Systematic Data Selection to Mine Concept-Drifting Data Streams

Wei Fan
IBM T.J.Watson Research
19 Skyline Drive
Hawthorne, NY 10532, USA
weifan@us.ibm.com

ABSTRACT

One major problem of existing methods to mine data streams is that it makes ad hoc choices to combine most recent data with some amount of old data to search the new hypothesis. The assumption is that the additional old data always helps produce a more accurate hypothesis than using the most recent data only. We first criticize this notion and point out that using old data blindly is not better than "gambling"; in other words, it helps increase the accuracy only if we are "lucky." We discuss and analyze the situations where old data will help and what kind of old data will help. The practical problem on choosing the right example from old data is due to the formidable cost to compare different possibilities and models. This problem will go away if we have an algorithm that is extremely efficient to compare all sensible choices with little extra cost. Based on this observation, we propose a simple, efficient and accurate cross-validation decision tree ensemble method.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications - Data Mining

General Terms

Algorithms

Keywords

data streams, concept-drift, decision trees

1. INTRODUCTION

One of the recent challenges facing traditional data mining methods is to handle real-time production systems that produce large amount of data continuously at unprecedented rate and with evolving patterns. Traditionally, due to limitation of storage and practitioner's ability to mine huge amount of data, it is a common practice to mine a subset of data at preset frequency. However, these solutions have been shown to be ineffective due to possibly oversimplified model as a result of sub-sampling as well as dynamically unpredictable evolving pattern of the production data. Knowledge discovery on data streams has become a research topic of growing interest. Much work has been done on modeling [Babcock et al., 2002], querying [Babu and Widom, 2001, Gao and Wang, 2002, Greenwald and Khanna, 2001], classification [Domingos and Hulten, 2000, Hulten et al., 2001, Street and Kim, 2001, Wang et al., 2003, Fan et al., 2004], regression analysis [Chen et al., 2002], clustering [Guha et al., 2000] as well as visualization [Aggarwal, 2003]. The fundamental problem we need to solve is the following: given an infinite amount of continuous measurements, how do we model them in order to capture possibly time-evolving trends and patterns in the stream, compute the optimal model and make time critical decisions?

At the present time, many existing methods to mine data streams "blindly" reuse some amount of old data to combine with new data to construct the models. The generally conceited reason on why to use old data is the hope to improve the current model's accuracy on the new data. There are mainly two approaches. One approach assigns a decreasing weight to older examples. A simpler approach always uses data from a fixed number of periods. For example, in [Hulten et al., 2001], they refine a decision tree by continuously incorporating new data from the data stream. In order to handle concept-drifts, they have chosen to retire old examples at a preset "fixed rate" besides discarding and re-growing subtrees under a node. Since old data is discarded at a fixed rate (no matter if they represent the changed concept or not), the learned model is supported arbitrarily much more by the current snapshot - a possibly very small amount of data. As a matter of fact, it is shown in [Hulten et al., 2001] that the prediction error of the tree rise quickly when the concept drift amplifies. Ideally, the prediction error should not be correlated to the amount of concept drift. In [Wang et al., 2003], they construct a weighted ensemble of classifiers. One classifier in the ensemble is trained from the most recent data chunk, and the others are trained from some old data chunks. Both the theorem and empirical analysis of [Wang et al., 2003] conclude that when there is concept drift, from the "same" data chunks (new and old), the weighted ensemble is more accurate than a single classifier trained from exactly the same amount of data. However, it didn't draw any conclusion about the relative accuracy of models trained from "different number of data chunks" or different amount of old data. In other words, it still remains an open problem whether it is more accurate to train from the new data only or train from old data plus with some amount of old data (and how much old data).

The unselective use of old data definitely helps improve model accuracy if there is *no* conceptual change in the data stream and new data stream is insufficient by itself. However, when there is no

KDD'04, August 22-25, 2004, Seattle, Washington USA

conceptual change, there may not be any utility to re-learn a new model unless the old model is trained from insufficient data. On the other hand, when there is indeed conceptual change, i.e., the underlying model in the new data stream is different from previous model, using older data unselectively is not better than gambling. In this situation, using old data unselectively helps only if the new concept and old concept still have some consistencies and the amount of old data chosen arbitrarily just happen to be right. The unrealistic approach would be to first know if the data has concept drift and if the data is sufficient by itself. Based on the combinations of whether there is concept drift and whether the data is sufficient, we would decide the correct decisions. Detection methods for both concept drift and data sufficiency could be proposed, but they could be wrong. Even if one is wrong, we will make the wrong decision. More importantly, a requirement for stream mining completely invalidate the need for sufficiency detection is that even if the data is insufficient for learning, we still need to find a model to best fit it.

All these problems will go away, if we can find an algorithm that is extremely efficient in training; we apply this extremely efficient approach on the data to compare all sensible choices using cross-validation, systematically select data, and make the data "speak for themselves." These sensible candidates could include new model trained from new data, model trained from new data combined with carefully selected old data, old model updated with new data, and old model itself. Besides comparing these choices, we also need a statistically reliable method to carefully select old data whenever necessary. The correct choice ought to be made by using cross-validation instead of making some "data-blind" assumptions. The basic framework proposed in this paper is based on this statistical test. Its implementation is based on an efficient multiple decision tree algorithm.

To solve the problem on how to systematically select old data to mine concept-drifting data streams, we propose a cross-validation decision tree ensemble approach. In the first step, the algorithm detects all features with information gain. In the second step, its builds multiple decision trees by randomly choosing from those features with information gain and ignore the irrelevant features. Discrete features can appear only once in a decision path, starting from the root of the tree to the current node. Continuous features can appear multiple times but with a different splitting point each time this feature is chosen. Internal nodes of the tree keep class distribution statistics. To classify an unknown instance, each decision tree outputs a membership probability (e.g, $p(fraud|\mathbf{x})$ or the probability that $\mathbf{x}$ is a fraud) computed at each leaf node level using the stored class distribution statistics. The probability outputs of multiple decision trees on the same example are then averaged as the final membership probability estimation. In order to make an optimal decision, the estimated posterior probability and a given loss function are used jointly in order to minimize the expected loss. For example, under traditional 0-1 loss, if the averaged probability $p(fraud|\mathbf{x}) > 0.5$, the best prediction is to predict $\mathbf{x}$ as fraud.

We justify our claim that using old data unselectively is like gambling by running on a synthetic dataset as well as credit card fraud dataset. We evaluated the proposed cross-validation decision tree ensemble and compared the results with some other models trained with either new data only or new data plus some ad hoc amount of old data.

2. ISSUES WITH DATA STREAM

There are two major issues with an incoming data stream, possible concept-drift and data insufficiency.

2.1 Concept Drift

Assume that $y = f(\mathbf{x})$ is the underlying true model that we aim to model. In order to do so, some number of training instances are randomly sampled, $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$. Most models are deterministic, i.e., for the same example, $f(\mathbf{x})$ produces the same prediction at different time. Some models can also be stochastic; in other words, for the same example, $f(\mathbf{x})$ may produce different class labels at the different times. For stochastic problems, the best we can do is to predict the label that minimizes a given loss function. Since in most applications, we don't actually know the true model. We normally discuss the optimal hypothesis or a hypothesis that minimizes a given loss function as the ultimate goal. A *true* model can be stochastic, however, an *optimal* model is generally deterministic.

