Proactive Drift Detection: Predicting Concept Drifts in Data Streams using Probabilistic Networks

Kylie Chen
Department of Computer Science
The University of Auckland
Email: kche309@aucklanduni.ac.nz

Yun Sing Koh
Department of Computer Science
The University of Auckland
Email: ykoh@cs.auckland.ac.nz

Patricia Riddle
Department of Computer Science
The University of Auckland
Email: pat@cs.auckland.ac.nz

***Abstract*—The application of current drift detection methods to real data streams show trends in the rate of change found by the detectors. We observe that these patterns of change vary across different data streams, and we use the term stream volatility pattern to describe change rates with a distinct mean and variance. First, we propose a novel drift prediction algorithm to predict the location of future drift points based on historical drift trends which we model as transitions between stream volatility patterns. Our method uses a probabilistic network to learn drift trends and is independent of the drift detection technique. We demonstrate that our method is able to learn and predict drift trends in streams with reoccurring stream volatility patterns. This allows the anticipation of future changes which enables users and detection methods to be more proactive. Second, we apply our drift prediction algorithm by incorporating the drift estimates into a drift detector, ProSeed, to improve its performance by decreasing the false positive rate.**

I. Introduction

Much of scientific research involves the generation and testing of hypotheses that can facilitate the development of accurate models for a system. In machine learning the automated building of accurate models is desired. However traditional machine learning often assumes that the underlying models are static and unchanging over time. In reality there are many applications where the underlying model or system changes over time. This may be caused by changes in the conditions of the system, or a fundamental change in how the system behaves. For example the development of antibiotic resistance changes how a pathogen responds to medication. This creates a need for systems that are able to detect and adapt to changes in the underlying model.

Many drift detection methods have been developed for the detection of changes in the streaming environment [1] [2] [3] [4]. Detectors have been shown to have a high rate of true positives on synthetic data streams, but most methods do not consider other information such as historical drift trends that could allow the anticipation of future change points. Huang et al. introduced a measure called stream volatility that represents the rate of change of the stream [5]. In their paper, they propose a volatility detector to detect changes in stream volatility and show that there are stream volatility trends for real data streams [5]. Unfortunately they did not use the information beyond finding the volatility trend. By incorporating volatility into traditional drift detectors we can use the additional information to proactively determine future changes. This is revolutionary as we are no longer monitoring historical event but anticipating future changes in the data stream. If we can map the frequency or interval of changes that occur to a particular pattern we are able to use this information to predict a future time range of having high probability of change occurrence.

We propose a drift predictor for learning stream volatility trends and a drift detector that can uses drift predictions to proactively search for drifts. Our drift predictor uses the output of a drift detector and volatility detector [5] to learn a probabilistic network. We assume these inputs to be relatively accurate as shown in Huang et al. [5]. Our drift detector, ProSeed, then adapts the compression of its data based on the estimated drift points from the drift predictor.

The main contributions of our work are: (1) a drift prediction algorithm that can accurately learn drift trends of a stream and (2) a drift detector which incorporates historical drift rate information that is accurate for streams with reoccurring volatility trends. We analyze our drift prediction technique by comparing it to ground truth in synthetic data streams and show that it can accurately capture trends for streams with reoccurring volatility patterns. We evaluated the performance of our drift detector by comparing it against detectors ADWIN2 [3], Seed [5] and DDM [2] on synthetic and real data streams and show that our technique is able to lower the rate of false positives for streams with these trends.

Relation to other research: Our research differs from research in drift detection with reoccurring patterns [6] as their methods are aimed at detecting models that reoccur whereas our method aims to learn the characteristics of drift rate trends. For example, suppose we are trying to learn the concept of seasons, research in reoccurring patterns focuses on the order that concepts reoccur such as: spring, summer, autumn, winter, spring. Our research aims to look at the rate of concept change, that is the time period between season changes. Unlike research in temporal forecasting for seasonal patterns, we do not assume there is any seasonal effect to the changes, and the trends do not necessarily occur periodically.

The following section describe the background and related work in the area. Section III discuss the key terminology used in our research. Section IV discuss the overview in the area. Sections V and VI discuss the inner workings of our technique. Section VII shows our experiments. Finally we conclude the paper and discuss future work in Section VIII.

II. Background and Related Work

Concept drift is the phenomena where there is a change in the underlying distribution of the data. In the context of classification, this can be formalized as a change in the posterior

$$p(y|X) = \frac{p(y)p(X|y)}{p(X)},$$

where X are the input features in the feature space R^P and y is the class label. Here $p(y)$ represents the prior probabilities of the classes, $p(X|y)$ represents the class conditional probabilities for all classes $y = 1, 2, \dots, c$ and $p(X)$ is the distribution of the input features [7]. Drift detectors often use numerical measures such as prequential statistics to detect changes in concepts [7]. The prequential is a binary stream of classification errors of a classifier, where a zero represents a correct classification and a one represents an error.

