Classification with a reject option under Concept Drift: the Droplets Algorithm

Pierre-Xavier Loeffel*, Christophe Marsala* and Marcin Detyniecki*†
*Sorbonne Universités, UPMC Univ Paris 06, CNRS, LIP6 UMR 7606, 4 place Jussieu 75005 Paris.
†Polish Academy of Sciences, IBS PAN, Warsaw, Poland.
{pierre-xavier.loeffel, christophe.marsala, marcin.detyniecki}@lip6.fr

***Abstract*—In this paper a new on-line algorithm is proposed (the *Droplets* algorithm) for dealing with concept drifts and to produce reliable predictions. The two main characteristics of this algorithm are that it is able to adapt to different types of drifts without making any assumptions regarding their type or when they occur, and can provide reliable predictions in a non-stationary environment without using a fixed confidence threshold. Experimental results on five datasets based on Random RBF and Rotating Hyperplane generators as well as a new semi-synthetic dataset based weather temperatures show that, by discarding difficult observations, the *Droplets* algorithm manages to obtain the best average accuracy against ten classifiers. The results also indicate that the algorithm manages to provide reliable prediction by accurately distinguishing which observations are easily classifiable.**

I. Introduction

State of the art machine learning algorithms obtain very good results when learning relationships linking the observations to their classes in stationary environments. However, many temporal processes evolve or change unpredictably in the real world, and thus, most of the real life problems require algorithms able to learn in a non-stationary environment. This is the goal of algorithms dealing with concept drift (when the relationship that links the observations to their classes change over time), which have managed to grab the attention of the community recently.

Furthermore, there is a need for algorithms producing reliable predictions at any time. This is achieved by devising predictors that can choose to abstain from prediction on a test example, decreasing the number of examples classified in order to achieve a better accuracy. Those classifiers can then be used to determine which observations are easy to classify and can find real life applications in fields where mistakes are costly like medical diagnosis for instance.

In this paper, we introduce a new on-line algorithm that can choose to abstain from predicting and which is able to adapt to non-stationary environments. This algorithm can deal with a broad range of drifts, without making any assumption regarding the type of drifts or when they occur. To our knowledge, this algorithm is the first "concept-drift handling algorithm" explicitly designed to abstaining from prediction.

The paper is organized as follows: Section 2 describes the classification problem in an online and non stationary environment. The field of concept drift is introduced along with some methods developed to deal with it. We also briefly introduce the topic of reliable prediction. Section 3 describes the proposed solution to the problem of providing reliable predictions under concept drift. Section 4 presents the experimental protocol and the characteristics of the datasets used to assess our algorithm. Section 5 presents the results of the experiments and the discussion. Finally Section 6 concludes.

II. Background and Related Work

A. Framework

The problem being addressed here is the one of classification, under supervised learning for an online algorithm. The supervised classification task aims at predicting the class y of a target variable given a set of input features $X \in \mathbb{R}^p$.

According to the Bayesian Decision Theory [21] the decision of a classifier is made according to

$$p(y|X) = \frac{p(X,y)}{p(X)} = \frac{p(y)p(X|y)}{p(X)} \tag{1}$$

with $p(y|X)$ the posterior probability of the classes given an observed set of features X, $p(y)$ the prior probabilities of the classes, $p(X)$ the distribution of the features, $p(X|y)$ the class conditional distributions of the features and $p(y,X)$ the joint distribution between the input features and the classes.

In online learning framework, a new set of features X_t is received at time *t* and the classifier uses it to output a prediction according to the previously learned model:

$$\hat{y}_t = \hat{h}_{t-1}\left(X_t\right) \tag{2}$$

The true label y_t is then received and the loss is computed. If necessary, the model is updated according to the latest observation and the previous model:

$$\hat{h}_t = train\left[\left(X_t, y_t\right), \hat{h}_{t-1}\right] \tag{3}$$

The additional challenge is that, because the learner has to deal with dynamically changing environments, the relationship between the observations and the target classes that the classifier is trying to learn can change over time. This is the focus of algorithms handling concept drift.

B. Concept drift

According to the literature [1][22], a formal definition of concept drift is: concept drift occurs between time t_0 and time t_1 if:

$$\exists X \in \mathbb{R}^p : p_{t_0}(X, y) \neq p_{t_1}(X, y) \tag{4}$$

with p the joint distribution between the features X and the class y.

According to (1)

$$p(X, y) = p(X)p(y|X) = p(y)p(X|y) \tag{5}$$

Hence a change in $p(X, y)$ is characterized by: either a change in one of the components of the right hand part of (5), either a change of both components at the same time. The drifts can be further categorized as abrupt, incremental, gradual or reoccurring.

C. Existing Methods to handle Concept Drift

We now review some of the main approaches proposed in the literature to deal with concept drift.

DWM (Dynamic Weighted Majority) [7] is an ensemble method able to dynamically add or remove on-line learners, weighted according to their past performances. The predictions are given by a weighted-majority vote of the learners.

Learn++.NSE [18] is an incremental ensemble classifier based on the same ensemble structure as DWM. Its novelties lie in the mechanism used to weight the vote of the classifiers. Here the weights are dynamically updated according to the classifier's current and past error and in its passive drift detection mechanism.