We generally describe the training data of data streams as chunks of labeled data at different time stamps. S_i is the data received at time stamp i and $FO_i(\mathbf{x})$ is its optimal model. Assume that $FO_{i-1}(\mathbf{x})$ is the older optimal hypothesis at the previous time stamp $i-1$. We say that there is concept drift from time stamp $i-1$ to time stamp i, if there are inconsistencies between $FO_{i-1}(\mathbf{x})$ and $FO_i(\mathbf{x})$. Formally, "under the same loss function", there exists $\mathbf{x}$ such that $FO_{i-1}(\mathbf{x}) \neq FO_i(\mathbf{x})$. If $\mathbf{x}$ is taken randomly from the universe of valid examples, with probability τ, $FO_{i-1}(\mathbf{x}) \neq FO_i(\mathbf{x})$. We call τ the rate of concept change.

2.2 Data Sufficiency

There is no formal definition of data sufficiency. In statistical sampling, we say that a data sample is sufficient if the observed statistics, such as sample mean, sample total, and sample proportion, have a variance smaller than predefined limits with high confidence. For example, under normal distribution, 99.7% confidence is at 3 times the standard variance interval. In practical terms of machine learning and data mining, a dataset is considered sufficient if adding more data into it will not increase the generalization accuracy. How much data is sufficient really depends on the combination of dataset, chosen learning algorithms and application related loss function. Given an infinite amount of training data, determining the sufficiency amount can be formidably expensive especially for hill-climbing based methods such as decision tree learner. One important requirement for streaming mining that completely invalidate the need for sufficiency test is that even if the dataset is insufficient, we still need to train a model that can best fit the changing data.

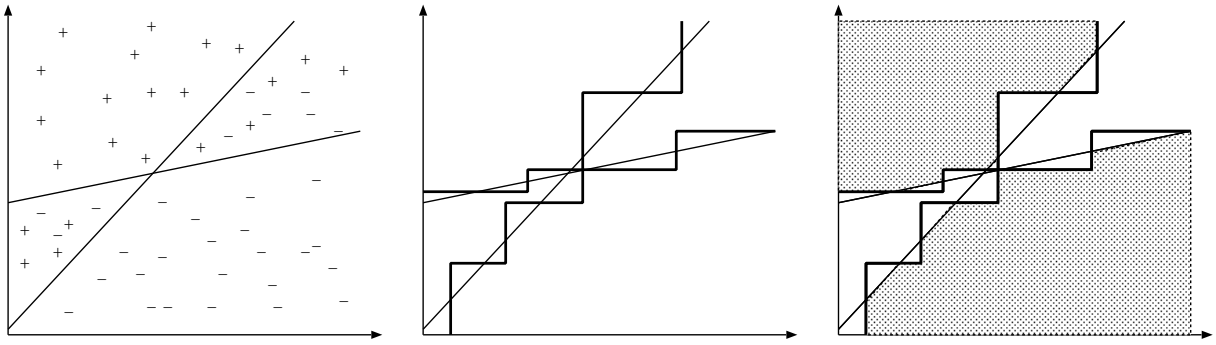
3. WILL OLD DATA REALLY HELP?

We analyze the effect of old data under two situations. The first situation is that the underlying model does not change. Obviously, older data will help improve accuracy if the recent data is insufficient and the combined old data and most recent data doesn't overfit the inductive learner. One important question to ask is: if the model doesn't change, what is the utility to update and train a new model? The answer is: it is only useful to combine older and new data to retrain a model, if the older data is insufficient by itself.

The second situation is that the underlying model does change. We discuss how the previous data chunks $SP = S_1 \cup \ldots \cup S_{i-1}$ might help to improve a model trained only from the most recent data chunk S_i. The data in SP can be one of the following three major categories.

- The first type of data are those where $FO_{i-1}(\mathbf{x}) \neq FO_i(\mathbf{x})$. They are a superset of the τ inconsistencies in the universe of all examples. The reason is that FO_i and FO_{i-1} are optimal models, but not perfect models, and they both make mistakes.

Figure 1: How to choose from old data



(a) Evolving hyperplane (b) Their optimal model (c) Those data that help

- The second type of data are those that both hypotheses make the correct prediction, i.e., $FO_{i-1}(\mathbf{x}) = FO_i(\mathbf{x}) = y$.
- The third type of examples are those data that both models make the same wrong predictions, i.e., $\Big(FO_{i-1}(\mathbf{x}) = FO_i(\mathbf{x})\Big) \neq y$. Obviously, τ inconsistent examples will not help. It will only cancel out the changing concept.

The only portion of data that may help is the portion that $FO_{i-1}(\mathbf{x})$ and $FO_i(\mathbf{x})$ agree and they both make the correct prediction. This is the portion of the data that doesn't change its concept. Please note that the third category of the data where both models agree but their predictions are wrong cannot be determined if they will help or not. Since these portion of data may be conceptual change (hence the inconsistency portion) or due to the learning error of the algorithm. Thus, when pattern does change, using older data unselectively can be dangerous and misleading. The only data that will help is those that are still consistent under the evolved models.

We illustrate the idea through a simple hyperplane example. Figure 1 shows an evolving hyperplane. Figure 1(a) shows the true model of the evolving hyperplane. An example is positive (+) if it is above the hyperplane; otherwise, it is negative (-). Although, it actually makes no difference in distinguishing which is earlier and which is later in its evolving process, we assume that the "flatter" hyperplane is earlier and the more "vertical one" is later. In Figure 1(a), we also plot both + and - instances. Obviously, the consistent portion in the universe of instances are the top left (all +) and bottom right (all -) areas. These are the only examples that one hyperplane can help the other. The two smaller areas on the bottom left and top right are inconsistent areas, where one hyperplane predicts + and the other predicts -. However, we do not know and usually will never know these true models. The best we can to is to find an optimal model. In Figure 1(b), we draw the decision tree optimal model (which are interpolated straight lines) for both hyperplanes. In Figure 1(c), the shaded areas are those that $FO_{i-1}(\mathbf{x}) = FO_i(\mathbf{x}) = y$, and they are a subset of the "agreement" between the true models. Examples from these shaded areas will help build optimal model for the newly evolved concept.

4. SIFTING THROUGH OLD DATA

So far, we have discussed the issues of concept drift and data insufficiency that are possibly present in data streams. We have also discussed the problem of using older data unselectively as well also what examples in the older data that may help to construct a better model. In this section, we first discuss a theoretically sound, however impractical method and then propose a practically useful framework as well as one efficient implementation.

