DDD: A New Ensemble Approach For Dealing With Concept Drift

Leandro L. Minku, *Student Member, IEEE,* and Xin Yao, *Fellow, IEEE*

Abstract—Online learning algorithms often have to operate in the presence of concept drifts. A recent study revealed that different diversity levels in an ensemble of learning machines are required in order to maintain high generalisation on both old and new concepts. Inspired by this study and based on a further study of diversity with different strategies to deal with drifts, we propose a new online ensemble learning approach called Diversity for Dealing with Drifts (DDD). DDD maintains ensembles with different diversity levels and is able to attain better accuracy than other approaches. Furthermore, it is very robust, outperforming other drift handling approaches in terms of accuracy when there are false positive drift detections. In all the experimental comparisons we have carried out, DDD always performed at least as well as other drift handling approaches under various conditions, with very few exceptions.

Index Terms—Concept drift, online learning, ensembles of learning machines, diversity.

◆

1 INTRODUCTION

ONLINE learning has been showing to be very useful for a growing number of applications in which training data are available continuously in time (streams of data) and/or there are time and space constraints. Examples of such applications are industrial process control, computer security, intelligent user interfaces, market-basket analysis, information filtering and prediction of conditional branch outcomes in microprocessors.

Several definitions of online learning can be found in the literature. In this work, we adopt the definition that online learning algorithms process each training example once "on arrival", without the need for storage or reprocessing [1]. In this way, they take as input a single training example as well as a hypothesis and output an updated hypothesis [2]. We consider online learning as a particular case of incremental learning. The latter term refers to learning machines that are also used to model continuous processes, but process incoming data in chunks, instead of having to process each training example separately [3].

Ensembles of classifiers have been successfully used to improve the accuracy of single classifiers in online and incremental learning [1]–[5]. However, online environments are often non-stationary and the variables to be predicted by the learning machine may change with time (concept drift). For example, in an information filtering system, the users may change their subjects of interest with time. So, learning machines used to model these environments should be able to adapt quickly and accurately to possible changes.

We consider that the term *concept* refers to the whole distribution of the problem in a certain point in time [6], being characterized by the joint distribution $p(\mathbf{x}, w)$,

where $\mathbf{x}$ represents the input attributes and w represents the classes. So, a *concept drift* represents a change in the distribution of the problem [7], [8].

Even though there are some ensemble approaches designed to handle concept drift, only very recently a deeper study of why, when and how ensembles can be helpful for dealing with drifts has been done [9]. The study reveals that different levels of ensemble diversity are required before and after a drift in order to obtain better generalisation on the new concept. However, even though diversity by itself can help to improve generalisation right after the beginning of a drift, it does not provide a faster recovery from drift in long term. So, additional strategies with different levels of diversity are necessary to better handle drifts.

This paper provides an analysis of different strategies to be used with diversity, which are then combined into a new approach to deal with drifts. In all the experimental comparisons we have carried out, the proposed approach always performed at least as well as other drift handling approaches under various conditions, with very few exceptions. Section 2 explains the research questions answered by this paper in more detail.

2 RESEARCH QUESTIONS AND PAPER ORGANIZATION

This paper aims at answering the following research questions, which are not answered by previous works:

1) Online learning often operates in the scenario explained by [10] and further adopted in many works, such as [2], [7], [11] and [12]:

    Learning proceeds in a sequence of trials. In each trial the algorithm receives an *instance* from some fixed *domain* and is to produce a binary *prediction*. At the end of the trial the algorithm receives a binary *label*, which can

L. Minku and X. Yao are with the Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA), School of Computer Science, The University of Birmingham, Edgbaston, Birmingham B15 2TT, UK, e-mail: {L.L.Minku,X.Yao}@cs.bham.ac.uk.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING

2

be viewed as the correct prediction for the instance.

