Decision Tree Evolution Using Limited Number of Labeled Data Items from Drifting Data Streams

Wei Fan[1]   Yi-an Huang[2]   Philip S. Yu[1]

[1]IBM T. J. Watson Research, Hawthorne, NY 10532

{weifan,psyu}@us.ibm.com

[2]College of Computing, Georgia Institute of Technology, Atlanta, GA 30332

yian@cc.gatech.edu

Abstract

Most previously proposed mining methods on data streams make an unrealistic assumption that "labelled" data stream is readily available and can be mined at anytime. However, in most real-world problems, labelled data streams are rarely immediately available. Due to this reason, models are reconstructed only when labelled data become available periodically. This passive stream mining model has several drawbacks. We propose a new concept of demand-driven active data mining. In active mining, the loss of the model is either continuously guessed without using any true class labels or estimated, whenever necessary, from a small number of instances whose actual class labels are verified by paying an affordable cost. When the estimated loss is more than a tolerable threshold, the model evolves by using a small number of instances with verified true class labels. Previous work on active mining concentrates on error guess and estimation. In this paper, we discuss several approaches on decision tree evolution.

1 Introduction

Most previous work on mining data streams concentrates on capturing time-evolving trends and patterns with "labeled" data. However, one important aspect that is often ignored or unrealistically assumed is the availability of "class labels" of data streams. Most algorithms make an implicit and impractical assumption that labeled data is readily available. Most works focus on how to detect the change in patterns and how to update the model to reflect such changes when there are labelled instances to be mined. However, for many applications, the class labels are not "immediately" available unless dedicated efforts and substantial costs are spent to investigate these labels right away. If the true class labels were readily available, data mining models would not be very useful - we might just wait. In credit card fraud detection, we usually do not know if a particular transaction is a fraud until at least one month later after the account holder receives and reviews the monthly statement. As another example, in large organizations, data mining engine normally runs on a data warehouse, while the real-time data streams are stored, processed and maintained on a separate production server. In most cases, the data on the production server is summarized, de-normalized, cleaned up and transferred to the data warehouse periodically such as over night or over the weekend. The true class labels for each transaction are usually kept and maintained in several database tables. It is very hard to provide the true labels to the learner at real time due to volume and quality issues. Most current applications obtain class labels and update existing models in preset frequency, usually synchronized with data refresh. As a summary, the passive life cycle of today's stream data mining is: "given labeled data → train initial model → classify data stream → passively wait for labeled data → re-train model . . .". The effectiveness of the passive mode is dictated by some application-oriented constraints, yet not by the demand for a better model with a lower loss. Such a passive mode to mine data streams results in a number of potential undesirable consequences that contradict the notions of "streaming and continuous". If either the concept or data distribution drifts rapidly at an un-forecasted rate that application-dependent constraints do not catch up, the models is likely to be out-of-date on the data stream and important business opportunities might be missed. If there is neither conceptual nor distributional change, periodic passive model refresh and re-validation is a waste of resources.

Demand-driven Active Mining of Data Streams The demand-driven active stream data mining process is originally proposed in [Fan et al., 2004]. There are three simple steps:

1. Detect potential changes of data streams on the fly

when the existing model classifies continuous data streams. The detection process does not use or know any true labels of the stream. One of the change detection methods is a guess of the actual loss or error rate of the model on the new data stream.

2. If the guessed loss or error rate of the model in Step 1 is much higher than an application-specific tolerable maximum, we choose a small number of data records in the new data stream to investigate their true class labels. With these true class labels, we statistically estimate the true loss of the model.

3. If the statistically estimated loss in Step 2 is verified to be higher than the tolerable maximum, we reconstruct or evolve the old model using the same true class labels sampled in the previous step.

We extend the classification tree algorithm to implement active mining. Previous work in [Fan et al., 2004] concentrates on the first two steps of the active mining or error estimation. In this paper, we describe a solution to the third step or an extension to evolve existing decision trees using limited number of labelled instances.