4.1 Optimal Models

There are a large number of possibilities that can happen when mining data streams. To clearly define our scope, we first make some reasonable assumptions. We assume that training data is collected without any known prior bias. In other words, if $\mathbf{x}$ has probability of p to be seen in the universe of valid examples, it has the same probability p to be sampled without replacement from the universe to form the training set. It is important to point out that we clearly exclude the rare and unrealistic situation that the sampling probability of $\mathbf{x}$ is significantly different from its true probability to appear in the data stream. One such an example is one data chunk with mostly positive examples and the second one with mostly negative examples.

Before we go into the details of the proposed algorithm, we enumerate all situations that we can think of and discuss the best choice in each case and how to find the optimal model. The conclusion that we will draw from this enumeration is that: "although there are a lot of possibilities, but if we have an extremely efficient learning algorithm that works the same way under all conceivable possibilities, it will allow us to compare all sensible choices in a reasonable amount of time and make the best choice."

The two main themes of our comparison is on possible data insufficiency and concept drift. We start from simple cases.

- **New data is sufficient by itself *and* there is no concept drift.** The optimal model should be the one trained from the new data itself since new data is sufficient. The older model may also be an optimal model if it is trained from sufficient data. However, the tricky issue is that we do not know and will usually never know if the data is indeed sufficient and the concept indeed remains the same. However, it doesn't hurt to train a new model from the new data, a new model from combined new data and old data, and compare with the original older model to choose the more accurate one if the learning cost is affordable.
- **New data is sufficient by itself *and* there is concept drift.** The optimal model should be the one trained from the new data itself. Similar to the previous situation, we do know and will never know if the data is indeed sufficient and the con-

cept indeed remains the same. Ideally, we should compare a few sensible choices if the training cost is affordable.

- **New data is insufficient by itself *and* there is no concept drift.**. If the previous data is sufficient, the optimal model should be the existing model. Otherwise, we should train a new model from new data plus existing data and choose the one with higher accuracy.
- **New data is insufficient by itself *and* there is concept drift.** Obviously, training a new model from new data only doesn't return the optimal model. However, choosing old data unselectively, as shown previously, will only be misleading. The correct approach is to choose only those examples from previous data chunks that have consistent concept with the new data chunk and combine those examples with the new data

4.2 Computing optimal models

We notice that the optimal model is completely different under different situations. The choice for optimal model completely depends on if the data is indeed sufficient and if there is indeed concept drift. The ideal solution would be to compare a few plausible optimal models statistically, and choose the one with the highest accuracy. In the end, the target of stream mining is to find a model that best fit the new data no matter there is a concept drift or the data is sufficient. Next we discuss a conceptual framework for this approach. We will propose an efficient algorithm to implement this framework afterwards. To clarify some notation conventions, $FN(\mathbf{x})$ denotes a new model trained from recent data. $FO(\mathbf{x})$ denotes an optimal model finally chosen after some statistical significance tests. i is the sequence number of each sequentially received data chunk.

1. Train a model $FN_i(\mathbf{x})$ from the new data chunk S_i only.
2. Assume that D_{i-1} is the dataset that trained the most recent "optimal" model $FO_{i-1}(\mathbf{x})$. It is important to point out that D_{i-1} may not be the most recent data chunk S_{i-1}. D_{i-1} is collected iteratively throughout the streaming data mining process. The exact way how D_{i-1} is collected will be clear next. We select these examples from D_{i-1} that both the trained new model $FN_i(\mathbf{x})$ and the recent optimal model $FO_{i-1}(\mathbf{x})$ make the correct prediction. We denote these chosen examples as s_{i-1}. In other words, $s_{i-1} = \{\forall(\mathbf{x}, y) \in D_{i-1}, \text{such that}, (FN_i(\mathbf{x}) = y) \wedge (FO_{i-1}(\mathbf{x}) = y)\}$.
3. Train a model $FN_i^+(\mathbf{x})$ from the new data *plus* the selected data in the last step or $S_i \cup s_{i-1}$.
4. Update the most recent model FO_{i-1} with S_i and call this model $FO_{i-1}^+(\mathbf{x})$. To update a model, we keep the "structure" of the model and update its internal statistics. Using decision tree as an example, every example in S_i is "classified" or sorted to each leaf node. The statistics, i.e., the number of examples belonging to each class label, are updated. Obviously, the training set for $FO_{i-1}^+(\mathbf{x})$ is $D_i \cup S_i$.
5. Compare the accuracy of all four models ($FN_i(\mathbf{x})$, $FO_{i-1}(\mathbf{x})$, $FN_i^+(\mathbf{x})$), and $FO_{i-1}^+(\mathbf{x})$) using "cross-validation" and choose the one that is the most accurate and we name it $FO_i(\mathbf{x})$.
6. D_i is the training set that computes $FO_i(\mathbf{x})$. It is one of $S_i, D_{i-1}, S_i \cup s_{i-1}$, and $S_i \cup D_{i-1}$.

For the moment, we address how the above framework finds the optimal model under all four previously discussed situations. Later, we will propose an extremely efficient algorithm to implement this "seemingly" expensive process.

1. **New data is sufficient by itself *and* there is no concept change**. Conceptually $FN_i(\mathbf{x})$ should be the optimal model. However, $FN_i^+(\mathbf{x})$, $FO_{i-1}(\mathbf{x})$ and $FO_{i-1}^+(\mathbf{x})$ could be its close match since there is no concept change.
2. **New data is sufficient by itself *and* there is concept change**. Obviously, $FN_i(\mathbf{x})$ should be the optimal model. However $FN_i^+(\mathbf{x})$ could be very similar in performance to $FN(\mathbf{x})$.
3. **New data is insufficient by itself *and* there is no concept change**. The optimal model should be either $FO_{i-1}(\mathbf{x})$ or $FO_{i-1}^+(\mathbf{x})$.
4. **New data is insufficient by itself *and* there is concept change**. The optimal model should be either $FN_i(\mathbf{x})$ or $FN_i^+(\mathbf{x})$.

4.3 Discussion

There are two important questions about this data selection process. One important question is that if more data from the history will help or not. Formally, in our algorithm, we *only* consider to include data from D_{i-1}, or the most recent chunk. The question is if the data from $(\bigcup_{j=1}^{i-2} D_j) - D_{i-1}$ will help or not. The answer is: it may or may not. Even if it may, it may not help much. First of all, one empirical assumption is that most recent data is closer to data of its closest periods. Even though we completely don't count on this, it is a good argument against using data that are too old. Second of all, the amount of data from the past cannot be overdone. When it is overdone, the learner may overfit on the unchanging part of the new concept and ignore the new part. In practical sense, choosing the exact number of old examples to have the maximal accuracy is not feasible. It is a combinatorial problem and the added benefits is hard to justify the cost to do so.

The second question to ask is "will the training data D_i become unnecessarily large?". The answer is no. D_i only grows in size (or includes older data) if and only if the additional data helps improve accuracy. In other words, D_i only grows in size whenever necessary.

5. CROSS VALIDATION DECISION TREE ENSEMBLE

We propose an efficient algorithm based on decision tree ensemble to "sift through" old data and combine with new data to construct the optimal model for evolving concept. The basic idea is to train a number of random and uncorrelated decision trees. Each decision tree is constructed by randomly selecting available features. The structure of the tree is uncorrelated. Their only correlation is on the training data itself.

