# Ensemble classifiers for drift detection and monitoring in dynamical environments

Imen Khamassi[1], Moamar Sayed-Mouchaweh[2], Moez Hammami[1] and Khaled Ghédira[1]

[1]*University of Tunisia, Search Laboratory SOIE, Higher Institute of Management of Tunis*
*imenkhamassi@yahoo.fr*
*moezhammami@gmail.com*
*khaled.ghedira@isg.rnu.tn*

[2]*Univ Lille Nord de France, F-59000 Lille, France, Mines-Douai, IA, F-59500 Douai, France*
*moamar.sayed-mouchaweh@mines-douai.fr*

## ABSTRACT

Detecting and monitoring changes during the learning process are important areas of research in many industrial applications. The challenging issue is how to diagnose and analyze these changes so that the accuracy of the learning model can be preserved. Recently, ensemble classifiers have achieved good results when dealing with concept drifts. This paper presents two ensembles learning algorithms BagEDIST and BoostEDIST, which respectively combine the Online Bagging and the Online Boosting with the drift detection method EDIST. EDIST is a new drift detection method which monitors the distance between two consecutive errors of classification. The idea behind this combination is to develop an ensemble learning algorithm which explicitly handles concept drifts by providing useful descriptions about location, speed and severity of drifts. Moreover, this paper presents a new drift diversity measure in order to study the diversity of base classifiers and see how they cope with concept drifts. From various experiments, this new measure has provided a clearer vision about the ensemble's behavior when dealing with concept drifts .

## 1. INTRODUCTION

Recently, research in machine learning has shown its usefulness for automatic monitoring and diagnostics in industrial applications, especially when data is continuously generated and it is unpractical to store them all. Another issue is caused by the high speed of arrival of these data which require real time treatments and high computational efforts. Hence, the learning model must be able to classify this huge amount of data when environments are non-stationary; and the main challenge occurs when the underlying distribution that generates the data streams changes over time; which is known as "concept drift". Formally, the term "concept" refers to the distribution of the joint probability $P(X, w)$ in a certain point of time, where $X$ represents the input attributes and $w$ represents the class labels. A *concept drift* is a real or virtual change in this distribution. As stated by Tsymbal (2004), the *real concept drift* affects the posterior probability $P(w = w_i/X)$ which means that the target concept of the same values of attributes changes; we note $w_i \in W_c$ the set of $c$ different class labels and $1 \leq i \leq c$.The *virtual concept drift* affects the class-conditional probability $P(X/w = w_i)$ which means that the distribution of the underlying data within the same class changes. It is worth underlining that a drift can also affects the prior probability $P(w = w_i)$ of a particular class; this is known by "concept evolution". This form of drift can be due to *merging concepts* which refer to the emergence of novel classes, or to *crossing concepts* which refer to the fusion of existing classes (Masud et al., 2011).

Three steps are required to handle a concept drift:

- Monitoring step: When data are considered as non-stationary, methods with triggered mechanisms, namely informed methods (Ikonomovska et al., 2009), are used in order to provide descriptions about location, width and severity of the change. These methods can monitor the performance indicators of a learner, the estimators of data distributions or the learner's structure and parameters.

- Updating step: During this stage, the updating strategies differ according to the methods used to handle concept drifts. The blind methods implicitly adapt the learner to the current concept at regular intervals without any drift detection (Kolter and Maloof, 2007). Whereas, the informed methods can either relearn the model from scratch,

or update it using recent data when a change is detected. So the question is: how much data to remember or to forget? And what is the optimal size of the data-window in order to contain the most recent and significant data? In many studies (Sobhani and Beigy, 2011), the window size was a priori fixed. However, if the size is too small, we may not have enough data to train the leaner which risks being over fitted; and if the size is too large, the learner may keep outdated concepts which risks to reduce its accuracy. The fixed size window can work well if the width and severity of the change are known before or if we have rigorous instructions provided by an expert, but this is rarely the case. Recent studies have opted to windows with dynamic size which is reduced whenever a drift is detected and enlarged otherwise (Gama et al., 2004; Baena-García et al., 2006). In other studies, Lazarescu et al. (2004) have used multiple windows with different sizes in order to progressively adapt and predict the change.

- Diagnostic step: This step aims at interpreting the detected changes in concepts or in the learner parameters and structure.This interpretation can then be used as a short-term prognosis about the future tendency of the current system situation. Notice that it is important to differentiate between noise in data and real changes (Sayed-Mouchaweh, 2010; Lughofer and Angelov, 2011). Ideally, a trade-off between robustness against noise and flexibility in tracking concepts drifts must be reached. But unfortunately these two requirements seem to be contradictory.

In the light of these challenges, we present two ensembles learning algorithms BagEDIST and BoostEDIST, which respectively combine the Online Bagging and the Online Boosting with the drift detection method EDIST. EDIST is a new drift detection method which monitors the distance (the number of instances) between two consecutive errors of classification, and tracks concept drifts through two adaptive data-windows $W_G$ and $W_0$. EDIST makes use of a statistical hypothesis test in order to compare $W_G$ and $W_0$ error distance distributions and checks whether the averages differ by more than the adjusted threshold $\varepsilon$ .

The idea behind BagEDIST and BoostEDIST is to develop an ensemble learning algorithm which explicitly handles concept drifts by providing useful descriptions about location, speed and severity of drifts.

Moreover, this paper presents a new Drift Diversity Measure (D) in order to study the diversity of base classifiers and see how they cope with concept drifts. From various experiments, this new measure has provided a clearer vision about the ensemble's behavior when dealing with concept drifts.

The rest of the paper is organized as follows. In Section 2, we briefly discuss related work. In Section 3, we describe the drift detection method EDIST. In Section 4, BagEDIST and BoostEDIST are presented. In Section 5, the Drift Diversity Measure is decribed. In Sections 6 and 7, we explain the experimental setup, the results and the analysis of the different improvement stages of our method. Finally, conclusions and some ideas for future work are discussed in Section 8.

## 2. DISCUSSION AND RELATED WORK

In this section, the blind and the informed methods are discussed. The blind methods handle concept drifts implicitly, while the informed methods detect drifts by monitoring some performance indicators of the model (See Figure 1).

### 2.1. Informed Methods

These methods are useful when we expect to provide descriptions about the occurrence, the severity and the width of the encountered drifts. They proceed by monitoring and diagnosing these changes so that the accuracy of the learner can be conserved whatever the nature of the encountered drifts. These methods can make use of single classifier based on instance selection or batch collection; or an ensemble of classifiers with change detector.

#### 2.1.1. Single classifier with performance monitoring

This kind of methods monitors the indicators of performance of the model such as accuracy, precision and recall (Klinkenberg, 2001). These indicators are monitored constantly and compared to a confidence level or an adjusted threshold. Two well-known performance-based methods can be cited: Drift Detection Method (DDM) proposed by Gama et al. (2004); and Early Drift Detection Method (EDDM) proposed by Baena-García et al. (2006).