Gama et al.'s survey [7] groups drift detection methods into four categories: (1) sequential analysis [1], (2) statistical process control [2], (3) monitoring two distributions [3] [5], and (4) contextual approaches [8]. Most methods are reactive in nature and are unable to protectively predict the location of the next drift point. Huang et al. are the first to use historical drift points for improving drift detection by introducing a predictive approach using the mean stream volatility to estimate the location of the next drift [4]. They use a relative measure called stream volatility which represents the rate at which drifts occur [5]. High volatility indicates that concepts in a stream change frequently, whereas low volatility indicates that changes are occurring less frequently. Although their method is predictive, it does not utilize information such as trends of temporal changes in stream volatility which may provide more accurate predictions for streams that show reoccurring patterns of volatility change. Other proactive drift detectors such as [6] aim to predict concept models for future drifts using historical trends of concept change. The authors use a Markov Chain to model the concept history and have shown that this allows the learner to adjust more quickly to concept change in streams with reoccurring concepts [6].

Work related to the characterization of changes in data streams include research on the magnitude of change [9], research on the rate of change [5] and Webb et al.'s review which formalizes different aspects of change [10]. A closely related area is temporal time series forecasting where the goal is to make predictions on numerical data based on historical trends and seasonality effects.

III. Definitions

In this section we define the key terms that are relevant to the scope of our research. We follow the stream volatility definition in the previous section and the volatility shift definition from [5]. We generalize streams with volatility changes into two categories - streams with abrupt volatility change, and streams with gradual volatility change, and define two terms - volatility pattern, and pattern transition to model these changes.

Volatility shift: A volatility shift is a change in the rate that drifts occur [5]. Let $c_1, c_2, c_3, ..., c_t$ represent the drift points of a stream, and $i_1, i_2, ...i_{t-1}$ represent the corresponding drift intervals between consecutive drifts. A volatility shift occurs when we can find two volatility windows $V_1 = (v_1, v_2, ..., v_k)$ and $V_2 = (v_k + 1, ..., v_{t-1})$ with sample variances σ_1, σ_2 such that $\frac{\sigma_1}{\sigma_2} \lesseqqgtr 1.0 + \beta$ where β is a user defined threshold.

Abrupt volatility change: Streams with abrupt volatility change represent streams where there are stable periods followed by sudden changes in stream volatility, for example a stream with the drift intervals 100, 100, 100, 300, 300, 300 represents a stream with abrupt volatility change.

Gradual volatility change: Streams with gradual volatility change are streams with incremental changes in stream volatility. For example a stream with drift intervals 100, 110, 120, 130, 140, 150 would be classified as a stream with gradual volatility change.

Volatility pattern: Following the drift point and drift interval definitions above. We define a volatility pattern as $p = \{v_m, v_{m+1}, ..., v_n\}$ with a mean μ_p and length l_p. The set of drift intervals $\{v_m, v_{m+1}, ..., v_n\}$ is a subset of the intervals in the volatility windows $V_1 \cup V_2$ and represents a snapshot of the stream volatility at time t that has a distribution D with a mean of μ and variance of σ^2. The pattern length l_p is the average number of time steps spent in pattern p prior to a volatility shift at t where t is an approximate location of the true volatility change point.

For example, given we observe the volatility windows $V_1 = (100, 150, 100)$ and $V_2 = (100, 150, 500)$. The pattern can be estimated to be $p_1 = \{100, 150, 100, 100, 150\}$ and has a distribution with mean $\mu_{\{100,150,100,100,150\}} = 120$ and variance $\sigma^2_{\{100,150,100,100,150\}} = 750$.

Pattern transition: A transition $p_1 \rightarrow p_2$ at time t indicates that a change in volatility has occurred at time t, where there is a change in the distribution generating the drift intervals from distribution D_1 corresponding to the volatility pattern p_1 to a distribution D_2 which corresponds to the volatility pattern p_2 such that the distributions are not equivalent $D_1 \neq D_2$.

Following from the previous example, suppose we have two patterns $p_1 = \{100, 150, 100, 100, 150\}$ and $p_2 = \{1000, 1500, 1000, 1000, 1500\}$. A transition $p_1 \rightarrow p_2$ at time step t means that there is a change in volatility at t from observing a change every 100 time steps to observing a change every 1000 time steps.

IV. Overview

In this section we provide a general overview of our drift predictor that estimates the next drift point based on historical drift trends and our proposed algorithm, ProSeed , a proactive drift detector that uses the estimates from the drift predictor to guide the search for changes.

A. Drift prediction using a probabilistic network

Our drift prediction method has three layers - a drift detector that provides the drift points and drift intervals, a volatility detector that is able to locate local volatility change points using the drift intervals, and a drift prediction algorithm that uses the location of volatility shifts to estimate the next drift. We use a pattern reservoir and probabilistic network to learn the volatility trends of a stream.

A pattern reservoir is a pool of size P that stores volatility patterns from the stream. Volatility patterns capture a snapshot of a period where the volatility of the stream is stable. We use these patterns to build an overall picture of stream volatility and assume that each pattern has some underlying distribution. Each pattern has a sample of drift intervals which we use to approximate the underlying distribution that generates the intervals, and a pattern length that denotes the number of time steps the pattern persists before transitioning to another pattern.