5.1 Training and Testing

The algorithm first sequentially scans the complete dataset once and finds out all features with information gain. To avoid noise in the data, we provide a parameter ϵ as its "cut off" value. After finding out f good features, it builds N "random decision trees" from only these f good features. Features without information gain will never be used. At each step, it chooses a "remaining" feature randomly. Each discrete feature can be used at most once in a particular decision path of the tree starting from the root of the tree. Each continuous feature can be chosen multiple times on the same decision path, but with a randomly chosen splitting threshold each time this continuous feature is chosen. The splitting threshold is a random value within the max and min of that feature. To handle missing values in the training data, each example $\mathbf{x}$ is assigned an initial weight of $w = 1.0$. When missing feature value is encountered, the current weight of $\mathbf{x}$ is distributed across its children nodes. If the prior distribution of known values are given, the

weight is distributed in proportion to this distribution. Otherwise, it is equally divided among the children nodes. The tree stops growing a branch if there are no more examples passing through that branch.

To classify an example, raw posterior probability is required. If there are n_c examples out of n in the leaf node with class label c, the probability that $\mathbf{x}$ is an example of class label c is $P(c|\mathbf{x}) = \frac{n_c}{n}$. Some leaf node, especially a branch from a discrete feature test, may not have any examples. When this happens, it carries the probability from its parent node. Some examples (such as those with missing values) will be classified by multiple decision paths. We count the number of examples in each leaf belonging to different classes along with their weights. Assume $\mathbf{x}$ is classified by paths A and B with weights 0.3 and 0.7 respectively. The leaf under path A has 100 out of 2000 examples belonging to class c. Similarly, path B has 200 out of 1000 examples belonging to class c. Then the probability that $\mathbf{x}$ is an instance of class $\mathbf{x}$ is simply, $P(c|\mathbf{x}) = \frac{0.3 \cdot 100 + 0.7 \cdot 200}{0.3 \cdot 2000 + 0.7 \cdot 1000}$. Each tree computes a posterior probability for an example and the probability outputs from multiple trees are averaged as the final posterior probability of the ensemble.

To make a decision, application specific loss function is required. For a binary problem under 0-1 loss, if $P(y|\mathbf{x}) > 0.5$, the best prediction is y. For cost-sensitive application such as credit card fraud detection, assuming that the cost to investigate a fraud is \$90 and $Y(\mathbf{x}$ is the amount of the transaction. We predict fraud if and only if $P(fraud|\mathbf{x}) \cdot Y(\mathbf{x}) > \$90$. In other words, we only save money if and only if the expected loss is more than the cost of doing business.

5.2 Cross Validation

We propose to use the decision tree ensemble trained from the training set for cross-validation test. Assuming that n is the size of the training set, n-fold cross validation leaves one example $\mathbf{x}$ out and uses the remaining $n-1$ examples to train a model and classify on the left-out example $\mathbf{x}$. If n is non-trivial, the exclusion of $\mathbf{x}$ is very unlikely to change the subset of features having information gain (those found out in the first step to train the decision tree ensemble). With the same seed, the random number function generates the same sequence of numbers. In this case, the structures of the trees remain the same even when $\mathbf{x}$ is excluded from the training set. The only difference is the class distribution statistics recorded in the nodes. Any node that classifies $\mathbf{x}$ will have one fewer example for the true class label of $\mathbf{x}$. When we compute the probability for the excluded $\mathbf{x}$ under n-fold cross validation using the original decision tree ensemble, we need to compensate this difference.

Assuming that we have two class labels, either fraud or non-fraud, to compute the probability of the excluded $\mathbf{x}$ being fraudulent is simply

$$\begin{cases} \frac{n_{fraud}-1}{n_{fraud}-1+n_{normal}} & \text{if } \mathbf{x} \text{ is indeed a fraud} \\ \frac{n_{fraud}}{n_{fraud}+n_{normal}-1} & \text{if } \mathbf{x} \text{ is a normal transaction} \end{cases}$$

The minimal number of examples in any node is generally set to 2. If a node originally has only 2 examples in total, the parent node is used to compute the probability for cross-validation to avoid over estimation. It is important to subtract 1 based on $\mathbf{x}$'s true class label. If we did not subtract 1 in the formula, the probability for being a member of the positive class would be over-estimated for true positives and under-estimated for negatives positives. For example, a leaf node has 10 examples with 7 frauds and 3 non-frauds. If we did not subtract 1, the probability for being a fraud would be $\frac{7}{10} = 0.7$. In fact, the probability to be fraud for a true fraud transaction is $\frac{7-1}{10-1} = 0.67$, and the probability to be fraud for a normal transaction is $\frac{7}{10-1} = 0.78$.

5.3 Update Decision Tree Ensemble

In Section 4.2, we discussed $FO^{+}_{i-1}(\mathbf{x})$ or the old model updated with new streaming data. To update the decision tree ensemble is similar to classification. For every example in the new data chunk, we simply increment the class label count in each classifying node.

5.4 Training and Memory Efficiency

The total time to choose the right data and compute the optimal model includes the time to compute a new ensemble from the new data chunk, update the recent ensemble, train a new ensemble from incremented dataset, as well as compare four candidate models on the new data. Obviously, in our particular implementation, comparing candidate models using n-fold cross-validation is the same as classifying the training dataset. Classification with decision tree is an efficient procedure. Updating the recent ensemble is the same as classifying on the new data. Computing information gain of features from the complete training set requires grouping of different feature values multiple times for all features and is an expensive procedure. However, this is done only once for multiple CV decision trees. We construct each tree by randomly selecting from the pool of candidate good features and do not compute any information gain; the only operation is to group training items once at each node. The training for multiple CV decision trees is an efficient procedure, especially when there are a lot of features or the training set contains a large number of data items.

Each tree in the CV decision tree ensemble is very likely larger in size than a best tree built by checking information gain at each step. The whole purpose of information gain is to find a smaller tree. In our experimental study, we will record the size of each tree in the ensemble and compare it with the single best tree trained from the same dataset.

6. EXPERIMENT

We conducted extensive experiments on both synthetic and real life data streams. Our goals are to demonstrate that the proposed method can efficiently and effectively compare all sensible choices and choose to build the most accurate model under all combinations of situations. Our framework was modified from C4.5 classification tree. We always compute 10 trees for each CV decision tree ensemble. The threshold for information gain, ϵ, is set to be 0.001. We have used both 0-1 loss and cost-sensitive loss to evaluate performance.

6.1 Streaming Data

Synthetic Data. We create synthetic data with drifting concepts based on a moving hyperplane. A hyperplane in d-dimensional space is denoted by equation: $\sum_{i=1}^{d} a_i x_i = a_0$. We label examples satisfying $\sum_{i=1}^{d} a_i x_i \geq a_0$ as positive, and examples satisfying $\sum_{i=1}^{d} a_i x_i < a_0$ as negative. Hyperplanes have been used to simulate time-changing concepts because the orientation and the position of the hyperplane can be changed in a smooth manner by changing the magnitude of the weights [Hulten et al., 2001].