Several real world applications operate in this sort of scenario, such as spam detection, prediction of conditional branches in microprocessors, information filtering, face recognition, etc. Besides, during drifts which take some time to be completed (gradual drifts), the system might be required to make predictions on instances belonging to both the old and new concepts. So, it is important to analyse the prequential accuracy [13], which uses the prediction done at each trial and considers examples of both concepts at the same time when the drift is gradual. The work presented in [9] considers only the generalisation calculated using a test set representing either the old or the new concept. So, how would a low and a high diversity ensemble behave considering the prequential accuracy in the presence of different types of drift?

2) Even though high diversity ensembles may obtain better generalisation than low diversity ensembles after the beginning of a drift, the study presented in [9] also reveals that high diversity ensembles present almost no convergence to the new concept, having slow recovery from drifts. So, is it possible to make a high diversity ensemble trained on the old concept converge to the new concept? How? A more general research question would be: is it possible to use information from the old concept to aid the learning of the new concept? How?

3) If it is possible, would that ensemble outperform a new ensemble created from scratch when the drift begins? For which types of drift?

4) How can we use diversity to improve the prequential accuracy in the presence of drifts, at the same time as we maintain good accuracy in the absence of drifts?

In order to answer the first 3 questions, we perform a study of different strategies with low/high diversity ensembles and analyse their effect on the prequential accuracy in the presence of drifts. The study analyses the ensembles/strategies presented in table 1 using artificial data sets in which we know when a drift begins and what type of drift is present.

Before the drift, a new low diversity and a new high diversity ensemble are created from scratch. After the drift begins, the ensembles created before the drift are kept and referred to as *old* ensembles. The old high diversity ensemble starts learning with low diversity, in order to check if it is possible to converge to the new concept. In addition, a new low and a new high diversity ensemble are created from scratch.

The analysis identifies what ensembles are the most accurate for each type of drift and reveals that high diversity ensembles trained on the old concept are able to converge to the new concept if they start learning the new concept with low diversity. In fact, these ensembles

TABLE 1
Ensembles Analized To Address Research Questions

Before the drift	After the beginning of the drift	Questions addressed
New low diversity →	Old low diversity	1
New high diversity →	Old high diversity now learning with low diversity	1, 2 and 3
	New low diversity	1 and 3
	New high diversity	1

are usually the most accurate for most types of drift. The study uses a technique presented in [9] (and explained here in section 3.2) to explicitly encourage more or less diversity in an ensemble.

In order to answer the last question, we propose a new online ensemble learning approach to handle concept drifts called Diversity for Dealing with Drifts (DDD). The approach aims at better exploiting diversity to handle drifts, being more robust to false alarms (false positive drift detections) and having faster recovery from drifts. In this way, it manages to achieve improved accuracy in the presence of drifts at the same time as good accuracy in the absence of drifts is maintained. Experiments with artificial and real world data show that DDD usually obtains similar or better accuracy than EDDM and better accuracy than DWM.

This paper is further organized as follows. Section 3 explains related work. Section 4 explains the data sets used in the study. Section 5 presents the study of the effect of low/high diversity ensembles on the prequential accuracy in the presence of drifts using the strategies presented in table 1. Section 6 introduces and analyses DDD. Section 7 concludes the paper and gives directions for further research.

3 RELATED WORK

Several approaches proposed to handle concept drift can be found in the literature. Most of them are incremental learning approaches [8], [14]–[21]. However, most of the incremental learning approaches tend to give little attention to the stability of the classifiers, giving more emphasis to the plasticity when they allow only a new classifier to learn a new chunk of data. While this could be desirable when drifts are very frequent, it is not a good strategy when drifts are not so frequent. Besides, determining the chunk size in the presence of concept drifts is not a straightforward task. A too small chunk size will not provide enough data for a new classifier to be accurate, whereas a too large chunk size may contain data belonging to different concepts, making the adaptation to new concepts slow.

