Evolving Stream Classification using Change Detection

Ahmad Mustafa*, Ahsanul Haque*, Latifur Khan*, Michael Baron†, Bhavani Thuraisingham*
*Department of Computer Science
†Mathematical Sciences Department
The University of Texas at Dallas, Richardson, Texas
{ahmad.mustafa, ahsanul.haque, lkhan, mbaron, bxt043000}@utdallas.edu

***Abstract*—Classifying instances in evolving data stream is a challenging task because of its properties, e.g., infinite length, concept drift, and concept evolution. Most of the currently available approaches to classify stream data instances divide the stream data into fixed size chunks to fit the data in memory and process the fixed size chunk one after another. However, this may lead to failure of capturing the concept drift immediately. We try to determine the chunk size dynamically by exploiting change point detection (CPD) techniques on stream data. In general, the distribution families before and after the change point are unknown over the stream, therefore non-parametric CPD algorithms are used in this case. We propose a multi-dimensional non-parametric CPD technique to determine chunk boundary over data streams dynamically which leads to better models to classify instances of evolving data streams. Experimental results show that our approach can detect the change points and classify instances of evolving data stream with high accuracy as compared to other baseline approaches.**

I. Introduction

Data streams are continuous flows of data. Typical example of data streams include network traffic, sensor data, and call center records, among others. The sheer volume and throughput speed pose a great challenge for the data mining community to extract useful knowledge from such streams. Data streams demonstrate several unique properties when compared with traditional data set, such as: infinite length, concept-drift, and concept-evolution. Concept-drift occurs in data streams when the underlying concept of data changes over time [1], [18], [25]–[28], [30]. In concept-evolution, a new class may emerge over stream [21]. Neither multi-step methodologies and techniques nor multi-scan algorithms suitable for typical knowledge discovery and data mining can be readily applied to data streams due to well-known limitations such as unbounded memory to handle infinite length, online data processing to handle concept drift, and the need for one-pass techniques (i.e., forgotten raw data).

Data stream classification has been a major research thrust for the past several years because of increasing demand in many business and security applications, such as credit card transaction monitoring, online blog or micro-blog (e.g., twitter messages) categorization, and evolving malicious code detection. Traditional batch classification techniques are not applicable to the aforementioned domains because of the evolving nature of the data. In particular for malware detection, signature-based techniques are widely used and dynamic signature updating is very common. Antivirus software can adapt to new malware threats by updating its signature database as soon as a single instance of the malware has been identified and analyzed by experts. In contrast, polymorphic malware can pose some significant challenges to signature updating because the malware modifies itself during propagation yielding many variants over time. A malware detector may fail to identify such malware due to the usage of outdated signatures. One way to address this problem is to update the signatures, which achieves superior adaptability over current polymorphic malware. While this advantage has kept antivirus products mostly ahead in the virus-antivirus co-evolution race [16] up to the present time, a malware detector needs to be adaptive to cope with the changes in the wild.

Stream classification falls into the following categories: single model, ensemble classification, and hybrid. Single model classification techniques maintain and incrementally update the single classification model and effectively respond to concept drift [29]. In ensemble based techniques, a number of classification models are maintained, and over time some outdated classification models are replaced by new models [21]. Hybrid methods combine the strength of the above two [10]. In current state of the art ensemble techniques, data stream is divided into a number of chunks so that each chunk can be accommodated in memory and processed online [21], [22]. Each chunk is used to train one classification model as soon as all the instances in the chunk are labeled. Concept-drift is handled by maintaining an ensemble of M such classification models. An unlabeled instance is classified by taking a majority vote among the classifiers in the ensemble. The ensemble is continuously updated so that it represents the most recent concept in the stream. The update is performed as follows. As soon as a new model is trained, one of the existing models in the ensemble is replaced by it, if necessary. The removed model is chosen by evaluating the error rate of each of the existing models in the ensemble on the latest labeled chunk, and discarding the one with the highest error rate.

Almost all the current state of the art stream classification techniques divide data stream in equal size (i.e., fixed size) [1], [21], [22]. So, these approaches fail to capture the concept drift/concept evolution immediately. Moreover, if the chunk is too small, these approaches may introduce poor quality model (few training data points) and/or additional computational overhead to update the ensemble. On the other hand, for large chunk sizes, these approaches have to wait much longer to build the next classifier. As a result, the ensemble is updated less frequently than desired, meaning the ensemble remains outdated for a longer period of time. This ultimately causes

COLLABORATECOM 2014, October 22-25, Miami, United States

DOI 10.4108/icst.collaboratecom.2014.257769

increased error rates.

To address the above mentioned problems, we have focused to determine the chunk size dynamically so that we can keep track of the change immediately. To do this, we have exploited change point detection (CPD) approaches over stream data. Most of the CPD algorithms assume that distributions of data before and after the change are known [3], [12], [17]. However, in real life, these assumptions may not be a valid one. If the distributions of data before and after are unknown entirely, these approaches will not be applicable. In this case, methods have been developed for detecting change points non-parametrically over single-dimensional data [2], [5], [9], [13], [14]. If data is multi-dimensional, these approaches have two major shortcomings. First, there is no efficient mechanism to combine all the individual one-dimensional score into one final score. Second, it is difficult to set the threshold for the multi-dimensional data that will control the rate of false alarms at the desired level.

In this paper, we have proposed a novel Stream Classification using Change Detection (SC^2D) method. We use decision tree to model changes across multiple dimensions and classify instances. The dimensions contribute to the prediction based on their normalized information gain using decision tree. So, using this approach we do not need to manually set a threshold. Decision tree partitions and selects the dimensions based on their information gain. This proposed approach is then used to determine chunk boundary over data streams dynamically which leads to better models to classify instances of evolving data streams. Since it uses dynamic chunk size based on change in the distribution of the data, it can capture drift in the concept or detect arrival of a new class immediately. In this paper, we focus on classifying instances of data streams having two classes e.g., benign and malicious. Our adaptive stream classification technique is beneficial to various communities, including those working in cyber security to analyze massive amounts of stream data e.g., differentiate between benign and malicious events.