Accuracy Updated Ensemble [17] also has a similar structure as DWM but processes the observations in chunk. One novelty is that after each data chunk it replaces the worst performing classifier by a new one, trained on the latest data chunk, no matter how good the worst performing one is. Another novelty is that the weights of each classifiers are now updated according to their prediction error and the mean square error of a randomly predicting classifier on each data chunk instead of simply decreasing their weight according to a pre-set parameter. Finally, it is equipped with a memory surveillance module that prune the Trees when the memory limit is reached.

DACC (Dynamic Adaptation to Concept Changes) [20] has been proposed as an ensemble algorithm able to adapt to concept changes. It is also based on the same principles as DWM in the sense that it also maintains a set of on-line learners, weighted according to their performance. One improvement is the deletion strategy which doesn't delete automatically the worst performers of the ensemble but select them randomly from the worst performing half of the committee. Another improvement is the final vote which instead of using a simple weighted vote, rely either on a weighted vote of the learners with weights belonging to the upper half of the committee either on the prediction of the best performer of the ensemble.

On-line Bagging [9] is an extension of the well known Bagging algorithm [10]. The authors noticed that when the size of the training set tends to infinity, the number of occurrence of each example in the training set tends to a *Poisson*(1) distribution. Thus, in an on-line framework, for every new example each base model is trained k times with $k \sim Poisson(1)$. The prediction is performed in the same way, by taking an unweighted vote of the base learners.

ADWIN Bagging [12] adds ADWIN (ADaptive sliding WINdow) [11] in order to detect changes and to estimate the weights in the On-line Bagging [9] algorithm.

Leveraging Bagging [8] has been proposed as an improvement of On-line Bagging [9]. The authors managed to improve the On-line Bagging algorithm by randomizing even more the classification process. Instead of using $\lambda = 1$ in the Poisson(λ) distribution to generate the weight of each new example, they use a larger value of λ. Their claim is that, by doing so, they increase the diversity of the weight given to every new example and thus modify the set on which the base classifiers will be trained. The second improvement comes from the use of output codes in order to add randomization at the output of the ensemble. According to them, the main benefits of this process are that it can reduce the effects of correlations between classifiers and further increase the diversity of the ensemble. Finally, they use ADWIN [11] to detect changes in error of the classifiers and replace the worst performing by new ones.

On-line Boosting [9] is designed to extend the AdaBoost.M1 algorithm [13] to the on-line framework. It is similar to On-line Bagging [9] except that it increases (respectively decreases) the parameter λ associated with the Poisson distribution for the next base learner if the latest example has been misclassified (respectively correctly classified).

Hoeffding Adaptive Trees [14] is an extension of the Hoeffding trees [15]. Hoeffding trees was initially developed in order to cope with a potentially infinite stream of data under the assumption that the distribution of the data wouldn't change. Hoeffding trees has been implemented in VFDT (Very Fast Decision Trees) with the addition of some heuristics. CVFDT (Concept-adapting Very Fast Decision Trees) [16] was developed as an extension of VFDT able to deal with change of concept. Finally, Hoeffding Adaptive Trees were developed to further improve CVFDT by getting rid of the parameter governing the size of the sliding window of instances. Instead of that, it uses ADWIN as a change detector to remove and replace the branches of the tree with poor accuracy.

D. The problem of Reliable Predictions under Concept Drift

The previous review of algorithms dealing with concept drift shows that none of them abstain from predicting after reception of an unlabeled observation, regardless of how confident they are in their prediction. Some algorithms (like Boosting [19]) associate a degree of confidence with their predictions, leaving to the end user the task of setting a meaningful threshold above which he is willing to trust the algorithm. However, setting a relevant threshold for a particular task might not always be easy and the use of a fixed value can be questioned when dealing with non-stationary environments which often require flexibility in the value of the parameters. Furthermore, most of the algorithms producing confidence estimate of their prediction rely on the assumption that the distribution linking the observations to their class is stationary. This assumption doesn't hold when concept drift occur and hence the validity of the confidence estimate produce can be questioned.

The idea of providing reliable predictions has been abundantly studied in the literature, under various names such as selective classification [2][19][25] or conformal prediction [3]. Usually, the performance of these classifiers is evaluated by assessing the trade-off between accuracy against coverage (the fraction of observations on which the classifier gave a prediction). Their goal, then, is to maximize the accuracy while achieving the highest possible coverage for a given confidence threshold.

In the following section, a solution is proposed in order to fix some of the shortfalls of the algorithms described above. The developed online classifier can be defined by 3 characteristics: (i) it doesn't make any assumption (regarding the type of drift, the distribution of the data, or when the drifts occur), (ii) it isn't forced to classify every new observation when "it doesn't know" and (iii) it is very risk-averse (in the sense that it refrains from generalizing too much).

III. The Droplets algorithm

To illustrate the Droplets algorithm, every new observation can be thought as a droplet falling on a plan (the plan would be the feature space in this case). We refer to this plan as the "Droplets' map" from now on. The "chemical" composition of the droplets is defined by a class and the Droplets are mutually repulsive, so 2 droplets that do not belong to the same class can't cover the same part of the Droplets' map. The algorithm is given in Figure 4.

The goal of the initializing set is twofold: (i) initialize the map and (ii) estimate the range of values that each feature can take. Once the minimum and maximum values of each feature on the initializing set have been determined, a vector $\overrightarrow{D}$ is computed and used to normalize all the observations so the initializing set becomes an hypercube.