The online learning algorithms which handle concept drift can be divided into two groups: approaches which use a mechanism to detect drifts [7], [12], [22]–[24] and approaches which do not explicitly detect drifts [25]–[27]. Both of them handle drifts based directly or indirectly on the accuracy of the current classifiers. The

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING

3

former approaches use some measure related to the accuracy to detect drifts. They usually discard the current system and create a new one after a drift is detected and/or confirmed. In this way, they can have quick response to drifts, as long as the drifts are detected early. However, these approaches can suffer from non accurate drift detections. The latter approaches usually associate weights to each ensemble's classifier based on its accuracy, possibly allowing pruning and addition of new classifiers. These approaches need some time for the weights to start reflecting the new concept.

Section 3.1 presents an example of drift detection method and two approaches to handle drifts: an approach based on a drift detection method and an approach which does not detect drifts explicitly. Section 3.2 briefly explains ensemble diversity in the context of online learning in the presence of concept drift.

3.1 Approaches to Detect and/or Handle Drifts

An example of drift detection method is the one used by the approach presented in [12]. It is based on the idea that the distance between two consecutive errors increases when a stable concept is being learnt. So, the distance is monitored and, if it reduces considerably according to a pre-defined constant which we call γ in this paper, it is considered that there is a concept drift.

The approach presented to handle concept drift in [12] is called Early Drift Detection Method (EDDM). It uses the drift detection method explained above with two values for γ: α and β, $\alpha > \beta$. When a concept drift is alarmed by the drift detection method using α, but not β, a warning level is triggered, indicating that a concept drift might have happened. From this moment, all the training examples presented to the system are used for learning and then stored. If a concept drift is alarmed using β, the concept drift is confirmed and the system is reset. If we consider that a new online classifier system is created when the warning level is triggered, instead of storing the training instances for posterior use, EDDM is considered a true online learning system.

An example of a well cited approach which does not use a drift detection method is Dynamic Weighted Majority (DWM) [26]. It maintains an ensemble of classifiers whose weights are reduced by a multiplier constant ρ,[1] $\rho < 1$, when the classifier gives a wrong prediction, if the current time step is a multiple of p. The approach also allows the addition and removal of classifiers at every p time steps. The removal is controlled by a pre-defined weight threshold θ. In this way, new classifiers can be created to learn new concepts and poorly performing classifiers, which possibly learnt old concepts, can be removed.

A summary of the parameters used by the drift detection method, EDDM and DWM is given in table 2.

---

1. The original greek letter to refer to this parameter was β. We changed it to ρ to avoid confusion with EDDM's parameter β.

3.2 Ensemble Diversity in the Presence of Drifts

Even though several studies of ensemble diversity can be found in the literature, e.g., [28]–[30], the study presented in [9] is the first diversity analysis regarding online learning in the presence of concept drift.

The study shows that different diversity levels are required before and after a drift in order to improve generalisation on the old or new concepts and concludes that diversity by itself can help to reduce the initial increase in error caused by a drift, but does not provide a faster recovery from drifts in long term. So, the potential of ensembles for dealing with concept drift may have not been fully exploited by the existing ensemble approaches yet, as they do not encourage different levels of diversity in different situations.

Intuitively speaking, the key to the success of an ensemble of classifiers is that the base classifiers perform diversely. Despite the popularity of the term diversity, there is no single definition or measure of it [30]. A popular measure is Yule's Q statistic [31]. Based on an analysis of ten measures, Q statistic is recommended by Kuncheva and Whitaker [29] to be used for the purpose of minimizing the error of ensembles. This measure is recommended especially due to its simplicity and ease of interpretation. Considering two classifiers D_i and D_k, the Q statistic can be calculated as:

$$Q_{i,k} = \frac{N^{11}N^{00} - N^{01}N^{10}}{N^{11}N^{00} + N^{01}N^{10}}$$

where $N^{a,b}$ is the number of training examples for which the classification given by D_i is a and the classification given by D_k is b, 1 represents a correct classification and 0 represents a misclassification.