2 The Approaches

If the loss of a decision tree on the data stream is suspected to be higher than an empirically tolerable threshold, it is necessary to reconstruct the tree in order to reduce its loss. Since the data stream doesn't have true class labels at the moment and verifying every single true class label is impossible, we choose to verify the true class labels of a small number of instances selected from the data stream. We then use these sampled true class labels to reconstruct the decision tree at the leaf node level by either replacing the class probability distribution or expanding a leaf node.

Class Distribution Replacement The basic idea is to use those examples with verified true class labels, that are classified by the leaf node, to estimate the current class probability distribution at this leaf node. When the estimated class distribution based on examples with verified true class labels is significantly different from the original class distribution at this leaf node, the class distribution will be replaced with the new distribution.

At a particular leaf, the probability distribution of class labels is a "proportion statistics". For many practical loss-function, such as 0-1 loss and cost-sensitive loss, we are only interested in the probability of one class, usually the positive or rare class. Assume that p (shorthand for $p(c|\mathbf{x})$) is the probability estimated from the sample that examples classified at this leaf node is of class c, the confidence limits

for examples classified by this leaf to be of class c is

$$p \pm \left[t\sqrt{1-f}\sqrt{pq/(n'-1)} + \frac{1}{2n'} \right] \qquad (1)$$

where $q = 1 - p$, N' is the total number of examples from the data stream that is classified by the leaf, n' is the number of examples among N' that are verified with their true class labels, and $f = \frac{n'}{N'}$. Exact methods exist, however this normal approximation is good enough in practice. It is obvious that the standard error is a function of both the estimated probability p and sample size n'. When the difference in distribution is significant with high confidence and the difference results in less loss, the distribution of the sample will replace that in the leaf.

To give an example under 0-1 loss, assume that we have two labels $\{-, +\}$. The original distribution of a leaf node is $P(+|\mathbf{x}) = 0.7$ and the prediction will be $+$ since it is the majority label. If the distribution of those examples with verified true class labels is $p(+|\mathbf{x}) = 0.45$ with confidence limit of 0.1 at 99.7% confidence interval, we will change the distribution and the majority label will be - instead of +. However, if the confidence limit is 0.3, we can not make a decision, but keep the original class distribution until we can afford to verify the true class labels of more instances.

Leaf Node Expansion Leaf node expansion is necessary if the replaced class label distribution cannot improve the loss at this node. For example, under 0-1 loss, if the new distribution is 51% positive, it means that 49% of examples classified by this node as positive are actually negative. However, this leaf node can be expanded or further grown by invoking the tree growing function. Decision tree algorithms, such as C4.5, expands a node as much as possible until the number of examples reaches the minimum threshold (2 by default in C4.5). However, if the cumulative loss by growing a node into a branch is the same as the original node, the original node is kept.

3 Experiments

We conducted extensive experiments on both synthetic and real life data streams. Our goals are to evaluate the impact of the frequency and magnitude of both the distribution and concept drifts on prediction accuracy of the extended classification tree, and to analyze the advantage of our approach over complete re-training on the new data streams. The decision tree algorithm was modified from C4.5.

Streaming Data We use both synthetic dataset and real life datasets in our experiments. Detailed description about the synthetic and credit card fraud datasets can be found

in [Fan et al., 2004]. The adult dataset from UCI repository. We use the natural split of training and test sets. In order to simulate pattern-drifting streams, we sample different portions of positive and negatives to generate the new data stream, i.e, {10/90,20/80,. . . ,90/10}.

Experiment Setup   Each of the three datasets have one training set and a series of data stream chucks with increasing percentage of either distribution or concept drifts from the original training set. An original classification tree, or sometimes called old decision tree in the rest of paper, is constructed from the original training set. Then the series of data stream chunks are applied on this original classification tree to compute its error. When the data stream chunks arrive, a small percentage of them are sampled to verify their true class labels to evolve the original decision tree. The resultant tree is called either evolved decision tree or reconstructed decision tree. For comparative studies, we also compute a new decision tree "from scratch" using every data item of the new data chunk.