The map is updated as follows: At first it is empty. A labeled observation from the initializing set is received and the observed values for its features are normalized (according to the normalization vector $\overrightarrow{D}$ found beforehand). The normalized observation is then inputed in the map as an hypersphere with a radius of size $R_{default}$ and with coordinates the normalized feature's values. The class of the observation is then attributed to this hyper-sphere. The next labeled observation from the initializing set is then released and inputed in the map. If the hypersphere of latest observation overlaps one or more existing hyperspheres, the classes of the overlapped hyperspheres are assessed. For all the hyperspheres belonging to the class of the latest observation, nothing happens, for each of the other hyperspheres, the overlap

$$\lambda_k = R_k + R_i - \|Center_k - Center_i\| \tag{6}$$

is computed, with $\|.\|$ the Euclidean distance. The hyperspheres are then ranked from smallest to largest overlap. The details of the ranking process are given in Figure 1.

```
Inputs: $map_i$,observation i
$J \leftarrow Number\ of\ hyperspheres\ in\ map_i$
$indexes \leftarrow \emptyset$
If $J \geq 2$ Then Foreach: $k \leftarrow 1, ..., J-1$ //Observation i is the $J^{th}$ one
  $\lambda_k \leftarrow R_k + R_i$-$\|Center_k - Center_i\|$
  If $\lambda_k > 0$ And $y_k \neq y_i$ Then
    $indexes \leftarrow indexes \cup obs_k$
  End If
End If
$indexes \leftarrow order(indexes; smallest\ to\ largest; criteria = \lambda)$
$indexes \leftarrow indexes \cup obs_i$
Return $indexes$
```

Figure 1. Ranking of overlapping hyperspheres belonging to different class

```
Inputs: $indexes$, $map_i$
$L \leftarrow Number\ of\ values\ in\ indexes$
If $L \geq 2$ Then Foreach:$k \leftarrow 1, ..., L-1$
  If $R_k = 0$ Or $R_i = 0$ Then
    next k
  End If
  $\Delta_k \leftarrow \frac{R_k + R_i - \|Center_k - Center_i\|}{2} + \varepsilon$
  Foreach:$m \leftarrow k, ..., L$
    $R_m \leftarrow max(R_m - \Delta_k; 0)$
End If
$map_i \leftarrow map_i - delete(obs; R = 0)$
Return $map_i$
```

Figure 2. Update of the radiuses sizes

For the first hyper-sphere belonging to the previously created set ($indexes$), the value

$$\Delta_1 = \frac{R_1 + R_i - \|Center_1 - Center_i\|}{2} + \varepsilon = \frac{\lambda_1}{2} + \varepsilon \tag{7}$$

used to remove the overlap between this hyper-sphere and the newest hyper-sphere is computed. This value includes an arbitrary small $\varepsilon > 0$, added to prevent the hyperspheres from being tangent. Δ_1 is then subtracted from the radius size of all the hyperspheres that have an overlap greater or equal to λ_1. A mechanism is added to prevent the size of the radiuses to drop bellow 0. The value Δ_2 is then computed for the hyper-sphere with the second smallest overlap and so on. If an hyper-sphere has a radius equal to 0, it is removed from the map. Figure 2 explains the process of updating the size of the radiuses.

The previous updating process is then repeated again with each new observation.

Once the initializing set is over, the first unlabeled observation is received. In order to give a prediction, the algorithm will simply check if the normalized coordinates of the center of the new hyper-sphere belong to an existing hypersphere. If it is the case the algorithm will predict that they both have the same class, otherwise, the algorithm won't predict and wait for the true label to be released. Note that the update of the map doesn't start until the true label is released. Figure 3 gives the details of the prediction process.

```
$J \leftarrow Number\ of\ hyperspheres\ in\ map_i$
$\hat{y_i} \leftarrow \emptyset$
**If** $J \geq 2$ **Then Foreach**:$k \leftarrow 1, ..., J-1$ // Observation i is the $J^{th}$ one
  **If** $\|Center_k - Center_i\| \leq R_k$ **Then**
    $\hat{y_i} \leftarrow Class_k$
    **Break**
  **End If**
**End If**
**Return** $\hat{y_i}$
```

Figure 3. Prediction

```
**Inputs**: $R_{default}$, N labeled observations
Divide the dataset in initializing/test set
$\overrightarrow{D} \leftarrow Normalize\ set\,(initializing\ set)$
$map_0 \leftarrow \emptyset$
**Foreach**: $i \leftarrow 1, ..., N$
  $Observation_i \leftarrow Normalize\ obs\left(Observation_i, \overrightarrow{D}\right)$
  $map_i \leftarrow Add\ to\ map\,(map_{i-1}, Observation_i)$
  **// Prediction**
  **If** $i \in Test\ Set$
    $\hat{y_i} \leftarrow Predict\,(map_i, Observation_i)$ // Figure 3
  **End If**
  **// True label released, perform map update**
  $indexes \leftarrow Rank\,(map_i, Observation_i)$ // Figure 1
  $map_i \leftarrow Update\ radiuses\,(map_i, indexes)$ // Figure 2
**End**
```

Figure 4. The Droplets algorithm

A. Main characteristics of the Droplets Algorithm

In this section, the two main characteristics of our algorithm are further explained.

1) Implicit handling of the stability-plasticity dilemma: One of the biggest challenge when trying to handle concept drift is to give sufficient flexibility to an algorithm so it can adapt to its new environment and yet at the same time make it robust to noise or outliers. Our algorithm provides an implicit forgetting mechanism that can cope efficiently with these two situations.