We generate random examples uniformly distributed in multi dimensional space $[0,1]^d$. Weights a_i $(1 \leq i \leq d)$ are initialized randomly in the range of $[0,1]$. We choose the value of a_0 so that the hyperplane cuts the multi-dimensional space in two parts of the same volume, that is, $a_0 = \frac{1}{2}\sum_{i=1}^{d} a_i$. Thus, roughly half of the examples are positive, and the other half negative. Noise is intro-

duced by randomly switching the labels of $p\%$ of the examples. In our experiments, the noise level $p\%$ is set to 5%.

We simulate concept drifts by a series of parameters. Parameter k specifies the total number of dimensions whose weights are changing. Parameter $t \in \mathcal{R}$ specifies the magnitude of the change (every N examples) for weights $a_1, \cdots, a_k$, and $s_i \in \{-1, 1\}$ specifies the direction of change for each weight a_i, $1 \leq i \leq k$. Weights change continuously, i.e., a_i is adjusted by $s_i \cdot t/N$ after each example is generated. Furthermore, there is a possibility of 10% that the change 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, we recompute $a_0 = \frac{1}{2}\sum_{i=1}^{d} a_i$ so that the class distribution is not disturbed.

Credit Card Fraud Data. We use real life credit card transaction flows for cost-sensitive mining. 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 $cost = \$90$, and let $t(y)$ be the transaction amount of y. The following is the benefit matrix to compute the overall loss:

	predict $fraud$	predict $\neg fraud$
actual $fraud$	$t(\mathbf{x})$ - \$90	0
actual $\neg fraud$	-\$90	0

The total benefit is the sum of recovered amount of fraudulent transactions less the investigation cost. To maximize benefits, we only predict fraud if and only if $p(fraud|\mathbf{x}) \cdot t(\mathbf{x}) > \$90$. To study the impact of concept drifts on the benefits, we derive stream by ordering the records with increasing transaction amount. In other words, the original decision tree is trained with transaction records of low transaction amount and the data stream has increasing transaction amount. It is then split into multiple chunks of equal size.

Donation Dataset. The third one is the famous donation dataset that first appeared in KDDCUP'98 competition. Suppose that the cost of requesting a charitable donation from an individual $\mathbf{x}$ is \$0.68, and the best estimate of the amount that $\mathbf{x}$ will donate is $Y(\mathbf{x})$. Its benefit matrix (converse of loss function) is:

	predict $donate$	predict $\neg donator$
actual $donate$	Y(**x**) - \$.0.68	0
actual $\neg donate$	-\$0.68	0

The accuracy is the total amount of received charity minus the cost of mailing. Assuming that $p(donate|\mathbf{x})$ is the estimated probability that $\mathbf{x}$ is a donor, we will solicit to $\mathbf{x}$ iff $p(donate|\mathbf{x}) \cdot Y(\mathbf{x}) > 0.68$. The data has already been divided into a training set and a test set. The training set consists of 95412 records for which it is known whether the person made a donation and how much the donation was. The test set contains 96367 records for which similar donation information was not published until after the KDD'98 competition. The feature subsets (7 features in total) were based on the KDD'98 winning submission. To estimate the donation amount, we employed the multiple linear regression method. The donation dataset has very small number of donors (less than 5% in total). It is difficult to use the same "sorting" approach as the credit card dataset. Instead, we shuffle the dataset 5 times. From each shuffled dataset, we sequentially sample different number of examples.

6.2 Experiment Setup

We have a number of dimensions to compare and evaluate.

1. We first need to justify our claims that using old data unselectively is the same as gambling; sometimes, it may increase accuracy, and other times, it may decrease accuracy.
2. The most important set of results is to show that the proposed framework and its CV decision tree ensemble implementation can indeed efficiently and accurately choose the most accurate sensible model under all different kind of situations. We evaluate both the accuracy and training time and memory consumption.
3. The accuracy of the n-fold cross validation is an important issue. We study if the estimated probability by n-fold cross validation is close the estimated probably on an unseen dataset.
4. Since in reality, chunksizes can be arbitrarily small, to show that the decision tree ensemble is resilient to data insufficiency, we measured the change in accuracy with increasing training data size. As a comparison, we show the accuracy result of single best decision trees.

6.3 Evidence of using data unselectively may hurt

We use both the hyperplane synthetic dataset and credit card dataset to illustrate that using old data unselectively may hurt. It is important to point out that we didn't run experiment with no conceptual change. It is obvious that when there is no conceptual change, using old data will most likely help unless it overfits the learner. The moral of this experiment is to show that when the concept does change, it really depends on the combination of chosen method, changing degree and, data size to decide if using old data unselectively will help increase the accuracy.

We ran a series of experiments with increasing data chunksize. For each chosen data chunksize, we construct a series of models using different amount of training data.

- **Use new data only**: G_1 is the single best unpruned C4.5 tree trained from the new data chunk only without using any previous data.
- **Different ways to use old data unselectively**: GA is a single decision tree trained from the complete dataset using all available data from the very beginning of the data stream. VFDT [Domingos and Hulten, 2000] builds a decision tree virtually the same as GA. G_i ($i \geq 2$) is a single decision tree trained from the the new data chunk plus the most recent $i-1$ data chunks. The CVFDT algorithm [Hulten et al., 2001] trains a model similar to G_i's. E_i is a decision tree ensemble trained from the same data chunks as G_i. Each tree in the ensemble is trained from one data chunk. A weight is assigned to each tree in the ensemble that is correlated to its accuracy on the new data chunk [Wang et al., 2003].

The results for the synthetic dataset with dimension $d = 10$ are in Table 1 under the columns of "use new data only" and "different ways to use old data unselectively". In our experiments, we incremented the data chunksize by 250. The concept drift is simulated by various parameters: the number of dimensions with changing weights ranges from 2 to 8, and the magnitude of the change t ranges from 0.1 to 1.0 for every 1000 examples. Each result is the average of different conceptual change with the same chunksize. We bold face a result if it is better than G_1, the model computed only from the new data. It is important to point out that results of "different ways to use old data unselectively" for chunksizes={250,500,750,1000} were reported in our previous work [Wang et al., 2003]. The brand new results are those in column "use old data selectively" as well as additional chunksizes = {1500, 2000,

5000, 20000} that were not tested previously. We use the same random seed sequence as in our previous work to generate the streaming data. For analytic purposes, we find out how much data is approximately sufficient for a fixed hyperplane with dimension $d = 10$. We increased the amount of data by 100 instances, reconstructed a new single unpruned C4.5 tree at every increment, and found that after about 2000, the error rate remained between 4% and 7%. In other words, a training set with size 2000 is probably sufficient. From the results in Table 1, when the data chunk is ≤ 1000, any methods that use old data help. After the size of data chunk is more than 1000, the difference between using new data only G_1 and all other models using some amount of older data unselectively (GA, E_i's and G_i's, $i \geq 2$), have all started to decrease. When the data chunk has 2000 instances, the advantage of using any amount of old data diminishes to nearly none. When the chunksize increases further more (i.e., 5000 and 20000), any methods that use any amount of old data unselectively are only detrimental; none of the methods that uses old data unselectively is more accurate than the simple model trained from new data itself.