*a. DDM: Drift Detection Method*

The Drift Detection Method (DDM) monitors the number of errors produced by the learner and considers that the error rate follows the binomial distribution. In a sample of $n$ instances, the distribution gives the probability of misclassification $p_i$ with standard deviation $s_i = \sqrt{p_i(1-p_i)/i}$ for each instance $i$ of the sampled sequence. According to probability approximately correct leaning model (PAC) used by Mitchell (1997), if the distribution of instances is stationary, the error rate decreases as the number of instances increases, thus a significant increase in the error rate during the training implies a change in the distribution. DDM stores $p_{min}$ and $s_{min}$ which correspond respectively to the minimum probability and the minimum standard deviation, and then it defines two levels as follows:

- The Warning level when $p_i + s_i \geq p_{min} + 2 . s_{min}$ ; beyond this level, instances are stored for a possible change of distribution.
- The Drift level when $p_i + s_i \geq p_{min} + 3 . s_{min}$; beyond this level, the drift is confirmed and the learner is reset using instances stored since the Warning level. Note that $p_{min}$ and $s_{min}$ are reset too.

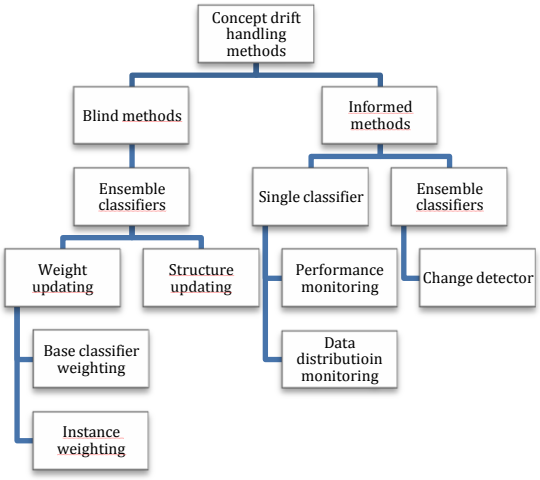


Figure 1. General scheme of classification methods handling concept drift

DDM has good ability to detect abrupt and global drifts which affect the whole dataset. However, it presents low adaptability to gradual and local drifts which slowly affect some parts of the dataset.

*b. EDDM: Early Drift Detection Method*

The idea behind the Early Drift Detection Method (EDDM) is to consider the distance between two consecutive errors of classification. Notice that the error distance is represented by the numbers of instances between two consecutive classification errors. This approach assumes that if the distribution of the instances is stationary, the learning model will improve its prediction and the error distance will increase as the number of instances increases. Thus, a significant decrease in the error distance implies a drift. EDDM calculates the average distance between two errors $p'_i$ and its standard deviation $s'_i$ for each instance $i$ of the sampled sequence and compare them to $p'_{max}$ and $s'_{max}$ reached before, and then defines two levels as follows:

- The Warning level when $(p'_i + 2.s'_i)/(p'_{max} + 2.s'_{max}) < \alpha$; beyond this level, instances are stored for a possible change of distribution.
- The Drift level when $(p'_i + 2.s'_i)/(p'_{max} + 2.s'_{max}) < \beta$ ; beyond this level, the drift is confirmed and the learner is reset using instances stored since the Warning level. Note that $p'_{max}$ and $s'_{max}$ are reset too. $\alpha$ and $\beta$ are respectively set to 0.95 and 0.9 after some experiments. The method considers the thresholds for detecting concept drifts when a minimum of 30 errors occur; then estimates the current error distance distribution and compares it with future ones.

EDDM is more suited to detect gradual and local drifts than DDM, but it presents sensitivity to the values of $\alpha$ and $\beta$ in the sense that bigger values are suited to detect gradual drifts whereas smaller values are more suited to detect abrupt drifts. Hence, a trade-off between these values and different types of drifts is required to reach good results.

### 2.1.2. Single classifier with data distribution monitoring

This kind of methods detects changes by measuring differences between consecutive batches of data. Sobhani and Beigy (2011) present a method which process chunk by chunk and measures differences between two consecutive batches of data. The idea is to find nearest neighbor in previous batch of data for each instance in current batch, then compare their corresponding class labels. The authors use the *heom distance* to quantify the similarity between data batches and the drift alarm is launched when there is a significant increase of the degree of drift (DoF) value (for more details please refer to Sobhani and Beigy (2011)).

### 2.1.3. Ensemble classifiers with change detector

The ensemble classifiers with change detector explicitly handle concept drifts by providing useful description about the change. This new kind of technique is becoming an interesting area of research, because:

- it combines the flexibility of the ensemble classifier to cope with different types of drifts and,
- it can provide useful descriptions about location, width and severity of the drift.

ADWIN bagging method (Bifet et al. 2009) combines the online version of bagging for data streams developed by Oza and Russell (2001) with ADWIN which represents a change detector (Bifet et al. 2007). The idea behind ADWIN is to track the average of a stream of bits or real-valued numbers and keep a dynamic sliding window of recently seen instances, with the property that the window has the maximal length statistically consistent with the hypothesis "there has been no change in the average value inside the window".
These approached have achieved good accuracy when dealing with different kind of drifts, but they present two main issues; the first is how to maintain the diversity of the classifiers overall the learning process? and the second is how can we improve the run time and the memory consumption when resources are scarce?

## 2.2. Blind methods

The ensemble classifiers approaches are considered as blind methods when they are implicitly adapted to changes without any drift detection tool (Minku et al. 2010).

### 2.2.1. Structure-updating ensemble classifiers

In the dynamic ensemble classifiers the structure of the ensemble evolves to deal with the concept drifts. A possible strategy is "replace the loser": the individual classifiers are

re-evaluated and the worst one is replaced by a new one trained on recent data (Kuncheva, 2004).

The Dynamic online Ensemble Learning Algorithm (DELA) is characterized by dynamic and continuous structural update of the ensemble classifiers as soon as the global accuracy decreases. DELA makes use of distinct and incremental base learners which handle drift detection by their nature, and then updates the structural in two ways:

- The addition of classifiers is made when (i) the ensemble fails to predict the correct label or when (ii) it makes as many as errors as half of a window of length Q.
- The removal of classifiers is made when (i) the ensemble fails to predict the correct label or when (ii) the base learner makes an error on each step over the last window of length Q.

### 2.2.2. Weight-updating ensemble classifiers

In the weight-updating ensemble classifiers the structure of the ensemble is fixed, whereas the weights of the base classifiers or of the instances may evolve.

*a. Weight-updating base classifiers*

This kind of approach makes use of ensemble classifiers where each component classifier is evaluated and receives a weight reflecting its performance on the most recent batch of data.

In the Accuracy Weighted Ensemble (AWE), Kolter and Maloof (2007) propose to train a new offline learning classifier on each incoming data batch, then use that batch to evaluate all the existing classifiers in the ensemble. To evaluate each classifier, they propose to derive weights by estimating the error rate on the most recent data batch. For the first *k* data chunks, AWE takes a set of all available classifiers, but it selects only the *k* best components to form an ensemble. The predictions of components are aggregated by a weighted voting rule. However, the main problem of AWE is the tuning of the batch size of the most recent data used to evaluate all the existing classifiers.

The Accuracy Updated Ensemble (AUE) is another ensemble approach which proceed by updating each classifier features as new instances are available. Brzezinski and Stefanowski (2011) use online learning classifiers updated with recent instances, then adjust their weights. During the weighting process AUE preserves only the *k* best classifiers according to a simple weighting function. When no concept drift occurs, each base classifiers can be trained on more instances as if it was built on bigger batch of data; and should update its features to become more accurate. Good results have been achieved; however AUE has to ensure additional diversity of the ensemble components.

*b. Instance-updating ensemble classifiers*

This kind of approach makes use of ensemble classifiers where the weights of the instances are modified in order to preserve diversity when dealing with concept drift.
The Leveraging bagging approach (Bifet et al. 2010a) is an ensemble approach that makes use of a modified bagging approach by adding more randomization on the weights of the instances of the input stream in order to improve the accuracy of the ensemble classifier.