For instance, if a noisy observation (belonging to class B) appears in a part of the map that is fully covered by observations of class A, its influence (characterized by the size of its radius) will be naturally limited by the surrounding observations of class A that it overlaps. The noisy observation will then vanish as new observations belonging to class A appear next to it and push its influence down to 0. On the other hand, if a drift occurs and more observations belonging to class B keep reappearing in this part of the map, it will have the effect of decreasing the influence of the former observations belonging to class A (some parts of the map that were previously fully covered will be left empty, reflecting the new uncertainty in those areas) up to the point where they are replaced by new observations belonging to class B.

2) Cautious stance leading to reliable predictions: By limiting the influence of each observations, the algorithm is prevented from making clue-less assumptions. The underlying idea is that if an observation appears in a part of the map that do not belong to an hypersphere, it is either because this part has not been explored yet and hence the algorithm doesn't want to make assumptions, either because, the influence of the nearby hyperspheres has been diminished as a result of conflicting informations and hence reflects the fact that this is a zone of uncertainty in which the algorithm adopts a cautious stance.

As a consequence, the algorithm is able to distinguish which observations are potentially hard to classify and can abstain from predicting in those cases, further increasing the faith that the end user can put in the predictions.

B. Identified Drawbacks

We now review the identified drawbacks.

1) Behavior in high dimensions: Like the KNN, the algorithm could suffer in high dimensions due to the increased volume of the map that needs to be filled by the hyperspheres. In this case, one solution could be to feed the algorithm with more data. Another solution has been proposed by T. Hastie and R. Tibshirani [23] in the case of the KNN which could be adapted to our algorithm. A third solution could be to use algorithms which would shrink the dimension of the feature space.

2) Percentage of unclassified observations: The algorithm can also potentially find itself in situations where it creates a high percentage of unclassified observations. This is a deliberate choice and comes as the price for reliable predictions (which can find useful applications in real-life, when wrong predictions are costly). The results of the experimental section show that the accuracy of the adaptive classifiers is lower on the unclassified observations, leading to the conclusion that the risk of being wrong increased on those particular observations.

3) Initial Normalization step: The normalization step that take place at the very beginning of the process can be troublesome in the cases where the drift imply a change of scale between the initializing and test set. We haven't yet studied this type of drift which has been left for futures researches.

IV. Experimental Framework

In this section, we describe the datasets on which experiments have been conducted, the drifts they include and the experimental protocol.

A. Datasets

We decided to include both semi-artificial and artificial datasets and tried to cover different types of drifts (abrupt, incremental and reoccurring).

1) Common characteristics of all the datasets: In order to assess the predictive performances of the classifiers at every single time step, we used the experimental protocol presented in [24]: For each dataset, a set of 1 000 temporally indexed observations is generated. Then, for each $t_i \in \{t_1, ..., t_{1000}\}$, 500 observations were randomly generated $t_i^j \in \{t_i^1, ..., t_i^{500}\}$ according to the concept in force at that time. The classifiers are then trained on $\{t_1, ..., t_i\}$ and their accuracy is assessed on $\{t_{i+1}^1, ..., t_{i+1}^{500}\}$. The initializing set is composed of the first 100 observations $\{t_1, ..., t_{100}\}$ and the test set composed of the remaining observations $\{t_{101}, ..., t_{1000}\}$ and each $t_i^j \in \{t_i^1, ..., t_i^{500}\}$ with $i \in [101, 1000]$.

The advantage of this evaluation process is that it gives a much more reliable idea of the accuracy of the classifiers at a given time than what would have been obtained with only one observation. The drawback is that it forces the use of datasets where the user can generate the labeled observations for each time step.

In order to have full control over the generating process of the observations, we used our own implementation of all the datasets described bellow. It should also be mentioned that the datasets were generated in small dimensions ($d \leq 3$) apart from RBF_3. The goal here was to show the adaptive capabilities as well as the reliability of the predictions given by the Droplets.

2) Random RBF: This dataset was initially introduced in [5]. The idea is to generate a fixed number of centroids in d-dimensions where each center will be characterized by random coordinates, a random standard deviation, a particular class label and a random prior probability of being selected. The new observations are then randomly generated according to the previous parameters. Drifts are introduced by offsetting the coordinates of the centers in a random direction according to a Gaussian distribution with zero mean and a given standard deviation.

We used 3 Random RBF datasets, two for incremental drifts: RBF_2 with 6 classes, RBF_3 with 30 classes and 12 dimensions and one for abrupt drifts: RBF_1 with 3 classes. In the case of incremental drifts, the centroids are offset at each time step. In the case of abrupt drifts, there are 4 different concepts evenly distributed on the dataset (the drifts happen at $t = \{250, 500, 750\}$). In each cases the data were generated without noise and constrained in a $[0, 10]^2$ square (or hypercube for RBF_3).

3) Rotating Hyperplane: This synthetic dataset was initially introduced in order to assess the performance of CVFDT against VFDR in [4]. The idea is to generate the data uniformly in a d-dimensional hyperplane according to the following decision boundary

$$\sum_{i=1}^{d} w_i x_i = w_0 = \sum_{i=1}^{d} w_i \tag{8}$$

with x_i the i^{th} feature and w_i the weight of this feature. If the left hand part of equation (8) is superior or equal to w_0 the label of the observation will be positive and negative otherwise. Concept drifts are introduced by changing the weights. d was set to 2 and some noise has been added to the dataset by randomly switching the class of each observation with a 15% probability. The abrupt drifts happen at times $t = \{166, 332, 498, 664, 830\}$.