We use a probabilistic network to learn the trends of pattern changes by using a transition matrix data structure to store the probability of transitioning to the next pattern given the current pattern. For example, suppose the error rate of the stream can

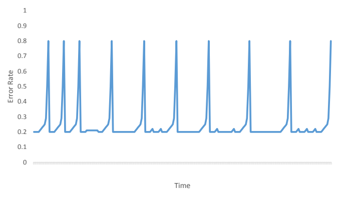

Fig. 1. Error rate of the stream

be represented by the graph above. Let the spikes represent increases in error rate caused by changes in concepts that are identified as cut points followed by an adaptation period where the classifier relearns. Let the time steps 100, 200, 300, 600, 900, 1200, 1600, 2000, 2400, 3000 represent the change points detected by the drift detector. The corresponding drift intervals are 100, 100, 100, 300, 300, 300, 300, 400, 400, 400. The volatility trend representing the rate of change over time is shown below. Now lets suppose we are able to identify

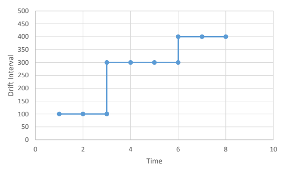

Fig. 2. Drift intervals of the stream

the first volatility shift using the volatility detector at time step 600 after observing the drift interval of 300. When there is a volatility shift, a pattern will be stored into our pattern reservoir. This pattern stores a sample of recent data close to the volatility change which represents a snapshot of the stable drift intervals preceding the volatility change.

Let the volatility windows be $V_1 = (100, 100)$ and $V2 = (100, 300)$. We construct a pattern p by removing outliers from $V_1 \cup V_2$ to get an approximation of the intervals in the stable period prior to the change. The first pattern $p_1 = \{100, 100, 100\}$ is added to the empty pattern reservoir. As this is the first volatility change we observed, the pattern length of p_1 is set to $l_{p_1} = 600$ which is the number of time steps between the volatility change point and the start of the stream.

When the next volatility shift is detected at time step 1600 following the first 400 interval. Assuming the volatility windows are now $V_1 = (300, 300)$ and $V_2 = (300, 400)$, removing the outliers the pattern becomes $p = \{300, 300, 300\}$. Since the time between consecutive volatility shifts is $1600 - 600 = 1000$, the pattern length for p is 1000. Then we attempt to add a new pattern $p = \{300, 300, 300\}$ into the pattern reservoir. First we perform pattern matching between the potential pattern p and the pattern reservoir by testing for equivalence using the KS test. If an equivalent pattern is found, we update the data samples in the pattern using the samples in p. Otherwise we add p as a new pattern p_2, and update the network by adding the transition $p_1 \to p_2$ to the probabilistic transition matrix. This transition can be illustrated by the network below.

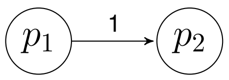

Fig. 3. Network of pattern transitions

Once we have learned a network, we make predictions by doing a linear projection of the possible pattern means from the mean of the latest pattern that we have detected. The details of this process is given in Section V .

To control the size of our probabilistic network, we merge similar transitions based on the similarity of the slopes between consecutive pattern means. For example, assume we now have three patterns and our network has the transitions $p_1 \to p_2 \to p_3$. Let the vertical axes represent the pattern's mean, and the horizontal axes represent time. Let the slope of $\mu_{p_1} \to \mu_{p_2}$ be 0.4, and the slope of $\mu_{p_2} \to \mu_{p_3}$ be 0.45. If the difference between the slopes are less than a user-defined slope parameter r, then the transition will be compressed. Suppose we use $r = 0.1$, since $|s_1 - s_2| = 0.05 < r$, we compress the transition $p_1 \to p_2 \to p_3$ into $p_1 \to p_3$.

B. ProSeed: Proactive drift detection

Our drift detection method uses a two phase approach: In the first phase we train our prediction algorithm. In the second phase we use the predictions to adjust the behaviour of our drift detector, ProSeed. First we use the drift intervals from the Seed detector [5] to train our drift prediction algorithm and learn the network. We use the Seed detector in the first phase to allow the exploration of the drift trends without any prior information. Then we use the network generated from our prediction algorithm to guide ProSeed by controlling the compression of the data.

ProSeed groups the error rate data into blocks of size b. For example, suppose the error stream is:

0001100010011011 0111

Using blocks of size four $b = 4$, the data becomes:

$$\underset{}{0001} \mid_{c_1} 1000 \mid_{c_2} 1001 \mid_{c_3} 1011 \mid_{c_4} 0111$$

Each vertical bar represents a block boundary and is a potential drift point c_1, c_2, c_3, c_4. We use the drift estimates from the drift predictor to find time steps where the next drift is likely to occur. For example, let $t_1, t_2, ..., t_{20}$ represent the time stamps associated with each binary input. If the drift predictor estimates the next likely drift to be at t_6 and t_{18}, the second and fifth blocks will not be compressed as they contain the estimated time steps. Blocks between the estimates will be compressed as drifts are less likely to occur during those periods. All blocks following the highest estimated drift time t_{18} will not be compressed. This would compress the third and fourth blocks, removing the second potential drift point from being checked.

$$0001 \mid_{c_1} 10001001 \mid_{c_2} 1011 \mid_{c_3} 0111$$

Then the drift detector will search for a drift by testing for a difference in means at the block boundaries using the Hoeffding bound detailed in Section VI . The novelty of our method is that it uses historical drift trends to adjust the compression of blocks in contrast to Seed which compresses based on a linear block similarity measure. This could be advantageous for data streams with reoccurring drift trends.

