Concept Drift Detection using Supervised Bivariate Grids

Christophe Salperwyck
EDF R&D
1 avenue du Général de Gaulle
92140 Clamart, France

Marc Boullé and Vincent Lemaire
Orange Labs
2 avenue Pierre Marzin
22300 Lannion, France

***Abstract*—We present an on-line method for concept change detection on labeled data streams. Our detection method uses a bivariate supervised criterion to determine if the data in two windows come from the same distribution. Our method has no assumption neither on data distribution nor on change type. It has the ability to detect changes of different kinds (mean, variance...). Experiments show that our method performs better than well-known methods from the literature. Additionally, except from the window sizes, no user parameter is required in our method.**

I. Introduction

Numerous actors from the "Information Technology" field have to deal with massive data as Google or Yahoo since they collect data coming from their user usage. Telecommunication companies also have millions of customers and have to deal with massive data to manage their telecommunication network and/or their customer relationship. The volume of these data continues to grow quickly and is not any more compatible with most of the off-line methods, which need to load all the data in memory. In these conditions, a solution could be to process data as they arrive. These data, called streaming data, are seen only once and in their order of arrival.

In supervised classification, the concept to learn $P(C|X)$ is the conditional probability of the class of C knowing the data observations X. A data stream could be non-stationary, includes concept drift, if the process which generate the data evolves over the time. In this case the classifier has to adapt gradually as the concept changes.

In this paper we propose a new method to detect the concept drift based on the monitoring of the data stream variables. Our method uses two sliding windows and is able to identify if the data of these two windows come from the same distribution or not. This method is able to detect different aspects (mean, standard deviation...) of the drift conditionally to the classes C of the supervised classification problem. This method has no assumption neither on the data distribution nor on the change type. Besides, except the window sizes, this method has no user parameter.

In the next section, the related work on change detections is described. The section III presents our detection method with an experimental validation on artificial data. The section IV shows how our detection method can be integrated on the top of a classifier to manage concept drifts. The last section concludes this paper.

II. Related work

The literature on concept drift detection or concept drift management is abundant [1], [2]. The methods on drift management can be split into several families: drift detection, ensemble of classifiers, samples weighting...

The goal of the supervised classification on data stream is to keep the performance of the classifier over the time, while the concept to learn is changing. In this case, literature contains two sub-families:

- methods without a classifier: authors use mainly the distributions of the explanatory variables of the data stream: $P(X)$, $P(C)$, $P(X|C)$
- methods using a classifier: authors use mainly the performances of the classifier related to the estimation of $P(C|X)$.

From a Bayesian point of view all these methods try to detect the variation of **a part of**:

$$P(C|X) = P(C)P(X|C)/P(X) \quad (1)$$

with

1) $P(C)$ the priors of the classes in the data stream
2) $P(X)$ the probability (distribution) of the data
3) $P(X|C)$ the conditional distribution of the data X knowing the class C.

The method proposed in this paper is able to detect the variation in the **three terms** $P(C)$, $P(X)$ and $P(X|C)$, which to our knowledge does not exist in the literature. The proposed method has few assumption on the data distribution, is resilient to outliers, does not use user parameter and has a good regularization to avoid over-fitting.

The literature on concept drift is large, the interested reader may find in [2] an overview of the existing methods. In the following subsections we present only a part of the literature related to drift detection methods applied on one of these three terms: $P(C)$, $P(X)$ and $P(X|C)$. The presented methods will be used in comparison to our method in this paper.

A. Methods without a classifier

The methods without a classifier are mainly methods based on statistical tests applied to detect a change between two observation windows.

a) Welch's t test: this test applies on two samples of data of size N_1 and N_2 and is an adaptation of the Student's t test. This test is used to statistically test the null hypothesis that the means of two population $\overline{X}_1$ and $\overline{X}_2$, with unequal variances (s_1^2 and s_2^2), are equals. The formula of this test is:

$$p\text{-}value = (\overline{X}_1 - \overline{X}_2) / \left(\sqrt{(s_1^2/N_1) + (s_2^2/N_2)} \right) \quad (2)$$

This test returns a $p\text{-}value$ which allows or not to reject the null hypothesis.

b) Kolmogorov-Smirnov's test: this test is often used to determine if a sample follow (or not) a given law or if two samples follow the same law. This test is based on the properties of their empirical cumulative distribution function. We will use this test to check if two samples follow the same law. Given two samples of size N_1 and N_2 having respectively cumulative distribution function $F_1(x)$ and $F_2(x)$, the Kolmogorov-Smirnov distance is defined as: $D = \max_x |F_1(x) - F_2(x)|$. The null hypothesis, assuming that the two samples follow the same law, is rejected with a confidence of α if: $\left(\sqrt{(N_1 N_2)/(N_1 + N_2)} \right) D > K_\alpha$. K_α can be found in the Kolmogorov-Smirnov table.

c) MODL $P(W|X_i)$: this method has been proposed by [3] to detect the change while observing a numerical variable X_i in an unsupervised setting. This method addresses this problem as a binary classification problem. The method uses two windows to detect the change: (1) W_{ref} contains the distribution of X_i at the beginning of the stream and is used as a reference window; (2) W_{cur} a current (sliding or jumping) window which contains the current distribution. The examples belonging to W_{ref} are labeled with the class "ref" and the ones belonging to W_{cur} are labeled with the class "cur". This two labels constitute the target variable $W \in \{W_{ref}, W_{cur}\}$ of the classification problem. All these examples are merged into a training database and the supervised MODL discretization [4] is applied to see if the variable X_i could be split in more than a single interval. If the discretization gives at least two intervals then there are at least two significantly different distributions for X_i conditionally to the window W. In this case the method detects that there is a change between W_{ref} and W_{cur}.

B. Methods using a classifier

Methods using a classifier monitor the performance of the classifier and detect a concept drift when the performance varies significantly. These methods assume that the classifier is a stationary process and data are independent and identically distributed (iid). Though, in case of a data stream these hypothesis are not valid [5], these methods proved their interest on diverse application [6], [7], [1]. The two most popular methods using a classifier of the literature are described below.

d) DDM: this method of Gama et al. [6], detects a concept drift by monitoring the classifier accuracy. Their algorithm assumes that the binary variable which indicates that the classifier has correctly classified the last example follows a binomial distribution law. This law can be approximated as a normal law when the number of observations is higher than 30. The method estimates after the observation of each sample of the data stream the probability of misclassification p_i (p_i corresponds also to the error rate) and the corresponding standard deviation $s_i = \sqrt{p_i(1-p_i)/i}$. A significant increase of the error rate is considered by the method as the presence of a concept drift.