Similar phenomenon is observed in the credit card dataset sorted with increasing transaction amount, as shown in Table 2. For analytic purpose, we find out sufficient training size by shuffling the data set completely, using 10% for testing, incrementing the training set by 1000 examples, and training a new unpruned decision tree at each increment. After approximately 15000 examples, the benefit ($) or accuracy on the 10% test data stabilizes. From the results in Table 2, we observe that it helps to use old data unselectively only when the chunksize is $\leq$ 24000. After 24000, using old data unselectively starts to drive down the overall dollar benefits.

6.4 Result of CV Decision Tree Ensemble

The results of cross-validation decision tree ensemble that systematically selects data to build optimal model are shown in the last column "Use old data selectively" of Tables 1 to 3. It is important to emphasize that the optimal CV decision tree model is chosen by comparing the accuracy of four models: a new CV decision tree trained from the new data chunk only ($FN_i(\mathbf{x})$), updated CV decision tree ($FO^+_{i-1}(\mathbf{x})$), a new CV decision tree trained from the new data chunk plus selected consistent examples that trained the most recent optimal CV decision tree ($FN^+_i(\mathbf{x})$), and the most recent optimal CV decision tree itself ($FO_{i-1}(\mathbf{x})$).

There are two important observations from the results of the synthetic dataset in Table 1. First, the error rate of the CV decision tree ensemble (under "Use old data selectively") is significantly lower than any other methods in comparison, either training from the new data only or some unselective use of old data. The difference is particularly big when the chunksize of the new dataset is small. The second observation is that the error of the CV decision tree ensemble remains relatively stable around 6%, while all other competitive methods are sensitive to the data chunksize.

The results on the credit card data set is shown in Table 2. Each reported result in dollar amount is the average of multiple runs. The chunksize is from 3000 to 48000 transactions per chunk. The benefits increase as the chunksizes increase, as more fraudulent transactions are discovered in the chunk. Similar to the synthetic dataset, the CV decision tree ensemble is consistently better than training from new data chunk alone and training from new data plus some ad hoc selection of recent data chunks. When the chunksize is as small as 3000, the best method (E_8) that uses previous data unselectively recovered $77735, but the CV decision tree ensemble that systematically selects previous data recovered $81354. When the chunksize is as big as 48000, none of the methods (GA, G_i's and E_i's, $i \geq 2$) that use old data unselectively recovered more money than training from the new data itself (G_1). However, CV decision tree ensemble still recovered $582918, which is $20000 more than G_1.

The results on the donation dataset are shown in Table 3. Each number is the average of 5 runs. Obviously, any methods that uses more data than the new data itself is better, and the most accurate model is GA, the model trained from all available data in history. The CV decision tree ensemble is the second highest after GA and very close to GA for all different chunksizes. Training with all available data is consistently better than the CV decision tree ensemble is due to very skewed distribution ($< 5\%$ donors) and small data size (95412). In this situation, using more data almost always helps. However, we conjecture that if we had more training data beyond 95412, the accuracy of CV decision tree ensemble will increase and eventually reach GA.

6.5 Accuracy of Cross-validation

To evaluate how accurate is the n-fold cross-validation in estimating the true probability on a completely unseen testing data, we use 90% of the credit card fraud data for training and 10% of data for testing. We use the formulas in Section 5.2 to estimate the probability on new data. The results are plotted using "reliability plots" shown in Figure 2. Reliability plot shows how reliable the score of a model is in estimating the empirical probability of an example $\mathbf{x}$ to be a member of a class y. To draw a reliability plot, for each unique score value predicted by the model, we count the number of examples (N) in the data having this same score, and the number (n) among them having class label y. Then the empirical class membership probability is simply $\frac{n}{N}$. Most practical datasets are limited in size; some scores may just cover a few number of examples and the empirical class membership probability can be extremely over or under estimated. To avoid this problem, we normally divide the range of the score into continuous bins and compute the empirical probability for examples falling into each bin. To summarize these results, we use *mean square error* or MSE to measure how closely the score matches the empirical probability. Assuming that n_j is the number of examples covered in bin j, s_j is the score or predicted probability and p_j is the empirical probability, then $MSE = \frac{\sqrt{n_j \cdot (s_j - p_j)^2}}{\sum n_j}$.

The reliability plot using cross validation is the left one in Figure 2 with the subtitle of "(a) cv probability estimate" and the reliability plot of the same model tested on unseen test data set is the middle one with the subtitle of "(b) testing probability estimate". The shape of these two reliability plots are very similar. On the other hand, on the right plot with the subtitle of "(c) training probability estimate", we draw the training reliability plot. The difference of cv reliability plot and training reliability plot is whether to subtract 1 depending on the true label of the data. Obviously, without subtracting 1 for true positives, the score or estimated probability tend to significantly under estimate its true probability. Comparing the MSE's, the CV probability estimate plot has MSE=0.041, while the training probability plot has a much higher MSE=0.081.

6.6 How big is the incremental training set

As discussed in Section 4.2, the dataset that trains the optimal model could increase when the concept does change and the chunksize is significantly insufficient. We recorded the biggest training set in our experiments. For the synthetic dataset, it is approximately 1500 to 2500 under all experimented chunksizes. A detailed plot for all test runs (i.e., different amount of change and the number of dimensions affected) with dimension $d = 10$ and chunksize = 250 is shown in Figure 3. Each point is the size of the incremental dataset that trained the optimal model $FO_i(\mathbf{x})$. 20 chunks of the

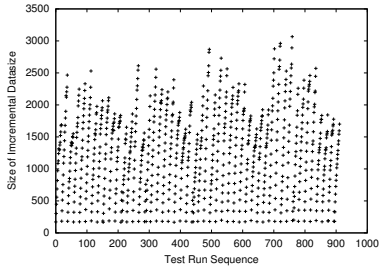
ChunkSize	Use new data only	Different ways to use old data unselectively									Use old data selectively
	G_1	GA	G_2	E_2	G_4	E_4	G_6	E_6	G_8	E_8	FO
250	18.76	**18.09**	**18.00**	**18.37**	**16.70**	**14.02**	**16.72**	**12.82**	**16.76**	**12.19**	**6.34**
500	17.59	17.65	**16.39**	**17.16**	**16.19**	**12.91**	**15.32**	**11.74**	**14.97**	**11.25**	**6.48**
750	16.47	17.18	**16.29**	**15.77**	**15.07**	**12.09**	**14.97**	**11.19**	**14.86**	**10.84**	**6.12**
1000	16.00	16.49	**15.89**	**15.62**	**14.40**	**11.82**	**14.41**	**10.92**	**14.68**	**10.54**	**6.03**
1500	10.81	16.03	13.43	11.98	12.88	10.82	12.78	**10.45**	12.13	**10.25**	**5.88**
2000	8.47	15.12	12.94	8.98	12.02	9.87	11.45	8.78	10.48	**8.34**	**5.98**
5000	5.72	14.24	10.86	7.13	11.02	7.45	10.78	7.35	10.84	7.45	5.78
20000	4.15	10.10	5.66	4.74	7.82	5.62	8.94	6.62	9.10	7.58	**4.11**