4) Weather Temperatures: This dataset is composed of one feature and 3 classes, created out of the historical temperatures of the city of Paris[1] from the 17th of January 2006 to the 15th of December 2014. 3 temperatures are available: the highest, lowest and average temperature of each day. After removing the outliers from the data, the average of the highest and lowest temperatures for each day of the year have been computed (the average of the highs and lows of every 1st of January, 2nd of January and so on ...). This gives 2 bands that are used as decision boundaries. If the observed temperatures is above the average of the highs it will be given the class “warm”. The temperatures below the average of the lows will be labeled as “cold” and the ones in between “medium”.

The underlying idea is to reproduce incremental, reoccurring drifts. In order to keep the size of the dataset consistent with the other datasets, only the first 1000 observations starting from the 17th of January 2006 were kept. For each day, 501 observations were generated by uniformly drawing temperatures between the maximum and the minimum observed that day.

B. Implementation of the algorithms

MOA[2] and its extension[3] have been used to conduct the experiments and provide the implementation of the classifiers. All the parameters of the classifiers were set to default. This is required because there is no assumptions regarding the structure of the data or the type of drifts the classifiers will have to deal with. Therefore, it wouldn’t be relevant to optimize parameters that would be suitable for a particular concept, at a particular time and for a particular dataset. In the case of the Droplets algorithm, the default radius was set to 0.1 for all the experiments. We also added the KNN algorithm with a fixed parameter of K=1 as a baseline.

V. Results and discussion

Experiments have been performed, underpinning our 2 statements: (i) the Droplets algorithm manages to perform well compared to other adaptive learners, regardless of the type of drift and (ii) the algorithm provides reliable predictions by discarding potentially difficult observations without relying on a fixed confidence threshold.

A. High accuracy, regardless of the drift type

In order to assess the performances of the Droplets against other algorithms, the results have been decomposed in two tables: the performances of the algorithms on the observations where the Droplets gave a predictions and their performances on all the observations where the Droplets didn’t predict. The underlying idea is to avoid producing misleading results with a table reporting the overall accuracy on the test set where the Droplets could have achieved a very high performance at the cost of a very high proportion of unclassified observations

[1]The data can be downloaded at the following address: http://www.wunderground.com/personal-weather-station/dashboard?ID=I75003PA1.

[2]http://moa.cms.waikato.ac.nz/

[3]https://sites.google.com/site/moaextensions/

(e.g. a performance of 100% by classifying correctly a single observation).

Table I. shows the average accuracy (in percentage) of all the classifiers on the observations where the Droplets algorithm gave a prediction.

The last line of the table indicates the percentage of unclassified observations by the droplets on each dataset. Bold numbers indicate which classifier didn't manage to over-perform a simple 1-NN by at least 5%.

The Droplets algorithm managed to get the top spot twice out of 5 datasets. The average accuracy and the average rank indicate that overall, the algorithm has a steady performance compared to the other classifiers, regardless of the type of drift. It should also be mentioned that on RBF_1, 4 classifiers didn't manage to outperform a simple 1-NN by at least 5% whereas on RBF_3, apart from the Droplets algorithm, none of them managed to equal the performance of the 1-NN.

The performance achieved on RBF_3 is an extreme case where the algorithm managed to get the perfect score at the cost of 98.15% of unclassified observations. The perfect score was achieved because of the high dimensions: the incrementally moving centroids never overlapped each other, which never gave the opportunity to output a wrong prediction. On the other hand, the very high percentage of unclassified observations comes from a perpetual discovery of unexplored parts of the map.

In order to assess the stability of the Droplets over different scenarios, we reruned the algorithm 45 times on each dataset. For each time step i, we randomly switched one of the 500 observation t_i^j with t_i. This has the effect of completely modifying the set of observations the algorithm uses to learn. Table II. gives an overview of the mean and variance achieved for the accuracy and the percentage of unclassified observations. The results indicate that the Droplets algorithm is very stable in its performances.

B. Reliable predictions under concept drift

In this section, we experimentally show that the Droplets algorithm overcomes the issue of producing reliable predictions in a non-stationary environment, without an explicitly fixed confidence threshold.

1) The radius doesn't act as a confidence threshold: In the selective classification framework, there is generally a trade-off between risk and coverage [2] and the performance of this type of classifier is usually assessed with the curve connecting these 2 quantities. Moreover, algorithms based on conformal prediction or confidence rated prediction tend to have a constant parameter managing the desired confidence threshold. In this experiment we show that the only parameter of the Droplets doesn't act as a confidence threshold. Figure 5 shows the evolution of the accuracy of the Droplets against the percentage of unclassified observations for different values of the default radius size (from 0.02 to 0.38 with a step of 0.04).

Figure 5 shows that the value of the parameter yielding the best accuracy is also the value minimizing the percentage of unclassified observations, regardless of the dataset and of the type of drifts encountered. It also shows there isn't a trade-off between accuracy and coverage because the accuracy of the Droplets increases with the coverage. This lead to draw the conclusion that the radius doesn't act as a confidence threshold which was our initial requirement as we stated that a fixed confidence threshold wouldn't be relevant in an evolving environment.

The default radius sizes that gave the best results (i.e. the highest accuracy and the lowest rate of unclassified observations) are respectively 0.22, 0.1, 0.18 and 0.1 for RBF1, RBF2, RH and Temperatures

2) Ability to produce reliable predictions by discarding potentially difficult observations: The results in table III. show that the Droplets managed to successfully avoid the difficult observations to produce reliable answers. Indeed, except in the case of the algorithms that were already performing poorly on the observations where the Droplets didn't predict (AUE on RBF_1, RBF_3 and Learn++.NSE on RBF_3), the performance of all learners decreased on the unclassified observations. Table III. also indicates that a majority of classifiers didn't manage to outperform by at least 5% a simple 1-NN on RBF_3 and Temperatures datasets.