Primary contributions of this paper are as follows. To the best of our knowledge, this is the first effort that develops an adaptive classification technique exploiting multi-dimensional non parametric change point detection to address concept-drift/concept-evolution problems in a timely manner. Our approach exploits dynamic sized chunk. In other words, chunk size is determined over stream on the fly based on multi-dimensional non-parametric change point detection. Once a change point is detected, the algorithm continues to find subsequent change points. Two subsequent change points create a chunk with variable length. In this paper, we focus on classifying data streams having instances from two classes. We consider a scenario where instances from each class come in a sequence and class of the sequences change alternatively. In this case, a dynamic chunk determined by detecting change point represents instances from the same class. Our approach determines the class label of a chunk by calculating the change point and taking into account previous chunk's class labels. It reduces false alarm rates and overall classification error. We test performance of our approach on several benchmark datasets. We compare the experimental results with both supervised and unsupervised techniques. Supervised techniques include various state of the art approaches implemented in MOA [7] framework. On the other hand, for unsupervised techniques, we use variants of multi dimensional change point detection approaches.

The rest of the paper is organized as follows. In Section II, we briefly discuss various approaches for change point detection and some related terms. In Section III, we point out the problems of change point detection approaches available currently. Then we describe our approach and how it can solve those problems. Section IV demonstrates the experimental results and efficiency of our proposed approach. Section V briefly describes some earlier works related to data stream mining and change point detection. Finally, Section VI concludes our discussion along with some future research directions.

II. Background

In this paper, we focus to classify instances in stream of data using change point detection technique. We use change point detection to capture concept drift in timely manner by determining chunk size dynamically. We intend to look into few details of both concept drift and change point detection in this section.

A. Concept Drift

Concept drift refers to scenario when the relation between input data X, e.g., feature values and the target variable y, e.g., class label changes over time. Formally, concept drift between time t_0 and time t_1 can be defined as

$$\exists X : p_{t0}(X, y) \neq p_{t1}(X, y) \tag{1}$$

where $p_{t0}(X, y)$ denotes the joint distribution between the set of input variables X and the target variable y at time t_0 and $p_{t1}(X, y)$ denotes the same at time t_1 [15]. The components of this relation are the prior probabilities of classes $p(y)$, the class conditional probabilities $p(X|y)$, distribution of incoming data $p(X)$ and the posterior probabilities of classes $p(y|X)$. $Real\ concept\ drift$ refers to the changes in posterior probabilities $p(y|X)$. It may happen for several reasons including changes in incoming data (i.e., $p(X)$). However, if there is a change in incoming data without affecting $p(y|X)$, it is called a $virtual\ drift$.

B. Change Point Detection Algorithms

Change point detection (CPD) is used to detect abrupt changes in characteristics of data at unknown time instants. Abrupt change means when changes in characteristics occur very fast with respect to the sampling period of the measurements. Change point detection is particularly very useful for quality control, system monitoring, and fault detection. In this paper, we use this technique to detect changes in distribution parameters of the data points collected from the system at different timestamps. This is then used to separate malicious instances from benign data instances.

Let $\{x_j\}$ be a sequence of data points where x_j has a distribution f for $j \leq \nu$ and a distribution g otherwise. The problem of change point detection is to discover a change and to estimate the change point parameter ν from these data. CPD

algorithms in general can be divided into two categories, i.e., parametric and non-parametric algorithms. Parametric CPD algorithms assume that the distribution families before and after the change point are known beforehand. However, in many applications, one may know the distribution before the change, when the process is "in control", but it is usually impossible to know the distribution after the change, when the process goes "out of control". Sometimes, it may also be the case that both distribution families before and after the change point are unknown. In those cases, non-parametric change point detection algorithms are used to detect the change point in data.

Several non-parametric change point estimation approaches have been proposed in the literature [2], [11], [14], [24]. Usually, the pre- and post-change empirical distributions are compared for each $k = 1, \ldots, n$, where n is the number of data points. Then the point $\hat{\nu}$ that maximizes some predefined "metric" or "measure of diversity" between these distributions is used as an estimator of the actual change point ν. For example, E. Carlstein [11] proposes a mean-dominant norm to be used as such a "metric" and achieves the accuracy rate of $|\hat{\nu} - \nu| = O_p(n^{1/2+\epsilon})$. L. Dümbgen [13] uses a seminorm and achieves the rate $O_p(1)$, which is similar to the maximum likelihood estimator (MLE) for the case of known distributions. U-type statistics are used by D. Ferger [14], achieving the rate of $O_p(\log n)$.

M. Baron [2] proposes both off-line and online versions of an efficient MLE-type non-parametric change point detection algorithm with an unbeatable $|\hat{\nu} - \nu|$ rate of $O_p(1)$. These algorithms are based on a log-likelihood ratio random walk

$$S_k = \sum_{j=1}^{k} \log \frac{f}{g}(x_j). \tag{2}$$

In non-parametric setup, it is assumed that any or both of distribution families f and g are unknown. In the algorithms proposed by M. Baron [2], histogram density estimation procedures are used to estimate f and g for any potential change-point k. Histogram density estimators are chosen as they allow to detect changes in the distribution of categorical and ordinal data as well. Let $\mathcal{X}$ be the united support of f and g with a measure μ on it, and let $\{A_m,\ m = 1 \ldots, r\}$ be a μ-measurable partition of $\mathcal{X}$ of rank r. If supports are not known, an intentionally larger set $\mathcal{X}$ can be taken. For $k = 1, \ldots, n$, introduce the histogram density estimates

$$\begin{aligned} \hat{f}_k(x) &= \frac{1}{k\mu(A_m)} \sum_{m=1}^{r} \sum_{j=1}^{k} I\left\{x \in A_m, x_j \in A_m\right\}, \\ \hat{g}_k(x) &= \frac{1}{(n-k)\mu(A_m)} \sum_{m=1}^{r} \sum_{j=k+1}^{n} I\left\{x \in A_m, x_j \in A_m\right\} \end{aligned} \tag{3}$$

The choice of a reference measure μ is arbitrary because the likelihood ratios are independent of it, and only the data counts in each bin A_m enter the equation. A counts-based maximum likelihood estimator of a change point can then be formed by substituting the estimates (3) in equation (2),

$$\hat{\nu} = \arg\max_k \hat{S}_k \tag{4}$$

where

$$\hat{S}_k = \sum_{j=1}^{k} \log \frac{\hat{f}(x_j)}{\hat{g}(x_j)} = \sum_{j=1}^{k} \log \frac{\sum_{i=1}^{k} \delta(\xi_i, \xi_j)/k}{\sum_{i=k+1}^{n} \delta(\xi_i, \xi_j)/(n-k)}, \tag{5}$$

Here $\xi_j = (\xi_j^{(1)}, \ldots, \xi_j^{(r)})$, $\xi_j^{(m)} = I\left\{x_j \in A_m\right\}$, for $m = 1, \ldots, r, j = 1, \ldots, n$, and $\delta(x, y) = I\left\{x = y\right\}$; therefore $\delta(\xi_i, \xi_j) = 1$ when observations x_i and x_j fall into the same bin of partition $\mathcal{X}$. It appears that when unknown densities are replaced by their histogram estimators, the change point problem reduces to estimating a change point for the multinomial distribution of ξ_j. Moreover, it can be shown that $\hat{S}_k$ is proportional to the Kullback-Leibler information κ between the estimated multinomial distributions before and after the time k. Hence, the non-parametric estimator $\hat{\nu}$ is actually the point that maximizes the Kullback-Leibler divergence between empirical pre- and post-change distributions.

