

# Active Mining of Data Streams

Wei Fan<sup>1</sup>

Yi-an Huang<sup>2</sup>

Haixun Wang<sup>1</sup>

Philip S. Yu<sup>1</sup>

<sup>1</sup>IBM T. J. Watson Research, Hawthorne, NY 10532

{weifan, haixun, psyu}@us.ibm.com

<sup>2</sup>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 refreshed periodically, that is usually synchronized with data availability schedule. There are several undesirable consequences of this “passive periodic refresh”. In this paper, we propose a new concept of demand-driven active data mining. It estimates the error of the model on the new data stream *without* knowing the true class labels. When significantly higher error is suspected, it investigates the true class labels of a selected number of examples in the most recent data stream to verify the suspected higher error.

## 1 State-of-the-art Stream Mining

State-of-the-art 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 learned*. 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. Due to these facts, most current applications obtain class labels and update existing models in preset frequency, usually synchronized with data refresh. The effectiveness of the passive mode is dictated by some “statuary and static constraints”, yet not by the “demand” for a better model with a lower loss. Such a pas-

sive mode to mine data streams results in a number of potential undesirable consequences that contradict the notions of “streaming” and “continuous”. First, it may incur possibly higher loss due to neglected pattern drifts. If either the concept or data distribution drifts rapidly at an un-forecasted rate that statuary constraints do not catch up, the models are likely to be out-of-date on the data stream and important business opportunities might be missed. Second, it may have unnecessary model refresh. If there is neither conceptual nor distributional change, periodic passive model refresh and re-validation is a waste of resources.

**1.1 Demand-driven Active Mining of Data Streams** We are proposing a demand-driven active stream data mining process that solves the problems of passive stream data mining. As a summary, our particular implementation of active stream data mining has 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 the old model by using the same true class labels sampled in the previous step.

In this paper, we concentrate on the first two steps. Our particular implementation extends on classification trees.

## 2 The Framework

We first discuss different types of possible pattern changes in the data stream, then propose a few statistics to monitor in

the decision tree to detect changes.

**2.1 Changes in Data Stream** There are three possible types of changes in the data stream, i) distribution change, ii) concept drift as well as iii) combined distribution and concept drift. In our work, we explicitly exclude new symbolic values and new class labels. Given an unknown target function  $y = f(\mathbf{x})$  over domain  $\mathbf{X}$ . A “complete” dataset  $D_c$  is defined over every possible  $\mathbf{x} \in \mathbf{X}$  with its corresponding  $y$ 's. A complete dataset is not always possible and is most likely infinite. A training set is typically a sample from the complete dataset. A dataset  $D$  of a given size is a sample (usually with replacement) from the complete dataset  $D_c$ , in which each data point has some prior probability to be chosen. A training dataset  $D$  and data stream  $S$  have different distribution if the same example has different probability to be chosen by  $S$  than by  $D$ . A concept drift refers to target function changes, i.e., assume  $y = g(\mathbf{x})$  is the target function of the data stream, there exists  $\mathbf{x}$  such that  $f(\mathbf{x}) \neq g(\mathbf{x})$ . In reality, data streams may have both distribution and concept drifts. Next, we discuss how distribution and concept changes are reflected in a decision tree's statistics.

**2.2 Observable Statistics in Decision Trees** Assume that  $dt$  is a decision tree constructed from  $D$ .  $S$  is a data stream. The examples in the data stream  $S$  are classified by a unique path from the root to some leaf node. Assume that  $n_\ell$  is the number of instances classified by leaf  $\ell$  and the size of the data stream is  $N$ . We define the statistics at leaf  $\ell$  as

$$(2.1) \quad P(\ell) = \frac{n_\ell}{N}$$

Obviously  $\sum P(\ell) = 1$  summed over all leaf nodes in a tree.  $P(\ell)$  describes how the instance space of the data stream  $S$  is shattered among the leaf nodes solely based on attribute test results of a given decision tree  $dt$ . It doesn't consider either the true class labels or attributes that is not tested by  $dt$ . If the combination of attributes values in the data stream  $S$  is different from the training set, it will be reflected in  $P(\ell)$ . The change of leaf statistics on a data stream is defined as

$$(2.2) \quad PS = \frac{\sum_{\ell \in dt} |P_S(\ell) - P_D(\ell)|}{2} \times 100\%$$

The increase in  $P(\ell)$  of one leaf is contributed by decrease in at least one other leaf. This fact is taken into account by dividing the sum by 2. When there is significant changes in the data stream, particularly distribution drifts, this statistic is likely to be high.