V. Drift prediction using a probabilistic network

In this section we provide the details of our drift prediction algorithm. We use a drift detector to find points of change and Huang et al.'s volatility detector [5] to locate the local peaks and troughs of volatility change. The pseudocode for our method is shown in Algorithm 1. We will discuss each of the different components to our technique separately.

A. Pattern construction

When we construct a pattern we store a pool of pattern data with mean μ_p to represent the distribution of the drift intervals, and the average pattern length l_p to represent the average time spent in the pattern before a volatility change was detected. We use the mean of the pattern μ_p to decide what the next drift interval may be, and l_p to determine when the change will occur. The average pattern length l_p is updated using the time between consecutive volatility changes each time we detect the pattern p.

We populate the pool of a pattern by sampling from a recent volatility window after a volatility change was detected. Given two volatility windows $V_1 = (v_1, v_2, ..., v_k)$ and $V_2 = (v_{k+1}, ..., v_{2k})$ of size k. Here V_2 represents the buffer that triggered the volatility change. First we compute the interquartile ranges of $V_1 \cup V_2$ and V_2, and choose the set of intervals that has the smaller interquartile range as the pool to sample our pattern data from. We assume that there are trends to the location of the true drifts, and false alarms are deviations from these trends. To filter out the possible false alarms and account for the period of volatility change we remove the outliers from the set of drift intervals. We assume the outliers contribute to the volatility change, and follow Tukey's definition for outliers

$$Y < (Q_1 - 1.5 \cdot IQR) \text{ or } Y > (Q_3 + 1.5 \cdot IQR), \qquad (1)$$

where Y is an outlier, Q_1 is the first quartile, Q_3 is the third quartile and IQR is the interquartile range.

B. Pattern matching

When we add a potential pattern p' to the pattern reservoir of size P, we first perform pattern matching to ensure that the pattern is not already present in the pattern reservoir. This is done by testing for pattern equality using the two sample KS test. If a matching pattern is found, we update the samples stored by the pattern by randomly replacing old data points with the new data samples from p'. This allows patterns to evolve over time, and can reduce the number of redundant patterns in the reservoir.

C. Transition compression

We use a transition compression scheme to compress transitions that are similar. An example of this method is presented in Section IV-A. The aim is to achieve a more compact representation of the network and also to reduce the number of pattern transitions to itself which has a dampening effect on the true transitions in the network.

D. Predicting the next drift

Given that the latest pattern we have seen is pattern p_x we can predict the next patterns that are likely to occur $p_{y1}, p_{y2}, ..., p_{yk}$ using the network, where k is a parameter for the number of predictions to use. We recommend setting this as the size of the pattern reservoir $k = P$. This will give us information on what drift interval is likely to change to. The average length of each predicted pattern $l_{y1}, l_{y2}, ..., l_{yk}$ will allow us to determine how long we expect the predicted pattern to persist.

For streams with abrupt volatility change, that have stable periods punctuated by abrupt changes we use the pattern means $\mu_{y1}, \mu_{y2}, ..., \mu_{yk}$ as the predicted drifts. For streams with gradual volatility change, that have incremental changes we calculate the next drift for each predicted pattern $p_{y_1}, p_{y_2}, ..., p_{yk}$ by

$$\frac{\mu_{yi} - \mu_x}{l_{yi}} \cdot t, \qquad (2)$$

where μ_{yi} is the mean of pattern p_{yi}, μ_x is the mean of the latest pattern p_x, l_{yi} is the average length of pattern p_{yi} and t is the current time step.

Our current algorithm assumes the nature of the volatility changes of the stream are known in advance, however in reality the nature of the stream is often unknown. We outline a method to address this below.

E. Characterizing volatility change

In real streams the characteristics of the streams are often unknown. This can lead to significant challenges in modelling and predicting the behaviour of a stream as well as setting appropriate parameters. We present a method for addressing the first issue, that allows us to determine the nature of volatility changes for streams given only the drift output from the drift detector.

Given drifts are detected at time steps $c_1, c_2, c_3, ..., c_t$, and $i_1, i_2, ...i_{t-1}$ are the corresponding drift intervals between consecutive drifts. The drift intervals can be grouped into blocks $B_1, B_2, B_3...$ of size $b \geq 32$. We can compute the difference in means between consecutive blocks as $|\mu B_i - \mu B_{i+1}|$. The volatility of the stream can be monitored using a histogram that is updated incrementally. This gives us a view of the overall volatility of the stream that can allow us to determine the volatility nature by matching the characteristics of the histogram with the characteristics of streams with abrupt or gradual volatility.

Streams with abrupt volatility changes are defined as having stable periods with abrupt changes in volatility so the distribution of means between consecutive blocks should be more concentrated at lower values due to the periods of stability, with a right hand tail that represents the sudden changes. In contrast streams with gradual volatility changes should have a relatively even distribution. However the shape of this distribution also depends on the variance of the drift intervals.