The method uses two decision levels: a "warning" level when $p_i + s_i \geqslant p_{min} + 2 \cdot s_{min}$ and a "detection level" when $p_i + s_i \geqslant p_{min} + 3 \cdot s_{min}$ (after the last detection, every time a new example i is processed p_{min} and s_{min} are updated simultaneously according to $(p_{min} + s_{min}) = \min_i(p_i + s_i)$). The examples seen between the warning level and the detection level are used to train a new classifier which will replace the current classifier if the detection level is reached. This mechanism allows not to start from scratch when the decision level is reached (if the concept drift is not too sudden).

e) EDDM: this method [7] uses the same algorithm but with another criterion to set the warning and detection levels. This method uses the distance between the classification error rather the error rate. This distance corresponds to the number of right predictions between two wrong predictions.

EDDM computes the mean distance between the errors p'_i and the corresponding standard deviation s'_i As for DDM a warning level and a detection level are defined, respectively of $(p'_i + 2 \cdot s'_i)/(p'_{max} + 2 \cdot s'_{max}) < \alpha$ and $(p'_i + 2 \cdot s'_i)/(p'_{max} + 2 \cdot s'_{max}) < \beta$. In the experimental part the authors of EDDM use $\alpha = 90\%$ and $\beta = 95\%$. On synthetic datasets EDDM detects faster than DDM the gradual concept drift.

III. A new supervised detection approach

A. Presentation

The concept drift detection method presented in this paper is dedicated to supervised classification. The goal is to propose a method that is able to differentiate if the examples belonging to a reference window, W_{ref} and the examples belonging to a current window, W_{cur}, come from the same distribution conditionally to the class. One contribution of this paper is to pose this problem, similarly as in [3], as a supervised classification problem. We define:

- each sample of the data stream is described by d explanatory variables ($X_i, i \in \{1, ..., d\}$) and a class variable C with J values
- the examples get a label W corresponding to their window ($W \in \{W_{ref}, W_{cur}\}$)
- the explanatory variables are assumed to be independent conditionally to the classes
- the test detection is performed for each variable X_i on the data coming from the two windows (W_{ref}, W_{cur})

The window W_{ref} contains observations of the initial concept and is not updated while there is no drift detected. The second window W_{cur} is a sliding (or jumping) window on the data stream for the current concept. The reference window represents the normal operation of the observed system. The current window characterizes the present state of the system. Defining these two window sizes is the only required adjustment from the user.

TABLE I. TABLE USED TO TRAIN THE CLASSIFIER

		X_1	X_2	...	X_i	...	X_d	C	W
W_{ref}	1	...	...	...	...	...	...	$C \in \{1, ..., J\}$	W_{ref}
	2	...	...	...	...	...	...	$C \in \{1, ..., J\}$	W_{ref}
	3	...	...	...	...	...	...	$C \in \{1, ..., J\}$	W_{ref}
	...	...	...	...	...	...	...	$C \in \{1, ..., J\}$	W_{ref}
	$\|W_{ref}\|$	...	...	...	...	...	...	$C \in \{1, ..., J\}$	W_{ref}
W_{cur}	1	...	...	...	...	...	...	$C \in \{1, ..., J\}$	W_{cur}
	2	...	...	...	...	...	...	$C \in \{1, ..., J\}$	W_{cur}
	3	...	...	...	...	...	...	$C \in \{1, ..., J\}$	W_{cur}
	...	...	...	...	...	...	...	$C \in \{1, ..., J\}$	W_{cur}
	$\|W_{cur}\|$	...	...	...	...	...	...	$C \in \{1, ..., J\}$	W_{cur}

The first step is to define the position and the size of the windows[1] and to collect information in a table as represented in Table I. The examples of the data stream are then labeled according to their window (column W in the Table I). The examples belonging to the reference window (respectively to the current window) are labeled to the class W_{ref} (respectively to the class W_{cur}). The purpose is to exploit a supervised classifier to quantify the change in the distribution.

The following situations give the intuition of this kind of approach: (i) Assuming that the joint distribution $P(X, C)$ of examples has not changed between the two windows, classes (in the sense of W) are not separable. In that case, any robust classifier is not able to differentiate the two classes; (ii) Assuming that the joint distribution $P(X, C)$ has changed, the examples of the classes W_{ref} and W_{cur} do not have the same distribution. In that case, the classifier should be able to differentiate the classes. We are interested in the probability of the window knowing simultaneously the class C and the explanatory variable X_i: $P(W|C, X_i)$. We need a method able to accurately quantify the difference between $P(W_{ref}|C, X_i)$ and $P(W_{cur}|C, X_i)$. Supervised bivariate discretization / grouping methods are methods which can be used in this context.

Among many methods in the literature, we chose to use the MODL grid [8] for its characteristic: no a priori on the data distribution, no prior knowledge, resilient to outliers, no user parameter[2] and a good regularization to limit over-fitting. The cross-product of the discretization / grouping on each feature forms a multi-dimensional data grid. The correlation between the cells of this data grid and the output values allows the joint predictive information to be quantified. The trade-off between information and reliability is established using a Bayesian model selection approach embedded in the MODL criterion.

The MODL criterion estimates the cost of a model $c(M)$ knowing the data D. The value of the criterion $c(M)$ is related to the probability that a data grid model M explains the output variable given the input variables: $c(M) = log(P(M|D))$. This criterion is not detailed in this paper but interested readers can find all the details in [8]. For our drift detection method we have two possibilities: either X_i is a numerical variable or X_i is a categorical variable. In both cases C is a categorical variable. The numerical case corresponds to a mixed case of one categorical input variable C with J categories and one numerical input variable X_i, in that case $P(M|D)$ is defined by the equation 5 page 15 in [8]. The second case corresponds to two categorical input variables: C and X_i, in that case $P(M|D)$ is defined by the equation 6 page 15 in [8].

The criterion $c(M)$ can also be interpreted as the ability of a data grid model to compress the output classes given the input values. When the joint distribution of (X_i, C) does not change between the reference and the current window, it is impossible to separate the type of stream regime exploiting the variables jointly. By contrast, if the joint distribution of (X_i, C) have significantly changed, the joint distribution of these variables can be exploited to detect the type of the stream regime: the grid has more than one cell.