An online version of the algorithm consists of a stopping time T, which signals a change in distribution "as soon as possible" after it occurs. An optimal stopping time T must achieve the best trade-off between minimizing the mean delay

$$\mathrm{MD}(T) = E_\nu (T - \nu)^+ \tag{6}$$

and maximizing the mean time between false alarms

$$\mathrm{MTBFA}(T) = E_{\nu=\infty}(T) = E\{T | \text{no change}\} \tag{7}$$

M. Baron [2] proposes a nonparametric analogue of the CUSUM algorithm ([3], sec. 5.2), whose asymptotic performance in the sense of (6) and (7) is similar to that in the case of known distributions.

First, in the case of known pre-change distribution f and completely unknown post-change distribution g, the density of g in (2) is replaced by a histogram density estimator (5). The CUSUM process is then estimated by

$$\hat{W}_n = \max_{1 \le k < n} \hat{S}_{k:n},$$

where

$$\hat{S}_{k:n} = (n-k) \sum_{m=1}^{r} \hat{p}_{k:n}^{(m)} \log \frac{\hat{p}_{k:n}^{(m)}}{p^{(m)}} \tag{8}$$

is the "tail" of the generalized log-likelihood ratio process. In the above equation, $\hat{p}_{u:v}^{(m)} = \left\{\xi_{u+1}^{(m)} + \ldots + \xi_v^{(m)}\right\}/(v-u)$ is a histogram density estimator computed from a sub-sample $x_{u+1}, \ldots, x_v$ for $0 \le u < v \le n$. Next, a positive threshold h is chosen and a stopping time is computed as follows,

$$\hat{T}(h) = \inf\left\{n :\ \hat{W}_n \ge h\right\} \tag{9}$$

Noticeably, relatively small samples may have no data in some of the bins, in which case $\hat{p}_{u:v}^{(m)} = 0$ and $\hat{S}_k = \pm\infty$. A smoothed version of density estimators can then be used. Let

s be the number of empty bins, which is the number of zeros among $\hat{p}_{k:n}(1), \ldots, \hat{p}_{k:n}(r)$. Then, define

$$\tilde{p}_{k:n}^{(m)} = \begin{cases} \hat{p}_{k:n}^{(m)}, & \text{if } s = 0 \\ \epsilon/s(n-k), & \text{if } \hat{p}_{k:n}^{(m)} = 0 \\ (1 - \epsilon/(n-k))\, \hat{p}_{k:n}^{(m)}, & \text{if } s > 0;\ \hat{p}_{k:n} \neq 0 \end{cases} \tag{10}$$

for some $0 < \epsilon < 1$. This definition still preserves $\sum_m \tilde{p}_{k:n}^{(m)} = 1$.

Finally, if both distributions f and g are unknown, the change point problem gets fundamentally more difficult. In this case, it is possible to miss the actual change point and proceed without detecting it. In order to achieve the optimal asymptotic performance in terms of (6) and (7), the process needs to be observed for a sufficient period before ν. Indeed, for any finite interval $[s; t]$, if f is estimated by $\hat{f}$ from $(x_s, \ldots, x_t)$, then with a positive probability $\hat{f} \approx g$, and no change can be detected. That is why $\hat{g}$ should not be compared with $\hat{f}$ for small k. In the method proposed by [2], this issue is addressed by redefining the CUSUM-type process as

$$\hat{W}_n = \max_{\gamma n \leq k < n} \hat{S}_{k:n} \text{ for some } \gamma \in (0; 1), \tag{11}$$

where a nonlinear function $\gamma(n)$ can also be used in place of γn. It signifies that when both of the distribution families are unknown, changes can be reported at most once per every γn observations. If a change at the time ν is not detected before the time ν/γ, then for all $k > \nu/\gamma$, the density f is estimated from a sample, at least $100(1 - \nu/\gamma n)$ percent of which comes from the distribution g. This portion increases and tends to 100% as $n \to \infty$, which makes it impossible to detect the change point if it occurred far in the past. So, for $n >> \nu$, $\hat{W}_n$ behaves like the no-change situation, when all data follow the distribution g. Therefore, if a change point is not detected before ν/n, $\hat{T}(h)$ will increase exponentially fast in h, like the mean time between false alarms. Thus, an important characteristic of a stopping time is $\text{pr}\{T > \nu/\gamma\}$, which is the probability to fail to detect the change point. For the proposed algorithm, this probability is exponentially small, when n is large.

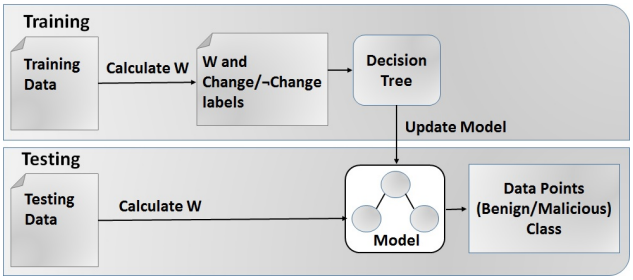

Fig. 1. Block diagram of SC^2D

With these modifications, in case when both f and g are unknown, the smoothed density estimator $\tilde{p}_{0:k}$ is defined similarly to equation (10), guaranteeing $\tilde{p}_{0:k}^{(m)} > 0$ and $\sum_m \tilde{p}_{0:k}^{(m)} = 1$. Then $p^{(m)}$ in equation (8) is replaced by $\tilde{p}_{0:k}^{(m)}$. CUSUM-type process $\hat{W}_n$ introduced in equation (11) along with the stopping time defined in equation (9) are then used to detect change point when both of the distribution families are unknown.

