Proceedings of Machine Learning Research 152:1–20, 2021 Conformal and Probabilistic Prediction and Applications

Using Inductive Conformal Martingales for addressing concept drift in data stream classification

Charalambos Eliades ST009072@STUD.FREDERICK.AC.CY
Computational Intelligence (COIN) Research Lab
Frederick University, Cyprus
Harris Papadopoulos H.PAPADOPOULOS@FREDERICK.AC.CY
Computational Intelligence (COIN) Research Lab
Frederick University, Cyprus

Editor: Lars Carlsson, Zhiyuan Luo, Giovanni Cherubin and Khuong An Nguyen

Abstract

In this paper, we investigate the use of Inductive Conformal Martingales (ICM) with the histogram betting function for detecting the occurrence of concept drift (CD) in data stream classification. A change in the data distribution will almost surely affect the performance of our classification model resulting in false predictions. Therefore, a reliable and fast detection of the point at which a CD occurs, allows effective retraining of the model to recover accuracy. Our approach is based on ICM with the histogram betting function, which is much more computationally efficient than alternative ICM approaches. To accelerate the process of detecting CD we also modify the ICM and examine different parameters of the histogram betting function. We evaluate the proposed approach on three benchmark datasets, namely STAGGER, SEA and ELEC, presenting different measures of its performance and comparing it with existing methods in the literature.

Keywords: Conformal, Martingales, Exchangeability,Drift

1. Introduction

In this study we deal with the problem of CD. When CD occurs while mining data streams the generating data distribution changes, this phenomenon leads to poor predictions and decision outcomes. A reliable tool that will raise an alarm as close as possible to the point at which the distribution changes will be beneficial. When a CD is detected the prediction algorithm will be adapted allowing as to recover accuracy.

Note that CD has also been defined by various authors as concept shift or dataset shift, in this study we use the term 'CD'(Lu et al., 2019).

Formally a CD at time stamp $t+1$ can be described as follows: given a set of examples we divide it into two subsets: $S_{0,t} = \{(x_0, y_0), (x_1, y_1), \dots, (x_t, y_t)\}$ $S_{t+1,\dots} = \{(x_{t+1}, y_{t+1}), \dots, \}$, where x_i is a feature vector and y_i a label. If the distribution of $S_{0,t}$ is different from the distribution of $S_{t+1,\dots}$, then we have a CD at time stamp $t+1$. For simplicity in this study we say that CD occurs at $t+1$ iff $P_t(X,Y) \neq P_{t+1}(X,Y)$ where (X,Y) is a pair of random feature vectors and labels respectively, with P_t and P_{t+1} the distribution of the consecutive instances contained in the sets $S_{0,t}$ and $S_{t+1,\dots}$.

CD can be produced from three sources. Recall that $P_t(X,Y) = P_t(X|Y) \cdot P_t(Y)$ and $P_{t+1}(X,Y) = P_{t+1}(X|Y) \cdot P_{t+1}(Y)$. To have a change in the joint distribution of (X,Y) one of the following might have happened: (a) $P_t(X|Y) = P_{t+1}(X|Y)$ and $P_t(Y) \neq P_{t+1}(Y)$ in this case we have a change in the labels distribution while the decision boundaries remain unchanged, this has also been considered as label shift (Vovk, 2020) and virtual drift (Bagui and Jin, 2020) (b) $P_t(X|Y) \neq P_{t+1}(X|Y)$ and $P_t(Y) = P_{t+1}(Y)$ here the decision boundaries change and lead to decrease in accuracy, it has also been considered as actual drift (Bagui and Jin, 2020) or concept shift (Vovk, 2020) and (c) $P_t(X|Y) \neq P_{t+1}(X|Y)$ and $P_t(Y) \neq P_{t+1}(Y)$; which is a combination of the two previously mentioned sources.

CD types can be classified in four categories: (a) sudden drift where the data generating mechanism changes instantly (b) gradual drift where the data distribution is replaced with a new one over time (i.e over time we experience fewer examples belonging to the initial distribution and more from the new distribution), (c) incremental where the a new data generating mechanism incrementally replaces the existing mechanism (i.e each example is generated by a mixture of distributions but over time the impact of the initial distribution disappears) d) reoccurring drift when a previously seen data distribution reappears (Lu et al., 2019),(Bagui and Jin, 2020).

The CD detection algorithms can be classified in three categories based on their statistics they apply (Lu et al., 2019). The first category is the 'Error rate-based algorithms', which they monitor increases or decreases of the online error rate, if these changes are statistically significant a drift alarm is raised. The second and biggest category is the 'Data Distribution-based', here the algorithms quantify the dissimilarity between the historical data and the new data. A distance function is used to measure the dissimilarity between the two distributions, a statistical hypothesis test with respect to a significance level will test if CD occurs. The last category 'Multiple Hypothesis Test', applies similar techniques with the ones mentioned above. Techniques of this category use multiple hypothesis tests to detect CD. They can be divided in two groups: parallel hypothesis tests and hierarchical multiple hypothesis tests for more information refer to (Lu et al., 2019).

In this paper our aim is to detect CD by testing the exchangeability assumption (EA), violation of this assumption is an indication that CD occurred. By the term exchangeability of the data we mean that the examples are invariant to any permutation. Many machine learning techniques are based on EA assumption, if this assumption is violated then predictions are likely to be misleading. Note that testing that the data is exchangeable is equivalent to testing the data for being independent and identically distributed (i.i.d) (Fedorova et al., 2012).

Vovk et al. (2003) proposed a way of testing exchangeability in an online manner based on Conformal Prediction (CP) and Conformal Martingales (CM). The technique proposed by (Vovk et al., 2003) consists of calculating a sequence of p-values by applying conformal prediction. The p-values are calculated in an online manner where the p-value of each new example is calculated from the new example and the previously seen examples. After the p-values are calculated a betting function is applied on each p-value and the product of the betting function outputs is the value of the Martingale. When the value M of the Martingale, becomes large enough we can reject the exchangeability assumption at significance level 1/M.

An important problem of using the original CM method for CD detection is its computational inefficiency since it requires retraining every time a new instance is observed. For this reason, in this work we use its inductive version, called ICM, which avoids this issue. However even with the inductive CM the whole process of detecting CD might be computationally inefficient. This is due to the fact that if we want the martingale to detect changes quickly in streaming data a betting function equal to the p-values density function is required. For this reason we use the histogram density estimator as a betting function which is more computationally efficient than the kernel density estimator (Eliades and Papadopoulos, 2020). Even though a well tuned kernel density estimator might perform better on CD detection, the computational time it requires is prohibitive for processing the huge amounts of data typical in CD detection tasks. Additionally another issue of the original CM and ICM is that it might need a lot of time to recover from a low value (Volkhonskiy et al., 2017), for this reason we propose a modification in which we restrict its lowest value to ensure that it will recover quickly from a low value and an alarm will be raised. Furthermore when calculating the histogram betting function we set the number of bins to be a function of the instances seen so far. This allows us to detect CD quickly and retrain our model again on the new data therefore recovering accuracy. We test the proposed approach on two synthetic and one real benchmark data sets. We present different measures of its performance and compare it with existing methods in the literature.