Let $M_\emptyset$ be the null model that discretizes the joint distribution (X, C) into a grid of a single cell. $M_\emptyset$ represents the coding length of the stream regime type without exploiting the input joint distribution. In the case where the joint distribution (X, C) does not change between the reference and the current window, it is impossible to separate the stream regime type: the null model is the best one. The compression gain [8] is defined as follows:

$$Gain(M) = 1 - \frac{c(M)}{c(M_\emptyset)} \tag{3}$$

In the experiments performed in this paper we use the value $Gain(Map)$: the compression gain of the most probable model given the data $Gain(Map)$ is 0 when it is not possible to separate the stream regime type. This value is strictly positive when there is a significant difference in the stream regime coming from the two windows.

Let the vertical axis of a grid represents the partition over C and the horizontal axis the partition over X_i. If $P(W|C, X_i)$ changes only conditionally to X_i the grid will be the same as in [3] and we will able to detect covariate shift over X_i: the grid will have only columns. If $P(W|C, X_i)$ changes only conditionally to C the grid will allow to detect changes over $P(C)$: the grid will have only lines. From a Bayesian point of view (see Section II) we will be able to detect the variation in the three parts of $P(C|X)$ (equation 1).

For our problem of concept drift handling, we detect a change if the target variable W can be separated using the values of X_i and C. In that case the detection is represented by a grid with more than one cell, otherwise there is no detection. Figure 1 illustrates the detection on a dataset with an explanatory numerical variable X_1 and 2 classes $C \in \{C_1, C_2\}$. The data taken from the W_{ref} are represented with green points and the ones from W_{cur} with red points. We search a MODL grid at two different periods of time, t_1 and t_2, in the data streams to check if the data distribution has changed. Data in W_{ref} stays the same for both periods and only data in W_{cur} changes. For the period t_1, on the top of the figure (where we detail the complete process), the MODL grid has just one cell since no split can be found to explain the data distribution conditionally to W. For the period t_2, on the bottom of the figure (where we show only the resulting grid), the MODL grid has 6 cells since knowing W can explain the data distribution.

B. Implementation details

1) Windows size: in the method proposed in this paper the size of the windows is a trade-off between reactivity and

[1] A temporal window is defined by a "start date" and an "end date", and includes the examples arrived during this time interval

[2] Making classifier simpler is an important point in the context of data stream. Constant manual adjustment of models is inefficient and with increasing amounts of data is becoming infeasible (see the discussion in [9])

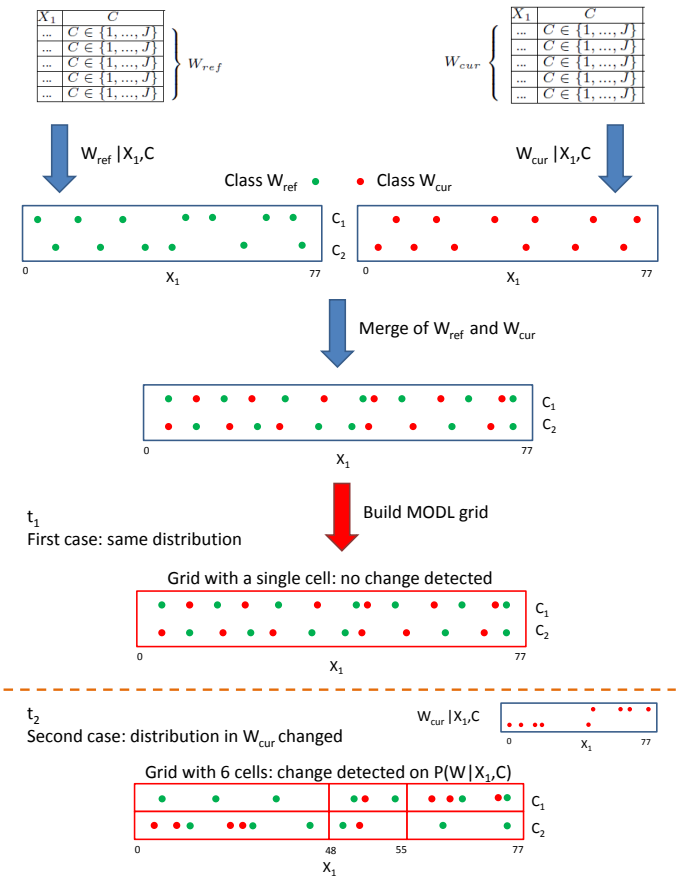

Fig. 1. On the top no change is detected in the data distribution as the grid has just one cell. On the bottom the grid has 6 cells and therefore a change on $P(W|C, X_i)$ is detected.

robustness. On one hand a large window provides a better confidence on a potential concept drift, on the other hand a small size is more reactive. From our point of view the window size setting is problem dependent. However the method could be applied using several sizes of windows in parallel.

2) Complexity: data grid models are optimized using a greedy bottom-up strategy. The algorithm starts from a fine grained partition of the variables and evaluates all merges between groups of values in the categorical case or intervals in the numerical case. The best merge is performed and the process iterates until there is no more improvement on the optimized criterion. A straightforward implementation of the greedy heuristic remains a hard problem with polynomial time complexity. Advanced optimizations combined with sophisticated algorithmic data structures are exploited to get a $O(N\sqrt{N}\log N)$ time complexity and $O(N)$ space complexity, where N is the number of data points in the window. These algorithms, fully detailed in [11], mainly exploits (i) the sparseness of the grid, (ii) the additivity property of the optimized criterion and (iii) starts from non-maximal grained grid models using pre and post-optimization heuristics.

3) Independence assumption: the potentially high rate of the input data stream leads to have a method with a low complexity. That is the reason why we have chosen to simplify the initial learning task into d univariate detection problems which are described above in this paper. While this simplification may seem drastic, this methodological choice provides a pragmatic response to the curse of dimensionality. In practice, few cases of change in the distribution cannot be detected by examining each variable individually. This naive Bayes assumption implies that the complexity grows linearly with the number of variables (d).

For example, the naive Bayes (NB) classifier has proved to be very effective in many real data applications [12], [13]. It is based on the assumption that the variables are independent within each class, and solely relies on the estimation of univariate conditional probabilities, which make its training time complexity linear w.r.t. the number of input variables. In our method, we solely rely on the estimation of $P(W|X_i, C)$ and expect to get the effectiveness of the naive Bayes classifier.