III. Problem and Approach

A. The Problem

Most of the state of the art approaches to classify instances in evolving data stream divide stream data into fixed size chunks to accommodate data into memory and to process without storing data. However, using fixed sized chunks has some disadvantages. First, this may lead to failure of capturing the concept drift/concept evolution immediately. Second, quality of model may depend on size of the data chunks. Based on the size of data chunk, it may cause under-fitting or over-fitting problem. So, in this paper we focus to use change point detection techniques to fix the chunk size dynamically.

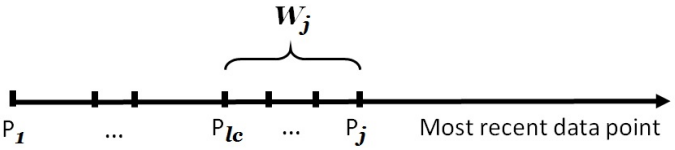

Fig. 2. Calculation of cusum type process score

As discussed in Section II-B, Change Point Detection (CPD) algorithms are generally efficient when used on one dimensional data. However, real life datasets typically have a large number of dimensions (e.g., network security datasets). In the case of multi-dimensional datasets, CPD algorithm is typically applied separately on each dimension. Thus, if there are d number of dimensions in the dataset, $w^{(1)} \ldots w^{(d)}$ are computed, where $w^{(i)}$ is the score of CUSUM-type process on dimension i using equation (11). A final score $w^{(total)}$ is then computed from the CUSUM-type process scores calculated on individual dimensions. There are several ways to compute $w^{(total)}$.

B. Our Approach

One way of calculating $w^{(total)}$ is to take the summation of all the one-dimensional scores $w^{(1)}, \ldots, w^{(d)}$

$$w^{(total)} = \sum_{i=1}^{d} w^{(i)}$$

which we refer to as CP-SUM. This is motivated by the joint log-likelihood of multi-dimensional data if components of the observed vectors are independent and their log-pseudo-likelihood in the general case. To continue controlling for the rate of false alarms, the threshold h will be replaced accordingly by $d * h$.

Another way of calculating $w^{(total)}$ is to take the maximum of $w^{(1)}, \ldots, w^{(d)}$

$$w^{(total)} = \max_{1 \leq i \leq d} w^{(i)}$$

which we refer as CP-MAX. This scheme is obtained by the Bonferroni approach for multiple comparisons, where the probability of a false alarm along at least one dimension is to be controlled at the given level. According to it, a change point

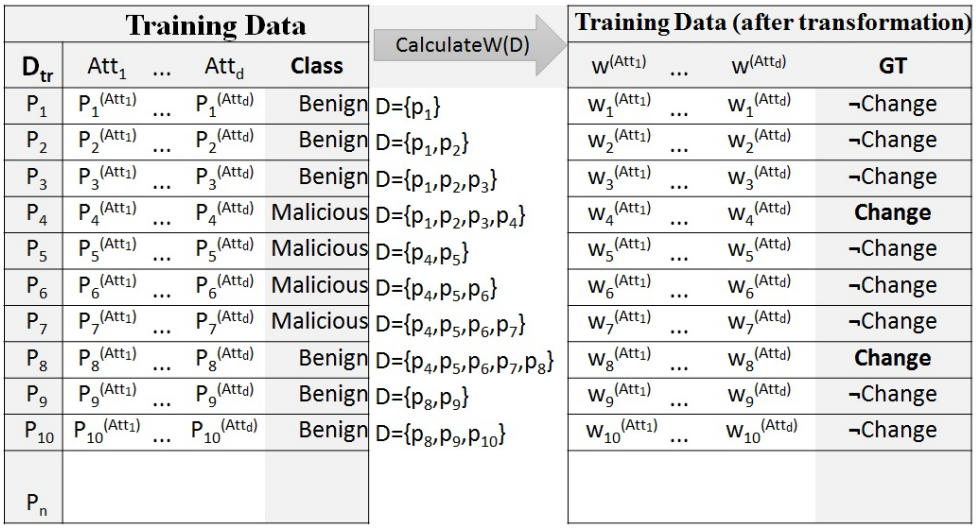

Fig. 3. Example of Training

will be detected when the most significant of the observed CUSUM values $w^{(1)}, \ldots, w^{(d)}$ exceeds the threshold h.

This final CUSUM-type score $w^{(total)}$ is then compared with a suitably chosen and pre-defined threshold to detect a change point. There are two major problems associated to these techniques. First, these techniques assume that all the dimensions have equal confidence to detect the change point which is not the case in practical situations. For example, in cyber security datasets, where malicious instances typically appear in between sequence of benign instances, different dimensions, i.e., features may have different discriminating abilities. Some features have similar values across all the benign and malicious instances. On the other hand, some features may have substantially different range of values in malicious instances than in benign instances. These later features are more useful to detect change between benign to malicious or vice versa for having more ability to discriminate between different classes. So, discriminating features should be given more importance, i.e., weight while detecting the change point. Second, we need to manually set a threshold to compare with the $w^{(total)}$ to detect the change point. However, it is difficult to set a threshold as both the range and the rate of change for $w^{(total)}$ are unpredictable.

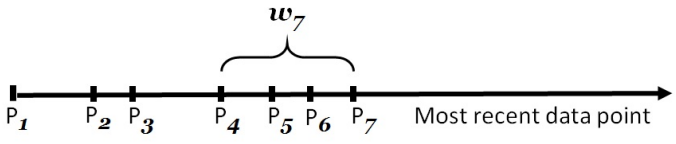

Fig. 4. Example of Training on timeline

We propose a novel Stream Classification using Change Detection (SC^2D) approach which uses the idea of non-parametric change point detection method as discussed in Section II-B. SC^2D has two phases, i.e., training and testing. Figure 1 shows the phases of the proposed approach. In the training phase, it calculates a matrix W on training data points. Each row of this matrix is a vector which corresponds to CUSUM type process scores calculated on different dimensions of a data point. For example, the i^{th} row of matrix W is a vector W_i which corresponds to training point P_i. W_i consists of scores $w_i^{(1)} \ldots w_i^{(d)}$ where d is the number of dimensions. Furthermore, our approach assigns either $change$ or $no\ change$ as a ground truth label to each training data point. If the class of data point P_i is same as the class of data point P_{i-1}, SC^2D assigns label $no\ change$ to point P_i to indicate this as not a change point. On the other hand, if the class of data point P_i is different than the class of data point P_{i-1}, SC^2D considers point P_{i-1} as a change point and assigns label $change$ to it. For instance, a malicious training data point is labelled as change point if it is preceded by a benign data point. While calculating W_j vector for any point P_j, SC^2D considers only the points P_{lc} to P_j as shown in Figure 2, where P_{lc} is the last change point.