Classifiers which tend to classify the same examples correctly will have positive values of Q, whereas classifiers which tend to classify different examples incorrectly will have negative values of Q. For an ensemble of classifiers, the averaged Q statistic over all pairs of classifiers can be used as a measure of diversity. Higher/lower average indicates less/more diversity. In this paper, we will consider that low/high diversity refers to high/low average Q statistic.

In online learning, an example of how to explicitly encourage more or less diversity in an ensemble is by using a modified version [9] of online bagging [1]. The original online bagging (algorithm 1) is based on the fact that, when the number of training examples tends to infinite in offline bagging, each base learner h_m contains K copies of each original training example d, where the distribution of K tends to a $Poisson(1)$ distribution. So, in online bagging, whenever a training example is available, it is presented K times for each base learner h_m, where K is drawn from a $Poisson(1)$ distribution. The classification is done by unweighted majority vote, as in offline bagging.

In order to encourage different levels of diversity, algorithm 1 can be modified to include a parameter λ for the $Poisson(\lambda)$ distribution, instead of enforcing

TABLE 2
Parameters Description.

Approach	Parameter	Description
Drift detection method	γ	Used to check whether the distance between two consecutive errors increased sufficiently.
EDDM	α	Value for γ which determines whether the warning level is triggered.
	$\beta, \alpha > \beta$	Value for γ which determines whether a drift is considered to be detected.
DWM	$\rho, \rho < 1$	Multiplier constant for base learner's weight decrease.
	p	Interval of time steps in which the system can decrease weights, add or remove learners.
	θ	Threshold for removing base learners based on their weights.
Modified Online Bagging	λ	Parameter used by $Poisson$ to encourage more or less diversity.
DDD	p_l, e.g., $p_l = \lambda_l$	Parameter for encouraging low diversity in the underlying ensemble learning algorithm.
	p_h, e.g., $p_h = \lambda_h$	Parameter for encouraging high diversity in the underlying ensemble learning algorithm.
	W (default 1)	Multiplier constant for the weight of the old low diversity ensemble.
	p_d, e.g., $p_d = \gamma$	Parameter for the drift detection method to be used with the approach.

Algorithm 1 Online Bagging

Inputs: ensemble $\mathbf{h}$; ensemble size M; training example d; and online learning algorithm for the ensemble members $OnlineBaseLearningAlg$;

1: **for** $m \leftarrow 1$ to M **do**
2:    $K \leftarrow Poisson(1)$
3:    **while** $K > 0$ **do**
4:       $h_m \leftarrow OnlineBaseLearningAlg(h_m, d)$
5:       $K \leftarrow K - 1$
6:    **end while**
7: **end for**
Output: updated ensemble $\mathbf{h}$

$\lambda = 1$. In this way, higher/lower λ values are associated to higher/lower average Q statistic (lower/higher diversity), as shown in section 5.4.1 of [9]. This parameter is listed in table 2.

4 DATA SETS

When working with real world data sets, it is not possible to know exactly when a drift starts to occur, which type of drift is present or even if there really is a drift. So, it is not possible to perform a detailed analysis of the behaviour of algorithms in the presence of concept drift using only pure real world data sets. In order to analyse the effect of low/high diversity ensembles in the presence of concept drift and to assist the analysis of DDD, we first used the artificial data sets described in [9]. Then, in order to reaffirm the analysis of DDD, we performed experiments using three real world problems. Section 4.1 describes the artificial data sets and section 4.2 describes the real world data sets.