The effect of drifts on decision trees can also be expressed in loss functions. If  $S$  and  $D$  have the same distribution, we can use the loss on the training set or hold-out validation set to estimate the “anticipated loss” or  $L_a$  on

the data stream “without” even looking at the data stream. Assume that the error rate (0-1 loss) on the hold-out validation set is 11%. If there is no drifts on the data stream, the error on the data stream is expected to be around 11%. This 11% error rate conjecture is the “anticipated” loss. For credit card fraud detection, assume that the total money recovered from fraud on a validation set of 10000 transactions is \$12000. Then the anticipated total money recovered from a data stream of 5000 transactions is approximately \$6000 ( $=\$12000 \times 5000/10000$ ).

$$(2.3) \quad L_a = \text{validation loss}$$

On the other hand, rather than a blind conjecture, a better guess takes both the decision tree itself and  $S$ 's attribute values into account. Consider the number of examples at some leaf node, without any prior knowledge about how the distribution or concept of the examples at this leaf node may have changed or in other words everything is possible, rather than a wild random guess, the best guess is to use the distribution on the training data of this leaf, i.e.,  $P(c|\mathbf{x})$  as the “expected” or “averaged” probability distribution to estimate the loss on this leaf for the streaming data. Then the loss on the data stream is the cumulative loss of all the leaf nodes of the tree, called “expected loss” or  $L_e$ . Assume 0-1 loss and the probability of the majority class at some leaf node is 0.7. If the portion of examples in the data stream classified by this node is 30%, the portion of examples in the data stream that are expected to be classified incorrectly is  $(1 - 0.7) \times 30\% = 9\%$ . We iterate this process for every leaf node in the tree and sum up the expected loss from every node to reach the overall expected loss.

$$(2.4) \quad L_e = \text{sum of expected loss at every leaf.}$$

Expected loss takes the attribute value of the data stream into account. Examples are sorted into leaf nodes by attribute tests. Leaves classifying more examples in the data stream contribute more to the overall loss. A leaf classifying more examples in the training set may not necessarily classify the same proportion of examples in the data stream due to drifts of the data stream. The difference of anticipated and expected loss is an indicator of the potential change in loss due to changes in the data stream.

$$(2.5) \quad LS = |L_e - L_a|$$

Although both  $PS$  (as defined in Eq(2.2)) and  $LS$  are indicators of the possible drifts in data streams,  $LS$  takes the loss function into account.

**2.3 Tolerable Maximum** First, we need to find out the tolerable maximum of each statistics. The exact values are completely defined by each application, i.e., the dataset itself and anticipated performance. We use overall expected loss

$L_e$  (Eq (2.4)) as an example. In credit card fraud detection, if 50c per transaction is the most we can afford to lose due to fraud, the tolerable maximum for  $L_e$  is at most 50c. If the business target is not to lose more than 40c, the tolerable maximum will be set to 40c. The threshold for leaf change statistics can be correlated with model loss.

**2.4 Loss Estimation** The above two methods are likelihood indicators since they do not use any true labels of the data stream. When the values of these two statistics are above their tolerable maximums, we use a statistically reliable method is to investigate true class labels of a selected number of examples in the data stream and estimate the expected loss and its standard error.

We randomly sample a small number of true labels from the data stream and compute its average loss and standard error. Assume that we have a sample of  $n$  examples out of a data stream of size  $N$ . The loss on each example is  $\{l_1, l_2, \dots, l_n\}$ . From these losses, we compute the average sample loss  $\hat{l}$  and its variance  $s^2 = \frac{\sum_1^n (l_i - \hat{l})^2}{n-1}$ . Then the unbiased estimate to the average loss on the stream and its standard error are  $\hat{l}$  and  $\frac{s}{\sqrt{n}}$  respectively.

**2.5 Investigation Cost** Investigating true class labels from current data stream is not for free. The number of examples that we can afford to investigate are dictated by an application specific cost model.

