Active Learning With Drifting Streaming Data

Indrė Žliobaitė, Albert Bifet, Bernhard Pfahringer, and Geoffrey Holmes

***Abstract*— In learning to classify streaming data, obtaining true labels may require major effort and may incur excessive cost. Active learning focuses on carefully selecting as few labeled instances as possible for learning an accurate predictive model. Streaming data poses additional challenges for active learning, since the data distribution may change over time (concept drift) and models need to adapt. Conventional active learning strategies concentrate on querying the most uncertain instances, which are typically concentrated around the decision boundary. Changes occurring further from the boundary may be missed, and models may fail to adapt. This paper presents a theoretically supported framework for active learning from drifting data streams and develops three active learning strategies for streaming data that explicitly handle concept drift. They are based on uncertainty, dynamic allocation of labeling efforts over time, and randomization of the search space. We empirically demonstrate that these strategies react well to changes that can occur anywhere in the instance space and unexpectedly.**

***Index Terms*— Active learning, concept drift, data streams, user feedback.**

I. Introduction

PREDICTIVE models are widely used in industry, finance, marketing, web analytics, surveillance, and many more areas. A model takes as input an observation (e.g., a credit applicant) and aims to predict an outcome (e.g., the credit risk of this applicant). Predictive models are built from historical data that record past observations (input variables) and the true outcomes for those observations (labels). In the standard setting, a predictive model is built from historical data once, and then continuously applied for new data. In the streaming data scenario, data continuously arrive in real time, and may be changing over time. In this setting, predictive models need to operate fast, fit into limited memory, and adapt online; otherwise their accuracy will degrade over time.

In predictive modeling, often unlabeled data is abundant but labeling is expensive. Labels can be costly to obtain due to required human input (labor cost). Consider, for example, blog posts arriving as a stream. The goal is to predict whether a post will be interesting to a given user at a given time. The interests of the user may change. To obtain training data, the historical posts need to be read and labeled as interesting or not interesting. This requires costly human labor. For example, Amazon Mechanical Turk[1] provides a marketplace for intelligent human labeling. Labeling can also be costly due to a required expensive, intrusive, or destructive laboratory test. Consider a production process in a chemical plant where the goal is to predict the quality of production output. The relationship between the input sensor readings and the output may change over time due to manual re-tuning, complementary ingredients, or replacement of physical sensors. In order to know the true quality, a laboratory test needs to be performed, which is costly. Under such conditions, it may be unreasonable to request true labels for all incoming instances.

Active learning studies how to label selectively instead of asking for all true labels. It has been extensively studied in pool-based and online settings [1]. In the pool-based setting, the decision concerning which instances to label is made after inspecting all historical data. This setting is designed for offline operation. In this paper, we explore active learning in data stream settings. As a motivation for the setting, consider again a chemical production example. In order to bring a sample to the laboratory for labeling, an operator physically collects this sample from the production line. The decision whether to sample needs to be made immediately, before the product is gone via production lines, for instance, for packaging. Moreover, it is infeasible to collect a large pool of samples and later decide which one to bring to the laboratory. Thus, for every incoming instance, a decision whether to label it needs to be made immediately, as there is no re-access.

The main difference between online active learning and active learning in data streams is in the expectations around changes. Online active learning typically fixes an uncertainty threshold and requests the true label if this threshold is breached. In data streams, the relations between the input data and their labels may change due to concept drift [2] and these changes can happen anywhere in the instance space. Existing online active learning strategies may never query instances from some regions and thus may never know that changes are happening and therefore never adapt. Moreover, in data streams we cannot keep the decision threshold or a region of uncertainty fixed, as eventually the system would stop learning and fail to react to changes. Finally, traditional active learning distorts the incoming data distribution expecting more effective learning, while active learning with data streams must preserve

Manuscript received May 12, 2012; revised December 7, 2012; accepted December 16, 2012. Part of the research leading to these results has received funding from the EC within the Marie Curie Industry and Academia Partnerships and Pathways (IAPP) programme under Grant Agreement 251617.

I. Žliobaitė is with the Smart Technology Research Center, Bournemouth University, Poole BH12 5BB, U.K. (e-mail: zliobaite@gmail.com).

A. Bifet is with Yahoo! Research Barcelona, Barcelona 08018, Spain (e-mail: abifet@yahoo-inc.com).

B. Pfahringer and G. Holmes are with the Department of Computer Science, University of Waikato, Hamilton 3240, New Zealand (e-mail: bernhard@cs.waikato.ac.nz; geoff@cs.waikato.ac.nz).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2012.2236570

[1]Available at https://www.mturk.com.

the incoming data distribution to the extent that changes could be detected as they happen.

This paper introduces a framework for active learning from drifting data streams. First, we formulate and justify requirements for active learning in this setting. Then, we propose three corresponding active learning strategies that can be integrated with an adaptive learning algorithm of a user's choice.

The problem setting is as follows. Data arrives in a stream, and predictions need to be made in real time. Concept drift is expected, thus learning needs to be adaptive. Since the instances are regularly discarded from memory, the true label can be requested either immediately or never. Our goal is to maximize prediction accuracy over time while keeping the labeling costs fixed within an allocated budget, expressed as a fraction of all incoming data. After scanning an instance and outputting the prediction for it, we need a strategy to decide whether or not to query for the true label so that the predictive model could train itself with this new instance. Regular retraining is needed because of changes in data distribution. Active learning strategies in data streams, in addition to being able to learn an accurate classifier in stationary situations, need to be able to:

1) balance the labeling budget over infinite time;
2) notice changes happening anywhere in the instance space;
3) preserve the distribution of the incoming data for detecting changes.

In this paper, we develop three such strategies, assuming that the adaptive learning technique is externally given. Experimental evaluation on real data streams demonstrates that our strategies effectively handle concept drift while saving labeling costs, as we do not need to label every instance. To the best of our knowledge, this study is the first to address active learning for instance-incremental streaming data (we preclude methods that learn from a stream in batches) where historical data cannot be stored in memory.

A short version of this paper appeared in conference proceedings [3]. The major novelties with respect to the conference version are the following: a new monitoring mechanism for label spending (with decay); theoretical analyses of the framework and the strategies; a new strategy (Split); experiments with alternative base learner (Hoeffding); and new experimental analyses (sensitivity and change detection).

The rest of this paper is organized as follows. Section II discusses related work. Section III presents our active learning framework for drifting data streams. In Section IV, we analytically investigate the proposed strategies. Section V presents experimental analysis, and Section VI concludes this paper.

II. Related Work

Online active learning has been a subject of studies, where the data distribution is assumed to be static (e.g., [1], [4]). However, static online active learning is not designed to handle changes. As we will demonstrate in our analysis and experiments, existing strategies are able to handle drifts if the changes happen to be gradual and close to the current decision boundary; however, the reaction to changes might be slow. When changes happen far from the decision boundary, such methods fail completely. Those situations require advanced strategies; we develop several such strategies in this study.

The problem of label availability in data streams with concept drift has been acknowledged in several recent works [5]–[8]. Most convert a stationary active learning or a semi-supervised learning method to an online setting by partitioning a stream into batches and applying stationary strategies within each batch. These works differ from our study in two major aspects. First, their strategies require to inspect a batch of instances at a time, thus they need to assume that limited re-access to data is possible. In contrast, our stream setting requires making labeling decisions online at the time of scanning each instance.

Moreover, the existing active learning or semi-supervised learning strategies only complement concept drift handling strategies; they are not tailored to adapt directly. It is assumed that the data within a batch is stationary, and the goal is to train an accurate classifier while minimizing labeling costs. Adaptation and drift detection is separate. Thus, these techniques help to learn an accurate current model with fewer labels, but they are not designed to adapt to changes faster. If changes happen far from the decision boundary, they are likely to be missed. In contrast, we tailor active learning to handle concept drift directly online and search the instance space explicitly.

Some studies are conceptually related to our approach. We highlight them below.

The first group uses active learning strategies [9], [10] with batches. Zhu *et al.* [10] build a classifier on a small portion of data within a batch at random and use uncertainty sampling to label more instances within this batch. A new classifier in each batch is needed to take into account concept drift. Lindstrom *et al.* [9] use uncertainty sampling to label the most representative instances within each new batch. They do not explicitly detect changes, instead, they use a sliding window approach, which discards the oldest instances. In summary, these approaches apply static active learning to batches, which is not possible in data streams where historical data cannot be stored in memory.

Note that, typically, (a real) concept drift refers to changes in the posterior distributions of data $p(y|X)$, where X is the observed input data and y is the label. In other words, real concept drift means that the unlabeled data distribution does not change, only the class conditional distributions change. In contrast, data evolution refers to changes in the unconditional distribution of data $p(X)$.