Table 1: Synthetic Dataset: Classification Error (%) using Single Best Decision Tree, Weighed Averaging Ensemble, and CV Decision Tree Ensemble

ChunkSize	Use new data only	Different ways to use old data unselectively									Use old data selectively
	G_1	GA	G_2	E_2	G_4	E_4	G_6	E_6	G_8	E_8	FO
3000	51943	**65470**	**55788**	**61793**	**59344**	**70403**	**62344**	**74661**	66184	**77735**	**81354**
4000	62181	**96879**	**66581**	**82663**	**72402**	**95792**	**74589**	**101930**	**76079**	**103501**	**126565**
6000	102099	**146848**	**102330**	**129917**	**113810**	**148818**	**118915**	**155814**	**123170**	**162381**	**193177**
12000	207392	**296144**	**233098**	**268838**	**248783**	**313936**	**263400**	**327331**	**275707**	**360486**	**387548**
24000	388646	356481	387464	**391024**	375292	**389368**	3723083	388342	361256	368839	**414124**
48000	561244	421455	517320	529543	498727	501492	465514	486814	452039	455830	**582918**

Table 2: Credit Card Dataset: Benefits (US $) using Single Classifiers, Weighted Averaging Ensembles, and CV Decision Tree Ensemble

Figure 3: Size of incremental training set for the synthetic data set with d=5 and chunksize = 250



same size (i.e., 250) are continuously generated with drifting concepts. As shown in Figure 3, for one complete test run, the size of the incremental training set nearly all monotonically increases. At the end, the biggest size of all tests settles down in between 1500 and 2500. It is evident that old data are being chosen judiciously to construct the new model to fit the changing concept.

For the credit card dataset, the biggest size is approximately 15000 to 20000 when chunksize $\leq$ 12000 and approximately 40000 when chunksize $\geq$ 24000. For the donation dataset, the biggest size nearly increases up to about 10000 to 20000 until there are no more data to run the experiment. We conjecture that this size would still grow if we had more training data beyond 95412.

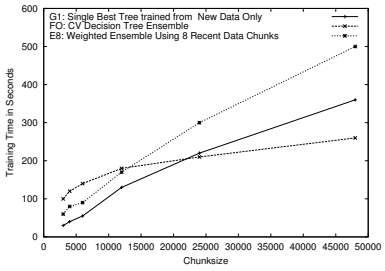
6.7 Optimal Models

One important aspect of our proposed algorithm is to choose the optimal model under different situations. As a particular study, we recorded the number of times that each of the four models, $FN_i(\mathbf{x})$, $FN_i^+(\mathbf{x})$, $FO_{i-1}(\mathbf{x})$ and $FO_{i-1}^+(\mathbf{x})$, is the actual optimal model with the lowest loss. For the synthetic dataset with

chunksize	$FN(\mathbf{x})$	$FN^+(\mathbf{x})$	$FO_{i-1}(\mathbf{x})$	$FO_{i-1}^+(\mathbf{x})$
250	289	517	99	407
2000	359	501	104	445
20000	879	212	45	30

Table 4: Optimal model counts for synthetic dataset

Figure 4: Training time of different models



dimension $d = 10$, chunksize = 250 is absolutely insufficient, chunksize = 200000 is absolutely sufficient, and chunksize=2000 is moderate. The number of times each of the four models is the optimal model is shown in Table 4. Two or more models can have exactly the same lowest error rate. When this happens, all these models are optimal and the counts for all of them are incremented. As a summary of Table 4, $FN^+(\mathbf{x})$ is the optimal model most of the times when the data is insufficient, and $FN(\mathbf{x})$ becomes the optimal model most of the time when the data is sufficient.

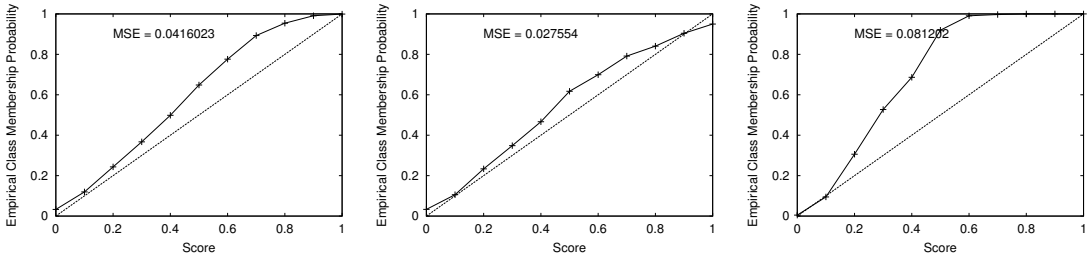
6.8 Training and Memory Efficiency

We recorded the running time to train different models. The results for the credit card dataset are shown in Figure 4. The x-

ChunkSize	Use new data only	Different ways to use old data unselectively									Use old data selectively
	G_1	GA	G_2	E_2	G_4	E_4	G_6	E_6	G_8	E_8	FO
94	5	**16**	**6**	**6**	**8**	**9**	**8**	**10**	**9**	**11**	**13**
188	7	**29**	**9**	**10**	**11**	**16**	**15**	**15**	**19**	**20**	**26**
376	15	**41**	**17**	**18**	**19.4**	**21**	**23**	**24**	**27**	**28**	**37**
752	28	**96**	**33**	**30**	**42**	**45**	**48**	**51**	**64**	**80**	**82**
2988	141	**412**	**171**	**182**	**201**	**245**	**237**	**320**	**284**	**370**	**381**
5976	401	**879**	**420**	**470**	**497**	**520**	**580**	**610**	**728**	**712**	**761**

Table 3: Donation Dataset: benefits (US $) using Single Classifiers, Weighted Averaging Ensembles, and CV Decision Tree Ensemble

Figure 2: Results of Cross-validation and Testing



(a) CV probability estimate (b) testing probability estimate (c) training probability estimate

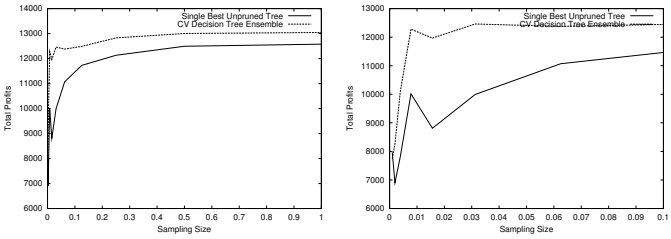
axis is the chunksize and y-axis is the training time. In this figure, we compare the time to construct a single best tree from the new dataset only (G_1), a CV decision tree ensemble (FO) as well as a weighted ensemble that always use fixed amount of old data (E_8). It is important to point out that the CV decision tree ensemble result includes the total time to compute a new CV decision tree from new data only, update most recent CV decision tree ensemble, train a new CV decision tree ensemble from new data and selected only data as well as to compare these models' and the most recent model's accuracy in the new data. The training time of the weighted ensemble E_8 includes the training time of G_1 as well as the time to assigns weights to each of the eight decision trees in the ensemble. When the chunksize is small (< 12000), it takes more time to train CV decision tree ensemble. This is due to the overhead of memory allocation and numerous file operations. However, after 12000, the training time for the single best tree G_1 as well as the weighted ensemble E_8 starts to shoot up significantly while the training time for the CV decision tree ensemble is less and grows at a lesser rate.