## 3. EDIST: ERROR DISTANCE BASED APPROACH FOR DRIFT DETECTION AND MONITORING

In this section, a new drift detection method EDIST is described. Then its performance is evaluated using datasets presenting concepts drifts of several width, severity and time change.

We consider the online learning framework where the instances arrive one at the time and we assume that the learning model is able to make a prediction as an instance is available. Once the prediction is made, the system can learn from instances and incorporate them to the learning model. Each instance is in the form of pairs $(\vec{x}_i, w_i)$ where $\vec{x}_i$ is the vector values of different attributes and $w_i$ is the class label. The model prediction $w_i'$ is correct when $w_i' = w_i$, false otherwise.

The idea behind EDIST draws its inspiration from the popular drift detection method EDDM which studies the distance between two consecutive errors of classification. Note that the distance is represented by the numbers of instances between two consecutive errors of classification. In EDIST, we track the concept drift through two data-windows. The first represents the global sliding window $W_G$ which is adaptively adjusted by containing the recent read instances. The second $W_0$ represents the batch of current collected instances. Note that $W_0$ is constructed from a fixed number of consecutive errors of classification, thus it could contain a variable amount of instances at each step. In EDIST, we want to estimate the error distance distribution of $W_G$ and $W_0$ and make a comparison between their error distance averages in order to check a difference.

In EDIST, we employ the same hypothesis used in EDDM and which assumes that if the distribution of the instances is stationary, the learning model will improve its prediction and the error distance will increase as the number of instances increases. Thus, a significant decrease in the error distance implies a change in the instances distribution and suggests that the learning model is not appropriate. Unlike EDDM which compares the current average of error distance and its standard deviation with the maximum average and standard deviation stored from previous instances, EDIST makes use of a statistical hypothesis test in order to compare $W_G$ and $W_0$ error distance distributions and check whether the averages differ by more than the threshold $\varepsilon$. The novelty of our method is that there is no a priori definition of the threshold $\varepsilon$, in the sense that it does

not require any a priori adjusting related to the expected width or severity of the change. $\varepsilon$ is adaptively adjusting itself according to the statistical test used.

### 3.1. Statistical hypothesis test

Let $d_1, d_2, d_3, \ldots d_t, \ldots$ be the sequence of the error distance values where each value $d_t$ is available only at time $t$ and independently generated from an error distance distribution $D_t$.

Let $X_G$ and $X_0$ be the random variables of the two error distance distribution $D_G$ and $D_0$ of respectively $W_G$ and $W_0$, we assume that $X_G \sim N\left(\mu_G, \frac{\sigma_G}{\sqrt{N}}\right)$ and $X_0 \sim N\left(\mu_0, \frac{\sigma_0}{\sqrt{n}}\right)$ are normally distributed where $N$ and $n$ are the numbers of errors of classification respectively occurred in $W_G$ and $W_0$. We pose $\mu_d = \mu_G - \mu_0$ and define:

- The null hypothesis ($H_0$): $\mu_d = 0$ which states that there is no change between the two distributions averages.
- The alternative hypothesis ($H_1$): $\mu_d > 0$ which states that there is a decrease in $D_0$'s average, hence we detect a change.

### 3.2. Region of acceptance

We suppose that $H_0$ is true, thus the random variable $X_d \sim N(\mu_d, \sigma_d)$ is normally distributed with $\mu_d = 0$ and $\sigma_d = \sqrt{\frac{\sigma_G}{\sqrt{N}} + \frac{\sigma_0}{\sqrt{n}}}$; and let α=0.05 be the test's probability of incorrectly rejecting $H_0$.

We want to calculate $\varepsilon$ such that:

$$P(X_d \leq \mu_d + \varepsilon) = 0.95 \qquad (1)$$

Let $T = \frac{X_d - \mu_d}{\sigma_d}$ be a random variable following the standard normal distribution $N(0,1)$

$$P\left(T \leq \frac{\varepsilon}{\sigma_d}\right) = 0.95 \qquad (2)$$

so, the cumulative distribution function of the normal distribution is defined as follows:

$$\varphi\left(\frac{\varepsilon}{\sigma_d}\right) = 0.95 \qquad (3)$$

and thus, according to the table of standard normal distribution, we can write:

$$\frac{\varepsilon}{\sigma_d} = t_{1-\alpha} \qquad (4)$$

Finally,

$$\varepsilon = t_{1-\alpha} * \sigma_d \qquad (5)$$

with $\sigma_d = \sqrt{\frac{\sigma_G}{\sqrt{N}} + \frac{\sigma_0}{\sqrt{n}}}$ and $t_{1-\alpha} = t_{0.95} = 1.65$

### 3.3. Decision rule

If $\mu_d \leq \varepsilon$ then we accept the null hypothesis $H_0$ with a risk of 5% to be wrong, else we accept $H_1$.

As in DDM and EDDM, our method defines three thresholds (see Fig.2):

- The In-Control level: $\mu_d \leq \varepsilon$ beyond this level we affirm that there is no change between the two distributions, so we enlarge $W_G$ by adding $W_0$ 's instances, and then we reset it in order to collect new ones.

- The Warning level: $\mu_d > \varepsilon + r * \sigma_d$; beyond this level, the instances are stored for an expected change. However, if the similarity between the two distributions increases, i.e. the drift is not confirmed after the Warning level, we consider that there is a false alarm and we remove the instances stored during this stage.

- The Drift level: $\mu_d > \varepsilon + s * \sigma_d$; beyond this level, the drift is confirmed and $W_G$ is reset by only containing the instances stored since the Warning level, then the learning model is reconstructed from the new $W_G$.

Note that $r$ and $s$ are integer values which represent the amounts of change for defining respectively the Warning and Drift levels; where $s > r$.

In static context, when we consider that $r = 2$, it means that $\mu_d$ has a variance smaller than $2 * \sigma_d$ with 95% of confidence; and when we consider that $s = 3$, it means that $\mu_d$ have a variance smaller than $3 * \sigma_d$ with 99.7% of confidence. In practice, we have varied $r$ and $s$ values from 0 to 3 in order to study the relationship between those values and different types of drifts in non-stationary context.