Several studies integrate active learning and change detection [8], [11], [12]. They first try to detect change, and only if change is detected do they ask for representative true labels. However, only a change in $p(X)$ can be handled this way. We address real concept drift, which means that $p(y|X)$ changes and these changes cannot be detected without labels. Thus, these works are not solving the same problem.

Another group uses semi-supervised learning approaches to label some of the unlabeled data automatically [5]–[7]. Klinkenberg [5] separates changes in $p(X)$ from changes in $p(y|X)$ and uses existing semi-supervised techniques to label when $p(X)$ drifts, while a separate non-active learning strategy handles changes in $p(y|X)$. Widyantoro and Yen [6] first verify that a concept is stable and only then apply semi-supervised techniques for labeling. Masud *et al.* [7] first label some instances within a batch at random, and then propagate those labels to other instances using microclustering. In both works, automated labeling concerns only the subsets of the learning problem — which are assumed to be stationary (no real concept drift) — which does not correspond to the data stream setting we are addressing, thus they are not directly comparable. Using semi-supervised learning for drifting data would be possible only if changes affect $p(X)$, while our strategies can handle changes in $p(y|X)$.

Several studies [13]–[15] address a similar problem, but they assume that only a part of the data in a stream is labeled and propose a method to learn from both labeled and unlabeled data. The problem setting is also different from that of this paper, as they do not perform active learning (active labeling).

A few studies relate to the variable active learning criterion, which we introduce as a part of our strategies. Attenberg and Provost [4] introduce active inference as an additional aspect of online active learning. They maintain a budget per time period in a stream setting, while, instead of uncertainty, they estimate the utility of labeling an instance, which also takes into account the expected frequency of seeing a particular instance again. It assumes a possibility of repeated examples. The labeling threshold is fixed, but it depends on more than just uncertainty. This paper is limited to the stationary setting.

Cesa–Bianchi *et al.* [16] develop an online active learning method for a perceptron based on selective sampling using a variable labeling threshold $b/(b+|p|)$, where b is a parameter and p is the prediction of the perceptron. The threshold itself is based on certainty expectations, while the labels are queried at random. This mechanism could allow adaptation to changes, although they did not explicitly consider concept drift.

III. Strategies

Next we present our active learning framework and labeling strategies for drifting data streams. We introduce our strategies in three steps, where each step aims to overcome a challenge posed by the data stream setting. We start with a formal definition of the setting and requirements for the strategies.

A. Setting

Let $\mathbf{D}$ be an infinite d-dimensional data stream $\ldots, X_i, X_{i+1}, X_{i+2}, \ldots$ and let y_t be the true label of X_t, where t indicates the time when an instance arrives. The labeling cost is the same for any instance. We impose a budget B to obtain the true labels, which is expressed as a fraction of the number of incoming instances. $B = 1$ means that all arriving instances are labeled, whereas $B = 0.2$ means that 20% of the arriving instances are labeled. B should satisfy $0 < B \leq 1$, as $B = 0$ would mean no labeling and no learning, and the upper bound holds because the budget is a fraction of all the incoming instances. The budget is motivated by online learning applications with limited resources, e.g., laboratory tests in the chemical industry.

Let $labeling_t$ be a Boolean indicator, where $labeling_t = true$ means that the true label y_t is queried, $labeling_t = false$ means that we do not ask for the true label.

A *labeling strategy* is a procedure for determining whether to query the true label for a given instance, i.e., $labeling_t = \text{Strategy}(X_t, parameters)$.

B. Requirements

We formulate three requirements for labeling strategies in order to achieve effective performance on drifting streaming data. First, a strategy should be able to balance labeling over infinite time so that at any time interval the expected labeling does not exceed the budget. Second, a strategy should query over all the instance space in order to be able to adapt to changes that may happen anywhere in the instance space. Third, a strategy should preserve the input data distribution for unbiased change monitoring and detection.

Formally, these requirements can be presented as follows.[2]

1) *Requirement 1*: Labeling at any time should obey the budget $\sum_{\mathbf{D}} p(label|X)p(X) \leq B$, where $p(label|X)$ is the probability of labeling given data and $p(X)$ is the probability density of data.
2) *Requirement 2*: For any instance $X \in \mathbf{D}$, the probability of labeling should not be zero: $p(label|X) > 0$.
3) *Requirement 3*: The probability density of labeled data should be equal to the probability density of unlabeled data: $p(X|label) = p(X)$.

Note that Requirement 3 can be softened. Our goal is to have unbiased change detection. Therefore, we only need to control the distribution of the data that is passed to the change detector, i.e., $p(X|label, detect) = p(X)$, where $detect = true$ means that this instance is used for change detection.

C. Framework

First, we need a generic framework in which the strategies would operate, which we build upon a standard incremental learning framework for data streams with change detection. Here we use the change detection technique of [17]: when the accuracy of the classifier begins to decrease, a new classifier is built and trained with new incoming instances. When a change is detected, the old classifier is replaced by the new one. We chose DDM [17] because it is simple, interpretable, and accepted in the community. In principle, any incremental change detector can be employed.

The framework for active learning is presented in Algorithm 1. The framework incorporates active learning strategies into learning from data streams. The main purpose of the framework is to govern the learning process and control utilization of the labeling budget.

At every time point, we make a prediction with the current model $\hat{y} = L(X)$ outside the learning framework. The framework then receives the instance X (line 1), and checks whether actual labeling expenses $\hat{b}$ do not exceed the allowed budget B (line 2). If the budget is exceeded or on the limit, then

[2]For probabilities we will use a simplified notation replacing $labeling = true$ by $label$.

Algorithm 1 Active Stream Framework

```
Input: labeling budget $B$, Strategy (parameters)
Output: at every time iteration output classifier $L$
Initialize: labeling expenses $\hat{b} \leftarrow 0$
1  repeat
2      receive incoming instance $X$;
3      if $\hat{b} < B$ then
           //budget is not exceeded
4          if Strategy ($X$,parameters) = true then
5              request the true label $y$ of instance $X$;
6              update labeling expenses $\hat{b}$ (Sec. III-D);
7              update classifier $L$ with $(X, y)$;
8              if $L_n$ exists then
9                  update classifier $L_n$ with $(X, y)$;
10             if change warning is signaled then
11                 start a new classifier $L_n$;
12             if change is detected then
13                 replace classifier $L$ with $L_n$;
14 until forever;
```

no learning happens, and the existing classifier L is brought forward. If budget is available, then an active strategy of the user's choice is called (line 3) to determine whether to request for the true label and use this instance for updating the classifier. Lines (7–12) manage classifier updates and, depending on whether a change is detected, may replace an old classifier with a new one.

D. Controlling the Budget

Since the framework is meant for infinite time operation, we need to ensure that label spending $\hat{b}$ does not exceed the allocated budget B over infinite time. We can count the exact spending or approximate it.

1) Exact Spending Estimate: A straightforward way is to keep two counters: u, a counter of how many labels we queried, and t, how much time has passed since the start. Then, at any time the total label spending is $b = u/t$. This gives us an exact answer of labeling expenses since the beginning of operation. However, over infinite time this approach will have a problem. The contribution of every next label to the spending will diminish over infinite time, and a single labeling action will become less and less sensitive.

Instead of keeping an exact count b since the start of the operation, one could keep a count b^w over a fixed memory window, e.g., the last 1000 instances. This approach, however, requires storing not only the counter but also all the labeling decisions within that window, contradicting the incremental learning setting. Therefore, we would rather keep an approximate count that would not require storing individual decisions.

2) Approximate Spending Estimate: Instead of keeping exact counts of label spending, we introduce a forgetting procedure. Let w be a parameter describing the size of the memory window; e.g., to control the budget within the last 100 time steps we set $w = 100$. Let $\hat{u}$ be an estimate of how many true labels were queried within the last w incoming data points. We define a recursive way to compute this estimate

$$\hat{u}_t = \hat{u}_{t-1}\lambda + labeling_t \tag{1}$$

where $\lambda = (w-1)/w$, or $\lambda = 1 - 1/w$. We define λ in such a way that $\lim_{t\to\infty} \hat{u}_t = wE[labeling]$.

Then the estimated label spending at time t is

$$\hat{b}_t = \frac{\hat{u}_t}{w}. \tag{2}$$

Next, we prove that $\hat{b}$ as defined in (2) is an unbiased estimate of label spending b. Let b be the true fraction of label spending. Let labeling $(label_t)$ be a random variable that follows the Bernoulli distribution $q \sim Bern(b)$. It can be shown from (1) that $\hat{u}_i = \sum_i q_i \lambda^i$. Then the expected value of the label spending estimate is

$$E[\hat{b}] = E\left[\frac{\hat{u}_i}{w}\right] = E\left[\frac{\sum_i q_i\lambda^i}{w}\right] = \frac{E[q]}{w}\frac{1}{1-\lambda} = E[q] = b. \tag{3}$$

Next we find the variance of the estimate $\hat{b}$

$$\begin{aligned} Var[\hat{b}] &= Var\left[\frac{\hat{u}_i}{w}\right] = Var\left[\frac{1}{w}\sum_i q\lambda^i\right] \\ &= \frac{1}{w^2}Var[q]\sum_i \lambda^{2i} = \frac{Var[q]}{w^2}\frac{1}{1-\lambda^2} = \frac{b(1-b)}{2w-1}. \end{aligned} \tag{4}$$

We see that $\lim_{w\to\infty} Var[\hat{b}] = 0$, i.e., $E[\hat{b}]$ converges to the true b independently of the initialization value of $\hat{u}$.