C. Experimental validation: detection without a classifier

In this section, we focus on how our new method performs on artificial datasets generated with different kinds of change. The detection is performed on only one variable X_a for those experiments. Since we made the assumption of independence between variables, testing on more variables would not give better insight on the capabilities of our method. The first section studies the robustness when the concept is stationary, the second section the time needed to detect abrupt changes and the last section the ability to deal with different kinds of change. The experiments compare the methods from the literature described previously and our new method:

- Welch's t test with statistical significance of 1%, 5% and 10%
- Kolmogorov-Smirnov (KS) with statistical significance of 1%, 5% and 10%
- supervised MODL method on $P(W|X_i)$
- supervised bivariate MODL method on $P(W|X_i, C)$

1) Robustness in the stationary case: The goal of this first experiment is to study the robustness of the methods when the data stream is stationary. In that case no change should be detected. The reference and current windows have the same size. The class distributions follow a Gaussian with the following parameters: $(\mu_1 = -1, \sigma_1 = 1)$ for the class with label 1, and $(\mu_2 = 1, \sigma_2 = 1)$ for the class with label 2. Different window sizes are used: from 10 to 5,000 examples. All experiments are repeated 1,000 times.

Results are presented in the Table II. They confirm the robustness of our bivariate MODL method on $P(W|X_a, C)$ as no detection were observed. The MODL method on $P(W|X_a)$ [3] has few false detections but they mainly happen on small window sizes. The other tests behave well with small statistical significance (1%). For higher significance value (10%), much more false detections occur for Welch's t test and Kolmogorov-Smirnov test.

2) Abrupt change detection: the goal of this experiment is to measure the number of examples needed to detect a change depending on the window sizes and methods. All these methods are able to detect the abrupt change used in this section but here we focus on how fast they can perform this detection.

TABLE II. NUMBER OF FALSE DETECTIONS PER METHOD AND WINDOW SIZE FOR 1,000 EXPERIMENTATIONS.

method→ ↓ size	Welch (1%)	Welch (5%)	Welch (10%)	KS (1%)	KS (5%)	KS (10%)	MODL $P(W\|X_a)$	MODL $P(W\|X_a, C)$
10	0	9	20	0	0	8	3	0
20	0	7	13	0	5	22	4	0
30	1	6	19	0	3	12	3	0
50	0	5	21	0	2	17	6	0
100	0	3	19	0	4	20	1	0
200	0	5	28	0	5	13	0	0
300	0	6	16	1	5	21	0	0
500	1	6	22	2	7	18	1	0
1000	1	8	25	0	5	24	0	0
2000	0	4	13	0	7	20	0	0
5000	0	6	26	0	7	19	0	0

TABLE III. MEAN DELAY TO DETECT A CHANGE FOR A GIVEN METHOD AND A GIVEN WINDOW SIZE ON 1,000 EXPERIMENTS.

method→ ↓ size	Welch (1%)	Welch (5%)	Welch (10%)	KS (1%)	KS (5%)	KS (10%)	MODL $P(W\|X_a)$	MODL $P(W\|X_a, C)$
10	15	14	14	15	15	15	15	15
20	29	26	24	29	28	26	28	29
30	41	36	33	43	40	36	41	40
50	62	53	49	67	57	54	65	58
100	103	90	86	110	96	90	109	96
200	178	160	150	185	166	160	195	163
300	241	218	211	252	227	214	269	218
500	367	337	321	375	344	330	404	325
1000	665	620	598	678	636	613	719	600
2000	1224	1188	1154	1260	1188	1174	1334	1120
5000	2886	2766	2741	2911	2781	2756	3081	2671

The reference and current windows have the same sizes. Only one feature X_a is used. Concept 1 is defined with (X_a|class 0) following $\mathcal{N}_0(\mu = -1, \sigma = 1)$ and (X_a|class 1) following $\mathcal{N}_1(\mu = 1, \sigma = 1)$. The change is simulated with a modification of the parameters. After the change we have the following concept 2 defined with (X_a|class 0) following $\mathcal{N}_0(\mu = 2, \sigma = 0.5)$ and (X_a|class 1) following $\mathcal{N}_1(\mu = 0, \sigma = 1)$.

The window sizes are from 10 to 5,000 examples and all the experiments are repeated 1,000 times. The change position within the window is taken randomly as in a real scenario this position is unknown. The obtained results show the mean delay to detect the change depending on the window size and the chosen method.

The results are presented in the Table III. With a window size smaller than 100 examples, we observe that it is difficult to detect the change. From a window size of 200 examples, the mean delay is less than the window size for all the methods. Increasing the statistical significance threshold to 5% and 10% for Welch's t test and Kolmogorov-Smirnov test decreases the delay. The method based on MODL $P(W|X_a)$ is slightly longer to detect than the parametric methods configured to 1%. However if we compare with our method MODL $P(W|X_a, C)$ it performs better than the two other methods configured with 1%.

3) Gradual Change Detection: the goal of this experiment is to study the behavior of the methods when the concept drift is gradual. This behavior is examined through the variation of the criteria used by each method versus the "quantity of change". These criterion are :

- for the Welch's test: the p-value of the test (we took the log to ease the comparison)

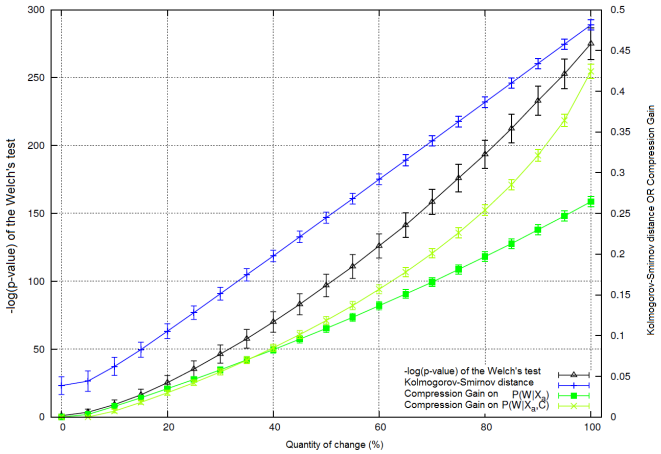

Fig. 2. Evolution of the different criterion versus the amount of change

- for Kolmogorov-Smirnov (KS): the Kolmogorov-Smirnov distance
- for the supervised MODL method on $P(W|X_i)$: the compression gain
- for the supervised bivariate MODL method on $P(W|X_i, C)$: the compression gain

For this experiment two concepts are used. Concept 1 is defined with (X_a|class 0) following $\mathcal{N}_0(\mu = 0, \sigma = 1)$ and (X_a|class 1) following $\mathcal{N}_1(\mu = 0, \sigma = 1)$. Concept 2 is defined with (X_a|class 0) following $\mathcal{N}_0(\mu = 0, \sigma = 1)$ and (X_a|class 1) following $\mathcal{N}_1(\mu = 4, \sigma = 1)$.