VI. ProSeed: Proactive drift detection

In this section we provide the details of our drift detection method. ProSeed extends the Seed detector [5] by using drift trends to proactively search for drifts. Our method differs from Seed in that we adjust the compression of data stored by the detector based on drift predictions whereas Seed does not use historical drift data. The compression of the data affects where and how often the drift bound is checked.

We use the Hoeffding bound with Bonferroni correction as a drift bound to determine whether a drift has occurred. Given a window of data W, we can partition this window into two subwindows W_L and W_R. These subwindows are partitioned according to the block boundaries and represent potential drift points. The drift bound for our detector is

$$|\hat{\mu}_{W_L} - \hat{\mu}_{W_R}| > \epsilon, \tag{3}$$

$$\epsilon = \sqrt{\frac{2}{m} \cdot \sigma_W^2 \cdot ln\frac{2}{\delta'}} + \frac{2}{3m} ln\frac{2}{\delta'}, \quad \delta' = \frac{\delta_d}{n} \tag{4}$$

where $\hat{\mu}_{W_L}$ and $\hat{\mu}_{W_R}$ represent the mean error rate of data in W_L and W_R respectively, ϵ_d is the Hoeffding bound with Bonferroni correction using a confidence parameter $\delta_d \in (0, 1)$, m is the harmonic mean of the lengths of W_L and W_R, and n is the length of W where $W = W_L + W_R$. A drift is detected when the difference between the means surpasses the drift threshold ϵ, and data from the left subwindow W_L is dropped. We use this bound as it has been shown to be more sensitive to small changes and gives strict theoretical guarantees on the rate of false positives.

Algorithm 1 Drift Prediction Algorithm

```
Input: $X_t \in \{0, 1\}$ classification result at time $t$
Output: estimated location of the next concept drift

  Initialize driftDetector $D$
  Initialize volatilityDetector $V$, with buffer $B$ of size $b$
  Initialize $driftInterval \leftarrow 0$
  Initialize a pattern reservoir $P$, and network $N$ of size $n$
  Initialize a large buffer $L$ of size $2b$

  for $t > 0$ do
      $estimatedInterval \leftarrow PredictNextDrift()$
      pass $X_t$ to $D$
      if $D$ detects a drift then
          update $L$ with $driftInterval$
          pass $driftInterval$ to $V$
          if $V$ detects a volatility shift then
              $AddPattern(L, B$ from $V)$
          end if
          $driftInterval \leftarrow 0$
      else  $driftInterval \leftarrow driftInterval + 1$
      end if
  end for

  function AddPattern(Large Buffer $L$, Buffer $B$)
      $data \leftarrow B$
      $l_r \leftarrow$ interquartile range of $L$
      $b_r \leftarrow$ interquartile range of $B$
      if $l_r \leq b_r$ then $data \leftarrow L$
      end if
      remove outliers in $data$ by Tukey's method
      if $PatternFound(data)$ then
          update pattern in $P$
      else
          $AddToReservoir(data)$
      end if
      update network $N$
  end function

  function PatternFound(data d)
      $found \leftarrow false$
      for each element $e$ in $P$ do
          if $d$ equals $e$ by KS-test then $found \leftarrow true$
          end if
      end for
      return $found$
  end function

  function AddToReservoir(data d)
      if $P$ is not full then add $d$ to $P$
      else   replace rarest element in $P$ with $d$
      end if
  end function
```

```
  function PredictNextDrift
      $E \leftarrow$ list of estimates
      $F \leftarrow$ list of top $k$ transitions from current pattern $p_x$
      for pattern $f$ in $F$ do
          if $state$ of $stream$ is gradual then
              $estimate \leftarrow \frac{\mu_f - \mu_{px}}{l_f} \cdot t$
          else if $state$ of $stream$ is abrupt then
              $estimate \leftarrow \mu_f$
          end if
          $E \leftarrow E \cup estimate$
      end for
      return $E$
  end function
```

VII. Experiments

We divide our experiments into two phases. In the first phase we evaluate the accuracy of our drift prediction algorithm. In the second phase we evaluate our drift detector by comparing it with Seed [5], ADWIN [3] and DDM [2]. We will briefly discuss our synthetic data generation process below.

Synthetic data streams: We generate streams with two types of volatility changes - streams with abrupt volatility change, and streams with gradual volatility change as described in Section III. We generate data with volatility trends by defining volatility patterns and a transition network for these patterns. We generate patterns with distinct means $\mu_H = \{100, 200, 300, ...\}$ each with variance $\sigma_H^2 = 100$ representing streams with high volatility, and $\mu_L = \{1000, 2000, 3000, ...\}$, $\sigma_L^2 = 1000$ for streams with low volatility. Based on these patterns we generate networks with cyclic trends. Consider the simple example with three patterns a, b, c, and the cyclic network $a \to b \to c \to a$. Transition probabilities are labelled as p and q. Here $p \in \{0.9, 0.75, 0.5\}$ is a transition probability,

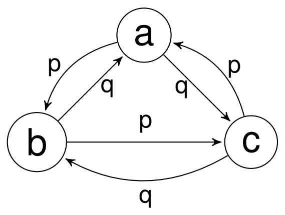

Fig. 4. Cyclic network with three patterns

and $q = 1 - p$.