In credit card fraud detection, the true label, i.e., fraud or non-fraud, of a transaction, is typically just a phone call away. However, there is a cost associated with each phone call. This cost is a function of the number of employees, their payroll and logistics cost, and the number of phone calls each one can make on a normal business day. Assume that this cost is \$5 per call. Calling to confirm every transaction is impossible, but calling to confirm a selected number of transactions is affordable and can actually save money. Assuming that the guessed loss is 60c per transaction, and the credit card company has a target of 40c loss per transaction and they have 0.1M transactions on a daily basis. Then the credit card company may lose \$20,000 (=20c  $\times$  0.1M) more per day. Instead of losing \$20,000 per day, we can spend some money to make phone calls to card holders to verify selected transactions to improve the model to prevent any further loss. When the model is improved, we will not incur \$20,000 loss per day. If we choose to spend \$10,000, we can make \$10,000 / \$5 = 2000 phone calls. Assuming with 2000 verified transactions, we can improve our fraud detection model to avoid the \$20,000 loss per day. The total amount of money saved for one month period (i.e., the frequency of model refresh) is \$20000  $\times$  30 - \$10000 = \$590000. For simplicity, the above analysis assumes that the guessed loss is exactly accurate, i.e., the 60c guesses loss is the exact loss

of the old model.

### 3 Experiments

**Streaming Data** The first one is a *Synthetic Dataset*. We create drifting concepts based on a moving hyperplane. A hyperplane in a  $d$ -dimensional space is denoted by  $\sum_{i=1}^d a_i x_i = a_0$ . We label examples satisfying  $\sum_{i=1}^d a_i x_i \geq a_0$  as positive, otherwise as negative. In a real life data set, examples are often not uniformly distributed in the multi-dimensional space  $[0, 1]^d$ . Some prior unknown distribution criteria may apply on whether certain combinations of features are more often than others. Only a subspace of combined attribute values may actually be seen at a time. During different times when the data stream flows in, it is likely not only concepts are drifted, but the distribution criteria may change as well. In our experiments, we simulate the distribution criteria change by introducing another  $d$ -dimensional hyperplane,  $\sum_{i=1}^d b_i x_i = b_0$ . Only examples satisfying  $\sum_{i=1}^d b_i x_i \geq b_0$  can appear in data stream. Similarly, by changing the hyperplane coefficient  $b_i$ , we can simulate the distribution criteria drift as well. Since the two hyperplanes are implemented independently, we can simulate different type of streaming data by changing the concept plane coefficients ( $a_i$ ), the distribution plane coefficients ( $b_i$ ) only, or a combination of both. In our results, we choose to change both the concept and distribution which simulates the characteristics of real streaming data to the maximal extent. Weights  $a_i (1 \leq i \leq d)$  and  $b_i (1 \leq i \leq d)$  are initialized randomly in the range of  $[0, 1]$ . We choose the values of  $a_0$  and  $b_0$  so that both hyperplanes cut the multi-dimensional space into two sub-spaces of roughly the same volumes. That is,  $a_0 = \frac{1}{2} \sum_{i=1}^d a_i$ , and  $b_0 = \frac{1}{2} \sum_{i=1}^d b_i$ . Noise is introduced by randomly switching the labels of  $p\%$  of the examples. In our experiments, the noise level is chosen to be 5%. We simulate concept and distribution drifts through a series of parameters. Parameter  $k$  specifies the total number of dimensions whose weights are involved in changing. Parameter  $s_i \in \mathcal{R}$  specifies the magnitude of the changes (every  $N$  examples) for both weights  $a_i$  and  $b_i$  where  $1 \leq i \leq k$ . Weights change continuously, i.e.,  $a_i$  and  $b_i$  are adjusted by  $s_i/N$  after each example is generated. Furthermore, there is a possibility of 10% that the changes would reverse direction after every  $N$  examples are generated, that is,  $s_i$  is replaced by  $-s_i$  with probability 10%. Also, each time the weights are updated,  $a_0$  and  $b_0$  are updated accordingly. In our experiments, we have chosen a dimension of 5. Both the initial weights ( $a_i$ 's and  $b_i$ 's) for two hyperplanes are generated randomly. To simulate the data stream, we choose to change all 5 features at 10% increment for each chunk of data stream. The 10% increment is based on the initial training set. In other words, the first data stream chunk has 10% change from the training set and second data stream

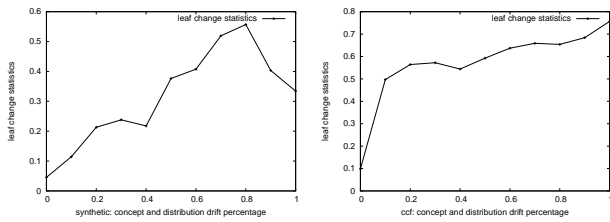


Figure 1: Correlation of change and leaf change statistics

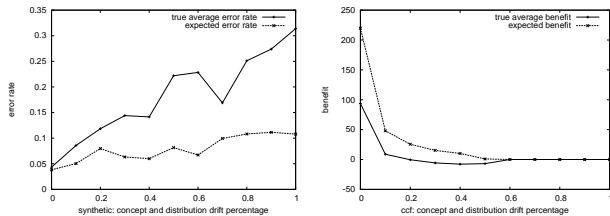


Figure 2: Correlation of change and expected loss

chunk has 20% from the training set. Though the magnitude of change is the same for both hyperplanes, but the exact change for each weight is generated randomly.