3.1   Experimental Results

Class Distribution Replacement   The results by replacing the class distribution at the leaf nodes are plotted in Figure 1. The total number of examples with verified true class labels is 10% of data item from the data stream. There are three curves in each plot. For the synthetic and adult datasets, the curve on the top is the loss of the original decision tree on the new data stream. The curve in the middle is the loss of the evolved decision tree, and the curve at the bottom is the loss of the new decision tree. Since we use benefits (average amount of recovered money per transaction) in the credit card datasets, the order of the curves reverses, i.e., the original decision tree is the one with the least benefits and it is at the bottom. To simulate concept-changing data streams, the transactions are sorted with decreasing transaction amount and the overhead to investigate a fraud is $90. The old decision tree actually starts to produce negative benefits after the x-axis is more than 5. This is because the extra overhead of false positives exceeds the amount recovered from true positives.

The general observation among all three datasets is that the evolved decision tree is lower in loss than the original decision tree. In addition, for the credit card fraud and adult datasets, the evolved decision tree has accuracy very close to that of the completely re-trained decision tree. Comparing the synthetic dataset result with that of credit card fraud and adult datasets, the evolved decision tree improved the accuracy of the old tree by a relatively small amount. We suspect this is probably due to the relative small size of the data chunk, and the algorithm cannot pass the significance test to replace the class probability distribution at most of

the leaf nodes.

Leaf Node Expansion   We first experimented to use the same 10% of true labels, as used in class probability distribution, to expand leaf nodes. However, most of the nodes do not have enough examples to justify the need for an expansion. We then increased the percentage to 20%. But there were still a lot of nodes that could not be expanded due to statistical insignificance. In the end, we increased the number of sampled true class labels to 30%. The plots on tree evolution via leaf node expansion are shown in Figure 2. There are 4 curves in each plot, including the loss of the original decision tree on the data streams, the loss of the reconstructed decision by leaf node expansion (either unpruned or pruned after expansion), and the loss of the newly trained decision tree on the complete data stream. The credit card dataset uses benefits (average amount of recovered money per transaction) rather than cost, and the order of the curves reverses. A pruned reconstructed decision tree removes statistically insignificant expansions of a reconstructed unpruned decision tree.

There are a few observations from the plots. Reconstructed decision tree is more accurate than the original decision tree. Pruned reconstructed decision tree is more accurate than unpruned reconstructed decision tree in general. Comparing with the results of class probability distribution replacement, as in Figure 1, evolving decision trees via leaf node expansion is more accurate than class distribution replacement only for the synthetic dataset. Leaf node expansion and class probability replacement results on credit card fraud detection and adult datasets are very similar.

4   Related Work

Data stream processing has recently become a very important research domain. Much work has been done on modeling [Babcock et al., 2002], querying [Gao and Wang, 2002], mining [Hulten et al., 2001], regression analysis [Chen et al., 2002], clustering [Guha et al., 2000] as well as visualization [Aggarwal, 2003]. A most recent updated list of stream processing related work appeared in a tutorial given in [Pei et al., 2004]. The most closely related work to this paper appears in [Fan et al., 2004]. The substantial difference of this paper from [Fan et al., 2004] is on the reconstruction of streaming models with limited number of labelled data, while the focus of [Fan et al., 2004] is on change detection and loss estimation either without labelled data or with very limited number of targeted data items. The work in this paper compliments the work of [Fan et al., 2004].
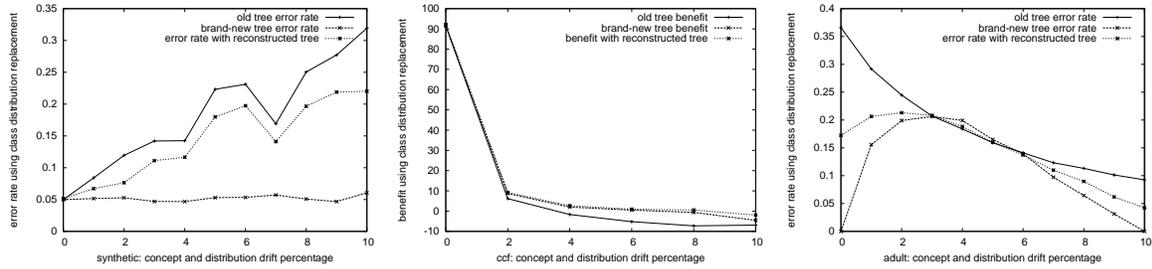
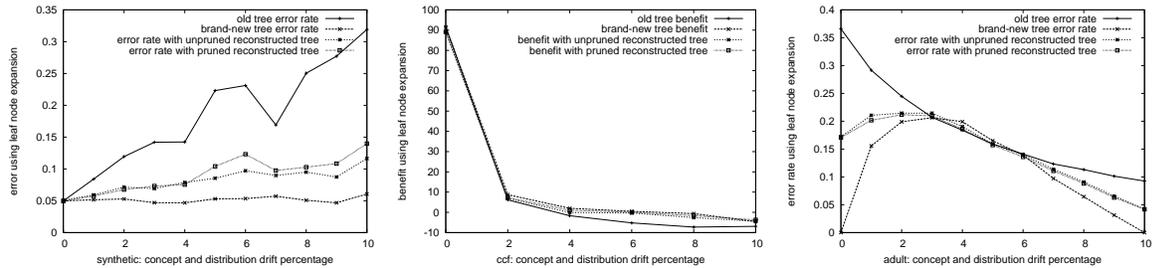**Figure 1. Loss on reconstructed tree via replacing class distribution**