Finally, an increment of the budget spending after one labeling decision is

$$\Delta = \frac{\hat{u}_{t+1}}{w} - \frac{\hat{u}_t}{w} = \frac{\hat{u}_t\lambda + 1}{w} - \frac{\hat{u}_t}{w} = \frac{1}{w} - \frac{\hat{u}_t}{w^2}. \tag{5}$$

In the limit

$$\begin{aligned} \lim_{t\to\infty}\left(\frac{1}{w} - \frac{\hat{u}_t}{w^2}\right) &= \frac{1}{w} - \frac{1}{w^2}\lim_{t\to\infty}\left(\sum_t q_t\lambda^t\right) \\ &= \frac{1}{w} - \frac{1}{w^2}\frac{E[q]}{(1-\lambda)} = \frac{1-b}{w} > 0 \end{aligned} \tag{6}$$

given that the intended label spending: $0 < b \leq 1$. We see that in the limit the contribution of one label is positive; thus, the system remains sensitive to labeling actions after a long period of operation.

Thus, in the remainder of this paper we will use this label spending estimate as given in (1) and (2).

E. Random Strategy (Baseline)

The Random strategy is naive in the sense that it labels the incoming instances at random instead of actively deciding which label would be more useful. The labeling decision does not depend on the actual incoming instance X_t. For any instance, the true label is requested with a probability B, where B is the given budget. Algorithm 2 gives a formal description.

Algorithm 2 Strategy RANDOM(X_t,B)

```
Input: labeling budget $B$
Output: $labeling \in \{\text{true}, \text{false}\}$ indicates whether to
        request the true label $y_t$ for $X_t$
1 generate a uniform random variable $\xi \in [0, 1]$;
2 return $labeling \leftarrow \mathbf{1}(\xi \le B)$
```

Algorithm 3 Strategy FIXED UNCERTAINTY(X_t,θ,L)

```
Input: incoming instance $X_t$, labeling threshold $\theta$,
       trained classifier $L$
Output: $labeling \in \{\text{true}, \text{false}\}$
1 $\hat{y}_t \leftarrow \arg\max_y P_L(y|X_t)$ prediction of $L$, where
  $y \in \{1, \ldots, c\}$ is the class label space;
2 return $labeling \leftarrow \mathbf{1}(P_L(\hat{y}_t|X_t) \le \theta)$
```

Properties of the Strategy: For the Random strategy, the probability of requesting the label for a given instance X is

$$p(label|X) = p(label) = \mathbf{1}(\xi \le B) = B \tag{7}$$

where $\xi \sim \mathcal{U}[0, 1]$, $\mathbf{1}()$ is the indicator function returning 1 if the argument is true and 0 otherwise. We show that the random strategy satisfies the requirements stated in Section III-B.

- ✓ *Requirement 1*: $\sum_{\mathbf{D}} p(label|X)p(X) = p(label)\sum_{\mathbf{D}} p(X) = p(label)1 = B$.
- ✓ *Requirement 2*: probability of labeling does not depend on X, thus $p(label|X) = p(label) = B > 0$, $\forall X \in \mathbf{D}$.
- ✓ *Requirement 3*: $p(X|label) = \frac{p(label|X)p(X)}{p(label)} = \frac{p(label)p(X)}{p(label)} = p(X)$.

F. Fixed Uncertainty Strategy (Baseline)

Uncertainty sampling is perhaps the simplest and the most common active learning strategy [18]. The idea is to label the instances for which the current classifier is the least confident. In an online setting, it corresponds to labeling the instances for which the certainty is below some fixed threshold. A simple way to measure uncertainty is to use the posterior probability estimates output by a classifier. If the maximum posterior probability for any class given the instance is less than a threshold, then the true label is requested and this instance is used for learning. The Uncertainty strategy with a fixed threshold is presented in Algorithm 3.

Properties of the Strategy: For the Fixed Uncertainty strategy, the probability of requesting the label is

$$p_L(label|X) = \mathbf{1}(P_L(\hat{y}|X) \le \theta) \tag{8}$$

where $P_L()$ indicates the probability output by the model L.

In the data-stream setting, fixing an optimal threshold θ so that the budget B is not violated is a challenging task. Let the distribution of the maximum posterior probabilities be a random variable $h = P_L(\hat{y}|X)$ for which $\max(h) = 1$ and $\min(h) = \frac{1}{c}$, where c is the number of classes. Thus, to satisfy the budget constraint we should set θ such that

$$p(h \le \theta) = p\left(h \in \left[\frac{1}{c}, \theta\right]\right) = \int_{1/c}^{\theta} p(h)\mathrm{d}h \le B. \tag{9}$$

In practice, the distribution of h is typically unknown, hence we need to make some simplifying assumptions. As a recommendation for setting θ, we can *assume* that the distribution of the posterior probabilities output by a classifier is uniform $h \sim \mathcal{U}[\frac{1}{c}, 1]$. In such a case, to satisfy (9) we can set

$$\theta = \frac{1}{c} + B\left(1 - \frac{1}{c}\right). \tag{10}$$

By the property of uniformity from (8)–(10), we get

$$p_L(label|X) = p(h \le \theta) = \int_{1/c}^{\theta} p(h)\mathrm{d}h = \frac{\theta - \frac{1}{c}}{1 - \frac{1}{c}} = B. \tag{11}$$

Next we show that the Fixed Uncertainty strategy satisfies none of the requirements, but Requirement 1 is satisfied by the budget constraint present in the framework (Algorithm 1).

First, let us formally define learning. The Fixed Uncertainty strategy uses the learning model L to determine whether to label an incoming instance. This model L in the online learning environment is updated with every new labeled instance arriving. Suppose L_t is the model at time t. Let us assume that the data distribution is *stationary*, i.e., $p_t(X) = p_{t+1}(X)$ and $p_t(y|X) = p_{t+1}(y|X)$. Then we define *learning* as a process that, given stationary data, does not increase uncertainty over time

$$P_{L_{t+1}}(\hat{y}|X) \ge P_{L_t}(\hat{y}|X). \tag{12}$$

- ✓ *Requirement 1*: At time $t = 1$, this requirement holds if θ is selected to satisfy (10). At times, $t > 1$ if learning condition (12) is satisfied. Independently of satisfying these conditions within the strategy, Requirement 1 is always enforced (outside the strategy) by the framework in Algorithm 1 line 2.
- × *Requirement 2*: Does not hold. Suppose we set a budget $B < 1$; otherwise no active learning is needed. We need to set $\theta < 1$, since otherwise every instance gets a label. In the worst case scenario, we may receive an instance X for which the model output is very certain $P_L(\hat{y}|X) = 1$. Let the input data distribution $p(X)$ remain stationary. Then for $\theta < 1$, the requirement is breached, since $p_L(label|X) = \mathbf{1}(P_L(\hat{y}|X) \le \theta) = 0$, and from (12) $p_{L_{t+1}}(label|X) \le p_{L_t}(label|X) = 0$. Thus, instance X never gets labeled.
- × *Requirement 3*: Does not hold, since $p(X|label) = \frac{p(label|X)p(X)}{p(label)} \neq p(X)$.

G. Variable Uncertainty Strategy

A challenge with the Fixed Uncertainty strategy for streaming data is that a classifier would either exhaust its budget if θ is set incorrectly, or learn enough for the uncertainty to remain above the threshold most of the time. In both cases, it will stop learning and thus fail to adapt to changes.