The rest of the paper starts with an overview of related work on CD in Section 2. Section 3 gives a brief overview of the ideas behind ICM. Section 4 describes the proposed approach and defines the betting function used for our ICM. Section 5 presents the experimental setting and performance measures used in our evaluation and reports the obtained experimental results. Finally, Section 6 gives our conclusions and plans for future work.

2. Related Work

In the literature there are many papers dealing with CD. Due to the large volume of research, we will present only few works related with the method we will follow and the datasets we used. We also present some work using CM and ICM which are directly connected with our approach.

Lu et al. (2019) surveyed reviews over 130 high quality publications in CD related research areas, they list and discuss 10 popular synthetic datasets and 14 publicly available benchmark datasets used for evaluating the performance of learning algorithms dealing with CD.

Bagui and Jin (2020) surveyed works dealing with CD, they presented a comprehensive study of public synthetic and real datasets that can be used to benchmark such a problem. They review the different types of drifts and approaches that used to handle such changes in the data.

Elwell and Polikar (2011) introduced an ensemble based approach for incremental learning of CD where the underlying data distributions change over time. The proposed algorithm learns from consecutive batches of data without making any assumptions on the nature or rate of drift, a new classifier is trained for each batch of data it receives and combines these classifiers using a dynamically weighted majority voting.

Li et al. (2015) implemented a classification algorithm based on Ensemble Decision Trees for Concept-drifting data streams (EDTC). Extensive studies on synthetic and on real streaming databases demonstrate that the EDTC performs very well compared to several known online algorithms based on single models and ensemble models.

Bu et al. (2017) proposed an incremental least squares density difference (LSDD) change detection method, their method is based on examining the difference between two distributions using two non overlapping windows. They tested their method on 6 synthetic datasets and on one real world dataset.

Fok et al. (2017) used a particle filter-based learning method (PF-LR), for training logistic regression models from evolving data streams. Here the step particles are sampled from the ones that maximize the classification accuracy on the current data batch. Their experiments show that this method gives good performance, even with relatively small batch sizes. They tested the proposed methods on both synthetic and real data sets and find that PF-LR outperforms some other state-of-the-art streaming mining algorithms on most of the data sets tested. Below we present two algorithms dealing with CD that according to Fok et al. (2017) can be considered as state of the art.

Kolter and Maloof (2007) presented an ensemble method for CD that creates, weights or remove online learners based on their performance, also based on the overall performance of the ensemble it can add new experts or remove them, they call their method Dynamic Weighted Majority (DWM), in this method if the base classifier is Naive Bayes then we call it (DWM-NB).

Bifet et al. (2010) proposed a new method called Hoeffding Tree Leveraging Bagging (LB-HT). This method combines the simplicity of bagging with adding more randomization to the input, and output of the classifiers.

CM introduced by Vovk et al. (2003) can be used as a tool for testing if a set of data satisfies the EA and for change point detection in time series. Some related techniques are presented below.

Ho et al. (2019) proposed a real time novel martingale-based approach driven by Gaussian process regression (GPR) to predict and detect anomalous flight behavior as observations are observed one by one. The authors here use multiple CM tests allowing them to reduce the number of false alarms and also the delay time required for anomaly detection.

Ho (2005) implemented a CM based on a simple betting mixture function extending the idea of detecting exchangeability online to detect concept changes in time-varying data streams. Two martingale tests were implemented based on: (i) martingale values and (ii) the martingale difference.

Fedorova et al. (2012) tested the exchangeability of data on two data-sets, USPS and Statlog Satellite data. The data is tested in an online manner i.e. the examples arrive one by one and then the value of the CM is calculated which is a valid measure indicating if the EA should be rejected. In this article the authors used a density estimator of the observed p-values as a betting function. The kernel density estimation has been employed and it has been shown that it outperforms the simple mixture betting function.

Volkhonskiy et al. (2017) implemented an Inductive version of CM to detect when a change occurs in a time series. In this study the underlying model is trained on the first observations of the time sequence. All the nonconformity scores are calculated via the underlying model. The authors tested their methods on synthetic data-sets and showed that their results are comparable with those of many other methods.

3. Inductive Conformal Martingales

In this study we test for CD using ICM and a modified version of ICM. At the point at which the CD occurs the EA will be violated thus a method that effectively tests the exchangeability of data will be sufficient for detecting CD. In this section we describe the basic concepts of ICM and how our nonconformity scores and p-values are calculated.

3.1. Exchangeability

Let $(Z_1, Z_2, \dots)$ be an infinite sequence of random variables. Then the joint probability distribution $\mathbb{P}(Z_1, Z_2, \dots, Z_N)$ (where N is natural number) is exchangeable if it is invariant under any permutation of these random variables. The joint distribution of the infinite sequence $(Z_1, Z_2, \dots)$ is exchangeable if the marginal distribution of $(Z_1, Z_2, \dots, Z_N)$ is exchangeable for every N. Testing if the data is exchangeable is equivalent to testing the data for being i.i.d, this is an outcome of de Finetti's theorem (Schervish, 1995) any exchangeable distribution on the data is a mixture of distributions under which the data is i.i.d.

3.2. Exchangeability Martingale

A test exchangeability Martingale is a sequence of random variables $(S_1, S_2, S_3, \dots)$ being equal to or greater than zero that keep the conditional expectation $\mathbb{E}(S_{n+1}|S_1, \dots, S_n) = S_n$.

To understand how a martingale works consider a fair game where a gambler with infinite wealth follows a strategy that is based on the distribution of the events in the game. The gain acquired by the gambler can be described by the value of a Martingale. Specifically Ville's inequality (Ville, 1939) indicates that the probability to have high profit(C) would be small, $\mathbb{P}\{\exists n : S_n \geq C\} \leq 1/C$.

According to Ville's inequality (Ville, 1939) for the case of the EA a large final value of the Martingale suggests rejection of the assumption with a significance level equal to the inverse of the Martingale value. Specifically, a Martingale value such as 10 or 100 rejects the hypothesis of exchangeability at 10% or 1% significance level, respectively.