First we generate drift intervals by using a cyclic network to determine the location of change points, then we simulating drifts at after each drift interval by changing the error rate or concept function for the stream based on the drift intervals from our network. We use two types of synthetic streams: (1) Bernoulli streams which simulates the error rate of a learner, and (2) SEA streams that has concepts which can be learned by a tree learner. We generate drifts in Bernoulli streams by alternating the mean error rate μ between values of 0.2 and 0.8 at each change point. We generate SEA streams by changing the threshold for the concept function, we choose the threshold values to be 7 and 9.5 to simulate a more subtle change.

Parameter selection: For the volatility detector we use $b = 32, c = 0.5$ for abrupt volatility streams, and $b = 32, c = 0.2$ for gradual volatility streams. We set this generously so that it can more accurately capture changes in the drifts. For ADWIN2, and Seed we use the recommended settings of $\delta = 0.05$ and Seed best from [5]. For ProSeed we use the same parameters as Seed, and $\alpha = 3, \beta = 2$ for DDM.

A. Drift Prediction

We evaluate our drift prediction algorithm by generating synthetic streams with cyclic volatility trends and compare the patterns and network produced by our predictor to the ground truth values for the streams.

Evaluation metrics: We use two metrics for measuring network similarity: The number of true transitions (TT) which shows how many transitions from the stream is captured by the drift predictor's network, and the number of additional transitions (AT) which measures how many additional transitions are present in the drift predictor's network that are not in the stream's network. For comparing different sized networks, we compress the larger network from the drift predictor so that it is the same size as the network from the stream. Network compression is performed by merging the most similar patterns according to the lowest D statistic value from the KS test until both networks have the same size. We present the average and standard deviation values over 100 runs.

In these experiments we present the results on streams with abrupt volatility change. For the experiments presented in Tables I, III, IV we generate synthetic streams with 500 transitions between patterns with high volatility μ_H and a volatility interval of 100. The volatility interval is the number of drift points between volatility changes, a higher value represents a more stable stream.

In Table I we show the effect of the drift predictor's pattern reservoir size on networks of various sizes. Here P is the pattern reservoir size, and N is the size of the synthetic network. As the number of true transitions is close to 500 for networks of size 3 and 5 this indicates that the predictor has been able to capture these transitions accurately. For networks of size 10, we observe a significant decrease in the number of true transitions when $P = 10$ because the total number of patterns detected is around 30, so setting the pattern reservoir size at 10 restricts the number of transitions the network can capture. A larger network can give a finer granularity of the trends whereas a smaller network may be better at capturing frequent patterns of the stream. We note that the high number of additional transitions is a result of false positives from the input of the volatility detector. This has a dampening effect on the probability of the transitions, however as we base the predictions on the top k transitions the effect is less prominent.

For the later experiments Tables II, III, IV we keep the stream's network size fixed at $N = 3$, and vary other parameters of the stream. In Table II we examine the effect of the stability of the drift rate on the accuracy of our computed network. Here V denotes the volatility interval which is the number of cut points between changes in the drift rate. The number of true transitions captured is relatively robust to the stability of the trends, but the number of additional transitions detected increases with respect to the size of the volatility interval.

In Table III we show the effect of adding Gaussian noise with a standard deviation of σ to the patterns. This increases the amount of overlap between between patterns. For example if 95% of pattern a falls within the bounds (80, 120) and 95% of pattern b falls between (180, 220). Adding noise to a and b increases the range of the patterns, so the patterns could become $a = (40, 180), b = (140, 260)$. The accuracy of the network degrades as the percentage of overlap between the patterns increases. Table IV compares the accuracy of the

TABLE I
NETWORK COMPARISON: ABRUPT BERNOULLI STREAMS WITH DIFFERENT NETWORK SIZES

Metric	N / P	10	30	100
	3	497.99 ± (0.90)	498.26 ± (0.66)	498.26 ± (0.66)
TT	5	496.94 ± (2.33)	498.29 ± (0.68)	498.29 ± (0.68)
	10	267.04 ± (61.59)	481.88 ± (20.83)	481.88 ± (20.83)
	3	603.59 ± (9.92)	603.89 ± (9.92)	603.89 ± (9.92)
AT	5	601.06 ± (10.37)	601.99 ± (10.32)	601.99 ± (10.32)
	10	482.50 ± (36.38)	624.43 ± (20.50)	624.43 ± (20.50)
	3	10498.80 ± (950.36)	17610.24 ± (1262.56)	92090.24 ± (1262.56)
Memory	5	10832.88 ± (630.05)	18493.68 ± (1420.28)	92973.68 ± (1420.28)
	10	11016.40 ± (78.48)	29319.20 ± (1777.46)	103799.20 ± (1777.46)

TABLE II
NETWORK COMPARISON: ABRUPT BERNOULLI STREAMS WITH DIFFERENT VOLATILITY LEVELS

V	TT (SD)	AT (SD)	Time (SD)
100	498.26 ± (0.66)	603.89 ± (9.92)	991.54 ± (50.64)
200	498.63 ± (0.49)	899.34 ± (20.13)	1345.64 ± (87.02)
500	498.91 ± (0.29)	1747.37 ± (44.31)	2368.44 ± (150.66)

TABLE III
NETWORK COMPARISON: ABRUPT BERNOULLI STREAMS WITH GAUSSIAN PATTERN NOISE

Pattern Noise (σ)	TT (SD)	AT (SD)
0	498.26 ± (0.66)	603.89 ± (9.92)
25	457.89 ± (51.06)	507.32 ± (52.77)
50	230.70 ± (57.88)	210.88 ± (58.71)

patterns detected. μ_p is the mean of the true pattern and σ_p^2 is the variance of the true pattern. In the best case, the computed means and variances are very accurate but the variance can be significantly higher in the worst case. This occurs when the drift prediction algorithm is unable to correctly separate patterns at a change point. The median value of the computed variances which we have not presented here is close to the theoretical values.