To overcome these issues, instead of labeling the instances that are less certain than the threshold, we aim to label the least certain instances within a time interval. More formally, given that the distribution of the model output $p(h)$ (9) is expected to change over time (due to learning more accurate models

Algorithm 4 Strategy VAR-UNCERTAINTY(X_t,L,s)

```
Input: incoming instance $X_t$, trained classifier $L$,
       threshold adjustment step $s \in (0, 1]$
Output: $labeling \in$ {true, false}
Initialize: initialize labeling threshold $\theta \leftarrow 1$ and store
            the latest value during operation
1 $\hat{y}_t \leftarrow \arg\max_y P_L(y|X_t)$, where $y \in \{1, \ldots, c\}$;
2 if $P_L(\hat{y}_t|X_t) < \theta$ then
      //uncertainty below the threshold
3     decrease the uncertainty region $\theta \leftarrow \theta(1 - s)$;
4     return $labeling \leftarrow$ true
5
6 else
      //certainty is good
7     make the uncertainty region wider $\theta \leftarrow \theta(1 + s)$;
8     return $labeling \leftarrow$ false
```

and adaptation to concept drift), we would like to update the threshold θ over time so that

$$p(h \leq \theta_t) = \int_{1/c}^{\theta_t} p_t(h)\mathrm{d}h = B. \tag{13}$$

Thus we introduce a time-variable threshold, which adjusts itself depending on the incoming data to align with the budget. If a classifier becomes more certain (stable situations), the threshold expands to be able to capture the most uncertain instances. If a change happens and suddenly many labeling requests appear, then the threshold is contracted to query the most uncertain instances first.

It may seem counterintuitive that we are asking for more labels at certain situations and fewer labels at changes. In fact, our dynamic threshold ensures that we are asking for the same number of labels in all situations. This is how we balance the budget, as we do not know when or how often changes will occur, so we aim to spend the budget uniformly over time.

The Variable Uncertainty strategy with a variable threshold is described in Algorithm 4. Note that for $s = 0$ the Variable Uncertainty would turn into the Fixed Uncertainty strategy.

The adjustment mechanism of the Variable Uncertainty resembles the trust-region mechanisms in numerical optimization [19]. If a good model of the objective function is found within the region, then the region is expanded. If the approximation is poor, then the region is contracted.

Properties of the Strategy: For the Variable Uncertainty strategy, the probability of requesting the label for an instance X is

$$p_L(label|X) = \mathbf{1}(P_L(\hat{y}|X) \leq \theta_t) \tag{14}$$

where θ_t varies over time as $\theta_t \leftarrow \theta_{t-1}(1 + update)$, where $update \in \{+s, -s, 0\}$ and $s \in (0, 1]$.

For large s, the Variable Uncertainty strategy turns into the Random strategy, and for small s the Variable Uncertainty strategy becomes the Fixed Uncertainty strategy. A complementary theoretical analysis to support this statement can be found in Appendix.

Since the behavior of Variable Uncertainty depends heavily on s, whether the requirements are satisfied also depends on s.

Algorithm 5 RAN-VAR-UNCERTAINTY(X_t,L,s,δ)

```
Input: incoming instance $X_t$, trained classifier $L$,
       threshold adjustment step $s \in (0, 1]$,
variance of the threshold randomization $\delta$
Output: $labeling \in$ {true, false}
Initialize: initialize labeling threshold $\theta \leftarrow 1$ and store
            the latest value during operation
1 $\hat{y}_t \leftarrow \arg\max_y P_L(y|X_t)$, where $y \in \{1, \ldots, c\}$;
2 $\theta_{randomized} \leftarrow \theta \times \eta$, where $\eta \in \mathcal{N}(1, \delta)$ is a random
  multiplier;
3 if $P_L(\hat{y}_t|X_t) < \theta_{randomized}$ then
      //uncertainty below the threshold
4     decrease the uncertainty region $\theta = \theta(1 - s)$;
5     return $labeling \leftarrow$ true
6
7 else
      //certainty is good
8     make the uncertainty region wider $\theta = \theta(1 + s)$;
9     return $labeling \leftarrow$ false
```

For large s, requirements are more likely to be satisfied. If s is small (e.g., recommended value $s = 0.01$), then Variable Uncertainty behaves as the Fixed Uncertainty in terms of satisfying the requirements.

✓ *Requirement 1.*
× *Requirement 2.*
× *Requirement 3.*

H. Uncertainty Strategy With Randomization

The Uncertainty strategy always labels the instances that are close to the decision boundary of the model. However, in data streams changes may happen anywhere in the instance space (thus Requirement 2). When concept drift happens in labels, the classifier will not notice it without the true labels. In order not to miss concept drift, we would like, from time to time, to label the instances about which the classifier is very certain. Thus, for every instance we randomize the labeling threshold by multiplying by a normally distributed random variable that follows $\mathcal{N}(1, \delta)$. This way, we will label the instances that are close to the decision boundary more often, but occasionally we will also label some distant instances.

This strategy trades off labeling some very uncertain instances for labeling very certain instances, in order not to miss changes. Thus, in stationary situations this strategy is expected to perform worse than the uncertainty strategy, but in changing situations it is expected to adapt faster. The uncertainty strategy with randomization is presented in Algorithm 5.

Properties of the Strategy: For the Random Variable Uncertainty strategy, the probability of requesting the label for a given instance X is

$$p_L(label|X) = \mathbf{1}(P_L(\hat{y}|X) \leq \theta_{\text{randomized}}) \tag{15}$$

where $\theta_{\text{randomized}} = \theta_t\eta$.

Algorithm 6 SPLIT(X_t,L,s,δ,B)

```
Input: incoming instance $X_t$, trained classifier $L$,
       threshold adjustment step $s \in (0, 1]$, proportion of
       random labeling $\upsilon \in (0, 1)$, budget $B$
Output: $labeling \in \{\text{true}, \text{false}\}$
Initialize: initialize labeling threshold $\theta \leftarrow 1$ and store
            the latest value during operation
1 if $\eta < \upsilon$, where $\eta \sim U[0, 1]$ is random then
2     if change detected then
3         cut the training window
4     return $labeling \leftarrow$ RANDOM($B$)
5
6 else
7     return $labeling \leftarrow$ VARUNCERTAINTY($X_t, L, s$)
```

The Random Variable Uncertainty strategy can satisfy Requirement 2 even with small s. Consider the worst case scenario $p_L(\hat{y}|X) = 1$. We will label X if $\theta_{\text{randomized}} = \theta\eta > 1$. For that, we need $\eta > \frac{1}{\theta}$. Since $\eta \sim \mathcal{N}(1, \delta)$, $p(\eta > \frac{1}{\theta}) > 0$ and thus Requirement 2 is satisfied in the worst case scenario. If s is small (e.g., recommended value $s = 0.01$), then Random Variable Uncertainty behaves as follows.

$\checkmark$ *Requirement 1.*
$\checkmark$ *Requirement 2.*
$\times$ *Requirement 3.*

I. Split Strategy

Many adaptive learning methods use change-detection mechanisms that monitor streaming error. Change detectors (e.g., [17]) are built with an implicit assumption that the errors are distributed uniformly over time unless a change has happened. The uncertainty strategy asks for labels based on a prediction. Since the predictive model adapts over time, the stream of *labeled* data is not distributed identically to the stream of unlabeled data. Thus, change detectors may have problems to distinguish a change in distribution due to active labeling from a change in distribution due to concept drift.

To overcome that problem, we introduce a labeling strategy that splits a stream at random into two streams. One of the new streams is labeled according to the uncertainty strategy, while the other is labeled according to the Random strategy. Both streams are used to train a classifier. But only the random stream is used for change detection. We sacrifice reactivity of the learning (labeling based entirely on the uncertainty strategy is expected to learn faster in stationary situations) for an opportunity to detect changes more accurately.

The Split strategy is presented in Algorithm 6.

Properties of the Strategy: The Split strategy is a mixture of the random and the Variable Uncertainty strategies in terms of labeling. Thus, for the Split strategy the probability of requesting the label for a given instance X is

$$p_L(label|X) = \eta B + (1 - \eta)\mathbf{1}(P_L(\hat{y}|X) \leq \theta_t). \quad (16)$$

Since the Split strategy includes the Random strategy, Requirements 2 and 3 are satisfied. Requirement 1 is enforced by the framework (Algorithm 1, line 2). Requirement 2 is satisfied due to the Random strategy. Requirement 3 is satisfied as change detection runs only on randomly labeled data.

$\checkmark$ *Requirement 1.*
$\checkmark$ *Requirement 2.*
$\checkmark$ *Requirement 3.*

J. Summary

Table I summarizes the five strategies with respect to the requirements indicated in the introduction. The Random strategy satisfies all three requirements. Randomized uncertainty satisfies budget and coverage, but it produces biased labeled data. The Variable Uncertainty satisfies only budget control, and the Fixed Uncertainty satisfies none.

TABLE I
SUMMARY OF STRATEGIES

	Budget Control Req. 1	Full space Coverage Req. 2	Preserving Distribution Req. 3
Random	Yes	Yes	Yes
Fixed Uncertainty	Handled	No	No
Variable Uncertainty	Handled	No	No
Randomized Uncertainty	Handled	Yes	No
Split	Handled	Yes	Yes

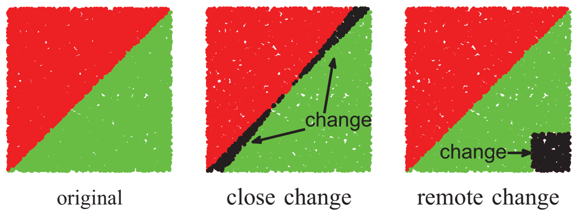

Fig. 1. Data with changes close to and far away from the decision boundary.