3.3. Calculating Non-conformity scores and p-values

As mentioned in the Section 1 instead of the original CM version we use the computational efficient inductive version ICM. Let $\{z_1, z_2, \dots, z_k, \dots, z_n\}$ be a sequence of examples, where $z_i = (x_i, y_i)$ with x_i an object given in the form of an input vector, and $y_i \in R$ the label of the corresponding input vector. The first k examples $(z_1, z_2, \dots, z_k)$ will be used to train our classification algorithm.

The examples $\{z_{k+1}, \dots, z_n\}$ arrive one by one so we want to examine how strange or unusual a new example z_j is compared to the training examples. To make this possible a numerical value will be assigned to each example called nonconformity score (NCS) denoted as α_i and equal to $(z_i, \{z_1, \dots, z_k\})$ with $i \in \{k+1, \dots, n\}$. The calculation of the NCS is based on the underlying algorithm. A big value of the nonconformity measure indicates a strange example and a small value indicates a less strange example.

For every new example z_j we calculate the sequence $H_j = \{\alpha_{k+1}, \alpha_{k+2}, \dots, \alpha_{j-1}, \alpha_j\}$ to find its p-value. Note that the NCS in H_j are calculated when the underlying algorithm is trained on $\{z_1, z_2, \dots, z_k\}$. Given the sequence H_j we can calculate the corresponding p-value (p_j) of the new example z_j with the function:

$$p_j = \frac{|\{\alpha_i \in H_j | \alpha_i > \alpha_j\}| + U_j \cdot |\{\alpha_i \in H_j | \alpha_i = \alpha_j\}|}{j-k}, \tag{1}$$

where α_j is the NCS of the new example and α_i is the NCS of the i^{th} element in the example series set and U_j is a random number from the uniform distribution (0,1).

For the three benchmark data-sets tested on this study we have used a simple tree classifier. For each example the tree classifier will output the posterior probability $\tilde{p}_j$ for each label y_j. Therefore we define the NCM: $\alpha_j = -\tilde{p}_j$.

3.4. Exchangeability Martingales

An ICM is an exchangeability test Martingale (see Subsection 3.2) which is calculated as a function of p-values such as the ones described in Subsection 3.3.

Given a sequence of p-values $(p_1, p_2, \dots)$ the martingale S_n is calculated as:

$$S_n = \prod_{i=1}^{n} f_i(p_i) \tag{2}$$

where $f_i(p_i) = f_i(p_i | p_1, p_2, \dots, p_{i-1})$ is the betting function.

Equation 2 should satisfy the constraint: $\int_0^1 f_i(p)dp = 1$ and also it follows that $\mathbb{E}(S_{n+1}|S_0, S_1, \dots, S_n) = S_n$.

The integral $\int_0^1 f_i(p)dp$ equals to 1 because $f_i(p)$ is the p-values $(p_1, p_2, \dots, p_{i-1})$ density estimator. We also need to prove that $\mathbb{E}(S_{n+1}|S_0, S_1, \dots, S_n) = S_n$ under any exchangeable distribution.

Proof $\mathbb{E}(S_{n+1}|S_0, S_1, \dots, S_n) = \int_0^1 \prod_{i=1}^{n} f_i(p_i) \cdot f_{n+1}(p)dp = \prod_{i=1}^{n} f_i(p_i) \cdot \int_0^1 f_{n+1}(p)dp = \prod_{i=1}^{n} f_i(p_i) = S_n$ ■

Using equation (2) it is easy to show that $S_n = S_{n-1} \cdot f_n(p_n)$ which allows us to update the martingale online. Let's say that the value of S_n is equal to M then Ville's inequality (Ville, 1939) suggests that we can reject the EA with a significance level equal to $1/M$.

4. Proposed Approach

In this section we describe the proposed approach i.e. the ICM, the modified ICM as well as the betting functions we used. We examine if a sequence of betting functions equal to the probability density estimation of the p-values seen so far combined with ICM and its modification is sufficient enough for the rejection of the EA. When the EA is rejected at a point with timestamp i a CD is detected, we retrain our prediction's model to regain accuracy. The density function estimator $\hat{f}_n$ we used is the simple histogram. Note that the ICM is able reject the EA with respect to a significance level while our modified version is evaluated empirically using empirical evidence.

```
Data: Training set $\{z_1, z_2, \dots, z_k\}$, Test set$\{z_{k+1}, \dots, z_n\}$, significance level $\delta$
Initialize $S_1 = 1$
for $i$=1,...,n-k do
    input $z_i$
    $\alpha_i = A(z_{k+i}, \{z1, \dots, z_k\})$
    $p_i = \frac{\#\{j:\alpha_j>\alpha_i\}+U_j\#\{j:\alpha_j=\alpha_i\}}{i}$
    Calculate density estimator $f_i = f(p_{i-1-L}, \dots, p_{i-1})$
    $S_i = S_{i-1} \cdot f_i(p_i)$
    if $S_k > \frac{1}{\delta}$ then
        Raise an Alarm
    end
end
```

Algorithm 1: Detect CD using ICM

4.1. Histogram Estimator

Our goal here is to calculate a density estimator $\hat{f}_n$ of the density distribution of the p-value series: $p_1, p_2, \dots, p_{n-1}$. It should be beneficial that the calculation of this estimator is fast and it is desirable to have a small number of parameters to tune. The p-values $p_i \in [0, 1]$, so we partition $[0, 1]$ into a predefined number of bins k and calculate the frequency of the observations that lie in each bin, dividing these frequencies by the total number of observations and multiplying it by the number of bins gives us the histogram estimator.

Let us take a fixed number of bins κ this will partition $[0, 1]$ into $B_1 = [0, 1/\kappa)$, $B_2 = [1/\kappa, 2/\kappa)$,..., $B_{\kappa-1} = [(\kappa-2)/k, (\kappa-1)/\kappa)$ and $B_\kappa = [(\kappa-1)/\kappa, 1]$. When a p-value $p_n \in B_j$ then the density estimator will be equal to:

$$\hat{f}_n(p_n) = \frac{n_j.\kappa}{n-1}, \tag{3}$$

where $n-1$ is the number of p-values seen so far and n_j is the number of p-values belonging to B_j. Note that when n is small it is possible that that $\exists x : \hat{f}_n(x) = 0$, in that case until a sufficient number of observations arrives we set $\hat{f}_n = 1$ so that the martingale value doesn't become zero. Also here when f_n is calculated we use the p-values of the last $L = 10000$ seen observations, this is due to the fact that we wanted to accelerate the process of calculating f_n since 10000 p-values are sufficient enough to estimate the p-value density function.

