFDT's, and the size of the tree of the EVFDT's is more small than VFDT's.

ACKNOWLEDGMENT

This work was supported by National Natural Science Foundation of China (60673092, 60775046,60873116); Important Project of Ministry of Education(207040); Postdoctor Science Foundation of China (20060390919); Natural Science Foundation of JiangSu (BK2008161); Higher Education Natural Science Foundation of JiangSu (06KJB520104).

REFERENCES

[1] Jiawei Han, M Kamber. Data Mining: Concepts and Techniques. Second Edition, 2007.

[2] J. Gehrke, F. Korn, and D. Srivastava. On computing correlated aggregates over continual data streams. In Proc. of the 2001 ACM SIGMOD Intl. Conf. on Management of Data, pages 13–24. acmpress, June 2001.

[3] A. Arasu, B. Babcock, S. Babu, J. McAlister, and J. Widom.Characterizing memory requirements for queries over continuous data streams. In Proc. of the 2002 ACM Symp. on Principles of Database Systems. ACM Press, June 2002.

[4] P.Domingos and G.Hulten Mining high-speed data streams. In Proceedings of the ACM Conference on Knowledge and Data Discovery (SIGKDD), 2000.

[5] P. Domingos and G. Hulten. Mining high-speed data streams. In Knowledge Discovery and Data Mining, pages 71-80, 2000.

[6] Jin R. and Agrawal G. Efficient Decision Tree Construction on Streaming Data. In Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD-2003)-poster, pages 571-576, 2003.

[7] J Gama , Pedro Medas, Pedro Rodrigues, Learning decision trees from dynamic data streams, Proceedings of the 2005 ACM symposium on Applied computing, March 13-17, Santa Fe, New Mexico, 2005.

[8] J. Gama, R. Rocha, and P. Medas. Accurate decision trees for mining high-speed data streams. In P.Domingos and C. Faloutsos, editors, Procs. of the 9th ACM SigKDD. ACM Press, 2003.