IV. Analysis of How the Strategies Label

In this section we analyze the behavior of the strategies with stationary data and under concept drift. We employ synthetic data in two-dimensional space, which presents a controlled setting with known distributions and drifts. The data is distributed uniformly at random in a square; the distribution $p(X)$ does not change over time, $p(y|X)$ changes. This data represents a binary classification problem. The initial decision boundary is set at $x_1 = x_2$, as illustrated in Fig. 1 (left).

We investigate two drift situations—a change happening close to the decision boundary, and a remote change. Fig. 1 (center and right) presents in black the regions in the instance space that are affected by a change. The center plot illustrates a change that happens close to the decision boundary. The right plot illustrates a remote change. In both cases, the same number of instances change, i.e., changes affect 5% of the space.

A. Instance Space Coverage

First we investigate a stationary case. Fig. 2 shows which instances would be labeled by the strategies given the initial

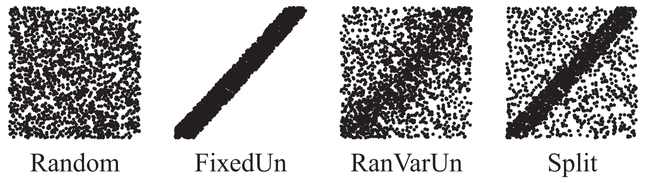

Fig. 2. Requirement 3: 20% of labels queried with different strategies.

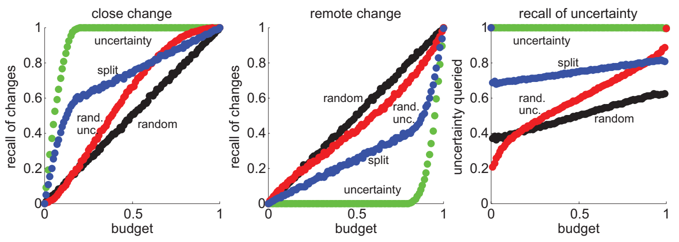

Fig. 3. Performance of the strategies.

data distribution. Each strategy labels the same number of instances (20%).

The Random strategy labels uniformly from the instance space, while the uncertainty strategy concentrates around the decision boundary. The randomized uncertainty as well as the Split strategy infuses randomization into the uncertainty sampling to cover the full instance space. The Split strategy assumes that the probability of a change happening anywhere in the instance space is uniform. The Randomized Uncertainty strategy samples more smoothly from the instance space than the Split strategy under the assumption that changes near the decision boundary are more likely. The Split strategy is a combination of the Random and the Uncertainty strategies.

B. Ability to Notice Changes

Next we investigate how well the strategies capture concept drift by analyzing what share of the changed labels would be revealed by our strategies. Fig. 3 (left and center) plots the proportion of the changed (black) instances queried by the strategies. The plots can be interpreted as recall of changes, which is computed as $H = q_{ch}/Q$, where Q is the total number of labeled instances and q_{ch} is the number of labeled instances that are affected by drift. In this evaluation, we aim to establish a point-in-time evaluation, thus we do not retrain the classifier after each instance. The Fixed Uncertainty strategy is omitted, because it does not have a mechanism to control the labeling budget. Besides, the Fixed Uncertainty strategy handles the changes in the same way as the Variable Uncertainty strategy; only the budget is handled differently.

Comparing the close and the remote change plots, we can see, as expected, that the Random strategy performs equally well independently of where changes occur. On the other hand, the uncertainty strategy is very sensitive to where the changes occur. If changes occur far from the decision boundary, the uncertainty strategy completely misses them and it will never know that the classifier continues making mistakes there. However, the uncertainty strategy captures changes perfectly well if they occur close to the decision boundary; it scores the best of all (100%). The randomized uncertainty reacts to the close changes worse than the uncertainty, but it does not miss the remote changes. The randomized uncertainty and the Split strategies perform similarly on the remote change, while the Split strategy is better at a close change.

C. Learning in Stationary Situations

Active learning strategies in data streams need not only handle changes but also aid the learning to be more accurate. We compare the strategies in terms of queried uncertainty, assuming that the most informative instances in stationary situations lie closest to the decision boundary. Fig. 3 (right) plots the queried uncertainty by each strategy against the labeling budget. The plot can be interpreted as recall of uncertainty; the higher the better. We measure it as

$$U = 1 - \frac{u_q - \min u}{\max u - \min u} \quad (17)$$

where $u_q = \sum_{X\text{queried}} \hat{p}(y|X)$ is the sum of the posterior probabilities of all the queried instances, $\min u$ and $\max u$ are the minimum and the maximum possible u_q from our dataset. First, we rank all $\hat{p}(y|X)$ in the dataset. Then, we query the top n instances to get $\min u$. Next we query n last instances to get $\max u$. Finally, we query n instances according to a chosen strategy to get u_q.

The plot confirms that the Variable Uncertainty always queries the most uncertain instances, thus it is expected to perform well in stationary situations. The Random strategy recalls the least, except for very small budgets, where the randomized Variable Uncertainty strategy recalls even less. This happens because at small budgets the threshold is very small, and therefore nearly all randomization attempts override the threshold and query further away from the decision boundary than random. The Randomized Variable Uncertainty strategy becomes more effective as the budget increases. The Split strategy recalls exactly the average of uncertainty as recalled by the Variable Uncertainty and the Random strategies, as these two strategies are combined into the Split strategy. Notice, that the higher the budget, the more similar the performance, since many of the queried instances overlap.

V. Experimental Evaluation

After analyzing our strategies, we empirically evaluate their performance along with the baselines. We compare five techniques: Random (baseline), Fixed Uncertainty (baseline), Variable Uncertainty, Variable Randomized Uncertainty, and Selective Sampling. Our implementation of Selective Sampling is based on [16], and uses a variable labeling threshold $b/(b+|p|)$, where b is a parameter and p is the prediction of the classifier. The threshold is based on certainty expectations, and the labels are queried at random. As they did not explicitly consider concept drift in [16], we add change detection to the base classifier to improve its performance.

We evaluate the performance on real streaming classification problems. We use as default parameters $s = 0.01$ and $\delta = 1$. All our experiments are performed using the MOA data stream software suite [20]. MOA is an open source software framework in Java designed for online settings as data streams.

TABLE II
SUMMARY OF THE DATASETS

	Instances	Attributes (nom + num)	Classes	Labels
Electricity	45 312	8 (1 + 7)	2	Original
Cover Type	581 012	54 (44 + 10)	7	Original
Airlines	539 383	7 (5 + 2)	2	Original
IMDB-E	120 919	1000	2	Assigned
IMDB-D	120 919	1000	2	Assigned
Reuters	15 564	47 236	2	Assigned

We use in our experiments an evaluation setting based on a prequential evaluation: each time we get an instance, first we test it, and if we decide to incur the cost of its label, then we use it to train the classifier.

A. Datasets

We use six classification datasets as presented in Table II. Electricity data [21] is a popular benchmark in evaluating streaming classifiers. The task is to predict a rise or a fall in electricity price (demand) in New South Wales (Australia), given recent consumption and prices in the same and neighboring regions. Cover Type data [22] is also often used as a benchmark for evaluating stream classifiers. The task is to predict forest cover type from cartographic variables. As the original coordinates were not available, we ordered the dataset using the elevation feature. Inspired by [23], we constructed an Airlines dataset using the raw data from U.S. flight control. The task is to predict whether a given flight will be delayed, given the information of the scheduled departure.

IMDB data originates from the MEKA repository.[3] The instances are TF-IDF representations of movie annotations. Originally, the data had multiple labels that represented categories of movies. We use binary labels. At a given time, we select categories of interest to an imaginary user, and the movies of that category get a positive label. After some time, the interest changes. We introduce three changes in the data stream (after 25 000, 50 000, and 75 000 instances). We construct two labelings: for IMDB-E (easy) only one category is interesting at a time; for IMDB-D (difficult) five related categories are interesting at a time, for instance, crime, war, documentary, history, and biography are interesting at the same time.

The Reuters data is from [24]. We formed labels from the original categories of the news articles in the following way. In the first half of the data stream, legal/judicial is considered to be relevant (+). In the second half, the share listings category was considered to be relevant. The categories were selected to make a large enough positive class (nearly 20% of instances had a positive label).

The first three datasets (prediction datasets) have original labels. Although we expect concept drift, it is not known with certainty when and if changes take place. The other three datasets (textual datasets) represent recommendation tasks with streaming data. Every instance is a document in TF-IDF representation. We form the labels of interest from the categories of the documents.