B. Drift detection

Tables V and VII show drift detectors evaluated on streams with abrupt volatility changes. Tables VI and VIII show results from evaluation on streams with gradual volatility change.

In the following experiments we use patterns with low volatility μ_L, and generate 10000 drift points in each stream.

TABLE IV
PATTERN COMPARISON: ABRUPT BERNOULLI STREAMS WITH PATTERN NOISE

Pattern Noise (σ)	μ_p	σ_p^2	μ_{Best}	σ_{Best}^2	μ_{Worst}	σ_{Worst}^2
0	100	100	99.84	105.40	120.96	3383.96
0	200	100	200.00	99.71	219.84	2448.81
0	300	100	299.20	108.71	263.36	5437.65
25	100	100	99.84	90.51	145.60	6699.54
25	200	100	200.00	464.52	254.72	5239.05
25	300	100	299.20	649.05	250.88	6915.21
50	100	100	100.16	530.96	176.64	8532.92
50	200	100	200.32	546.40	128.00	10382.64
50	300	100	299.84	309.79	217.6	8578.74

For the SEA streams, we first pass the data through a Hoeffding Tree to generate the error rate input to the detectors.

In Tables V, VI, VII and VIII, we observe that Seed and ADWIN are comparable in terms of the number of true drifts found. However ADWIN is more sensitive to false positives. This may be due to the way the data is stored which facilitates the detection of small changes. We show that ProSeed is able to accurately detect many true drifts, and is comparable to Seed and ADWIN in terms of true positives for the Bernoulli streams. Although ProSeed has a small reduction in true positive, it is able to effectively lowest the number of false positives detected. This gives promise to the use of drift trends for achieving a more accurate drift detector.

We also evaluated our drift detector on real data streams: Sensor stream, and Forest Covertype. As we do not know the location of true drifts for real streams we can not assess the true positive and false positive rates. We show the number of drifts detected by ProSeed is lower than the number of drifts detected by Seed. This suggests that it may have also reduced the number of false positives.

VIII. CONCLUSIONS

We proposed a drift prediction algorithm that can accurately learn drift trends of a stream using a probabilistic network. We then proposed a new drift detector which incorporates historical drift rate information that is accurate for streams with reoccurring volatility trends. The highlight of our new technique is that it allows us to proactively determine when we would see future drift point. This in itself changes the landscape of how drift detectors are currently developed and used. To show the accuracy and feasibility of our technique, we analyze our drift prediction technique by comparing it to ground truth in synthetic data streams and show that it can accurately capture trends for streams with reoccurring volatility patterns. In our experiments we compared the performance of our drift detector against other benchmark detectors such as ADWIN2 [3], Seed [5] and DDM [2] on synthetic and real data streams and show that our technique is able to lower the rate of false positives for streams with these trends.

One current limitation of our technique is that it is currently chose to predict the network patterns using the by matching the patterns using the information from the drift interval. In the future we can map both concept models and network patterns together. This would provide us with more knowledge about

TABLE V
DRIFT DETECTION ON ABRUPT BERNOULLI STREAMS

Detector	TP (SD)	FP (SD)	Delay (SD)	Memory (SD)	Time (SD)
ProSeed	9998.89 ± 0.31	33.10 ± 5.93	34.64 ± 10.09	683.84 ± 157.90	598.78 ± 14.57
Seed	9999.00 ± 0.00	213.34 ± 16.90	34.51 ± 10.15	2207.04 ± 974.11	1539.03 ± 41.15
ADWIN2	9999.00 ± 0.00	12689.68 ± 78.04	34.01 ± 10.14	1747.36 ± 137.13	5262.68 ± 108.51
DDM	5000.01 ± 0.22	97.41 ± 10.25	201.51 ± 52.57	128.00 ± 0.00	322.62 ± 15.52

TABLE VI
DRIFT DETECTION ON GRADUAL BERNOULLI STREAMS

Detector	TP (SD)	FP (SD)	Delay (SD)	Memory (SD)	Time (SD)
ProSeed	9998.92 ± 0.27	44.32 ± 6.43	34.53 ± 10.10	852.56 ± 349.62	558.01 ± 20.51
Seed	9998.99 ± 0.10	210.50 ± 15.21	34.46 ± 10.14	1968.48 ± 871.98	1479.92 ± 45.38
ADWIN2	9999.00 ± 0.00	12698.08 ± 68.53	33.98 ± 10.13	1740.64 ± 132.91	5254.83 ± 116.28
DDM	5000.04 ± 0.20	100.98 ± 10.03	203.29 ± 38.28	128.00 ± 0.00	299.54 ± 15.22