4.2. Detecting CD using ICM

To reject the EA and thus detecting CD for a pre-specified significance level δ the value of the Martingale must exceed $1/\delta$. In Algorithm 1 we summarize this process.

Now if the value of the Martingale S_k at point k exceeds 10 or 100 then a CD is detected with a significance level equal to 10% and 1% respectively.

4.3. Detecting CD with the modified ICM

Here in Algorithm 2 we show a modified version of the ICM. We propose this modification because S_n will recover fast from low values ensuring that the CD will be detected faster.

```
Data: Training set $\{z_1, z_2, \ldots, z_k\}$, Test set$\{z_{k+1}, \ldots, z_n\}$, siginficance level $\delta$, $\epsilon$
Initialize $S_1 = 1$
for i=1,...,n-k do
    input $z_i$
    $\alpha_i = A(z_{k+i}, \{z1, \ldots, z_k\})$
    $p_i = \frac{\#\{j:\alpha_j>\alpha_i\}+U_j\#\{i:\alpha_j=\alpha_i\}}{i}$
    Calculate density estimator $f_i = f(p_{i-1-L}, \ldots, p_{i-1})$
    $S_i = max(S_{i-1} \cdot f_i(p_i), \epsilon)$
    if $S_k > \frac{1}{\delta}$ then
        Raise an Alarm
    end
end
```

Algorithm 2: Detect Concept Drift using modified ICM

Although we might have a potential loss in theoretical validity, our experiments show that the use of this method is still practical. The difference between Algorithm 1 is that we restrict the value of S_n such it will not take a value below ϵ.

4.4. Betting Function Computational Efficiency Comparison

In this paper we use the histogram betting function to calculate the value of the martingale S_i. To show why we have chosen this betting function instead of the kernel betting function we focus on the calculation of the density estimator $f_i = f(p_{i-1-L}, \ldots, p_{i-1})$ and its evaluation on a number of p-values $(f_i(p_i))$ equal to n. Where L is a the size of the sliding window (i.e. we use the last L p-values to calculate the density estimator). As mentioned in Section 1 the histogram betting function is more computationally efficient than the kernel betting function, note that both approaches have a computational complexity of $O(N)$, however the kernel density estimator is several times slower due to fact that it needs more operations. Note that the purpose of L is to control the number of operations and thus reducing computational time.

To examine the time that each betting function needs to be calculated we have generated ten vectors containing $n = 10000, \ldots, 100000$ random numbers uniformly distributed. For the histogram betting function we have used $L = 10000$ and a number of bins $\kappa = 22$ while for the kernel betting function we set L to 100 as in Volkhonskiy et al. (2017). This process is repeated ten times.

In Table 1 we show a comparison of the time needed to calculate and evaluate a histogram and a kernel density estimator ten times. The times reported here were obtained on a machine with a Ryzen 7 5800x cpu and 32gb ram. Table 1 results show that the kernel density estimator is several times slower than the histogram estimator. However to perform experiments on big datasets with multiple simulations to ensure reliability of the experimental results is an extremely time consuming task using the kernel density estimator. For this reason while detecting concept drift we use the histogram betting function. Another drawback of the kernel density estimator is to adjust and test the bandwidth parameter which again makes things even harder to implement a kernel density estimator on stream data. For example for the case of the SEA dataset (see Subsection 5.1) which consists of

12000000 instances and a drift occurs every 100000 to evaluate the p-values would take about 1.5 hours for every scenario.

Table 1: Computational time(sec.) of Histogram and Kernel betting function evaluation

n	**histogram**	**kernel**
10000	6.01	56.09
20000	19.52	112.13
30000	32.96	168.63
40000	46.63	224.78
50000	60.03	279.83
60000	73.26	336.03
70000	86.68	392.63
80000	100.23	448.78
90000	113.81	504.61
100000	127.50	560.94

5. Experiments and Results

This section describes the performance of our proposed approach on two synthetic datasets and on one real world dataset. We compare the proposed approach with existing methods in the literature Li et al. (2015), Bu et al. (2017), Fok et al. (2017).

5.1. Datasets

The synthetic benchmark data-sets we used are the STAGGER and the SEA, while the real world dataset we used is the ELEC.

The STAGGER Schlimmer and Granger (1986) is a standard benchmark data-set widely used for evaluating CD detection. For our simulations we have generated 1200000 instances, where each example is described by 3 categorical attributes and has binary target output. The drift type here is sudden and we have 4 concepts, a drift occurs every 10000 examples(i.e. every chunk consists of 10000 examples). Here while training the tree classifier the training set size is set to 300. We used a training set window of 300 because each categorical attribute can take up to 3 different values, also taking into account that it a binary problem we will at most 3.3.3=27 unique instances(noise free). Thus using a test set that contains 300 instances increases the possibility for our classifier to be trained on every possible attribute and label combination. Here the training set size compared to number of instances until a drift occurs is very small, specifically it is 3% of the total number of instances.

The SEA Street and Kim (2001) data-set is a popular synthetic dataset that contains sudden CD and is used for evaluating CD detection algorithms. For our simulations we have generated 12000000 instances, where each example is described by 3 numeric attributes and binary labels. Here we have 4 concepts, a drift occurs every 100000 examples(i.e. every chunk consists of 10000 examples). We have performed 30 simulations and for each one we have used 400000 instances. In this dataset while training the tree classifier the training set size is set to 1000. In this dataset the three variables take random values from 0 to 10. Particularly if the sum of the first two variables is less or equal to a pre-specified number then the instance will belong to class 1, the third variable is irrelevant. Thus by taking a training set that has 1000 instances will be sufficient for the tree classifier to construct the decision boundary. Also compared to the number of observations till a CD occurs its size is negligible. Note that here we test the transition from concept $a \to b \to c \to d$ while for the case of STAGGER we test the transition from concept $a \to b \to c \to d \to a, \ldots,$ also for the needs of this study we also inject in each of the above datasets 10% noise.

The ELEC data-set Harries et al. (1999) is a time series containing 45312 instances recorded at 30-min intervals, it's a binary problem (the class label identifies if we have a rise or a drop in the price compared with a moving average of the last 24 hours). Each example contains eight variables. This data has been collected from the Australian New South Wales Electricity Market. In our experiments we have excluded the variables time, date, day, and period. We ignore these three variables and we have only used nswprice, nswdemand, transfer, vicprice, and vicdemand. While assessing results we have performed 30 simulations and the training set size we use is set to 300. Here we try to make predictions for the future when we use as training set the observations of about only 1 week time.

Note that when a CD is detected we wait until a pre-specified number of observations arrives, we retrain the model and then we apply Algorithms 1 and 2.