The second one is a *Credit Card Fraud Dataset*. The data set is sampled from credit card transaction records within a one year period and contains a total of 5 million transactions. Features of the data include the time of the transaction, the merchant type, the merchant location, past payments, the summary of transaction history, etc. We use the benefit matrix shown in the table below with the cost of disputing and investigating a fraud transaction fixed at  $cost = \$90$ . Let  $t(y)$  be the transaction amount of  $y$ . The following is the benefit matrix to compute the overall loss:

	predict <i>fraud</i>	predict $\neg$ <i>fraud</i>
actual <i>fraud</i>	$t(y) - \$90$	0
actual $\neg$ <i>fraud</i>	$-\$90$	0

The total benefit is the sum of recovered amount of fraudulent transactions less the investigation cost. To study the impact of concept drifts on the benefits, the data stream is ordered with decreasing transaction amount. In other words, the original decision tree is trained with transaction records of high transaction amount and the data stream has decreasing transaction amount. After the original data is sorted with decreasing transaction amount, it is split into 10 chunks of equal size. We use the first chunk (with highest transaction amount) for training, and the remaining 9 chunks to simulate data stream with evolving patterns. Since our data stream is sorted with decreasing transaction amount, the maximal amount of money that is recoverable will become less and less. For a chunk with transactions all less than  $\$90$ , there is no money to recover at all.

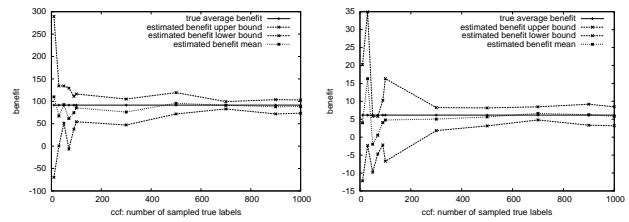


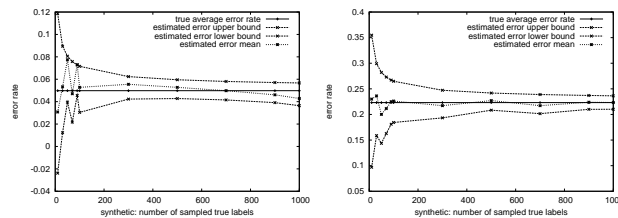
Figure 4: Loss estimation on credit card fraud data

### Change detection based on leaf changing statistics

The correlation of leaf change statistics and percentage of change in data on the two data streams is plotted in Figure 1. In each plot, the x-axis is the change percentage of the data stream. In other words, a change percentage of 20% means that 20% of the data in the data stream have pattern drifts from the training data of the original decision tree. The y-axis is the calculated percentage change of leaf statistics ( $PS$ ) as defined in Eq(2.2). The plot is computed by using the “same decision tree” to classify data streams with evolving drifts. As shown in the two plots, the  $PS$  statistics is well correlated with the change in the data stream for all three data streams. This empirical evaluation means that leaf changing statistics is a good indicator of the amount of change in the data stream.

**Change detection based on expected loss** The correlation between expected loss  $L_e$  (as defined in Eq(2.4)) and the true loss as a function of the percentage of change is shown in Figure 2. Please be noted that we have chosen to draw  $L_e$  instead of  $|L_e - L_a|$  since  $L_a$  or anticipated loss (as defined in Eq(2.3)) is the same value when normalized. In all of these plots, the x-axis is the percentage of change in the data stream and y-axis is the loss (or average benefits for credit card fraud dataset). As clearly shown in the plots, expected loss and true loss are positively correlated consistently when the changing ratio increases. This means that in practice, monitoring the change in expected loss is an effective heuristics to decide the likelihood of change in the data stream. The reason that the average benefits for the credit card dataset is decreasing is that we sort the data with decreasing transaction amount; and there are less and less transaction amount to be recovered. At the last data stream chunks where every transaction is less than  $\$90$  or the cost to dispute a fraud, the maximum amount of money that can be recovered is  $\$0$ . At some point, the original decision tree has too many false positives and their total benefits may fall below zero; and that’s exactly reconstruction is important and necessary.