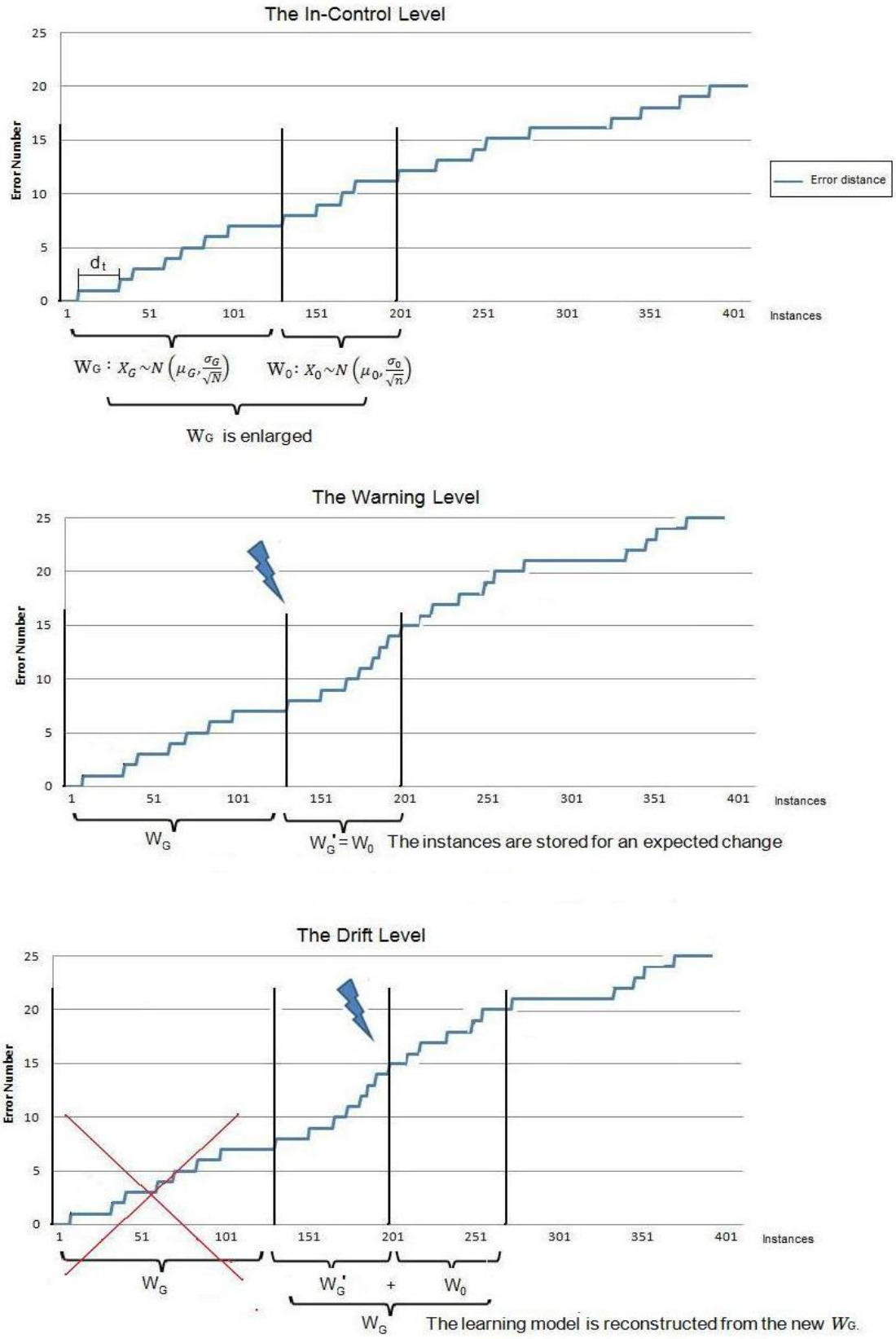


Figure 2. The three levels for drift detection and monitoring in EDIST

## 4. BAGEDIST AND BOOSTEDIST

In this section we describe the two ensembles learning algorithms BagEDIST and BoostEDIST, which respectively combine the Online Bagging and the Online Boosting with the drift detection method EDIST.

### 4.1. Online Bagging

The Off-line Bagging (Oza and Russell, 2001) builds a set of M base classifiers, training each classifier with a bootstrap sample of size N created by drawing random samples with replacement from the original training set. Each base classifier's training set contains each of the original training example k times where k is drawn from a Poisson$(1)$ distribution for large values of N.

In Online Bagging (Oza and Russell, 2001), each example is given a weight k drawn from a Poisson$(1)$ distribution, in order to be represented k times for each base classifier $h_m$. The ensemble prediction is done by unweighted majority vote (see Algorithm 1).

**Algorithm 1: Online Bagging Algorithm**

For all training examples $(x, y)$ do
  For each base model $h_m$ $(m \in \{1, \dots, M\})$ do
    **1.** Set $k \leftarrow$ *Poisson(1)*
    **2.** Do $k$ times
      $updateModel(h_m, x)$
  end For
$Prediction_{ensemble}(x) = argmax_{y \in Y} (\sum_{m=1}^{M} [y = Prediction_m(x)])$
end For

### 4.2. BagEDIST

BagEDIST is an ensemble learning algorithm which combines the Online Bagging algorithm with the drift detection method EDIST. The idea behind this combination is to develop an ensemble learning algorithm which explicitly handles concept drifts by providing useful descriptions about location, speed and severity of the changes. Unlike Online Bagging which assures the diversity by only weighting the training examples, BagEDIST proceeds by weighting the training examples and the base classifiers in the same time. As presented in algorithm 2, the weighting process of each classifier depends on its prediction performance and its ability to detect drifts. The accuracy $acc_m$ of the classifier $h_m$ represents the correct prediction rate, and $sumDrift_m$ is the number of drifts detected by $h_m$. Notice that every classifier is equipped by EDIST as drift detection mechanism and when a drift is confirmed the classifier is relearned from the examples stored during the warning level of EDIST (more details are presented in section 2). The weight $w_m$ is used to promote the classifier which presents good accuracy and high number of detected drifts. The parameter $\alpha$ is set to 2 to give more importance to classifier that achieved the best accuracy. The final ensemble prediction is done by weighted majority vote (see Algorithm 2).

**Algorithm 2: BagEDIST Algorithm**

For all training examples $(x, y)$ do
  For each base model $h_m$ $(m \in \{1, \dots, M\})$ do
    **1.** Set $k \leftarrow$ *Poisson(1)*
    **2.** Do $k$ times
      $updateModel(h_m, x)$
    **3.** If $correctPrediction(h_m, x)$ then
      $acc_m \leftarrow computeAccuracy(h_m)$
    **4.** If $driftDetection(h_m, EDIST)$ then
      $sumDrift_m \leftarrow sumDrift_m + 1$
      reset $h_m$ using examples stored since the $Warning\ level$ defined by $EDIST$
    end If
    **5.** $w_m \leftarrow computeWeightBagEDIST(h_m, sumDrift_m, acc_m)$
  end For
$Prediction_{ensemble}(x) = argmax_{y \in Y} (\sum_{m=1}^{M} w_m [y = Prediction_m(x)])$
end For

$computeWeightBagEDIST(h_m, sumDrift_m, acc_m)$

If $(sumDrift_m = 0)$ then
  $w_m \leftarrow acc_m$
else If $(sumDrift_m = 1)$ then
  If $accuracyMax(acc_m)$ then
    $w_m \leftarrow 1 + \alpha * acc_m$
  else
    $w_m \leftarrow 1 + acc_m$
  end If
else
  If $accuracyMax(acc_m)$ then
    $w_m \leftarrow log(sumDrift_m) + \alpha * acc_m$
  else
    $w_m \leftarrow log(sumDrift_m) + acc_m$
  end If
end If
Return $(w_m)$

### 4.3. Online Boosting

The Off-line Boosting (Oza and Russell, 2001) builds a set of M classifiers sequentially $\{h_1, \dots, h_M\}$ such that the training examples misclassified by classifier $h_{m-1}$ are given more weight for the next classifier $h_m$. Hence, the idea behind Boosting algorithms is to combine multiple base classifiers to obtain a small generalization error.

In Online Boosting (Oza and Russell, 2001) each example is given a weight k drawn from a Poisson$(\lambda_d)$ distribution, where $\lambda_d$ is increased when the training example is misclassified by the previous classifier; and decreased otherwise. $\lambda_m^{sc}$ and $\lambda_m^{sw}$ are the sum of $\lambda_d$ values scaled by the half of the total weight N for respectively corrected and uncorrected examples. At final stage, each classifier $h_m$ is weighted according to its $\lambda_m^{sc}$ and $\lambda_m^{sw}$ parameters then the final classification is done by weighted majority vote (see Algorithm 3).