The reference window contains the concept 1. The current window contains the concept 1 and a percentage of the concept 2 to simulate the quantity of change (from 0% to 100% with an incremental step of 5%).

The results of this experiment are presented in the Figure 2. We can observe that the variation of the four criteria are proportional to the amount of change. For this gradual detection change experiment with a simple drift of the mean of class 1, all methods have a similar behavior. Next section investigates on more complex pattern changes.

4) Detection of different types of change: the purpose of the experiments presented in this section is to observe which types of change (mean, variance, classes inversion) the studied methods are able to detect. For all types of change, the concept to learn at the beginning is the concept 1 defined with two classes: (X_a|class 0) follows a distribution $\mathcal{N}_0(\mu = 0, \sigma = 0.5)$ and (X_a|class 1) follows a distribution $\mathcal{N}_1(\mu = 2, \sigma = 1)$. The different types of change are applied on concept 1 to experiment the behavior of the different methods (see Figure 3):

- **Change in the mean (Figure 3-a):** this change is simulated by changing the mean in the class 0 distribution: from 0 to 1. This change produces the concept 2: (X_a|class 0) follows a distribution $\mathcal{N}_0(\mu = 1, \sigma = 0.5)$ and (X_a|class 1) follows a distribution $\mathcal{N}_1(\mu = 2, \sigma = 1)$.

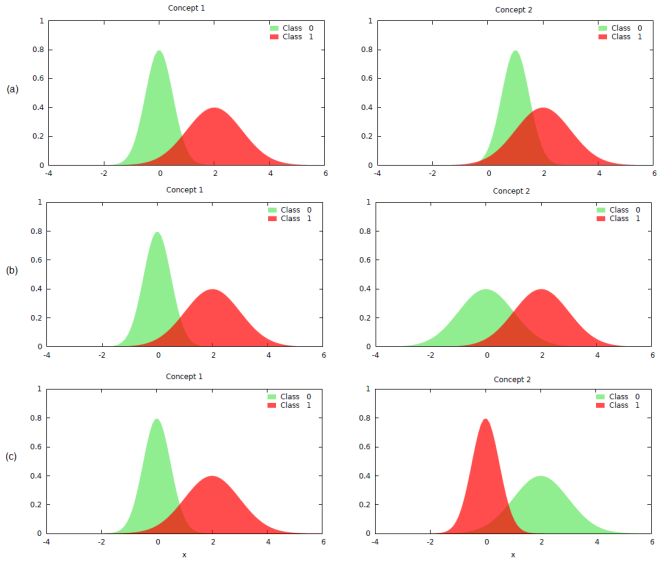

Fig. 3. Types of change studied: (a) change in the mean, (b) change in the variance, (c) class inversion.

- **Change in the variance (Figure 3-b):** this change is simulated by changing the variance in the class 1 distribution: from 0 to 0.5. This change produces the concept 2: $(X_a|\text{class 0})$ follows a distribution $\mathcal{N}_0(\mu = 0, \sigma = 1)$ and the $(X_a|\text{class 1})$ follows a distribution $\mathcal{N}_1(\mu = 2, \sigma = 1)$.
- **Class inversion (Figure 3-c):** this change is simulated by the inversion of the labels of the concept 1

TABLE IV. NUMBER OF DETECTIONS PER CHANGE TYPE FOR THE DIFFERENT METHODS.

Method	Mean	Variance	Inversion
Welch (1%)	1000	0	0
Welch (2%)	1000	0	0
Welch (10%)	1000	19	10
KS (1%)	1000	998	0
KS (2%)	1000	1000	0
KS (10%)	1000	1000	15
MODL $P(W\|X_a)$	1000	1000	1
MODL $P(W\|X_a, C)$	1000	1000	1000

In these experiments, for all the change types, the sizes of the windows are set to 1,000. This value corresponds to a good trade-off between reactivity and robustness according to the results above (see Table III and Table II). The reference window contains the concept 1 and the current window contains the concept 2 The results obtained in terms of number of detection over 1,000 experiments are presented in Table IV. The best methods are those able to find most of the changes.

Changes only on X: all methods are able to detect in a reliable way a change in the mean. As expected the Welch's test is not able to detect a change in the variance. A small number (19) of detections appears but they correspond to false detections and not to a change in the variance. The Kolmogorov-Smirnov's test and the MODL detection $P(W|X_a)$ are able to detect changes in the mean and variance.

Changes on X, Y: the Welch's test, the Kolmogorov-Smirnov's test and the MODL detection $P(W|X_a)$ are not able to detect class inversion. Contrary to these detection methods, the MODL detection $P(W|X_a, C)$ based on a supervised grid is able to detect the inversion and these experiments confirm its ability to detect a change on X, C, or X, C.

TABLE V. SYNTHETIC SUMMARY OF THE COMPARISONS

Criteria → Method ↓	False detections	Change in the mean	Change in the variance	Class inversion
Welch (1%)	+	+	-	-
Welch (2%)	-	+	-	-
Welch (10%)	- -	+	-	-
KS (1%)	+	+	+	-
KS (2%)	-	+	+	-
KS (10%)	- -	+	+	-
MODL $P(W\|X_a)$	+	+	+	-
MODL $P(W\|X_a, C)$	+ +	+	+	+

D. Discussion

The proposed method based on MODL $P(W|X_a, C)$ is robust: it does not detect changes on a stationary data stream. The methods based on the Welch's test and the Kolmogorov-Smirnov's test are interesting when they are used with a high confidence value. However the false detection increases quickly when the confidence value is lower (see Table IV). Our method is also faster to detect the changes than the two methods based on statistical tests (see Table III). To obtain the same reactivity, the confidence value has to be decreased but this leads to a loss of robustness. Our method is both robust and fast to detect a change. Besides it is able to detect different types of change: in the mean, in the variance and in inversion of the pattern. A synthetic summary of the comparisons is presented in the Table V.