**Loss Estimation** The plots on loss estimation are shown in Figures 3, and 4. For each dataset, we have chosen 2 different changing ratios in the data stream, one minor and one major change corresponding to the top and bottom plots. We have sampled up to 1000 instances in the data stream. There are 4 curves in each plot: the true loss on the data



Chunk size	loss	Change Ratio				
		0.1	0.3	0.5	0.7	0.9
5000	true	0.08	0.14	0.22	0.17	0.28
	estimate	0.09 ± 0.03	0.13 ± 0.03	0.23 ± 0.04	0.15 ± 0.04	0.30 ± 0.05
10000	true	0.07	0.18	0.22	0.41	0.30
	estimate	0.06 ± 0.02	0.21 ± 0.04	0.22 ± 0.04	0.43 ± 0.05	0.33 ± 0.05
15000	true	0.09	0.27	0.65	0.61	0.51
	estimate	0.10 ± 0.03	0.31 ± 0.05	0.64 ± 0.05	0.61 ± 0.05	0.52 ± 0.05

Figure 3: Loss estimation on synthetic data

stream, the estimated mean loss; and the upper and lower bounds at 99.7% confidence or three times the standard error. As we can see from the plots, the estimated mean loss at sample size of around 200 to 300 already give very close estimation to the true loss on the complete data stream. For the synthetic datasets, we also show in the table of Figure 3 the detailed results on loss estimation with chunk size of 5000, 10000 and 15000. Each test run is independent. In other words, the initial weights ( $a_i$ 's and  $b_i$ 's) for the three tests are different. The sampling size is 500 and the error bound has 99.7% confidence significance. The results in the table show that the estimation method is not sensitive to the size of the population.

#### 4 Related Work

Data stream processing has recently become a very important research domain. Much work has been done on modeling [1], querying [4], mining [3, 6], regression analysis [2], as well as clustering [5]. However, one thing in common among these previous work is the unrealistic assumption on the availability of labelled data.

#### 5 Conclusion

We proposed and evaluated a new framework of stream data mining with both synthetic and real-life datasets. Our new framework solves one important problem that the true labels of the data stream are not immediately available, but change detection need to be done immediately, and model reconstruction need to be done whenever estimated loss is higher than tolerable maximum. Our framework extends the traditional classification tree algorithm by using no true class labels or investigating the true labels of a small number of instances chosen from the new data stream. We define two statistics on the decision tree that are likely correlated with change and loss due to pattern drifts in the data stream.

Both statistics do not use any true class labels. We then introduce statistical sampling based method to estimate the range of true loss of the decision tree on the data stream. Experimental studies have found that i) the two statistics are very well correlated with the amount of change in the data stream; the guessed loss without any true labels are accurate estimates of the actual loss, ii) with approximately a few hundred labeled instances, the statistically estimated loss range is very close to the true value of the complete data stream.

**Future Work** In our immediate work, we propose a few methods to reconstruct the original decision tree with limited number of examples.

#### References

- [1] B. Babcock, S. Babu, M. Datar, R. Motawani, and J. Widom. Models and issues in data stream systems. In *ACM Symposium on Principles of Database Systems (PODS)*, 2002.
- [2] Y. Chen, G. Dong, J. Han, B. W. Wah, and J. Wang. Multi-dimensional regression analysis of time-series data streams. In *Proc. of Very Large Database (VLDB)*, Hongkong, China, 2002.
- [3] P. Domingos and G. Hulten. Mining high-speed data streams. In *Int'l Conf. on Knowledge Discovery and Data Mining (SIGKDD)*, pages 71–80, Boston, MA, 2000. ACM Press.
- [4] L. Gao and X. Wang. Continually evaluating similarity-based pattern queries on a streaming time series. In *Int'l Conf. Management of Data (SIGMOD)*, Madison, Wisconsin, June 2002.
- [5] S. Guha, N. Milshra, R. Motwani, and L. O'Callaghan. Clustering data streams. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 359–366, 2000.
- [6] Haixun Wang, Wei Fan, Philip Yu, and Jiawei Han. Mining concept-drifting data streams with ensemble classifiers. In *Proceedings of ACM SIGKDD International Conference on knowledge discovery and data mining (SIGKDD2003)*, pages 226–235, 2003.