[3]Available at http://meka.sourceforge.net/.

B. Accuracies on Prediction Datasets

First we test on the three prediction datasets using the Naive Bayes and Hoeffding tree as base classifiers. Here and in the next experiments we use the default decay parameter $w = 100$. Figs. 4 and 5 plot the accuracy of a given strategy as a function of the labeling budget. Fixed Uncertainty is not included in this figure, since it fails by a large margin. In the data stream scenario, it gives around 50%–60% accuracy, which is equivalent to predicting that all labels are negative.

There are several implications following from these results. First, the strategies that use a variable threshold (Selective Sampling, Variable Uncertainty, Randomized Uncertainty, and the Split strategy) outperform the fixed threshold strategies (Fixed Uncertainty), as expected in the data stream setting. Second, the strategies with randomization mostly outperform the strategies without randomization, which suggests that the changes that occur are far from the decision boundaries and there is a justified need for querying tailored to data streams rather than conventional uncertainty sampling. That supports our strategies. Note that, as the Selective Sampling strategy is implemented in our experiments using change detection, it shows good performance on these datasets. However, there is always at least one of the new strategies that outperforms it. The results are consistent when tested with two adaptive classifiers (Naive Bayes with change detector and the Hoeffding tree).

We observe that the Split strategy, which is an intelligent strategy satisfying all three requirements, performs well at small budgets. This is consistent with our expectations that at small budgets the intelligent labeling efforts would concentrate around the decision boundary and easily miss concept drift happening further from the decision boundary. The Split strategy effectively prevents that. The Variable Uncertainty strategy dominates with the Airline dataset. We have observed that this dataset does not contain much of concept drift, thus this performance is consistent with our expectations, since the Variable Uncertainty strategy does not waste labeling resources for querying very certain instances in order not to miss the drift. The Electricity dataset is known for having concept drift. On this dataset, the strategies with randomization (including the random baseline) perform well, particularly at small budgets. That is explainable, as at small budgets Variable Uncertainty samples only a few instances that are very close to the decision boundary. In such a case, randomization helps to capture changes better. All the plots exhibit rising accuracy as the budget increases, which is to be expected. If there was no upward tendency, then we would conclude that we have excess data and we should be able to achieve a sufficient accuracy by a simple random subsampling.

C. Accuracies on Textual Datasets

Textual datasets are very different in nature from the previous prediction datasets. The former are high dimensional and sparse. We are interested in the performance of our active

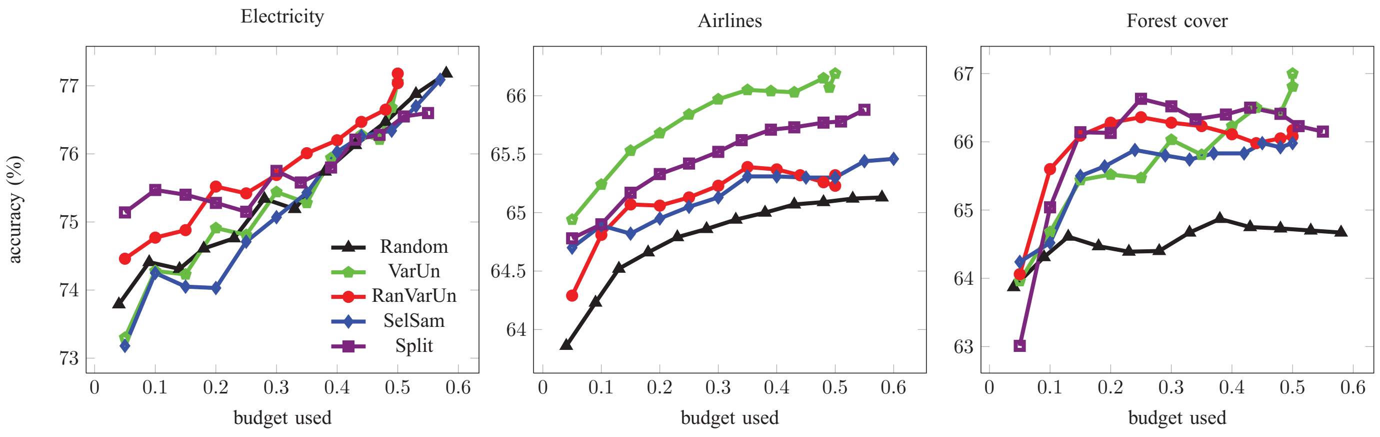

Fig. 4. Accuracies given a budget on prediction datasets (Naive Bayes).

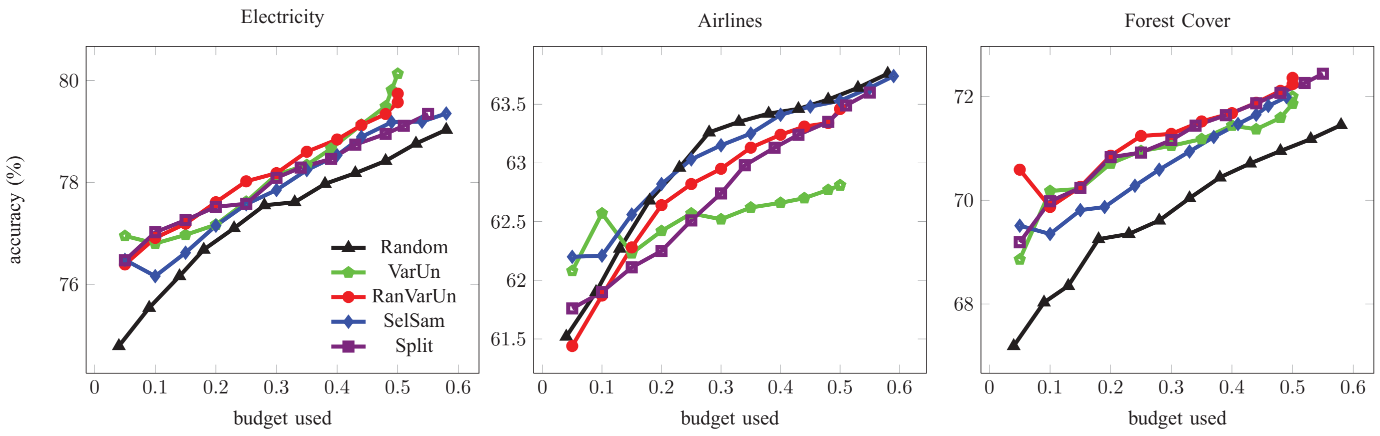

Fig. 5. Accuracies given a budget on prediction datasets (Hoeffding tree).

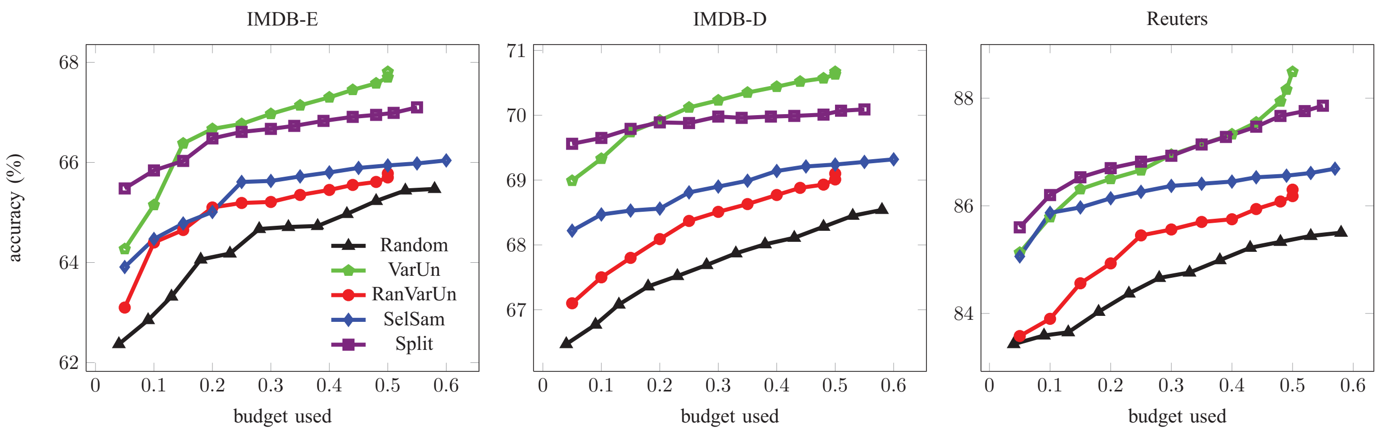

Fig. 6. Accuracies given a budget on textual drifting datasets (Multinomial Bayes).

learning strategies on such datasets, since labeling such data is typically very labor intensive and such learning tasks in stationary settings often use active learning.