TABLE VII
DRIFT DETECTION ON ABRUPT SEA STREAMS

Detector	TP (SD)	FP (SD)	Delay (SD)	Memory (SD)	Time (SD)
ProSeed	9895.42 ± 47.34	37.02 ± 11.47	135.59 ± 92.23	641.60 ± 138.40	624.50 ± 16.47
Seed	9997.14 ± 1.83	165.69 ± 11.45	117.15 ± 65.97	1845.12 ± 818.37	1578.80 ± 38.42
ADWIN2	9997.00 ± 1.98	24920.11 ± 173.14	127.01 ± 74.95	1750.72 ± 139.07	8576.84 ± 171.26
DDM	2453.68 ± 2507.21	945.90 ± 966.72	436.25 ± 231.54	128.00 ± 0.00	357.25 ± 47.99

TABLE VIII
DRIFT DETECTION ON GRADUAL SEA STREAMS

Detector	TP (SD)	FP (SD)	Delay (SD)	Memory (SD)	Time (SD)
ProSeed	9820.32 ± 22.33	12.70 ± 3.63	217.24 ± 246.87	874.16 ± 315.96	599.08 ± 19.62
Seed	9997.18 ± 1.94	169.41 ± 12.37	116.85 ± 65.93	1876.32 ± 946.58	1536.42 ± 46.24
ADWIN2	9996.92 ± 2.04	25391.84 ± 183.48	126.12 ± 76.20	1740.64 ± 132.91	8547.74 ± 192.83
DDM	2301.24 ± 2501.38	906.03 ± 985.18	440.90 ± 221.87	128.00 ± 0.00	342.29 ± 42.40

TABLE IX
DRIFT DETECTION ON SENSOR STREAM

Detector	Drifts Detected	Memory	Time
ProSeed	952.00	440.00	313.73
Seed	1152.00	456.00	266.60
ADWIN2	3348.00	1240.00	1107.10
DDM	197.00	128.00	60.95

TABLE X
DRIFT DETECTION ON FOREST COVERTYPE STREAM

Detector	Drifts Detected	Memory	Time
ProSeed	2309.00	1520.00	45.02
Seed	2562.00	1512.00	43.66
ADWIN2	3218.00	1576.00	268.70
DDM	1374.00	128.00	22.81

predictions and enhance the drift detector capability of only detecting true changes and reduces the false positives further.

REFERENCES

[1] E. Page, "Continuous inspection schemes," *Biometrika*, pp. 100–115, 1954.

[2] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Advances in Artificial Intelligence - SBIA 2004*, ser. Lecture Notes in Computer Science, A. Bazzan and S. Labidi, Eds. Springer Berlin Heidelberg, 2004, vol. 3171, pp. 286–295. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-28645-5_29

[3] A. Bifet and R. Gavaldà, "Learning from time-changing data with adaptive windowing," in *Proceedings of the Seventh SIAM International Conference on Data Mining, April 26-28, 2007, Minneapolis, Minnesota, USA*, 2007, pp. 443–448. [Online]. Available: http://dx.doi.org/10.1137/1.9781611972771.42

[4] D. T. J. Huang, Y. S. Koh, G. Dobbie, and A. Bifet, "Drift detection using stream volatility," in *Machine Learning and Knowledge Discovery in Databases*, ser. Lecture Notes in Computer Science, A. Appice, P. P. Rodrigues, V. Santos Costa, C. Soares, J. Gama, and A. Jorge, Eds. Springer International Publishing, 2015, vol. 9284, pp. 417–432. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-23528-8_26

[5] D. T. J. Huang, Y. S. Koh, G. Dobbie, and R. Pears, "Detecting volatility shift in data streams," in *2014 IEEE International Conference on Data Mining, ICDM 2014, Shenzhen, China, December 14-17, 2014*, 2014, pp. 863–868. [Online]. Available: http://dx.doi.org/10.1109/ICDM.2014.50

[6] Y. Yang, X. Wu, and X. Zhu, "Mining in anticipation for concept change: Proactive-reactive prediction in data streams," *Data mining and knowledge discovery*, vol. 13, no. 3, pp. 261–289, 2006.

[7] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, pp. 44:1–44:37, Mar. 2014. [Online]. Available: http://doi.acm.org/10.1145/2523813

[8] R. Klinkenberg, "Learning drifting concepts: Example selection vs. example weighting," *Intell. Data Anal.*, vol. 8, no. 3, pp. 281–300, Aug. 2004. [Online]. Available: http://dl.acm.org/citation.cfm?id=1293831.1293836

[9] P. Kosina, J. Gama, and R. Sebastião, "Drift severity metric," in *ECAI 2010 - 19th European Conference on Artificial Intelligence, Lisbon, Portugal, August 16-20, 2010, Proceedings*, 2010, pp. 1119–1120. [Online]. Available: http://dx.doi.org/10.3233/978-1-60750-606-5-1119

[10] G. Webb, R. Hyde, H. Cao, H. Nguyen, and F. Petitjean, "Characterizing concept drift," *Data Mining and Knowledge Discovery*, 2016. [Online]. Available: http://arxiv.org/abs/1511.03816