After calculating matrix W and assigning corresponding ground truth labels to each of the training data points, a decision tree model is built using both W and assigned ground truth labels. To build the decision tree, each dimension of the matrix W is divided into multiple partitions. For each of these partitions, decision tree computes the information gain. Then the decision tree is built by selecting the dimensions in the ascending order of information gain along with the corresponding partitions.

An example of a training phase on a network security dataset is shown in Figure 3. Let's assume that this dataset contains data points from two classes, i.e., Benign and Malicious. Each point P_i in D_{tr} is represented by a vector of attribute values $P_i^{Att_1} \ \ldots \ P_i^{Att_d}$ and a class label either $Benign$ or $Malicious$. As stated earlier, our proposed approach assigns either $no\ change$ or $change$ as label to each data point. $no\ change$ is assigned as ground truth label to data point P_i if its class label is different from previous data point P_{i-1}. Ground truth label $change$ is assigned if class label of data point P_i is different than class label of point P_{i-1}. In the given example, data points P_4 and P_8 are change points since these have different class labels than the previous data points. This is why the proposed approach assigns label $change$ to

each of these data points. These labels are assigned under column GT. Beside assigning labels, the proposed approach calculates CUSUM type process score for each attribute at each point. While calculating this score for a specific point, all the points from the last change point to the current point are considered. For all attributes i of point P_7 $w_7^{Att_i}$ is calculated by considering all the points from last change point P_4 to the current point P_7.

As shown in Figure 4, During the testing phase, the decision tree model formed in training phase is used to predict the $change/no\ change$ labels of the testing data points. SC^2D then predicts the class label for each data point based on the prediction of $change/no\ change$ label and class label predicted to the previous data point. Since we focus on data streams containing two classes in this paper, if a point is predicted as a change point then its class label is the opposite of the class of preceding data point. Otherwise, current data point is predicted to have the same class as the previous data point.

As described above, a decision tree model is built during the training phase. However, as the data stream evolves, the feature values and class boundaries may change. So, we update the decision tree frequently using the new data. Several approaches can be followed to update the model. One possibility is to append the new labeled data to the old points then use both of them to build a new decision tree. However, the number of data points in the stream is huge. Therefore, storing all the data is not possible. So, instead of keeping all the historical data we store a sample of the old data. Then we use the sample along with the new data to build a new decision tree. Keeping an ensemble of updated models is infeasible because the number of new $change$ points is significantly less than the number of $no\ change$ points. In other words, the new data is skewed and do not contain enough $change$ points to learn from.

We assume that there exist at least three $no\ change$ points between each two $change$ points. Therefore, we do not calculate the change score w for the first three points in training and testing phases. In other words, we leave a cushion of three points because change detection is very sensitive when the distributions are calculated from few points.

Algorithm 1: Algorithm for Training

input : Training Dataset D_{tr}, that contains data points from two classes
output: The Decision tree Model M

```
1  begin
2      D ← ∅;
3      for each point P ∈ D_tr do
4          D ← D ∪ P;
5          w_p ← CalculateW (D);
6          gt_p ← IsChangeOfClass (P);
7          if gt_p is true then
8              D ← {P};
9          W ← W ∪ w_p;
10         GT ← GT ∪ gt_p;
11     M ← TrainDecisionTree (W, GT);
```

Pseudocode for training phase is shown in Algorithm 1. The input is the training dataset D_{tr}. It contains the attribute values and the class labels of each data point. A vector of CUSUM type process scores w_p corresponding to point P is calculated at line 5 using all the points in D where D contains all the data points starting from the last change point to the current point P as discussed above. $IsChangeOfClass\,(P)$ function checks if the class label of point P is different from the class label of the previous point. In other words, this function returns true if the current data point is a change point, otherwise, it returns false. The returned value is stored in gt_p (line 6). If gt_p is true then current data point P is a change point and D is re-initialized containing only point P (i.e., a new D starting from P) in line 8. On the other hand, if $false$ is assigned to gt_p, it means that point P is not a change point in the training data. In this way, values of w_p and gt_p are stored in W and GT respectively (lines 9 and 10). Finally, a decision tree model is trained using W and GT (line 11).

Algorithm 2: Algorithm for Testing

input : Decision Tree Model M and set of newly arrived points D_{new}
output: label of newly arrived points

```
1  begin
2      PrevLabel ← −1;
3      D ← ∅;
4      for each point P ∈ D_new do
5          D ← D ∪ P;
6          W_p ← CalculateW (D);
7          IsChange ← ApplyModel (M, W_p);
8          if IsChange then
9              Label_p ← −1 ∗ PrevLabel;
10         else
11             Label_p ← PrevLabel;
12         PrevLabel ← Label_p;
13         D ← ResetD (LastTrueChangePoint);
14         if IsUpdateNeeded then
15             S ← Sample(HistoricalData);
16             R ← MostRecentpoints);
17             M ← UpdateModelbyAlgo.1 (R + S);
```

Pseudocode for testing phase is shown in Algorithm 2. We represent the testing data class labels by $+1$ and -1. In data streams, data points continuously enter into the system. Let D_{new} in line 4 be the set of newly arrived data points which need classification. For each point P in D_{new}, score W_P is computed in line 6. While calculating these scores, all the points from the last true change point to the current point are taken into account (see D in line 5). we use $ResetD$ in line 13 to make sure that D starts from the true change point. W_P is then fed into the decision tree model M to detect if P is a change point (line 7). If it is a change point, P is assigned a label by toggling the label of the previous data point (line 9). On the other hand, if it is not a change point, then P is assigned the same label as the previous data point (line 11). Please note that we focus on classifying data streams containing instances from only two classes in this paper. Classifying instances of a data stream containing more than two classes is beyond the scope of this paper. We intend to investigate it in future work.