We saved each unpruned random tree in the file system for all three datasets. For each dataset, the size of each CV decision tree is very close. Comparing their size with the single best tree, the size of random tree is approximately 2 to 3 times the size of the single best tree. The saved size of each tree is a good indicator of its relative size in memory. Since we construct each tree in the ensemble one at at time, 2 to 3 times the size of the single best tree in main memory is reasonable.

6.9 Tolerance to Data Insufficiency

We ran experiments with sampling size ranging from $\frac{1}{2^7}$, $\frac{1}{2^6}$ to the full size of the "completely shuffled" original dataset on all three datasets. We generated a decision tree ensemble with 10 decision trees and 1 unpruned best single decision tree for comparison. It is important to point out that the decision ensemble and the best tree uses exactly the same dataset for this study. The purpose is to show that the ensemble approach has high tolerance for data insufficiency with the same amount of training data. Each test was run 5 times and the results are the average of 5 runs. Figure 5 is the result on the donation dataset. The x-axis is the sampling size and the y-axis is the profits, i.e., the amount of donation minus mailing cost. Obviously, the decision tree ensemble tree has very high tolerance to insufficient amount of data. When the training data size is about 0.007812% of the original training set (or approximately 900) examples, the total profit of the ensemble is already very close to the total profit of the same model trained from 100% training data.

Figure 5: Data Sufficiency Test on Donation Dataset



(a) full range (b) enlarged area up to 0.1

7. RELATED WORK

The original idea of generating random structured decision tree is proposed in [Fan et al., 2003, Fan, 2004a]. Our algorithmic extension is on the initial scan of the data to figure out a candidate pool of good features. The obvious advantage of this step is the ability to build only relevant and good trees. The number of trees required to approximate the optimal model will be much smaller. On the other hand, the introduction of random tree into streaming mining is completely new. On the empirical part, the introduction of n-fold cross-validation is new and particular important for streaming min-

ing. Breiman has proposed the "random forests" method [Breiman, 2001]. In random forests, randomness is injected by randomly sampling a subset of remaining features (those not chosen yet by a decision path) and then choosing the best splitting criteria from this feature subset. The chosen size of the subset has to be provided by the user of random forests. However, in CV decision tree ensemble, the splitting feature is randomly chosen from any remaining features not chosen yet in the current decision path. Besides the initial step to choose candidate features with information gain, there is no additional information gain check involved in choosing this feature and when it will be chosen. The data is only used to update the class distribution in each node. However, in Breiman's random forests, information gain or other criteria is still used to choose the best feature among randomly chosen feature subsets. Another important distinction is that Breiman's random forests performance simple voting on the final prediction. In other words, each tree votes 1 on one of the class labels. The class labels with the highest vote is the final prediction. However, in CV decision tree ensemble, each tree output raw probability and the probability outputs from multiple tree are averaged as the final probability.

8. CONCLUSION

Three important points are made in this paper. First, we argue that using old data unselectively is like gambling. It definitely helps build a more accurate model if the there isn't any concept drift and the new data chunk is insufficient. When there is concept drift, using old data unselectively helps if the new concept and old concept still have consistencies and the amount of old data chosen arbitrarily just happen to be right. We justify this claim through a synthetic dataset and ran several stream mining models that use old data unselectively. Based on this observation, we discuss how to choose old data and the best hypothesis under different situations, i.e., whether there is concept drift and whether the new data is indeed sufficient. However, we show that without having an oracle, detecting concept drift and data sufficiency is difficult and non-quantitative. A useful framework is one that is still able to select good old examples and compute the optimal model even without knowing if there is indeed concept drift or if the data chunk is indeed sufficient. Second, we proposed a cross-validation-based framework to choose data and compare sensible choices. Third, we proposed an implementation of this framework using cross-validation decision tree ensemble. The cross-validation decision tree ensemble is built by first reading the complete dataset once to find out those candidate features with information gain. We then construct multiple decision trees by randomly choosing from those features with information gain found in the previous step and ignoring any other features. Each node of the tree keeps class distribution statistics. To classify an example, the posterior probability outputs of multiple trees in the ensemble are averaged. The best decision is made by using the averaged probability and a dataset specific loss function. Cross-validation is implemented by using the class distribution statistics in each node without physically splitting the dataset and re-training the ensemble. We evaluated our approach on three streaming data, synthetic hyperplane dataset, credit card fraud detection as well as charity donation. With various amount of new data and various degrees of concept change, we have found that cross-validation decision tree ensemble consistently has significantly lower error rate than all compared existing approaches that use old unselectively. This is particularly true when the size of the new data set is small. The error rate by the proposed decision tree ensemble also remains relatively stable independent of the amount of new data and degree of concept-drifts. A demonstration of the software will be given in VLDB'04 [Fan, 2004b].

Acknowledgement

We thank Dr. Haixun Wang's generous help to transform our paper drawing into the plot to illustrate the problem of data selection.

9. REFERENCES

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

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)*.

Babu, S. and Widom, J. (2001). Continuous queries over data streams. *SIGMOD Record*, 30:109–120.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

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)*, Hong Kong, China.

Domingos, P. and Hulten, G. (2000). Mining high-speed data streams. In *Int'l Conf. on Knowledge Discovery and Data Mining (SIGKDD)*, pages 71–80, Boston, MA. ACM Press.

Fan, W. (August 2004b). StreamMiner: A classifier ensemble-based engine to mine concept-drifting data streams. In *Proceedings of 2004 International Conference on Very Large Data Bases (VLDB'2004)*, Toronto, Canada.

Fan, W. (July 2004a). On the optimality of probability estimation by random decision trees. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI'2004)*, San Jose, California, USA.

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.

Fan, W., Wang, H., Yu, P. S., and Ma, S. (2003). Is random model better? on its accuracy and efficiency. In *Proceedings of Third IEEE International Conference on Data Mining (ICDM'2003)*.

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.

Greenwald, M. and Khanna, S. (2001). Space-efficient online computation of quantile summaries. In *Int'l Conf. Management of Data (SIGMOD)*, pages 58–66, Santa Barbara, CA.

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, 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.

Street, W. N. and Kim, Y. (2001). A streaming ensemble algorithm (SEA) for large-scale classification. In *Int'l Conf. on Knowledge Discovery and Data Mining (SIGKDD)*.

Wang, H., Fan, W., Yu, P., and Han, J. (2003). 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.