**Algorithm 3: Online Boosting Algorithm**

For all training examples $(x, y)$ do
  $\lambda_d \leftarrow 1$
  For each base model $h_m$ $(m \in \{1, \dots, M\})$ do
    **1.** Set $k \leftarrow Poisson(\lambda_d)$
    **2.** Do $k$ times
      $updateModel(h_m, x)$
    **3.** If $correctPrediction(h_m, x)$ then
      $\lambda_m^{sc} \leftarrow \lambda_m^{sc} + \lambda_d$
      $\lambda_d \leftarrow \lambda_d * \frac{N}{2*\lambda_m^{sc}}$
    else
      $\lambda_m^{sw} \leftarrow \lambda_m^{sw} + \lambda_d$
      $\lambda_d \leftarrow \lambda_d * \frac{N}{2*\lambda_m^{sw}}$
    end If
    **4.**
  $w_m \leftarrow computeWeightBoosting(\lambda_m^{sc}, \lambda_m^{sw})$
  end For
  $Prediction_{ensemble}(x) = argmax_{y \in Y} (\sum_{m=1}^{M} w_m[y = Prediction_m(x)])$
end For

$computeWeightBoosting(\lambda_m^{sc}, \lambda_m^{sw})$

Calculate

$$\epsilon_m \leftarrow \frac{\lambda_m^{sw}}{\lambda_m^{sw} + \lambda_m^{sc}}$$

$$\beta_m \leftarrow \frac{\epsilon_m}{1 - \epsilon_m}$$

$$w_m \leftarrow log\left(\frac{1}{\beta_m}\right)$$

Return $(w_m)$

### 4.4. BoostEDIST

BoostEDIST is an ensemble learning algorithm which combines the Online Boosting algorithm with the drift detection method EDIST. The first originality of BoostEDIST is that the training examples misclassified by classifier $h_{m-1}$ and which triggered a drift are given more weight for the next classifier $h_m$. Hence, a new parameter $\lambda_m^{drift}$ is used to compute the sum of $\lambda_d$ values when a drift is detected, and it is scaled by the total number of drifts (TotalDrifts) detected by the ensemble. Like in BagEDIST, every classifier is equipped by EDIST as drift detection mechanism and when a drift is confirmed the classifier is relearned from the examples stored during the warning level of EDIST. The second originality of BoostEDIST is that the weighting process of base classifier depends on its prediction performance and its ability to detect drifts. The weight $w_m$ is computed such that the classifier which presents high $\lambda_m^{sc}$ and $\lambda_m^{drift}$ values is promoted. The final ensemble prediction is done by weighted majority vote (see Algorithm 4).

**Algorithm 4: BoostEDIST Algorithm**

For all training examples $(x, y)$ do
  $\lambda_d \leftarrow 1$
  For each base model $h_m$ $(m \in \{1, \dots, M\})$ do
    **1.** Set $k \leftarrow Poisson(\lambda_d)$
    **2.** Do $k$ times
      $updateModel(h_m, x)$
    **3.** If $correctPrediction(h_m, x)$ then
      $\lambda_m^{sc} \leftarrow \lambda_m^{sc} + \lambda_d$
      $\lambda_d \leftarrow \lambda_d * \frac{N}{2*\lambda_m^{sc}}$
    else
      $\lambda_m^{sw} \leftarrow \lambda_m^{sw} + \lambda_d$
      $\lambda_d \leftarrow \lambda_d * \frac{N}{2*\lambda_m^{sw}}$
    end If
    **4.** If $driftDetection(h_m, EDIST)$ then
      $\lambda_m^{drift} \leftarrow \lambda_m^{drift} + \lambda_d$
      $\lambda_d \leftarrow \lambda_d * \frac{TotalDrifts}{\lambda_m^{drift}}$
      reset $h_m$ using examples stored since the Warning level defined by $EDIST$
      end If
    **5.**
  $w_m \leftarrow computeWeightBoostEDIST(\lambda_m^{sc}, \lambda_m^{sw}, \lambda_m^{drift})$
  end For
$Prediction_{ensemble}(x) = argmax_{y \in Y} (\sum_{m=1}^{M} w_m[y = Prediction_m(x)])$
end For

$computeWeightBoostEDIST(\lambda_m^{sc}, \lambda_m^{sw}, \lambda_m^{drift})$

Calculate

$$\epsilon_m \leftarrow \frac{\lambda_m^{sw}}{\lambda_m^{sw} + \lambda_m^{sc} * \left(1 + \lambda_m^{drift}\right)}$$

$$\beta_m \leftarrow \frac{\epsilon_m}{1 - \epsilon_m}$$

$$w_m \leftarrow log\left(\frac{1}{\beta_m}\right)$$

Return $(w_m)$

## 5. DRIFT DIVERSITY MEASURE

Diversity in ensemble learning algorithms is an issue that has received lots of attention in the literature. In off-line and online ensemble classifiers, the diversity could be expected to be one of the features that help to achieve good accuracy. Only some empirical studies have highlighted this hypothesis and have revealed that there could be a positive correlation between the accuracy of the ensembles and the diversity of its base classifiers (Kuncheva and Whitaker, 2003; Minku et al., 2010) However, there are still not theoretical explanations of how the diversity can enhance the performance of ensemble classifiers. Moreover, some results in the studies of Kuncheva and Whitaker (2003) have raised some doubts about the relationship between diversity

and accuracy and the authors have concluded that large diversity may not always correspond to better ensemble results.

In this investigation, we intuitively ought to study the diversity of the base classifiers in BagEDIST and BoostEDIST algorithms, and see how they cope with concept drifts. For this purpose, a new diversity measure, namely Drift Diversity Measure, is proposed. Unlike the most existing diversity measures, which are often related to the accuracy of base classifiers, the Drift Diversity Measure takes into consideration the performance of each base classifier and its ability to detect drifts. This new measure may help us to have a clearer vision about the ensemble's behavior when dealing with concept drifts.

The formulation of Drift Diversity Measure (D), which is inspired from the Entropy Diversity Measure (E) (Cunningham and Carney, 2000) is as follows:

$$D = \frac{1}{N} * \sum_{i=1}^{N} \sum_{y=1}^{Y} \left(-p_{iy}^{drift} log_2(p_{iy}^{drift})\right) \qquad (6)$$

Where N is the number of examples, Y is the number of class labels and $p_{iy}^{drift}$ is the ratio of base classifiers that have classified the example i as class y and detected a drift. Let assume the number of base classifiers is M. If three base classifiers classify the first example as class 1 and only two classifiers have detected a drift, $p_{11}^{drift} = \frac{2}{M}$.
The Drift Diversity Measure (D) $\in [0, 1]$ and larger value means larger diversity of base classifiers.

## 6. EXPERIMENTAL EVALUATION

BagEDIST and BoostEDIST were implemented in the java programming language by extending the Massive Online Analysis (MOA) software (Bifet et al., 2010) MOA is an online learning framework for evolving data streams. It derives from the well-known WEKA framework, and supports a collection of offline and online machine learning methods for both classification and clustering.

To evaluate EDIST, we have used the Hoeffding Tree (HT) learning algorithm. HT is an incremental decision tree induction algorithm that is able to learn from a massive data streams. (for more details please refer to (Hulten et al., 2001)). We have used HT implemented in MOA with information gain split criterion and Adaptive Naive Bayes classification at leaves.