IV. APPLICATION TO CONCEPT DRIFT MANAGEMENT

The concept drift detection is not intrinsically a final goal. The detection aims to be used to react to the changes. As described in [2], [15], having a detection method on a data stream gives several possibilities to deal with a change:

i: retrain the model from scratch;

ii: adapt the current model;

iii: adapt data statistics or data summaries on which the model is based on;

iv: weight, adapt or use a "sequence" of models trained over the time.

The purpose of this section is to propose a method which integrates the detection method presented above in this paper in an algorithm able to manage the concept drift. The objective is to replace the current model when the method detects a change.

A. Algorithm MDD

We integrated our detection method in an algorithm named MDD: MODL Drift Detection. This algorithm is presented below in Algorithm 1.

In this algorithm the replacement of the current classifier ($M_{current}$) is done after a detection on at least one of the explanatory variables using the bivariate discretization method

Notations:

- x: an example of the data stream
- W_{ref}: reference window of size $|W_{ref}|$
- W_{cur}: current jumping window of size $|W_{cur}|$
- $size(W)$: number of elements in W
- $ER_{.}$: error rate of the considered classifier
- `computeER`: compute error rate of the considered classifier
- $M_{current}$: currently used classifier with its error rate $ER_{M_{current}}$
- M_{new}: new classifier trained after a detection with its error rate $ER_{M_{new}}$
- n: number of examples used to train M_{new}
- n_{min}: minimal number of examples before comparing $ER_{M_{current}}$ with $ER_{M_{new}}$

$n \leftarrow 0$
while $x \leftarrow$ `data stream ()` **do**
  `// if a` M_{new} `exists: is it better than` $M_{current}$`?`
  if `isStarted` (M_{new}) **then**
    $n \leftarrow n + 1$
    $ER_{M_{current}} \leftarrow$ `computeER` $(ER_{M_{actual}}, M_{current}, x)$
    $ER_{M_{new}} \leftarrow$ `computeER` $(ER_{M_{new}}, M_{new}, x)$
    `train` (M_{new}, x)
    if $n > n_{min}$ *and* $ER_{M_{new}} < ER_{M_{current}}$ **then**
      $W_{ref} \leftarrow \emptyset$, $W_{cur} \leftarrow \emptyset$, $M_{current} \leftarrow M_{new}$
      $ER_{M_{current}} \leftarrow 0$, $ER_{M_{new}} \leftarrow 0$
      $n \leftarrow 0$
  `// fill the windows`
  if $size(W_{ref}) < |W_{ref}|$ **then**
    $W_{ref} \leftarrow W_{ref} \cup x$
  else if $size(W_{cur}) < |W_{cur}|$ **then**
    $W_{cur} \leftarrow W_{cur} \cup x$
  else
    `// check if there is a change`
    $G \leftarrow 0$
    for $i \leftarrow 1$ *to* d **do**
      `// compute` $G_i(Map)$ `(eq. 3) for all explanatory variables`
      $G \leftarrow G + G_i(Map)$
    if $G > 0$ **then**
      `// drift detected: training of a new classifier`
      `start` (M_{new})
    else
      `// data stream is stationary: remove the new classifier`
      `delete` $(M_{current})$
    `// jumping window: empty it so that it will be refilled`
    $W_{cur} \leftarrow \emptyset$
  `train` $(M_{current}, x)$

Algorithm 1: MDD (MODL Drift Detection) algorithm integrating the detection method to replace (or not) the current classifier when a concept drift is detected.

"MODL $P(W|X_i, C)$". This replacement is effective only when the error rate of the new classifier (M_{new}) is lower than the error rate of the current classifier ($M_{current}$). In our experiments the error rate is computed with an exponential moving average with the parameter α set to $\alpha = 1/|W_{cur}|$. The parameter n_{min} corresponds the minimum numbers of examples to get before comparing the two classifiers accuracy. The value $n_{min} = 30$ is the same as the one used by DDM and EDDM before they try to find a concept drift and is related to the approximation of the considered distribution by the normal distribution. Contrary to DDM or EDDM (see Section II-B) our algorithm MDD is not based on the performance of the classifier, thus it is not "classifier dependent". The `train()` method in this algorithm corresponds to the train method of the classifier chosen by the user. For example, below in the experiments we use an Hoeffding tree or a Naive Bayes classifier. This algorithm has a low memory footprint as it only needs to keep the data stored in the reference and current windows ($((|W_{ref}| + |W_{cur}|)\, d)$).

B. Experimental protocol

In our MDD algorithm, the size of the two windows W_{ref} and W_{cur} is set to 1,000 samples. Two different types of classifier were used:

- a naive Bayes classifier (NB) using an estimation of $P(X|C)$ taken from a two layers incremental discretization method based on order statistics [16]. The first layer is a summary per variable which contains an estimation of 100 quantiles. The second layer is the MODL discretization for the numerical variables or the MODL grouping for the categorical variables.
- a Hoeffding Tree [17] (HT) configured with a summary per variable based on 10 quantiles. This summary is used to split the leaf into decision nodes and to provide estimations of $P(X_i|C)$, using MODL method, for the naive Bayes local model in each leaf.

The results presented below are those obtained on:

- the "Rotating Hyperplane" problem suggested in [14]: the data stream contains ten explanatory variables, a change speed of 10^{-3} and 10% of noise on the class labels.
- the Random RBF problem: the data stream contains 50 centers, ten explanatory variables and a change speed of 10^{-3} (as suggested in [1]).

Among the methods available in the literature to evaluate the accuracy of a classifier on a labeled data stream [18], we chose to use the method based on the prequential error which interleaves the train and the test (Test-Then-Train). The error plotted in the Figure 4 gives the mean prequential error of the classifier between the beginning of the data stream and the current instant t. The MDD algorithm has been implemented in the MOA framework [19] and has a dependence to our in-house Khiops[3] software to compute the MODL grid.

C. Results

The results for the "Rotating Hyperplane" presented in the Figure 4 show that all the 3 methods (DDM, EDDM and MDD) on both classifiers, NB or HT, have the same performances until respectively 100,000 (NB) and 200,000 (HT) training examples. Then

- for the Naive Bayes classifier:
  - between 100,000 and 200,000: (i) DDM is better (ii) the performance of MDD is stable (iii) EDDM has a decreasing performance;
  - between 200,000 and 680,000: (i) DDM has a decreasing performance (ii) the performance of MDD and EDDM are stable;
  - between 680,000 and 1,000,000: (i) MDD and EDDM have a small improvement in their performance (ii) EDDM has an increasing performance but its accuracy is still lower from 2% to 4% than DDM.