5.2. Performance Measures

To evaluate the performance of the proposed approach on the synthetic datasets we consider four measures:

(a) Accuracy: Average accuracy of the predictive classifier (excluding the training set).

(b) Mean delay: Is the average number of observations needed to detect a CD after it occurs.

(c) Portion of true alarm: it represents the average portion of CD's that have been correctly detected per chunk.

(d) Portion of false alarm: It represents the average portion that Algorithms 1, 2 erroneously detected a change when no CD is present per chunk.

On the real life dataset we will use the accuracy as defined below and the number of CD detected.

5.3. Empirical Results

For each data-set we compare the results obtained with the ICM and the modified ICM when using the histogram betting function with its different parameters (see Algorithms 1 and 2). In the case of Algorithm 1 we tested its performance for $\delta = \{10, 100\}$ and for Algorithm 2 we used $\delta = \{10, 100\}$, $\epsilon = \{0, 1, 0, 01\}$. For both algorithms we applied different values of the number of histograms κ in the betting function, these are $\kappa = \{3, 4, 5, floor(i^{1/3} + 0.5)\}$. For simplicity we define $g(c) = floor(c^{1/3} + 0.5)$ and $c = max(i, 10000)$, where i is the number of data stream observations seen so far after the last training. Here as the value of δ increases we expect the portion of false alarms to decrease but at the expense of the time required to detect a CD, on the other hand as the value of δ decreases the time required to detect a CD decreases, the number of true alarms increases but we have a raise in false alarms. The purpose of the parameter ϵ in Algorithm 2 is to help the value of the modified ICM to recover quickly when it falls to a small value. As the value of epsilon increases we expect the number of true and false alarms to increase but the time required to detect a CD will be reduced, when we have a decrease in the value of ϵ the number of true and false alarms will be decreased but the required time to detect a CD will be increased. Note that while using $\kappa = g(i)$, κ increases as observations arrive and takes a maximum value of 22. We have used this formula because as the number of observations increases more histogram bins are required to approximate the p-value distribution. On the other hand, when the number of observations is small a smaller number of histogram bins is more appropriate making it more robust to statistical fluctuations.

5.3.1. STAGGER Data-set

In this subsection we present the results for the stagger dataset. The results shown in Table 2 are obtained using Algorithm 1. Here when the number of histogram bins increases the time required to detect a CD is reduced. Also the portion of false alarms stays low enough while keeping the detection of CD 100% accurate. The best accuracy and CD detection time is obtained when we set the number of bins to $g(i)$, while we also manage to keep the portion of false alarms low.

Table 2: STAGGER DATASET ALGORITHM 1

Noise	δ	**No of Bins**	**Accuracy**	**Mean delay**	**Portion of true alarm**	**Portion of false alarm**
0	10	3	0.97	547.43	1	0.12
0	10	4	0.97	516.15	1	0.11
0	10	5	0.97	507.34	1	0.11
0	100	3	0.96	596.21	1	0.01
0	100	4	0.97	558.86	1	0.02
0	100	5	0.97	529.91	1	0.01
10	10	3	0.92	515.33	1	0.05
10	10	4	0.92	505.31	1	0.04
10	10	5	0.92	489.43	1	0.07
10	100	3	0.92	579.77	1	0.00
10	100	4	0.92	541.87	1	0.01
10	100	5	0.92	514.77	1	0.01
0	10	g(i)	0.98	301.37	1	0.13
0	100	g(i)	0.98	320.45	1	0.02
10	10	g(i)	0.93	350.99	1	0.02
10	100	g(i)	0.93	364.84	1	0.01

The results shown in Table 3 are obtained using Algorithm 2 with $\epsilon = 0.1$. When comparing the results here with the ones shown in Table 2 we can see that the accuracy is improved, the time required to detect a CD is reduced, the portion of false alarms slightly increases especially for values of $\delta = 10$, the detection of CD stays at the level of 100%.

The best accuracy and CD detection time is obtained when we set number of bins equal to $g(i)$ while using $\delta = 100$ instead of 10 to keep the portion of false alarms low at acceptable levels.

Table 3: STAGGER DATASET ALGORITHM 1 ($\epsilon = 0.1$)

Noise	δ	**No of Bins**	**Accuracy**	**Mean delay**	**Portion of true alarm**	**Portion of false alarm**
0	10	3	0.98	361.88	1	0.12
0	10	4	0.98	300.45	1	0.12
0	10	5	0.98	261.24	1	0.20
0	100	3	0.97	453.24	1	0.01
0	100	4	0.98	373.99	1	0.01
0	100	5	0.98	320.74	1	0.01
10	10	3	0.93	357.68	1	0.05
10	10	4	0.94	290.16	1	0.06
10	10	5	0.94	255.96	1	0.10
10	100	3	0.93	433.34	1	0.01
10	100	4	0.93	365.69	1	0.01
10	100	5	0.93	310.48	1	0.02
0	10	g(i)	0.99	123.69	1	0.41
0	100	g(i)	0.99	146.04	1	0.04
10	10	g(i)	0.94	119.69	1	0.16
10	100	g(i)	0.94	155.35	1	0.01

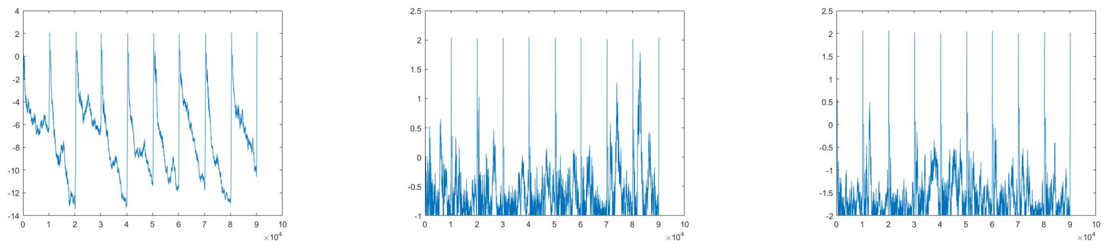
In Table 4 we present our results while using Algorithm 2 with $\epsilon = 0.01$. When comparing the results here with the ones shown in Table 3 we can see that there are cases in which accuracy is reduced, but still better than that reported in Table 2. The time required to detect a CD increases but improved compared with the ones shown in Table 2 especially when setting the number of bins equal to $g(i)$, the portion of false alarms is slightly decreased but is slightly worse with the ones shown in Table 2. We can observe that the best accuracy and CD detection time is obtained when the number of bins is set to $g(i)$. If we want to keep the portion of false alarms low we can use $\delta = 100$ keeping the practicability of this method.