Taken together, the last 2 experiments show that the Droplets managed to produce a result similar to the results that would have been achieved with algorithms producing reliable answers in the stationary case, without having to rely on a fixed confidence threshold. To our knowledge, this is the first time an algorithm is explicitly designed to produce reliable answers while dealing with drifts.

The right hand part of figure 6 shows the evolution of the accuracy of the top 4 classifiers. Despite being one of the dataset were our algorithm performed the worse, the results show that it managed to recover quickly from the abrupt drift. The left hand side figure shows that the uncertainty of the Droplets algorithm over time. As the observations are constrained in an hypercube (and hence no new parts of the map are explored), the increase in uncertainty, reflects the conflicting observations that the algorithms received just after the drift.

VI. Conclusion

Learning with concept drifts is a challenging task. Many different types of unexpected drifts can potentially diminish the accuracy of a classifier trying to learn in such environment. Algorithms exhibiting good performances in this framework must be robust to noisy observations and flexible to changes, regardless the type of change. When dealing with so much uncertainty, an algorithm outputting reliable predictions can prove particularly useful, especially when wrong predictions are costly. However, in general, algorithms able to provide reliable predictions have a fixed confidence threshold, set by the user beforehand. This could be troublesome, as it might not be always easy to set a relevant threshold when dealing with unexpected changes.

Throughout this paper, the Droplets algorithm has been introduced. This algorithm can deal with various types of drifts (whether they are abrupt, incremental or reoccurring)

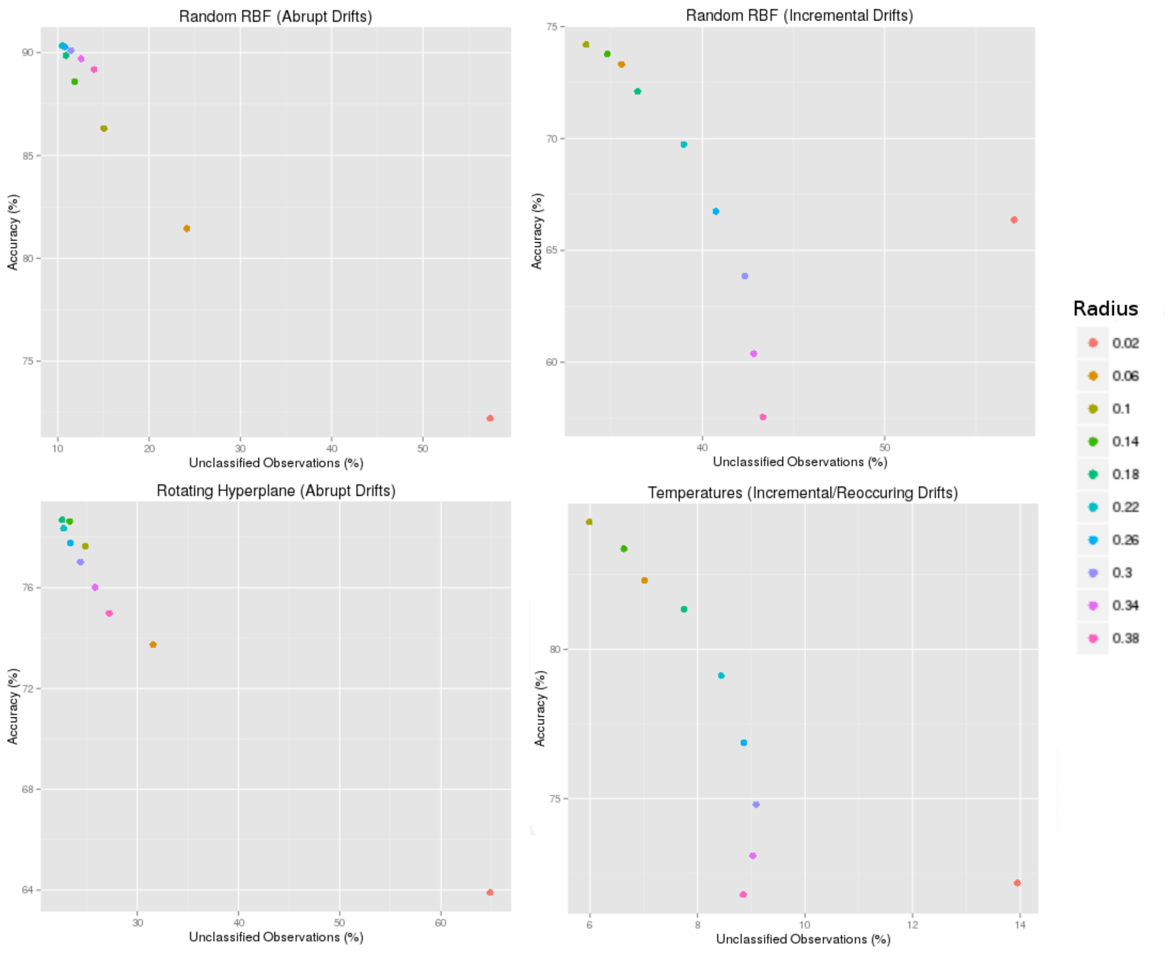
Table I. AVERAGE ACCURACY (IN PERCENT) ON THE OBSERVATIONS FOR WHICH THE DROPLETS GAVE A PREDICTION