The classification tasks in these textual datasets are hard, and often the results are close to the majority class, which would mean no recommendation is given by a classifier. For textual datasets we use the Multinomial Naive Bayes classifier [25]. This version is tailored to sparse textual data.

Fig. 6 presents the performance on the three textual datasets. Experimental results demonstrate that the new techniques are especially effective when the labeling budget is small. We see that the Split strategy consistently performs well at small budgets, and the Variable Uncertainty strategy takes over at higher budgets. Since we know that the textual data has concept drift, this performance is consistent with our expectations. At small budgets, randomization helps not to miss the changes happening far from the decision boundary.

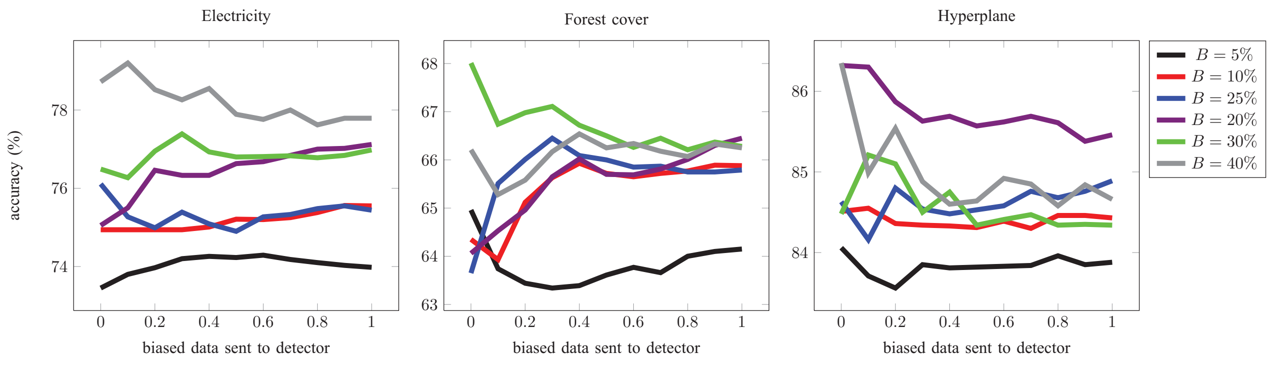

Fig. 7. Accuracies with different rate of the Variable Uncertainty labeled data sent to the change detection (Naive Bayes).

The randomized Variable Uncertainty does not perform that well likely due to a suboptimal variance of the randomization.

D. Change Detection

Requirement 3 presented for the active learning strategies requires the preservation of the incoming data distribution for the purpose of change detection. In this section we empirically test whether preserving distribution matters when learning from real data streams. The challenge for this test is that we do not know if the real datasets actually contain changes. We expect the Electricity data to contain more abrupt seasonal changes with virtual concept drifts and the Forest Cover data (which has been sorted according to elevation) to contain incremental changes. Moreover, in real datasets we are not certain if real or virtual concept drift happens. For completeness of the experiment we also include a synthetic hyperplane dataset where we have only real concept drift; the input data distribution does not change over time. For that we generate a data stream in a square in two-dimensional space, where $x_1 \in [-1, 1]$, $x_2 \in [-1, 1]$. The initial decision boundary is set at $x_1 = 0$. We generate 5000 instances from this distribution, then change the decision boundary to $x_2 = 0$, and generate another 5000 instances. This way, the decision boundary rotates 90° after each change.

The experimental protocol is as follows. We use a modification of the Split strategy (which originally is a combination of the random and the Variable Uncertainty strategies), where we can control how much of the Variable Uncertainty labeled data (from 0% to 100%) we send to the change detector. All labeled data is always used for learning the model. The results are presented in Fig. 7. In all cases, in particular on the synthetic hyperplane data, we observe a tendency that accuracy decreases when we send more Variable Uncertainty labeled data to the detector. As this data is sampled non-uniformly from the incoming data distribution, it distorts change detection. This empirically justifies Requirement 3 about preserving incoming data distribution for change detection.

E. Sensitivity of the Label Spending Estimate

We introduce a new label spending control via approximation, where we use a forgetting window w as a parameter. In this section we experimentally explore sensitivity of the

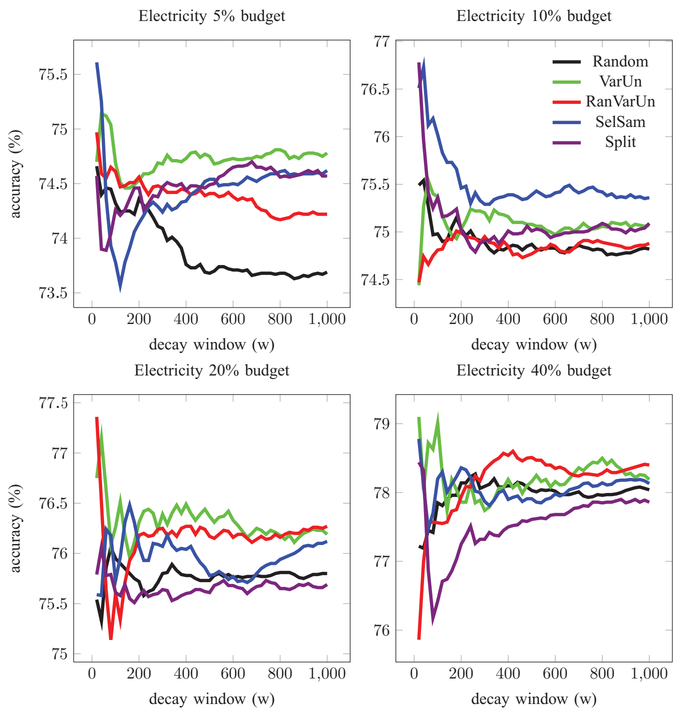

Fig. 8. Sensitivity of performance to the decay window w (Naive Bayes).

performance to various choices of w. We test the variation on the Electricity dataset at different levels of budget $B = 5\%$, $B = 10\%$, $B = 20\%$, and $B = 40\%$. The resulting accuracies of the five strategies (Random, Variable Uncertainty, Random Variable Uncertainty, Selective Sampling, and Split) are presented in Fig. 8. We see that at smaller w the results vary, but the performance becomes more stable at around $w = 300$. Overall, the dependence of the performance on selecting the right value of the parameter w does not seem to be critical.

F. Efficiency

These active strategies reduce the time and space needed for learning, as the classifiers are trained with a smaller number of instances. We can see these active learning strategies as a way to make training of classifiers in a more resource-efficient way. Using 10% or 20% of the instances, we may incur no loss in accuracy but a substantial win in costs of learning a model. These results show that the active learning strategies may be a

TABLE III
ACCURACIES WITH AND WITHOUT ADAPTATION

		Non-adaptive	Adaptive	
Dataset	Classifier	small B	$B = 100\%$	$B = 10\%$
Electricity	Naive Bayes	44%	83%	75%
Electricity	Hoeffding	44%	86%	77%
Airlines	Naive Bayes	49%	65%	65%
Airlines	Hoeffding	55%	65%	62%
Forest	Naive Bayes	39%	67%	66%
Forest	Hoeffding	39%	75%	70%
IMDB-E	Mult. Bayes	63%	68%	66%
IMDB-D	Mult. Bayes	63%	70%	70%
Reuters	Mult. Bayes	82%	88%	86%

good way to speed up the training process of classifiers 5–10 times without losing accuracy.

Table III presents a comparison of accuracies achieved by a non-adaptive classifier pretrained with 500 instances at the beginning [which makes $B \in (0.1\% - 1\%)$] and then training is stopped, with an adaptive classifier with full labeling (no active learning) and a classifier using active learning.

We observe that we can substantially reduce labeling costs at a small or even nearly no loss in accuracy. On the other hand, if we did not employ an adaptive learning procedure, the accuracies would degrade dramatically over time, making the outputs of such classifiers useless. The results suggest that these strategies may be a good way not only to save labeling costs but also to speed up the training process of classifiers while maintaining good accuracy.

We introduced new strategies for active learning tailored to data streams when concept drift is expected. Different strategies perform best in different situations. Our recommendation is to use the Variable Uncertainty strategy if mild to moderate concept drifts are expected. If significant drifts are expected, then we recommend using the Split strategy. In practice we find that drifts can be captured even though the assumption of *iid* data is violated.

VI. CONCLUSION

We proposed active learning strategies for streaming data when changes in the data distribution are expected. Our strategies are equipped with mechanisms to control and distribute the labeling budget over time, to balance the labeling for learning more accurate classifiers, and to detect changes.

Experimental results demonstrated that the new techniques are especially effective when the labeling budget is small. The best performing technique depends on what changes are expected. If concept drift is expected, then the Split strategy performs very well, in particular when only small labeling resources are available, as it is able to effectively spread the labeling efforts to query over all the instance space. Variable Uncertainty performs well in many real cases where the drift is not that strongly expressed.