IV. Experimental Results

We have evaluated performance of our approach SC^2D on a number of datasets which are shown in Table I. Name of the datasets are System calls, Synthetic, KDD and PAMAP. In the $Systemcalls$ dataset, each instance is a snapshot of system calls collected from a running executable during its first two minutes of execution. Each instance is a vector of continuous real values representing the frequency of system calls during execution. There are a total of 284 possible system calls. Each instance is also labeled as either benign (negative) or malicious (positive). Synthetic dataset is generated using $RandomRBFGenerator$ of MOA [7]. KDD dataset is formed by considering the normal instances as negative and other instances as positive from KDD Cup 99 [4] dataset. The last dataset PAMAP is formed by considering cycling as negative and lying as positive instances from Physical Activity Monitoring dataset (PAMAP) [23] dataset.

TABLE I. Sizes of Datasets

Dataset	Training			Testing		
	#Negative	#Positive	#Change Points	#Negative	#Positive	#Change Points
System calls	64,185	64,185	2567	32,092	32,092	1283
Synthetic	23,324	12,016	706	9,900	5,100	300
KDD	8,356	7,092	309	4,190	3,557	155
PAMAP	8,184	8,178	327	4,216	4,212	169

We have compared performance of our approach (SC^2D) with number of supervised and unsupervised approaches. To compare with supervised approaches, we have used MOA [7] implementation of Hoeffding Adaptive Tree (HAT-ADWIN), Active Classifier, OzaBag, Weighted Majority Algorithms. On the other hand, we have also compared performance of our approach with two unsupervised approaches, i.e., CP-SUM and CP-MAX. We have used two different variations of CP-SUM and CP-MAX approaches. One of the variations is change detection with resetting, where W is calculated by considering the points from the last **true** change point. These variations are named as (CP-SUM$_{wr}$ and CP-MAX$_{wr}$). The other variation is change detection without resetting, where W is calculated by considering the points from the last **detected** change point. These variations are named as (CP-SUM$_{wor}$ and CP-MAX$_{wor}$). Change detection without resetting variations may include cumulative error because the classification is done by toggling whenever a change point is detected. If there is any delay in detecting current change point, this may contribute in delayed detection of subsequent change points as well. For example in Figure 3, assume that P_4 was not detected as a change point. All the points from P_4 to P_7 will be misclassified because we did not toggle the class label in P_4. Moreover, missing a true change point may result in missing subsequent change points because the examined data points may now include multiple distributions.

Both of CP-SUM and CP-MAX approaches need a threshold to detect change points. Threshold for CP-MAX approach h_{max} is calculated by taking $-\log(\alpha)$ where α is the probability of false alarm before $W_n = 0$. On the other hand, threshold for CP-SUM approach h_{sum} is calculated by taking $-\log(\alpha) * d$ where d is the number of dimensions. In our experiments, we have used $\alpha = 0.05$ to set the thresholds of the variations of CP-SUM and CP-MAX approaches.

For performance evaluation, we have used True Positive Ratio (TPR), False Positive Ratio (FPR), False Negative Ratio (FNR), and True Negative Ratio (TNR). To evaluate the overall performance of the methods we have used total accuracy and F_β-measure (equation (12)) with $\beta = 2$ as in equation (13). In network security domain, the ability to classify malicious data is more important than the ability to classify benign data. Because the misclassified malicious data may have harmful results. Using F_2 allows us to put more emphasis on the successful and failure attempts to protect the host from the malicious data because it weighs both true positive (TP) and false negative (FN) higher than false positive (FP).

$$F_\beta = \frac{(1+\beta^2) * TP}{(1+\beta^2) * TP + \beta^2 * FN + FP} \tag{12}$$

$$F_2 = \frac{5 * TP}{5 * TP + 4 * FN + FP} \tag{13}$$

Overall Performance: Performance of all approaches in terms of accuracy is shown in table II. It shows that, our proposed approach SC^2D outperforms all the other approaches by a large margin in case of $System\ calls$ dataset. SC^2D also shows better performance than other approaches on PAMAP dataset, where it classifies all the instances correctly. On KDD and Synthetic datasets also, SC^2D shows superior performance than the unsupervised approaches, i.e., CP-SUM, CP-MAX and competitive performance comparing with the supervised approaches, i.e., HAT-ADWIN, Active Classifier, OzaBag, Weighted Majority Algorithms. Finally, we present the average accuracy of these approaches on all the datasets. SC^2D outperforms all the other approaches by a large margin in terms of average accuracy.

Recall that, in this paper we focus on a specific scenario where instances from each of the classes come in a sequence, i.e., sequence of instances from a class followed by sequence of instances from another class. $System\ calls$ dataset has instances from two classes, i.e., benign and malicious. Instances from each of the classes come in a sequence. Original Physical Activity Monitoring dataset (PAMAP) dataset also contains sequence of instances from different classes since each of the physical activity, i.e., class label in this case is performed for some time. So, $System\ calls$ and PAMAP datasets comply with the specific scenario we are considering in this paper. This is the reason why our proposed approach SC^2D shows very good performance on $System\ calls$ and PAMAP datasets. Moreover, in PAMAP dataset each class has a unique distribution which differentiates each class from other classes. Recall that change point detection is used to find concept drift/concept evolution immediately over evolving stream in our approach. As a result, the number of discovered true positives has increased without adding false positives in case of SC^2D. On the other hand, in original KDD Cup 99 [4] and Synthetic datasets do not comply with the above scenario. In these datasets, instances from different classes do not come in sequence, still SC^2D shows competitive performance comparing with the other supervised approaches.

V. Related Works

In this paper, we have proposed an approach which exploits change point detection technique along with data mining