When we use Algorithm 2 with $\epsilon = 0.1$, $\delta = 100$ and the number of bins set to $g(i)$ our results for this dataset are comparable with those of other studies. Li et al. (2015) while performing simulations on this dataset reported a portion of false alarms equal to 0.07, a portion of true alarms equal to 0.86, a CD detection time equal to 1877 and an accuracy equal to 0.82 and 0.60 with 0% and 10% noise respectively. Bu et al. (2017) reported a portion of false alarms equal to 0.004, a portion of true alarms equal to 1 while the CD detection time is 96.66 and an accuracy equal to 0.996. In their simulations no noise is injected.

Figure 1 shows the S_n growth when using Algorithms 1 with $\delta = 100$, Algorithm 2 with $\delta = 100$, $\epsilon = \{0.1, 0.01\}$ and the histogram betting function with the number of bins set to $g(i)$.

Table 4: STAGGER DATASET ALGORITHM 2 ($\epsilon = 0.01$)

Noise	δ	**No of Bins**	**Accuracy**	**Mean delay**	**Portion of true alarm**	**Portion of false alarm**
0	10	3	0.97	451.24	1	0.13
0	10	4	0.98	367.13	1	0.10
0	10	5	0.98	316.89	1	0.09
0	100	3	0.97	518.80	1	0.01
0	100	4	0.97	427.34	1	0.01
0	100	5	0.98	373.81	1	0.00
10	10	3	0.93	435.49	1	0.05
10	10	4	0.93	356.71	1	0.05
10	10	5	0.93	313.98	1	0.06
10	100	3	0.92	501.08	1	0.01
10	100	4	0.93	412.98	1	0.01
10	100	5	0.93	365.08	1	0.01
0	10	g(i)	0.99	148.62	1	0.11
0	100	g(i)	0.99	174.79	1	0.01
10	10	g(i)	0.94	154.56	1	0.04
10	100	g(i)	0.94	181.50	1	0.00



(a) $\delta = 100$ (b) $\delta = 100$, $\epsilon = 0.1$ (c) $\delta = 100$, $\epsilon = 0.01$

Figure 1: $log(S_n)$ growth as instances observed for the STAGGER dataset using Algorithm 1, 2

5.3.2. SEA Data-set

In this subsection we present the results for the SEA dataset. The results shown in Table 5 are obtained using Algorithm 1. Here roughly speaking when the constant number of histogram bins increases we have an increase in the value of accuracy and the time that required to detect a CD is reduced. Also the portion of false alarms stays low enough while the detection of CD in most cases is greater than 0.97, we have a drop in the true alarm portion when the dataset is injected with 10% noise and the number of bins is less or equal to 3. The best accuracy and CD detection time is obtained when we set the number of bins equal to 5 when we test on noisy data, while when we test on non noisy data the best accuracy and CD detection time is obtained when set the number of bins equal to $g(i)$.

Table 5: SEA DATASET ALGORITHM 1

Noise	δ	**No of Bins**	**Accuracy**	**Mean delay**	**Portion of true alarm**	**Portion of false alarm**
0	10	3	0.97	6044.30	1	0.05
0	10	4	0.97	5362.63	1	0.08
0	10	5	0.97	5451.50	1	0.03
0	100	3	0.97	6002.53	1	0.00
0	100	4	0.97	5096.03	1	0.00
0	100	5	0.97	5387.00	1	0.00
10	10	3	0.86	16213.38	0.70	0.03
10	10	4	0.87	7631.00	1	0.00
10	10	5	0.87	4698.17	1	0.00
10	100	3	0.86	18274.89	0.80	0.08
10	100	4	0.86	7876.97	1	0.00
10	100	5	0.87	4844.63	1	0.03
0	10	g(i)	0.97	3847.53	0.97	0.15
0	100	g(i)	0.97	3887.07	0.97	0.03
10	10	g(i)	0.87	7498.70	0.87	0.00
10	100	g(i)	0.87	7746.03	0.87	0.05

In Table 6 we show the results obtained using Algorithm 2 with $\epsilon = 0.1$. When comparing the results here with the ones shown in Table 5 we can see that the accuracy does not change much except for the case of non noisy data when we set the number of bins equal to $g(i)$, the time required to detect a CD greatly reduces, the portion of false alarms increases especially for values of $\delta = 10$, the detection of CD stays in most cases at the level of 100%.

The best accuracy and CD detection time is obtained when we set number of bins equal to $g(i)$ while using $\delta = 100$ instead of 10 to help us to keep the portion of false alarms low.

In Table 7 we present our results while using Algorithm 2 with $\epsilon = 0.01$. When comparing the results here with the ones shown in Table 6 we can see that in most cases the accuracy remains the same, the time required to detect a CD increases but is still less than the reported in Table 5, the portion of false alarms is slightly decreased, but is still more than that shown in Table 5. We can observe that the best accuracy and CD detection time

Table 6: SEA DATASET ALGORITHM 2 ($\epsilon = 0.1$)

Noise	δ	**No of Bins**	**Accuracy**	**Mean delay**	**Portion of true alarm**	**Portion of false alarm**
0	10	3	0.97	2636.97	1	0.13
0	10	4	0.97	2384.67	1	0.20
0	10	5	0.97	1983.77	1	0.38
0	100	3	0.97	2912.73	1	0.00
0	100	4	0.97	2598.50	1	0.05
0	100	5	0.97	2186.43	1	0.03
10	10	3	0.87	5140.73	1	0.18
10	10	4	0.87	2310.00	1	0.23
10	10	5	0.87	1553.73	1	0.28
10	100	3	0.87	7156	1	0.00
10	100	4	0.87	2625.23	1	0.03
10	100	5	0.87	1985.93	1	0.00
0	10	g(i)	0.97	614.27	1	1.48
0	100	g(i)	0.97	804.17	1	0.10
10	10	g(i)	0.87	1177.70	1	1.33
10	100	g(i)	0.87	1272.43	1	0.15

is obtained when setting number of bins to $g(i)$ this holds for both noisy and non noisy data.

The results indicate that setting the number of bins to $g(i)$ and $\delta = 100$ in Algorithm 2 greatly reduces the time required to detect CD while the portion] of false alarms is still comparable with that obtained when using Algorithm 1. Additionally we were able to detect all CD. These results are comparable with other methods found in bibliography. While using the PF-LR algorithm Fok et al. (2017) reported an average accuracy on the noiseless dataset equal to 0.98. Li et al. (2015) report an accuracy of 0.92 on noiseless data and an accuracy of 0.86 on 10% noise data, additionally they report that the CD was detected in almost all simulations and a minimum average time of detection equal to 6700, also they report a minimum portion of false alarm equal to 0.06. According to Fok et al. (2017) the two state of the arts algorithms DWM-NB, LB-HT mentioned in 2 give an accuracy of 0.965 and 0.963 respectively on the noiseless dataset.