### 6.1. Parameter Settings

To evaluate BagEDIST and BoostEDIST algorithms, we have used the Prequential Evaluation method presented by Gama et al. (2009). This method evaluates a classifier on a stream by testing then training with each example in sequence and may use a sliding window or a fading factor forgetting mechanism. We have used the Prequential Evaluation method implemented in MOA with sliding window of size 5000.

The number of base classifiers for BagEDIST and BoostEDIST was fixed to 10, and each classifier used the Hoeffding Tree (HT) as learning algorithm and EDIST as a drift detection method.

The parameter settings of the drift detection method EDIST are as follows:

- NB represents the minimum number of examples to initialise the learning algorithm and it is set to 30
- n represents the number of errors of classification occurred in $W_0$ and it is set to 90
- r represents the amount of change for the warning level and it varies from 0 to 1 and,
- s represents the amount of change for the drift level and it varies from 1 to 3; with the constraint s>r.

For the comparison, we have developped BagDDM, BoostDDM, BagEDDM and BoostEDDM which use the same BagEDIST and BoostEDIST algorithms with changing EDIST by the two well-known drift detection methods DDM and EDDM. We have also compared our approaches to BagADWIN and BoostADWIN proposed by Bifet et al. (2009) and which use ADWIN (Bifet et al. 2007) as a change detector in the original version of Online Bagging and Online Boosting developed by Oza and Russell (2001).

### 6.2. Synthetic data sets

Synthetic data sets are primordial for studying the behaviour of the proposed algorithms where severity, width and time of change of concept drifts are known.

***Rotating Hyperplane*** has been widely used to simulate a changing concept based on moving hyperplane (Hulten et al. 2001). A hyperplane in d-dimensional space is represented by: $\sum_{i=1}^{d} w_i x_i = w_0$ . The examples which satisfy $\sum_{i=1}^{d} w_i x_i \geq w_0$ are labelled as positive, otherwise negative. The concept drift is simulated either by the orientation or the position of the hyperplane which can be changed by varying the relative size of the weights.

***Agrawal Generator*** was introduced by Agrawal et al. (1993). It generates one of ten different pre-defined loan functions. The generator produces a stream containing nine attributes, six numeric and three categorical. This generator is based on ten functions defined for generating binary class labels from the attributes in order to determine whether the loan should be approved.

### 6.3. Real world datasets

Hereafter, the real world datasets used for evaluating our method.

***Electricity dataset*** is a real world dataset from the Australian New South Wales Electricity Market. This dataset was first described by Harries (1999). In this electricity market, the prices are not fixed and may be affected by demand and supply. This dataset contains 45312 instances, dated from May 1996 to December 1998.

***Forest Covertype*** contains the forest cover type for 30 x 30 meter cells obtained from US Forest Service (USFS) Region 2 Resource Information System (RIS) data. It contains 581012 instances and 54 attributes, and the task is to predict the specific forest cover types.

***Poker-Hand*** consists of 1000000 instances and 11 attributes. Each record of the Poker-Hand dataset is an instance of a hand consisting of five playing cards drawn from a standard deck of 52. Each card is described using two attributes (suit and rank), for a total of 10 predictive attributes; and one class attribute that describes the "Poker Hand".

***AirlinesDataset*** consists of a large amount of records, containing flight arrival and departure details for all the commercial flights within the USA, from October 1987 to April 2008. It contains 13 attributes and the task is to predict whether a given flight will be delayed or not.

## 7. RESULTS

In this section, the performance of BagEDIST and BoostEDIS is evaluated using data sets presenting concepts drifts of several severity, width and time change.

### 7.1. The diversity of base classifiers in BagEDIST and BoostEDIST algorithms

In this subsection, we use the Drift Diversity Measure to study the diversity evolution of base classifiers in BagEDIST and BoostEDIST algorithms on the Agrawal dataset of 300000 examples.

In Figure 3, the scatter plots correspond to detected drifts of every base classifier, the curves correspond to the evolution of diversity measure of the ensemble; and vertical dotted lines represent the occurrence positions of concept drifts.

From this experiment, it is noticeable that (i) some base classifiers make early detection of drifts where others detect drifts with acceptable delay. This implies that there is a good combination of base classifiers in the ensemble, in the sense that a classifier's early detection can make up for the delay of another. This confirms that the ensemble proceeds through an effective cooperation between base classifiers in order to optimally detect drifts. (ii) When we observe the diversity measure curves, we remark that the diversity increases as soon as a drift is encountered and stabilizes when it ends. This confirms that the Drift Diversity Measure is an adequate measure for ensembles that handle drifts and can provide effective information about width and time change of drifts.

### 7.2. The Drift Diversity Measure Vs. accuracy

For Hyperplane datasets, we have added two concept drifts at $t_0$=50000 and $t_0$=150000 and varied the widths from 5000 to 100000 by generating 300000 examples. Results are exposed in tables 1 and 2 where columns represent the Prequential accuracies (Acc), the Drift Diversity Measure (D) and the total number of drifts detected by the ensemble (num Drifts).

As shown, BagEDIST and BoostEDIST have found the best accuracy for both abrupt and gradual drifts; and presented more robustness to noise than the others methods. In the same time, it is worth underlining that there is no clear correlation between the accuracy and the diversity for different methods, in the sense that best accuracies may not correspond to highest amounts of diversity. Another important point is that the diversity measures obtained are positively correlated with the total number of detected drifts. Hence, we can define the Drift Diversity Measure as the degree of base classifiers disagreement about detecting drifts, in the sense that the Drift Diversity Measure increases as much as classifiers disagree about the detected drifts. This definition has two interpretations. The first is that the Drift Diversity Measure is an adequate measure for drift handling ensembles because it is obvious that a set of base classifiers that identically detect drifts does not bring any advantages. The second is that when the ensemble presents high diversity and low accuracy, it may be explained by its overreacting to concept drifts or to its sensitivity to false alarms. Once again, we are facing the issue of how quantifying the ensemble diversity in order to enhance the overall accuracy.

### 7.3. Results on real world data sets

For the comparison, we have developped BagDDM, BoostDDM, BagEDDM and BoostEDDM which use the same BagEDIST and BoostEDIST algorithms with changing EDIST by the two well-known drift detection methods DDM and EDDM.
BagEDIST and BoostEDIST have also been tested through real world data sets widely used in similar studies. Despite the different features of each real data set, encouraging results have been found comparing to BagADWIN, BoostADWIN, and to the original version of Online Bagging and Online Boosting (OzaBag and OzaBoost).
These results reaffirm the effectiveness of the adaptation of the original version of Bagging and Boosting for drift handling with the use of any drift detection method DDM, EDDM or EDIST (see tables 3 and 4).