TABLE II. EXPERIMENTAL RESULTS

Data set	Method	Accuracy	F_2	TPR	FPR	FNR	TNR
System calls	SC^2D	**0.97**	**0.97**	**0.98**	0.04	**0.02**	0.95
	CP-SUM$_{wr}$	0.66	0.66	0.68	0.37	0.32	0.63
	CP-MAX$_{wr}$	0.55	0.54	0.55	0.456	0.443	0.543
	CP-SUM$_{wor}$	0.49	0.38	0.49	0.50	0.50	0.49
	CP-MAX$_{wor}$	0.50	0.39	0.49	0.49	0.50	0.50
	Act-Class	0.74	0.60	0.57	0.10	0.42	0.89
	HAT-ADWIN	0.76	0.68	0.66	0.15	0.33	0.84
	Ozabag	0.66	0.36	0.31	0.03	0.68	0.96
	W-Majorit	0.55	0.03	0.02	0.00	0.97	0.99
Synthetic	SC^2D	0.97	0.97	0.98	0.05	0.01	0.95
	CP-SUM$_{wr}$	0.59	0.54	0.619	0.448	0.38	0.55
	CP-MAX$_{wr}$	0.55	0.49	0.569	0.479	0.43	0.52
	CP-SUM$_{wor}$	0.51	0.46	0.51	0.49	0.49	0.51
	CP-MAX$_{wor}$	0.50	0.45	0.51	0.51	0.49	0.49
	Act-Class	0.98	0.97	0.96	0	0.03	1
	HAT-ADWIN	0.99	0.99	0.99	0	0.00	1
	Ozabag	0.99	0.99	0.99	0	0.00	1
	W-Majorit	0.99	0.99	0.99	0	0.00	1
KDD	SC^2D	0.94	0.94	0.94	0.059	0.058	0.94
	CP-SUM$_{wr}$	0.61	0.603	0.602	0.392	0.398	0.608
	CP-MAX$_{wr}$	0.54	0.55	0.544	0.457	0.456	0.543
	CP-SUM$_{wor}$	0.50	0.50	0.49	0.49	0.50	0.50
	CP-MAX$_{wor}$	0.50	0.50	0.50	0.49	0.49	0.50
	Act-Class	0.98	0.98	0.99	0.01	0.00	0.98
	HAT-ADWIN	0.99	0.99	0.99	0.00	0.00	0.99
	Ozabag	0.99	0.99	0.99	0.00	0.00	0.99
	W-Majorit	0.99	0.99	0.99	0.00	0.00	0.99
PAMAP	SC^2D	**1.0**	**1.0**	**1.0**	**0**	**0**	**1.0**
	CP-SUM$_{wr}$	0.59	0.569	0.56	0.386	0.437	0.614
	CP-MAX$_{wr}$	0.54	0.535	0.534	0.447	0.466	0.553
	CP-SUM$_{wor}$	0.46	0.47	0.47	0.56	0.53	0.44
	CP-MAX$_{wor}$	0.50	0.49	0.49	0.49	0.50	0.50
	Act-Class	0.99	0.98	0.98	0.00	0.01	0.99
	HAT-ADWIN	0.99	0.99	0.99	0	0.00	1
	Ozabag	0.99	0.99	0.99	0.00	0.00	0.99
	W-Majorit	0.99	0.99	0.99	0.00	0.00	0.99
Average	SC^2D	**0.97**	**0.97**	**0.97**	0.03	**0.02**	0.96
	CP-SUM$_{wr}$	0.61	0.59	0.61	0.39	0.38	0.60
	CP-MAX$_{wr}$	0.54	0.52	0.54	0.45	0.44	0.53
	CP-SUM$_{wor}$	0.49	0.45	0.49	0.51	0.50	0.48
	CP-MAX$_{wor}$	0.50	0.45	0.49	0.49	0.49	0.49
	Act-Class	0.92	0.88	0.87	0.03	0.12	0.96
	HAT-ADWIN	0.93	0.91	0.91	0.03	0.08	0.96
	Ozabag	0.91	0.83	0.82	0.00	0.17	0.99
	W-Majorit	0.88	0.75	0.75	0.00	0.24	0.99

techniques to classify data instances from data streams. In this section, first we intend to look into some related works on change point detection techniques. We will also look into some related data stream classification techniques.

There are many change point detection (CPD) algorithms available in the literature. However, most of the work is based upon certain assumptions. For example, CPD method proposed by Hinkley [17] assumes that distribution families before and after the change point belong to known parametric families, e.g., normal, binomial. On the other hand, CPD algorithm proposed by Bhattacharyya et. al. [5] assumes that the variables after the change are stochastically larger than those before. In other words, distribution families before and after the change point differ only in their levels, e.g., mean or median. M. Baron proposes an online non-parametric change point detection approach in [2] which is free from above assumptions. This approach uses histogram density estimation procedures to estimate distribution families before and after the change. These assumptions are then used in a CUSUM-type process to estimate the change point with a stopping time.

These above mentioned approaches work on uni-dimensional data only. However, most of the real life datasets are multi-dimensional. In the literature, several change detection methods exist to deal with multidimensional data also. Song et. al. [24] propose a log-likelihood change detection method, called density test, which makes use of EM algorithm and kernel density estimator to infer the distribution. However, it is efficient for low dimensional data. Kuncheva et. al. [20] apply principle component analysis (PCA) to multidimensional data and keep the components with the smaller variance. Then the approach uses semi-parametric log-likelihood detector (SPLL) [19] to detect the distribution change. However, SPLL loses theoretical precision in order to achieve computational simplicity as the assumptions are rarely met. This makes it difficult to set up a threshold.

Extracting useful information efficiently from data stream has become a research focus since data streams are becoming more and more common in today's connected digital world. Masud et. al. propose an efficient stream mining approach DXMiner [21], where the authors proposed techniques to find novel classes over stream. It uses hyper-spheres to capture the decision boundary for classes as the stream is processed. They divide the stream into equal sized chunks (fixed length chunk). In other words, they have not considered dynamic length chunk size to address concept drift issues. SluiceBox [22] is a method for data stream mining which builds a hierarchy of ensemble classifiers. They divide stream data into equal chunk size (fixed one) and do classification.

Bifet et. al. propose MOA: Massive Online Analysis [7], which is a framework for stream classification and clustering. It implements a wide range of classification and clustering methods for data streams. Bifet et. al. [8] present a framework to implement multiple change point detection methods using MOA. ADWIN [6] is a change detection method that uses only one parameter (confidence bound). (HAT-ADWIN) is an adaptively learning method that uses ADWIN to detect changes and update the model. SC^2P differs from ADWIN in the following ways: First, SC^2P exploits entirely nonparametric change point detection; ADWIN uses CUSUM and EWMA as non-parametric methods for detecting specific changes in the location parameter - drifts. However, the drift detection method uses Binomial distribution so it is parametric. Second, on one hand, SC^2P does not use threshold and strive to detect change point immediately or with minimum delay. On the other hand, ADWIN may encounter delay to reach the threshold or confidence. Experimental results also show that SC^2P outperforms HAT-ADWIN.

Our approach presented in this paper uses Change Point Detection (CPD) to classify instances in data streams. Our approach incorporates both statistical CPD algorithm and classical data mining methods. It does not need to divide the stream into equal chunks to address infinite length which may lead to poor performance. The proposed approach detects change point on multi-dimensional data without reducing dimensionality or using subspace. Moreover, as our approach uses decision tree algorithm, each dimension contributes to the final label prediction according to its information gain. Experimental results show that this approach can efficiently be used to detect security threats, e.g., classify malicious or spam instances.