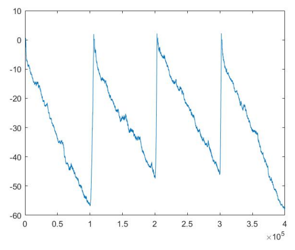
Figure 2 presents the S_n growth when using Algorithm 1 with $\delta = 100$, Algorithm 2 with $\delta = 100$, $\epsilon = \{0.1, 0.01\}$ and the histogram betting function with the number of bins set to $g(i)$.

5.3.3. ELEC Data-set

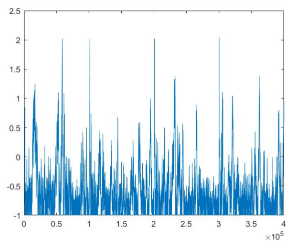
In this subsection we present the results for the ELEC dataset. In table Table 8 we show the results obtained using Algorithm 1. Here as the number of bins increases the accuracy improves, also the number of detected CD increases. When we set the number of bins to g(i) we have an improvement in accuracy and also an increase in the number of detected

Table 7: SEA DATASET ALGORITHM 2 ($\epsilon = 0.01$)

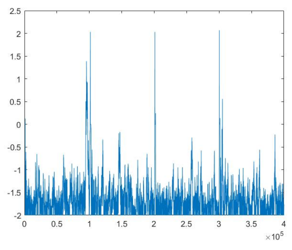
Noise	δ	**No of Bins**	**Accuracy**	**Mean delay**	**Portion of true alarm**	**Portion of false alarm**
0	10	3	0.97	3057.73	1	0.05
0	10	4	0.97	2624.23	1	0.13
0	10	5	0.97	2356.73	1	0.08
0	100	3	0.97	3293.27	1	0.03
0	100	4	0.97	2943.63	1	0.03
0	100	5	0.97	2498.17	1	0.03
10	10	3	0.87	7289.40	1	0.03
10	10	4	0.87	2684.43	1	0.03
10	10	5	0.87	1960.47	1	0.23
10	100	3	0.87	8880.6	1	0.00
10	100	4	0.87	3474.47	1	0.00
10	100	5	0.87	2283.87	1	0.00
0	10	g(i)	0.97	837.07	1	0.28
0	100	g(i)	0.97	1195.73	1	0.05
10	10	g(i)	0.87	1211.67	1	0.05
10	100	g(i)	0.87	1430.03	1	0.05



$(a)\ \delta = 100$



$(b)\ \delta = 100,\ \epsilon = 0.1$



$(c)\ \delta = 100,\ \epsilon = 0.01$

Figure 2: $log(S_n)$ growth for the SEA dataset using Algorithms 1, 2

CD. The best accuracy is obtained when we set $\delta = 10$ and the number of bins is set to $g(i)$.

Table 8: ELEC DATASET ALGORITHM 1

δ	**No of Bins**	**Accuracy**	**Number of cd detected**
10	3	0.72	34.87
10	4	0.72	35.93
10	5	0.73	34.73
100	3	0.71	20.60
100	4	0.72	26.87
100	5	0.73	27.74
10	g(i)	0.74	43.07
100	g(i)	0.73	32

In Table 9 we present our results while using Algorithm 2 with $\epsilon = 0.1$. When comparing the results here with the ones shown in Table 8 we can see that there is an improvement in the accuracy, also the number of detected CD increases. We can observe that the best accuracy is obtained when we set number of bins to $g(i)$ with a $\delta = 10$

Table 9: ELEC DATASET ALGORITHM 2 ($\epsilon = 0.1$)

δ	**No of Bins**	**Accuracy**	**Number of cd detected**
10	3	0.75	63.27
10	4	0.75	71.87
10	5	0.75	76.40
100	3	0.73	36.53
100	4	0.75	45.70
100	5	0.75	49.63
10	g(i)	0.76	84.13
100	g(i)	0.74	56.87

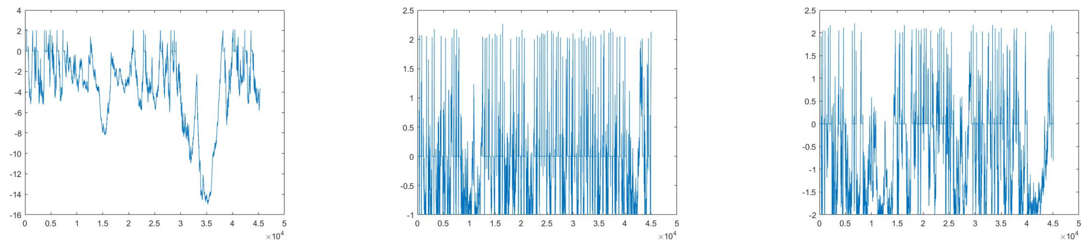
Table 10 presents the results while using Algorithm 2 with $\epsilon = 0.01$. When we compare the results here with the ones shown in Table 9 we can see a slight drop in accuracy and in the number of detected CD. Comparing these results with the ones shown in Table 8, the value of accuracy and the number of detected CD is greater. Here the best accuracy is obtained when we set the number of bins equal to $g(i)$ with a $\delta = 10$.

While performing simulations on this dataset the best averaged accuracy we have obtained equals to 0.76. This accuracy is achieved when we use Algorithm 2 with a $\delta = 10$ and by setting the number of bins equal to $g(n)$. In bibliography Fok et al. (2017) reported an average accuracy of 0.88 while using PF-LR, also this study reports that the two state of the art CD algorithms (DWM-NB and LB-HT) mentioned in Section 2 obtained an accuracy of 0.81, 0.86. Although the accuracy here is relatively low compared with some state of the arts algorithm and Fok et al. (2017) we believe that our algorithm can be improved. This

Table 10: ELEC DATASET ALGORITHM 2 ($\epsilon = 0.01$)

δ	**No of Bins**	**Accuracy**	**Number of cd detected**
10	3	0.73	46.33
10	4	0.75	52.87
10	5	0.75	56.60
100	3	0.72	28.80
100	4	0.73	36.27
100	5	0.74	38.70
10	g(i)	0.75	67.93
100	g(i)	0.74	48.03

is because we have used a tree classifier with default setting and we didn't optimize it's performance. Also the fact that our algorithm uses only 300 training instances each time a CD is detected makes is practically useful, this is because to obtain an accuracy of 0.76 only 56% of the dataset is used for training, furthermore for an accuracy of 0.75 only 30% of the dataset is used for training.