Figure 2. Loss on reconstructed tree via leaf node expansion

5 Conclusion

We extended a new framework to mine data streams that solves one important problem, that the true labels of the data stream are not immediately available. However, model evolution is necessary whenever loss of the current model is suspected to be higher than a tolerable maximum. We proposed two approaches to evolve the old decision tree by either i) replacing existing class probability distribution or, ii) expanding the leaf nodes. Experimental studies were conducted on both synthetic and real-life data streams. We have found that the evolved decision tree using from 10% to 30% of true class labels is significantly more accurate than the original decision tree on concept-drifted data streams, and is comparable to a brand new decision tree constructed using all available data items of the new data chunk. Between the two decision tree evolution methods, we also find that leaf node expansion requires more examples with true class labels than class probability distribution replacement, in order to pass the statistical significance test.

References

[Aggarwal, 2003] Aggarwal, C. C. (2003). A framework for diagnosing changes in evolving data streams. In *Proceedings of ACM SIGMO 2003*, pages 575–586.

[Babcock et al., 2002] Babcock, B., Babu, S., Datar, M., Motawani, R., and Widom, J. (2002). Models and issues in data stream systems. In *ACM Symposium on Principles of Database Systems (PODS)*.

[Chen et al., 2002] Chen, Y., Dong, G., Han, J., Wah, B. W., and Wang, J. (2002). Multi-dimensional regression analysis of time-series data streams. In *Proc. of Very Large Database (VLDB)*, Hongkong, China.

[Fan et al., 2004] Fan, W., an Huang, Y., Wang, H., and Yu, P. S. (April 2004). Active mining of data streams. In *Proceedings of 2004 SIAM International Conference on Data Mining*, pages 457–461.

[Gao and Wang, 2002] Gao, L. and Wang, X. (2002). Continually evaluating similarity-based pattern queries on a streaming time series. In *Int'l Conf. Management of Data (SIGMOD)*, Madison, Wisconsin.

[Guha et al., 2000] Guha, S., Milshra, N., Motwani, R., and O'Callaghan, L. (2000). Clustering data streams. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 359–366.

[Hulten et al., 2001] Hulten, G., Spencer, L., and Domingos, P. (2001). Mining time-changing data streams. In *Int'l Conf. on Knowledge Discovery and Data Mining (SIGKDD)*, pages 97–106, San Francisco, CA. ACM Press.

[Pei et al., 2004] Pei, J., Wang, H., and Yu, P. S. (August 2004). Tutorial: Online mining data streams: Problems, applications and progress. In *A tutorial for 2004 ACM SIGKDD International Conference on Knolwedge Discovery and Data Mining(KDD'2004)*, Seattle, Washington, USA.