[3] www.khiops.com

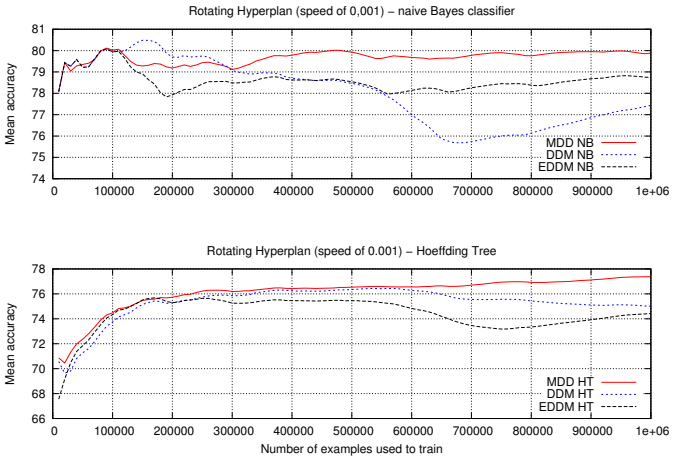

Fig. 4. Mean accuracy of the three tested methods on the "Rotating Hyperplane" data set.

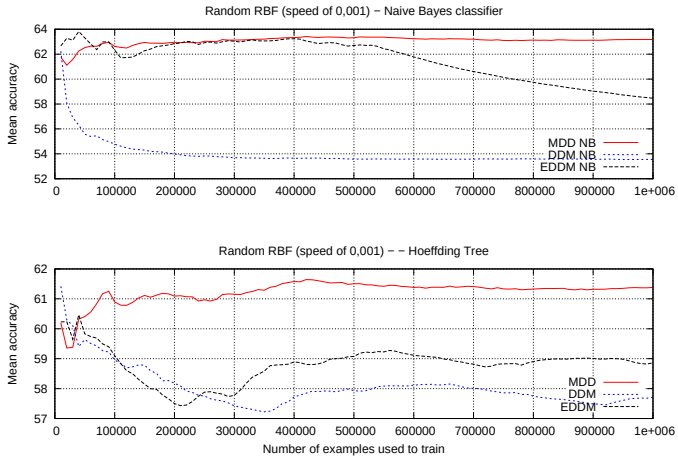

Fig. 5. Mean accuracy of the three tested methods on the "Random RBF" data set.

- for the Hoeffding Tree classifier:
  - between 200,000 and 600,000: the three methods are close with a small advantage to MDD;
  - between 600,000 and 1,000,000: (i) DDM and EDDM have globally a decreasing performance (ii) the performance of MDD has a small improvement in its performance.

The results for the "Random RBF" presented in the Figure 5 show that all the 3 methods (DDM, EDDM and MDD) on both classifiers, NB or HT, have quite different performances:

- for the Naive Bayes classifier:
  - between 0 and 400,000: (i) the performance of EDDM and MDD are stable (iii) DDM has a decreasing performance;
  - between 400,000 and 1,000,000: (i) the performance of DDM and MDD are stable, but the performances of DDM is low (iii) EDDM has a decreasing performance;
- for the Hoeffding Tree classifier:
  - between 0 and 200,000: (i) EDDM and DDM have a decreasing performance (ii) MDD has an increasing performance;
  - between 200,000 and 350,000: (i) DDM has a decreasing performance (ii) EDDM has a increasing performance (iii) MDD is stable;
  - between 350,000 and 1,000,000: DDM, EDDM and MDD are stable.

These figures shows that the proposed algorithm exhibits stable and better performances.

To understand and explain where the differences between the three methods come from, the Figure 6 presents the detection (red curves) versus the time t on the top for MDD, on the middle for DDM and on the bottom for EDDM. For DDM and EDDM which use an "alert" mechanism before the detection represented by the blue curves. This figure helps to understand that DDM and EDDM do not have regular detection, even though the hyperplane has a constant rotation speed. These two methods seems to often be "in alert" but the threshold to be "in detection" is not reached. The assumptions that data are i.i.d seem to be not valid here and/or the threshold settings for the alert and detection difficult to tune. This conduces to a lack of robustness. Our method, which has no threshold to tune and no alert mechanism before the detection and do not use directly the classifier, performs better on these experiments. The subfigure on the top (MDD) shows regular detections while the hyperplane rotates.

V. Conclusion

In this paper we have presented an on-line method for concept change detection on labeled data streams. Our detection method uses a bivariate supervised criterion to determine if the data in two windows come from the same distribution. Our method has no assumption neither on the data distribution nor on the change type and has the ability to detect changes of different kinds (mean, variance...) and velocity.

Our method is able to detect a change (i) in the joint distribution of an explanatory variable and the class variable; (ii) in the distribution of an explanatory variable and (iii) in the distribution of the class variable. Thus all the changes in the conditional distribution of the class variable knowing an explanatory variable can be detected. This method has no threshold to tune, no alert mechanism before a detection and do not use directly the classifier.

Experiments show that our method has better results than well-known methods from the literature and exhibits a good robustness and reactivity. Besides, except from the window sizes, no user parameter is required in our method.

On-line learning, which processes instances one-by-one and builds models incrementally, has the virtue of being fast, both in the processing of data and in the adaptation of models. Off-line (or batch) learning has the advantage of allowing the use of more sophisticated mining techniques, which might be more time-consuming or require a human expert. While the first allows the processing of "*fast* data" that requires real-time processing and adaptivity, the second allows processing of "*big* data" with longer processing time and potentially more complex and accurate models. Their combination can take place in many steps of the mining process, such as the data preparation