(a) $\delta = 100$ (b) $\delta = 100$, $\epsilon = 0.1$ (c) $\delta = 100$, $\epsilon = 0.01$

Figure 3: $log(S_n)$ growth as instances observed for the ELEC dataset using Algorithm 1, 2

Figure 3 presents the S_n growth when using Algorithm 1 with $\delta = 100$, Algorithm 2 with $\delta = 100$, $\epsilon = \{0.1, 0.01\}$ and the histogram betting function with the number of bins set to $g(i)$.

6. Conclusions

In this study we examined the use of ICM and a slightly modified version of ICM combined with a histogram betting function to examine their practicability on detecting CD. We have tested our approaches on two synthetic and one real life datasets. Our results show that the modification of the ICM improves accuracy and reduces the time needed to detect a CD. An increase in parameter δ can reduce the portion of false alarms and an increase in parameter ϵ can help the modified Martingale to recover from a low value. We also tested different settings on the bin numbers of the histogram function, we have seen that the best results are obtained when the number of bins increases as more observations arrive. As a future work we intend to improve the proposed approach by combining it with more advanced classifiers such as ensemble classifiers, other multiple hypothesis tests or other parallel tests to reduce the portion of false alarms.

References

Sikha Bagui and Katie Jin. A survey of challenges facing streaming data. *Transactions on Machine Learning and Artificial Intelligence*, 8(4):63–73, Aug. 2020. doi: 10.14738/tmlai.84.8579. URL https://journals.scholarpublishing.org/index.php/TMLAI/article/view/8579.

Albert Bifet, Geoff Holmes, and Bernhard Pfahringer. Leveraging bagging for evolving data streams. In José Luis Balcázar, Francesco Bonchi, Aristides Gionis, and Michèle Sebag, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 135–150, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-15880-3.

Li Bu, Dongbin Zhao, and Cesare Alippi. An incremental change detection test based on density difference estimation. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(10):2714–2726, 2017. doi: 10.1109/TSMC.2017.2682502.

Charalambos Eliades and Harris Papadopoulos. A histogram based betting function for conformal martingales. In Alexander Gammerman, Vladimir Vovk, Zhiyuan Luo, Evgueni Smirnov, and Giovanni Cherubin, editors, *Proceedings of the Ninth Symposium on Conformal and Probabilistic Prediction and Applications*, volume 128 of *Proceedings of Machine Learning Research*, pages 100–113. PMLR, 09–11 Sep 2020. URL http://proceedings.mlr.press/v128/eliades20a.html.

Ryan Elwell and Robi Polikar. Incremental learning of concept drift in nonstationary environments. *IEEE Transactions on Neural Networks*, 22(10):1517–1531, 2011. doi: 10.1109/TNN.2011.2160459.

Valentina Fedorova, Alex Gammerman, Ilia Nouretdinov, and Vladimir Vovk. Plug-in martingales for testing exchangeability on-line. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, page 923–930, Madison, WI, USA, 2012. Omnipress. ISBN 9781450312851.

Ricky Fok, Aijun An, and Xiaogang Wang. Mining evolving data streams with particle filters. *Computational Intelligence*, 33(2):147–180, 2017. doi: https://doi.org/10.1111/coin.12071. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/coin.12071.

Michael Harries, U Nsw cse tr, and New South Wales. Splice-2 comparative evaluation: Electricity pricing. Technical report, 1999.

Shen-Shyang Ho. A martingale framework for concept change detection in time-varying data streams. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML 05, page 321–327, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595931805. doi: 10.1145/1102351.1102392. URL https://doi.org/10.1145/1102351.1102392.

Shen-Shyang Ho, Matthew Schofield, Bo Sun, Jason Snouffer, and Jean Kirschner. A martingale-based approach for flight behavior anomaly detection. In *2019 20th IEEE International Conference on Mobile Data Management (MDM)*, pages 43–52, 2019. doi: 10.1109/MDM.2019.00-75.

J. Zico Kolter and Marcus A. Maloof. Dynamic weighted majority: An ensemble method for drifting concepts. *J. Mach. Learn. Res.*, 8:2755–2790, December 2007. ISSN 1532-4435.

Peipei Li, Xindong Wu, Xuegang Hu, and Hao Wang. Learning concept-drifting data streams with random ensemble decision trees. *Neurocomputing*, 166:68–83, 2015. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2015.04.024. URL https://www.sciencedirect.com/science/article/pii/S0925231215004713.

Jie Lu, Anjin Liu, Fan Dong, Feng Gu, João Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12): 2346–2363, 2019. doi: 10.1109/TKDE.2018.2876857.

Mark J. Schervish. *Theory of Statistics*. Springer, New York, 1995.

Jeffrey C. Schlimmer and Richard H. Granger. Incremental learning from noisy data. *Mach. Learn.*, 1(3):317–354, March 1986. ISSN 0885-6125. doi: 10.1023/A:1022810614389. URL https://doi.org/10.1023/A:1022810614389.

W. Nick Street and YongSeog 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, page 377–382, New York, NY, USA, 2001. Association for Computing Machinery. ISBN 158113391X. doi: 10.1145/502512.502568. URL https://doi.org/10.1145/502512.502568.

J. Ville. Étude critique de la notion de collectif. by j. ville. pp. 144. 75 francs. 1939. monographies des probabilités, calcul des probabilités et ses applications, publiées sous la direction de m. Émile borel, fascicule iii. (gauthier-villars, paris). *The Mathematical Gazette*, 23(257):490–491, 1939. doi: 10.2307/3607027.

Denis Volkhonskiy, Evgeny Burnaev, Ilia Nouretdinov, Alexander Gammerman, and Vladimir Vovk. Inductive conformal martingales for change-point detection. In Alex Gammerman, Vladimir Vovk, Zhiyuan Luo, and Harris Papadopoulos, editors, *Proceedings of the Sixth Workshop on Conformal and Probabilistic Prediction and Applications*, volume 60 of *Proceedings of Machine Learning Research*, pages 132–153, Stockholm, Sweden, 13–16 Jun 2017. PMLR. URL http://proceedings.mlr.press/v60/volkhonskiy17a.html.

Vladimir Vovk. Testing for concept shift online, 2020.

Vladimir Vovk, Ilia Nouretdinov, and Alexander Gammerman. Testing exchangeability on-line. pages 768–775, 01 2003.

Vladimir Vovk, Alex Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. 01 2005. doi: 10.1007/b106715.