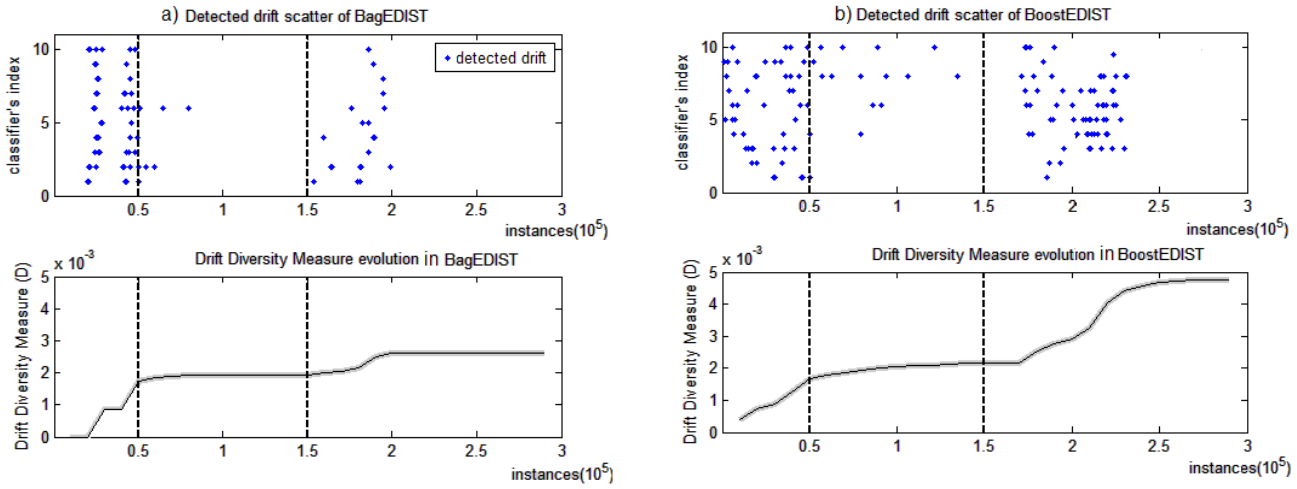


Figure 3. The diversity of base classifiers for repectively BagEDIST (a) and BoostEDIST (b) algorithms in Agrawal dataset containing 300000 examples, 2 concept drifts at $t_0$=50000 and $t_0$=150000 with width=50000.

Table 1. Prequential accuracy (Acc), the Drift Diversity Measure (D) and the total number of drifts detected by the ensemble (num Drifts) in Hyperplane datasets containing 300000 examples, 2 concept drifts at $t_0$=50000 and $t_0$=150000, 20% then 40% of severity and 30% of noise.

| | Drift width | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 5000 | | | 10000 | | | 50000 | | | 100000 | | |
| | acc | (D) | num drifts | acc | (D) | num drifts | acc | (D) | num drifts | acc | (D) | num drifts |
| BagEDIST | 68,400 | 8,969E-05 | 81 | 68,800 | 8,083E-05 | 73 | 68,700 | 5,979E-05 | 54 | 67,100 | 5,426E-05 | 49 |
| BagDDM | 67,800 | 7,308E-05 | 66 | 68,400 | 6,312E-05 | 37 | 68,100 | 6,422E-05 | 58 | 66,000 | 2,215E-05 | 20 |
| BagEDDM | 67,800 | 1,484E-04 | 134 | 68,300 | 1,440E-04 | 130 | 68,400 | 1,583E-04 | 143 | 67,200 | 1,532E-04 | 139 |

Table 2. Prequential accuracy (Acc) , the Drift Diversity Measure (D) and the total number of drifts detected by the ensemble (num Drifts) in Hyperplane datasets containing 300000 examples, 2 concept drifts at $t_0$=50000 and $t_0$=150000, 20% then 40% of severity and 30% of noise.

| | Drift width | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 5000 | | | 10000 | | | 50000 | | | 100000 | | |
| | acc | (D) | num drifts | acc | (D) | num drifts | acc | (D) | num drifts | acc | (D) | num drifts |
| BoostEDIST | 69,900 | 7,530E-05 | 68 | 68,400 | 8,083E-05 | 73 | 68,800 | 1,882E-05 | 17 | 66,200 | 3,986E-05 | 36 |
| BoostDDM | 68,000 | 3,100E-05 | 28 | 68,400 | 3,322E-05 | 30 | 68,000 | 1,661E-05 | 15 | 65,900 | 1,107E-05 | 10 |
| BoostEDDM | 67,200 | 8,637E-05 | 78 | 67,100 | 1,207E-04 | 109 | 66,900 | 1,008E-04 | 91 | 66,100 | 6,755E-05 | 61 |

Table 3. Prequential accuracy of different Bagging ensemble algorithms in real world data sets.

| | ELEC2 | Covtyp | Poker | Airline |
|---|---|---|---|---|
| BagEDIST | 91,800 | 96,800 | 67,600 | 63,500 |
| BagDDM | 87,000 | 97,400 | 64,300 | 62,100 |
| BagEDDM | 81,400 | 97,200 | 78,000 | 63,300 |
| BagADWIN | 75,300 | 97,100 | 65,500 | 61,600 |
| OzaBag | 88,500 | 91,000 | 63,300 | 64,200 |

Table 4. Prequential accuracy of different Boosting ensemble algorithms in real world data sets.

| | ELEC2 | Covtyp | Poker | Airline |
|---|---|---|---|---|
| BoostEDIST | 89,800 | 96,500 | 80,000 | 59,900 |
| BoostDDM | 89,300 | 96,400 | 81,100 | 60,100 |
| BoostEDDM | 88,400 | 97,100 | 90,800 | 62,800 |
| BoostAdwin | 88,000 | 97,200 | 85,800 | 58,200 |
| OzaBoost | 87,400 | 91,500 | 88,800 | 61,300 |

## 8. CONCLUSION

This paper has presented two ensembles learning algorithms BagEDIST and BoostEDIST, which respectively combine the Online Bagging and the Online Boosting with the drift detection method EDIST. EDIST is a new drift detection method which monitors the distance between two consecutive errors of classification, and tracks concept drifts through two adaptive data-windows $W_G$ and $W_0$. EDIST makes use of a statistical hypothesis test in order to compare $W_G$ and $W_0$ error distance distributions and checks whether the averages differ by more than the adjusted threshold $\varepsilon$.

BagEDIST and BoostEDIST are two ensemble learning algorithms which explicitly handle concept drifts by providing useful description about severity, width and time of change of drift. The originality behind these approaches is that the weighting process of each base classifier depends on its prediction performance and its ability to detect drifts. Moreover, in BoostEDIST, the instances which trigger drifts are given more importance for next base classifiers. During the experiments we have noticed that there is an effective cooperation between base classifiers to optimally cope with concept drifts, in the sense that a classifier's early detection can make up for the delay of another.

This paper has also presented a new Drift Diversity Measure in order to study the diversity of base classifiers and see how they cope with concept drifts. Unlike the existing diversity measures, which are often related to the accuracy of base classifiers, the Drift Diversity Measure takes into consideration the performance of each base classifier and its ability to detect drifts. Through previous experiments, this new measure has helped us to have a clearer vision about the ensemble's behavior when dealing with concept drifts.

## REFERENCES


Agrawal, R., Imielinski, T., & Swami., A. (1993). Database mining: A performance perspective. *IEEE Transactions on Knowledge and Data Engineering*, vol. 6, pp. 914–925.

Baena-García, M., Campo-Avila, J. D., Fidalgo, R., Bifet, A., Gavaldà, R., & Morales-Bueno, R. (2006). Early drift detection method. In *Proceedings of the Fourth International Workshop on Knowledge Discovery from DataStreams*, Berlin, Germany, pp. 77-86.

Bifet, A., & Gavald, R.(2007) Learning from time-changing data with adaptive windowing. In *Proceeding of 7th International Conference on Data Mining*, Minnesota, USA, pp. 443-448.

Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., & Gavalda, R,. (2009). New ensemble methods for evolving data streams. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, Paris, France, pp 139-148.

Bifet, A., Holmes, G., & Pfahringer, B.,(2010) Leveraging bagging for evolving data streams machine learning and knowledge discovery in databases. In *Proceedings of European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, Barcelona, Spain, pp 135-150.