VI. Conclusion

In this paper, we have presented an approach to classify data points in data streams having two classes. To do this, we have proposed an efficient method to detect change point in a multi-dimensional data set without using any pre defined threshold value. Moreover, our approach does not have the criteria of dividing the infinite length data stream into fixed size chunks. It uses the idea of change point detection in the data stream by sequential partitioning that determines an appropriate length for a chunk on the fly. Experimental result shows that the proposed approach is an efficient framework to classify data points of data stream containing two classes, e.g., data stream containing data points having classes benign/malicious. In the future, we intend to continue our work to classify data points and separate novel classes from concept drift in data stream having multi classes by exploiting ensemble based techniques.

Acknowledgments

This Research is based upon work supported by the National Science Foundation under Grants DMS-1322353, DUE-1129435, and CNS-1229652.

References

[1] C. C. Aggarwal and P. S. Yu. On classification of high-cardinality data streams. In *SDM*, pages 802–813. SIAM, 2010.

[2] M. I. Baron. Nonparametric adaptive change point estimation and on line detection. *Sequential Analysis*, 19(1-2):1–23, 2000.

[3] M. Basseville and I. V. Nikiforov. *Detection of Abrupt Changes: Theory and Application*. PTR Prentice-Hall, Inc., Englewood Cliffs, NJ, 1993.

[4] S. D. Bay, D. F. Kibler, M. J. Pazzani, and P. Smyth. The uci kdd archive of large data sets for data mining research and experimentation. *SIGKDD Explorations*, 2:81, 2000.

[5] G. K. Bhattacharyya and R. A. Johnson. Nonparametric tests for shift at an unknown time point. *The Annals of Mathematical Statistics*, 39(5):1731–1743, 10 1968.

[6] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà. New ensemble methods for evolving data streams. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 139–148, New York, NY, USA, 2009. ACM.

[7] A. Bifet, G. Holmes, B. Pfahringer, P. Kranen, H. Kremer, T. Jansen, and T. Seidl. Moa: Massive online analysis, a framework for stream classification and clustering. In *Journal of Machine Learning Research (JMLR) Workshop and Conference Proceedings, Volume 11: Workshop on Applications of Pattern Analysis*, pages 44–50. Journal of Machine Learning Research, 2010.

[8] A. Bifet, J. Read, B. Pfahringer, G. Holmes, and I. Zliobaite. Cd-moa: Change detection framework for massive online analysis. In *IDA*, pages 92–103, 2013.

[9] B. E. Brodsky and B. S. Darkhovsky. *Nonparametric Methods in Change-Point Problems*. Kluwer Academic Publishers, The Netherlands, 1993.

[10] D. Brzezinski and J. Stefanowski. Combining block-based and online methods in learning ensembles from concept drifting data streams. *Information Sciences*, 265(0):50 – 67, 2014.

[11] E. Carlstein. Nonparametric estimation of a change-point. *Ann. Statist.*, 16:188–197, 1988.

[12] J. Chen and A. K. Gupta. *Parametric Statistical Change Point Analysis: With Applications to Genetics, Medicine, and Finance*. Birkhäuser, Boston, MA, 2012.

[13] L. Dumbgen. The asymptotic behavior of some nonparametric change-point estimators. *The Annals of Statistics*, 19(3):1471–1495, 09 1991.

[14] D. Ferger. *Nonparametric change-point detection based on U-statistics*. PhD thesis, University of Giessen, 1991.

[15] J. a. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4):44:1–44:37, Mar. 2014.

[16] K. W. Hamlen, V. Mohan, M. M. Masud, L. Khan, and B. Thuraisingham. Exploiting an antivirus interface. *Computer Standards & Interfaces*, 31(6):1182–1189, April 2009.

[17] D. V. Hinkley. Inference about the change-point in a sequence of random variables. *Biometrika*, 57(1):pp. 1–17, 1970.

[18] J. Z. Kolter and M. A. Maloof. Dynamic weighted majority: An ensemble method for drifting concepts. *J. Mach. Learn. Res.*, 8:2755–2790, Dec. 2007.

[19] L. I. Kuncheva. Change Detection in Streaming Multivariate Data Using Likelihood Detectors. *Knowledge and Data Engineering, IEEE Transactions on*, 25(5):1175–1180, May 2013.

[20] L. I. Kuncheva and W. J. Faithfull. PCA feature extraction for change detection in multidimensional unlabelled data. *IEEE Transactions on Neural Networks and Learning Systems*, 2013.

[21] M. M. Masud, J. Gao, L. Khan, J. Han, and B. M. Thuraisingham. Classification and novel class detection in concept-drifting data streams under time constraints. *IEEE Trans. Knowl. Data Eng.*, 23(6):859–874, 2011.

[22] B. Parker, A. M. Mustafa, and L. Khan. Novel class detection and feature via a tiered ensemble approach for stream mining. In *ICTAI*, pages 1171–1178, 2012.

[23] A. Reiss and D. Stricker. Introducing a new benchmarked dataset for activity monitoring. In *Wearable Computers (ISWC), 2012 16th International Symposium on*, pages 108–109, June 2012.

[24] X. Song, M. Wu, C. Jermaine, and S. Ranka. Statistical change detection for multi-dimensional data. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 667–676, New York, NY, USA, 2007. ACM.

[25] W. N. Street and Y. Kim. A streaming ensemble algorithm (sea) for large-scale classification. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 377–382, New York, NY, USA, 2001. ACM.

[26] H. Wang, W. Fan, P. S. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 226–235, New York, NY, USA, 2003. ACM.

[27] H. Wang, J. Yin, J. Pei, P. S. Yu, and J. X. Yu. Suppressing model overfitting in mining concept-drifting data streams. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 736–741, New York, NY, USA, 2006. ACM.

[28] P. Wang, H. Wang, X. Wu, W. W. 0009, and B. Shi. A low-granularity classifier for data streams with concept drifts and biased class distribution. *IEEE Trans. Knowl. Data Eng.*, 19(9):1202–1213, 2007.

[29] Y. Yang, X. Wu, and X. Zhu. Combining proactive and reactive predictions for data streams. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pages 710–715. ACM, 2005.

[30] P. Zhang, J. Li, P. Wang, B. J. Gao, X. Zhu, and L. Guo. Enabling fast prediction for ensemble models on data streams. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 177–185, New York, NY, USA, 2011. ACM.