This framework provides a foundation for incremental active learning from drifting data streams. Following the new framework, more advanced strategies can be developed. A promising extension of the strategies would be to redistribute dynamically the labeling budget to the regions where changes are suspected.

APPENDIX
BEHAVIOR OF THE VARIABLE UNCERTAINTY STRATEGY

This appendix analyzes the behavior of the Variable Uncertainty strategy (Section III-G). We claim that for large s the strategy approaches the Random strategy, while for small s it approaches the Fixed Uncertainty strategy.

Large s: Let s be large so that $s \to 1$. Recall that $\theta \in [\frac{1}{c}, 1]$, where c is the number of classes. Let us limit this analysis to a binary classification case; then $0.5 \le \theta \le 1$. Three scenarios of operation at time t are possible:

1) labeling budget is not exceeded and certainty of X_t is good;
2) labeling budget is not exceeded and X_t is uncertain;
3) labeling budget is exceeded and X_t is skipped.

Let us start analyzing from option 1, where labeling budget is not exceeded so that $\hat{b}_t \le B$, and the certainty of X_t is good so that $p_L(label|X_t) = 0$. Thus, we do not label X_t, and as a result make $\theta_{t+1} = \theta_t(1+s)$. As $s \to 1$ and $\theta_t \ge 0.5$, then $\theta_{t+1} \to 1$. As a result, (14) becomes

$$p_L(label|X) = \mathbf{1}(P_L(\hat{y}|X) \le \theta) = 1 = p_L(label) \qquad (18)$$

and labeling does not depend on X as in the Random strategy.

Next we may encounter two situations. First, the labeling budget is exceeded, i.e., $\hat{b}_{t+1} > B$. In such a case, we wait j time steps until $\hat{b}_{t+1} \le B$. During that time θ does not change ($\theta_{i+1} = \theta_i$). At time $t+1+j$, a new instance X_{t+1+j} arrives, and it is labeled independently of its uncertainty (18). Thus, the strategy behaves like the Random strategy. Afterwards, the threshold decreases and we are in situation 1 (if θ decreases too much, more iterations will be required).

Second, the labeling budget is *not* exceeded (option 2), i.e., still $(u_t + 1/t + 1) \le B$. However, since $\theta_{t+1} \ge 1$, we are going to label any further incoming instance independently of X. Thus, the strategy behaves like the Random strategy.

In the other limit $\lim_{s\to 0} \theta_{t+1} = \theta_t$ stays fixed, thus the Variable Uncertainty strategy behaves like the Fixed Uncertainty one.

REFERENCES

[1] D. Cohn, L. Atlas, and R. Ladner, "Improving generalization with active learning," *Mach. Learn.*, vol. 15, no. 2, pp. 201–221, May 1994.

[2] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Mach. Learn.*, vol. 23, no. 1, pp. 69–101, Apr. 1996.

[3] I. Zliobaite, A. Bifet, B. Pfahringer, and G. Holmes, "Active learning with evolving streaming data," in *Proc. 2011 Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, Sep. 2011, pp. 597–612.

[4] J. Attenberg and F. Provost, "Online active inference and learning," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2011, pp. 186–194.

[5] R. Klinkenberg, "Using labeled and unlabeled data to learn drifting concepts," in *Proc. Workshop Notes IJCAI'01 Learn. Temporal Spatial Data*, Aug. 2001, pp. 16–24.

[6] D. Widyantoro and J. Yen, "Relevant data expansion for learning concept drift from sparsely labeled data," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 3, pp. 401–412, Mar. 2005.

[7] M. Masud, C. Woolam, J. Gao, L. Khan, J. Han, K. Hamlen, and N. Oza, "Facing the reality of data stream classification: Coping with scarcity of labeled data," *Knowl. Inf. Syst.*, vol. 33, no. 1, pp. 213–244, Oct. 2011.

[8] S. Huang and Y. Dong, "An active learning system for mining time-changing data streams," *Intell. Data Anal.*, vol. 11, no. 4, pp. 401–419, Sep. 2007.

[9] P. Lindstrom, S. J. Delany, and B. M. Namee, "Handling concept drift in a text data stream constrained by high labelling cost," in *Proc. 23rd Florida Artif. Intell. Res. Soc. Conf.*, May 2010, pp. 1–7.

[10] X. Zhu, P. Zhang, X. Lin, and Y. Shi, "Active learning from data streams," in *Proc. 7th IEEE Int. Conf. Data Mining*, Oct. 2007, pp. 757–762.

[11] W. Fan, Y. Huang, H. Wang, and P. Yu, "Active mining of data streams," in *Proc. 4th SIAM Int. Conf. Data Mining*, Apr. 2004, pp. 457–461.

[12] P. Lindstrom, B. M. Namee, and S. J. Delany, "Drift detection using uncertainty distribution divergence," in *Proc. IEEE 11th Int. Conf. Data Mining Workshops*, Dec. 2011, pp. 604–608.

[13] H. Borchani, P. Larranaga, and C. Bielza, "Mining concept-drifting data streams containing labeled and unlabeled instances," in *Proc. 23rd Int. Conf. Ind. Eng. Appl. Appl. Intell. Syst. Trends Appl. Intell. Syst.*, Jun. 2010, pp. 531–540.

[14] P. Zhang, X. Zhu, and L. Guo, "Mining data streams with labeled and unlabeled training examples," in *Proc. 9th IEEE Int. Conf. Data Mining*, Dec. 2009, pp. 627–636.

[15] P. Li, X. Wu, and X. Hu, "Mining recurring concept drifts with limited labeled streaming data," *ACM Trans. Intell. Syst. Technol.*, vol. 3, no. 2, pp. 29-1–29-32, Feb. 2012.

[16] N. Cesa-Bianchi, C. Gentile, and L. Zaniboni, "Worst-case analysis of selective sampling for linear classification," *J. Mach. Learn. Res.*, vol. 7, no. 7, pp. 1205–1230, 2006.

[17] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Proc. 17th Brazilian Symp. Artif. Intell. Adv. Artif. Intell.*, Sep. 2004, pp. 286–295.

[18] B. Settles, *Active Learning* (Synthesis Lectures on Artificial Intelligence and Machine Learning). San Francisco: Morgan Claypool, 2012.

[19] Y. Yuan, "A review of trust region algorithms for optimization," in *Proc. 4th Int. Congr. Ind. Appl. Math.*, Jun. 1999, pp. 271–282.

[20] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: Massive online analysis," *J. Mach. Learn. Res.*, vol. 11, pp. 1601–1604, Jun. 2010.

[21] M. Harries, C. Sammut, and K. Horn, "Extracting hidden context," *Mach. Learn.*, vol. 32, no. 2, pp. 101–126, 1998.

[22] A. Frank and A. Asuncion. (2010). *UCI machine learning repository* [Online]. Available: http://archive.ics.uci.edu/ml

[23] E. Ikonomovska, J. Gama, and S. Dzeroski, "Learning model trees from evolving data streams," *Data Mining Knowl. Discovery*, vol. 23, no. 1, pp. 128–168, Jul. 2011.

[24] D. Lewis, Y. Yang, T. Rose, and F. Li, "RCV1: A new benchmark collection for text categorization research," *J. Mach. Learn. Res.*, vol. 5, pp. 361–397, Apr. 2004.

[25] A. McCallum and K. Nigam, "A comparison of event models for naive bayes text classification," in *Proc. AAAI/ICML'98 Workshop Learn. Text Categorizat.*, Jul. 1998, pp. 41–48.

Indrė Žliobaitė received the Ph.D. degree from Vilnius University, Vilnius, Lithuania, in 2010.

She is currently a Lecturer of Computational Intelligence with Bournemouth University, Poole, U.K., where she is a Research Task Leader of the INFER.eu project. Her current research interests include mining evolving streaming data, change detection, adaptive and context-aware learning, and predictive analytics applications.

Albert Bifet is currently a Researcher with Yahoo! Research Barcelona, Barcelona, Spain. He has authored the book entitled *Adaptive Stream Mining and Pattern Learning and Mining from Evolving Data Streams*. He is one of the core developers of MOA software environment for implementing algorithms and running experiments for online learning from evolving data streams.

Bernhard Pfahringer received the Ph.D. degree from the University of Technology, Vienna, Austria, in 1995.

He is currently an Associate Professor with the Department of Computer Science, Waikato University, Hamilton, New Zealand. His current research interests include data mining and machine learning subfields, with a focus on streaming, randomization, and complex data.

Geoffrey Holmes received the Ph.D. degree from the University of Southampton, Southampton, U.K., in 1986.

He is currently the Dean of the Faculty of Computing and Mathematical Sciences, Waikato University, Hamilton, New Zealand. His current research interests include machine learning, data stream mining, and development of machine learning applications.