	RBF_1	RBF_2	RH	Temp	RBF_3	Avg Acc	Avg Rank
Droplets	86.32	74.20	77.65	84.27	**100.00**	84.49	1.6
DWM	73.90	69.40	77.16	69.11	**97.36**	77.39	5.2
Leveraging Bagging	79.17	74.88	71.94	72.36	**96.18**	78.91	4.0
Hoeffding Adaptive Trees	**67.61**	73.74	69.95	65.10	**95.22**	74.32	7.6
ADWIN Bagging	78.32	73.89	72.64	68.00	**95.59**	77.69	5.2
Online Bagging	**72.29**	73.87	69.68	66.55	**95.41**	75.56	6.8
Online Boosting	81.58	73.35	73.92	69.10	**99.03**	79.40	4.2
Accuracy Updated Ensemble	**20.34**	**42.75**	**59.57**	**34.29**	**29.87**	37.36	10.8
Learn++.NSE	**40.07**	**60.44**	**65.67**	**43.29**	**2.05**	42.30	10.0
DACC	88.97	71.08	79.25	80.40	**98.75**	83.69	3.0
1-NN	68.07	62.91	64.56	56.23	100.00	70.35	7.4
Unclassified observations (%)	15.05	33.62	24.89	5.98	98.15	-	-

Table II. AVERAGE PERFORMANCE AND VARIANCE OF THE DROPLETS OVER 45 RUNS

	RBF_1	RBF_2	RH	Temp	RBF_3
Accuracy	87,79±0,31	73,32±0,36	77,47±0,40	84,28±0,37	100±0.00
Unclassified	15,43±0,91	34,12±1,89	25,23±3,13	5.00±0,34	98,15±0,01

Figure 5. Evolution of the accuracy against the percentage of unclassified observations for different default radius



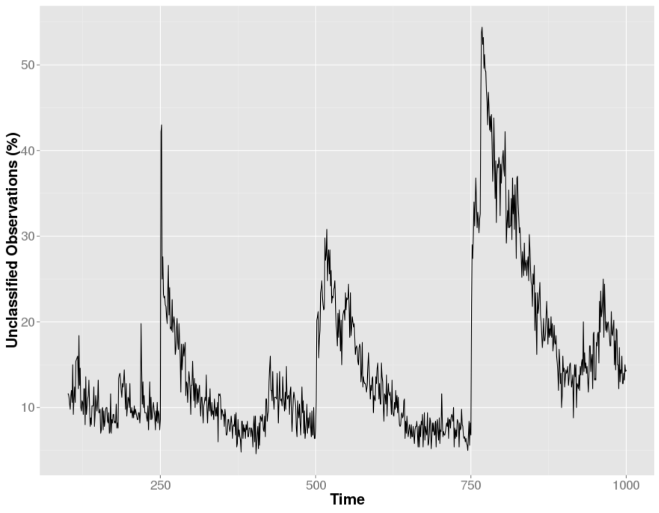
and produce reliable predictions by abstaining from predicting on potentially difficult observations. The reliable predictions are provided without a fixed confidence threshold.

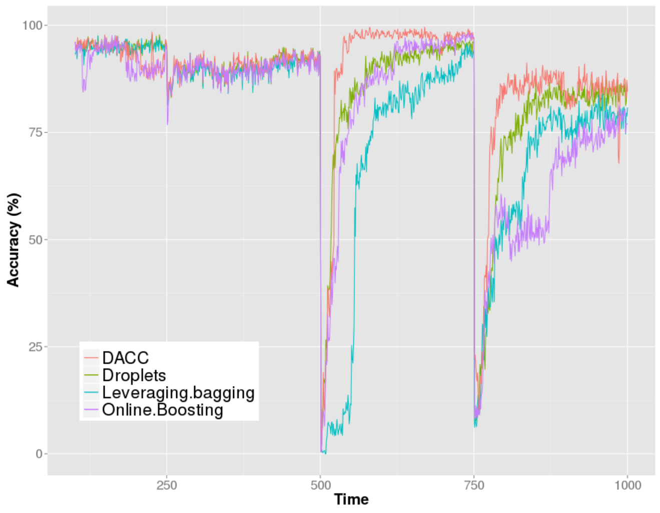
Experimental results show that the Droplets algorithm has

Table III. Average accuracy (percentage) on the observations where the Droplets algorithm didn't predict

	RBF_1	RBF_2	RH	Temp	RBF_3	Avg Acc	Avg Rank
DWM	50.76	51.30	72.65	**53.39**	**86.64**	62.95	3.8
Leveraging Bagging	46.74	55.86	63.65	**46.53**	**87.03**	59.96	4.6
Hoeffding Adaptive Trees	44.72	54.94	60.77	**48.55**	**86.37**	59.07	6.0
ADWIN Bagging	49.09	55.05	64.34	**50.70**	**85.40**	60.92	4.2
Online Bagging	43.29	55.03	63.59	**49.60**	**85.38**	59.38	5.8
Online Boosting	52.50	54.80	65.26	55.72	**88.37**	63.33	3.0
Accuracy Updated Ensemble	**25.22**	**37.18**	**54.84**	**32.43**	**31.78**	36.29	9.4
Learn++.NSE	**26.57**	**47.00**	**52.97**	**36.54**	**2.93**	33.20	9.0
DACC	70.57	50.83	75.85	66.36	**88.99**	70.52	2.4
1-NN	37.55	42.99	54.87	49.31	98.30	56.60	6.4

Figure 6. Evolution of the percentage of unclassified observations (left) and evolution of the accuracy of the top 4 classifiers on RBF1 (right)





the best average rank on 5 datasets (including abrupt, incremental and reoccurring drifts) and against 10 classifiers. They also show that it is able to accurately distinguish which observations are easily classifiable, as the performance of the other learners drops on the observations where the Droplets algorithm chose to abstain.