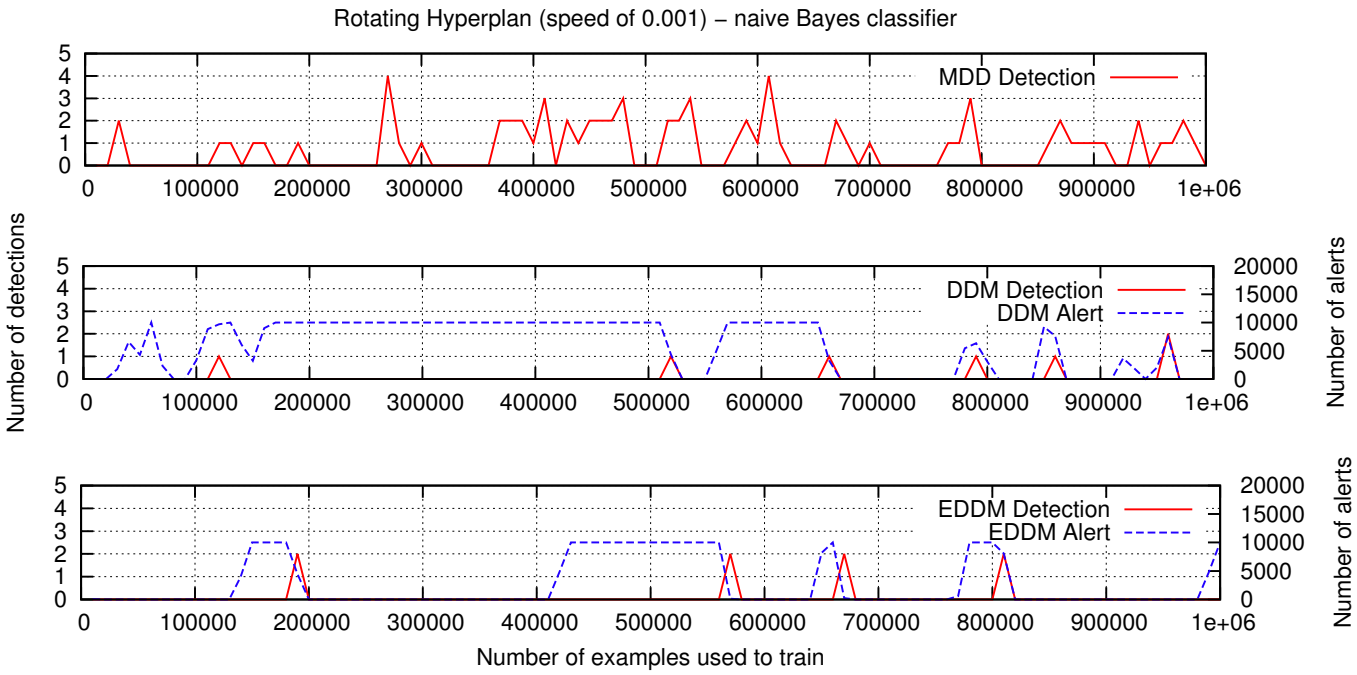

Fig. 6. Number of detections (red curves) and alerts (blue curves) versus the number of examples used to train for the different tested methods.

and the preprocessing steps. For example, off-line learning on big data could extract fundamental and sustainable trends from data using batch processing and massive parallelism. On-line learning could then take real-time decisions from on-line events to optimize an immediate pay-off.

In future work, we plan to investigate on combining off-line learning, drift detection and on-line learning, to solve problems such as on-line advertising or network attack detection.

REFERENCES

[1] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà, "New ensemble methods for evolving data streams," *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '09*, p. 139, 2009.

[2] I. Žliobaite, "Learning under Concept Drift: an Overview," https://sites.google.com/site/zliobaite/Zliobaite_CDoverview.pdf, Tech. Rep., Oct. 2010. [Online]. Available: https://sites.google.com/site/zliobaite/Zliobaite_CDoverview.pdf

[3] A. Bondu and M. Boullé, "A supervised approach for change detection in data streams," *International Joint Conference on Neural Networks (IJCNN)*, 2011.

[4] M. Boullé, "MODL: A Bayes optimal discretization method for continuous attributes," *Machine Learning*, vol. 65, no. 1, pp. 131–165, 2006.

[5] P. Matuszyk, G. Krempl, and M. Spiliopoulou, "Correcting the usage of the hoeffding inequality in stream mining," in *In Proceedings of the Twelfth International Symposium on Intelligent Data Analysis*, 2013.

[6] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," *Advances in Artificial Intelligence - SBIA 2004*, pp. 286–295, 2004.

[7] M. Baena-García, J. Del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavaldà, and R. Morales-Bueno, "Early Drift Detection Method," *Fourth International Workshop on Knowledge Discovery from Data Streams*, vol. 6, pp. 77–86, 2006.

[8] M. Boullé, "Optimum simultaneous discretization with data grid models in supervised classification: a Bayesian model selection approach," *Advances in Data Analysis and Classification*, 2009.

[9] V. Lemaire, "Real world issues in supervised classification for data stream," slides of a talk given at ECML 2013 - workshop RealStream, September 2013, http://perso.rd.francetelecom.fr/lemaire/publis/ECML-Prague-vf-2013.pdf.

[10] C. L. Blake and C. J. Merz, "UCI Repository of machine learning databases," p. http://archive.ics.uci.edu/ml/, 1998. [Online]. Available: http://www.ics.uci.edu/˜mlearn/MLRepository.html

[11] M. Boullé, "Bivariate Data Grid Models for Supervised Learning," France Telecom R&D, No NSM/R&D/TECH/EASY/TSI/4/MB, Tech. Rep., 2008.

[12] P. Langley, W. Iba, and K. Thompson, "An analysis of Bayesian classifiers," in *Proceedings of the National Conference on Artificial Intelligence*, no. 415, 1992, pp. 223–228.

[13] D. J. Hand and K. Yu, "Idiot's Bayes?Not So Stupid After All?" *International Statistical Review*, vol. 69, no. 3, pp. 385–398, Dec. 2001.

[14] G. Hulten, L. Spencer and P. Domingos, "Mining time-changing data streams" in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 97–106 2001.

[15] J. Gama, *Knowledge Discovery from Data Streams*. Chapman and Hall/CRC Press, 2010.

[16] Anonymous, "Anonymous," *Advances in Data Analysis and Classification*, 2012.

[17] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. pp. 71–80, ACM New York, NY, USA, 2000.

[18] J. Gama, P.P. Rodrigues, R. Sebastiao and P.P. Rodrigues, "Issues in evaluation of stream learning algorithms" in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp 329–338, ACM New York, NY, USA, 2009

[19] A Bifet, G Holmes, R Kirkby, B Pfahringer, "Moa: Massive online analysis" in The Journal of Machine Learning Research, pp 1601–1604, Volume 11, 2010