Bifet, A., Holmes, G., Kirkby, R., & Pfahringer, B.(2010). MOA: Massive Online Analysis. *Journal of Machine Learning Research*, vol. 11, pp. 1601-1604.

Brzezinski, D., & Stefanowski, J.,(2011). Accuracy Updated Ensemble for Data Streams with Concept Drift. In *Proceedings of the 6th international conference on Hybrid artificial intelligent systems*, Wroclaw, Poland, pp 155-163.

Cunningham, P., & Carney, J., (2000). Diversity versus Quality in Classification Ensembles based on Feature Selection. In *Proceedings 11th European Conference on Machine Learning.* Barcelona, Spain, pp. 109-116.

Domingos, P., & Hulten G.,(2000). Mining high-speed data streams. In *the Proceedings of the 6th ACM SIGKDD international conference on Knowledge discovery and data mining*, Boston, MA, USA, pp. 71-80.

Gama, J., Medas, P., Castillo, G., & Rodrigues. P.,(2006) Learning with local drift detection. In *Proceedings of the Second International Conference on Advanced Data Mining and Applications*, Xi'an, China. pp. 42-55.

Gama, J., Sebastião, R., & Rodrigues, P.,(2009). Issues in evaluation of stream learning algorithms. In *the Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, Paris, France, pp. 329–338.

Harries, M.,(1999). Splice-2 comparative evaluation: Electricity pricing. Technical Report, The University of South Wales, Autralia.

Hulten, G., Spencer, L., & Domingos, P.,(2001). Mining time-changing data streams. In *Proceedings of the Seventh ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, California, USA, pp. 97-106.

Ikonomovska, E., Gama, J., Sebastio, R., & Gjorgjevik, D.,(2009). Regression trees from data streams with drift detection. In *Proceedings of the 12th International Conference on Discovery Science*, Berlin, Germany, pp. 121–135.

Klinkenberg, R. (2001). Learning drifting concepts: example selection vs. example weighting. *Intelligent Data Analysis*, vol. 8 , pp. 281–300.

Kolter, J., & Maloof, M., (2007). Dynamic weighted majority: a new ensemble method for tracking concept drift. *The Journal of Machine Learning Research*, vol. 8. pp. 2755-2790.

Kuncheva, L., (2004). Classifier Ensembles for Changing Environments. In *Proceedings of the 5th International Workshop on Multiple Classifier Systems*, Cagliari, Italy, pp. 1-15.

Kuncheva, L. I., & Whitaker, C. J, (2003). Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Journal of Machine Learning*, vol. 51, pp. 181–207.

Lazarescu, M., Venkateshand, S.,& Bui, H., (2004). Using multiple windows to track concept drift. *Intelligent data analysis*, vol. 8, pp. 29-59.

Lughofer, E., & Angelov, P.,(2011). Handling Drifts and Shifts in On-Line Data Streams with Evolving Fuzzy Systems. *Applied Soft Computing*, vol. 11, pp. 2057-2068.

Masud, M., Gao, J., Khan, L., Han, J., & Thuraisingham, B., (2011). Classification and novel class detection in concept-drifting data streams under time constraints. *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, pp. 859–874.

Mitchell, T,(1997). *Machine Learning*. McGraw Hill, New York, USA.

Minku, L., White, A., & Yao, X.,(2010). The impact of diversity on online ensemble learning in the presence of concept drift. *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, pp. 730–742.

Oza, N., & Russell, S., (2001). Online bagging and boosting. In *Proceedings of the Eighth International Workshop of Artificial Intelligence and Statistics*, Florida,USA, pp. 105-112.

Sayed-Mouchaweh, M., (2010). Semi-supervised classification method for dynamic applications. *Fuzzy Sets and Systems*, vol. 4, pp. 544–563.

Schlimmer, J. C., & Granger, R. H. (1986). Incremental learning from noisy data. *Journal of Machine Learning*, vol. 3, pp. 317-354.

Sobhani, P., & Beigy, H., (2011). New drift detection method for data streams. In *Proceedings of the second international conference on Adaptive and intelligent systems*, Berlin, Germany, pp. 88-97.

Tsymbal, A., (2004). The problem of concept drift: definitions and related work. Technical Report TCD-CS-2004-15, Trinity College, Dublin, Ireland.

## BIOGRAPHIES

**Imen Khamassi** is a Ph.D. student at the High Institute of Management of the University of Tunis in Tunisia (ISG-Tunis). She received his Master degree in Computer Science and Business Intelligence from (ISG- Tunis) in 2010. She is member in the research laboratory SOIE (Tunisia) and member in the Tunisian Association of Artificial Intelligence (ATIA-Tunisia). Her research interests include combinatorial optimization, learning control, meta-heuristics, intelligent systems and their applications to decision making.

**Moamar Sayed-Mouchaweh** received his Master degree from the University of Technology of Compiegne-France in 1999. Then, he received his PhD degree from the University of Reims-France in December 2002. He was nominated as Associated Professor in Computer Science, Control and Signal processing at the University of Reims-France in the Research center in Sciences and Technology of the Information and the Communication (CReSTIC). In December 2008, he obtained the Habilitation to Direct Researches (HDR) in Computer science, Control and Signal processing. Since September 2011, he is working as a Full Professor in the High National Engineering School of Mines "EcoleNationaleSupérieure des Mines de Douai" at the Department of Automatic Control and Computer Science (Informatique&Automatique IA). He supervised several defended Master and PhD thesis as well as research projects in the field of Modeling, Monitoring and Diagnosis in non-stationary environments. He published more than 100 journal and conference papers. He served as International Program Committee member for several International Conferences as well as a member in IEEE and IFAC technical committees. He also (co-) organized several special sessions and presented several tutorials. He is an associate

Editor of the Springer international journal "Evolving Systems".

**Moez Hammami** has received the M.S. and Ph.D. degrees in Computer Science from the High Institute of Management of the University of Tunis in Tunisia (ISG-Tunis). In 2005, he was nominated as Associated Professor in Computer Science in (ISG- Tunis). He is member in the research laboratory SOIE (Tunisia) and member in the Tunisian Association of Artificial Intelligence (ATIA-Tunisia). His research interests include combinatorial optimization, meta-heuristics, constraint satisfaction problem and multi agent systems.

**Khaled Ghédira** Khaled Ghédira has received the bachelor degree in mathematics (FS-Tunisia); the Engineer degree in hydraulic (ENSEEIHT-France); the specialized Engineer degree in computer science and applied mathematics (ENSIMAG-France), both the Master and the Ph.D. degrees in Artificial Intelligence (ENSAE-France) and finally the Habilitation degree in computer science (ENSI-Tunisia). He was Research Fellow at IIIA (Switzerland 1992-96), expert consultant at BT (England 1995), the head of ENSI (2002-08) and from April 2011, the general managing director of the Tunis Science City. He is also Professor at ISG (University of Tunis), the founding president of the Tunisian Association of Artificial Intelligence (ATIA-Tunisia) and the founding director of the research laboratory SOIE (Tunisia). He is member of several international scientific networks/program committees and is often invited as keynote speaker/ visiting professor at national and international. He was also member of the think national committee for higher education (training of trainers, LMD) and member/president of several committees: evaluation of higher education institutions, research projects reviewing, teachers recruiting. His research areas include multi agent system, transport and production logistics, meta-heuristics and security in M/E-government. He has led several national and international research projects. He has supervised twenty PhD thesis and more than 50 master thesis. He has co/authored about 300 journal/conference/book research papers. He has written two text books on metaheuristics and production logistics and co-authored three others.