The algorithm can still be further improved. Future work will focus on assessing and optimizing the time complexity as well as the memory usage of the algorithm. Further investigation on performance metrics that take into account the cost of unclassified observation should also be considered. Finally, it could be interesting to assess the performances of the Droplets algorithm against an algorithm equipped with a fixed confidence threshold.

Acknowledgements.

We thank Vincent Lemaire, Marie-Jeanne Lesot as well as the reviewers for their comments and suggestions which helped to improve this paper.

References

[1] Gama, J., Bifet, A., Pechenizkiy, M., Bouchachia, A., & Zliobaite, I. (2014). A survey on concept drift adaptation. ACM Computing Surveys, 46(4).

[2] Zhang, C., Chaudhuri, K. (2014). Beyond Disagreement-based Agnostic Active Learning, NIPS 2014.

[3] Balasubramanian, V. N., Ho, S.-S., Vovk, V., Wechsler, H., & Li, F. (2014). Conformal Prediction for Reliable Machine Learning. Morgan Kaufmann Publishers Inc.

[4] G. Hulten, L. Spencer, and P. Domingos. 2001. Mining Time-Changing Data Streams. In Proc. of the 7th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD). ACM, 97–106.

[5] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavalda. 2009. New ensemble methods for evolving data streams. In Proc. of the Int. Conf. on Knowl. Discov. and Data Mining. ACM, USA, 139–148.

[6] W. Street and Y. Kim. 2001. A Streaming Ensemble Algorithm SEA for Large-Scale Classification. In Proc. 7th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD). ACM, 377– 382.

[7] Kolter, J., & Maloof, M. (2007). Dynamic Weighted Majority : An Ensemble Method for Drifting Concepts. Journal of Machine Learning Research, 8, 2755–2790.

[8] A. Bifet, G. Holmes, and B. Pfahringer. 2010. Leveraging Bagging for Evolving Data Streams. In Proc. of the Eur. Conf. on Mach. Learn. and Knowledge Discovery in Databases (ECMLPKDD).Springer-Verlag, Berlin, 135–150.

[9] N. Oza and S. Russell. Online bagging and boosting. In Artificial Intelligence and Statistics 2001, pages 105–112. Morgan Kaufmann, 2001.

[10] Breiman, L. (1996). Bagging predictors. Machine Learning, 24(421), 123–140.

[11] Bifet, A., & Gavaldà, R. (2007). Learning from Time-Changing Data with Adaptive Windowing. In SIAM International Conference on Data Mining

[12] Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., & Gavaldà, R. (2009). New ensemble methods for evolving data streams. Proceedings of the

15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '09, 139.

[13] Freund, Y., & Schapire, R. R. E. (1996). Experiments with a New Boosting Algorithm. International Conference on Machine Learning, 148–156.

[14] A. Bifet, R. Gavalda. Adaptive Learning from Evolving Data Streams. IDA '09 Proceedings of the 8th International Symposium on Intelligent Data Analysis: Advances in Intelligent Data Analysis VIII Pages 249 - 260

[15] Hulten, G. and Domingos, P. "VFML – A toolkit for mining high-speed time-changing data streams" http://www.cs.washington.edu/dm/vfml/. 2003.

[16] Hulten, G., Spencer, L., & Domingos, P. (2001). Mining time-changing data streams. Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '01, pages 97-106, ACM.

[17] Brzezinski, D., & Stefanowski, J. (2014). Reacting to different types of concept drift: the Accuracy Updated Ensemble algorithm. IEEE Transactions on Neural Networks and Learning Systems, 25(1), 81–94.

[18] R. Elwell and R. Polikar. 2011. Incremental Learning of ConceptDrift in Nonstationary Environments. IEEE Trans. on Neural Networks 22, 10 (2011), 1517–1531.

[19] Schapire, R. E., & Schapire, R. E. (1999). Improved Boosting Algorithms Using Confidence-rated Predictions. Computer, 336(1997), 297–336.

[20] Jaber, G. (2013). An approach for online learning in the presence of concept changes. PhD thesis, Université Paris Sud

[21] R. O. Duda, P. E. Hart, and D. G. Stork. 2001. Pattern Classification.Wiley

[22] Gao, J., Fan, W., Han, J., & Yu, P. S. (2007). A general framework for mining concept-drifting data streams with skewed distributions BT - 7th SIAM International Conference on Data Mining, April 26, 2007 - April 28, 2007, 3–14.

[23] Hastie, T., & Tibshirani, R. (1996). Discriminant adaptive nearest neighbor classification. Pattern Analysis and Machine …, 18(6).

[24] Bouchachia, A. (2011). Incremental learning with multi-level adaptation. Neurocomputing, 74(11), 1785–1799.

[25] El-Yaniv, R., & Wiener, Y. (2010). On the Foundations of Noise-free Selective Classification. Journal of Machine Learning Research, 11, 1605–1641.