4.1 Artificial Data Sets

The artificial data sets used in the experiments (table 3) comprise the following problems [9]: circle, sine moving vertically, sine moving horizontally, line, plane and Boolean. In the equations presented in the table, a, b, c, d, r, a_i, eq and op can assume different values to define different concepts. The examples of all problems but Boolean contain x/x_i and y as the input attributes

TABLE 3
Artificial Data Sets

Probl.	Equation	Fixed Values	Before→After Drift	Sev.
Circle	$(x-a)^2 +$ $(y-b)^2 \leq$ r^2	$a = .5$ $b = .5$	$r = .2 \rightarrow .3$	16%
			$r = .2 \rightarrow .4$	38%
			$r = .2 \rightarrow .5$	66%
SineV	$y \leq$ $a \sin(bx + c) +$ d	$a = 1$ $b = 1$ $c = 0$	$d = -2 \rightarrow 1$	15%
			$d = -5 \rightarrow 4$	45%
			$d = -8 \rightarrow 7$	75%
SineH	$y \leq$ $a \sin(bx + c) +$ d	$a = 5$ $d = 5$ $b = 1$	$c = 0 \rightarrow -\pi/4$	36%
			$c = 0 \rightarrow -\pi/2$	57%
			$c = 0 \rightarrow -\pi$	80%
Line	$y \leq -a_0 +$ $a_1 x_1$	$a_1 = .1$	$a_0 = -.4 \rightarrow -.55$	15%
			$a_0 = -.25 \rightarrow -.7$	45%
			$a_0 = -.1 \rightarrow -.8$	70%
Plane	$y \leq -a_0 +$ $a_1 x_1 +$ $a_2 x_2$	$a_1 = .1$ $a_2 = .1$	$a_0 = -2 \rightarrow -2.7$	14%
			$a_0 = -1 \rightarrow -3.2$	44%
			$a_0 = -.7 \rightarrow -4.4$	74%
Bool	$(color\ eq_1\ a$ op_1 $shape\ eq_2\ b)$ op_2 $size\ eq_3\ c$	$c =$ $S \vee M \vee L$ $op_2\ \wedge$ $eq_{1,2,3} =$	$a = R, op_1\ \wedge$ $b = R \rightarrow R \vee T$	11%
			$a = R, b = R,$ $op_1\ \wedge \rightarrow \vee$	44%
			$a = R \rightarrow R \vee G,$ $b = R \rightarrow R \vee T,$ $op_1 \wedge \rightarrow \vee$	67%

and the concept (which can assume value 0 or 1) as the output attribute.

The Boolean problem is inspired by the STAGGER problem [32], but it allows the generation of different drifts, with different levels of severity and speed. In this problem, each training example has three input attributes: color (red R, green G or blue B), shape (triangle T, rectangle R or circle C) and size (small S, medium M or large L). The concept is then represented by the Boolean equation given in table 3, which indicates the color, shape and size of the objects which belong to class 1 (true). In that expression, a represents a conjunction or disjunction of different possible colors, b represents shapes, c represents sizes, eq represents $=$ or $\neq$ and op represents the logical connective $\wedge$ or $\vee$. For example, the first concept of the Boolean data set which presents a drift with 11% of severity in table 3 is represented by:

$$(color = R\ \wedge shape = R) \wedge size = S \vee M \vee L .$$

Each data set contains one drift and different drifts were simulated by varying among three amounts of

severity (as shown in table 3) and three speeds, thus generating nine different drifts for each problem. Severity represents the amount of changes caused by a new concept. Here, the measure of severity is the percentage of the input space which has its target class changed after the drift is complete. Speed is the inverse of the time taken for a new concept to completely replace the old one. The speed was measured by the inverse of the number of time steps taken for a new concept to completely replace the old one and was modelled by the following linear degree of dominance functions:

$$v_n(t) = \frac{t - N}{drifting\_time}, \ N < t \leq N + drifting\_time$$

and

$$v_o(t) = 1 - v_n(t), \ N < t \leq N + drifting\_time \ ,$$

where $v_n(t)$ and $v_o(t)$ are the degrees of dominance of the new and old concepts, respectively; t is the current time step; N is the number of time steps before the drift started to occur; and $drifting\_time$ varied among 1, $0.25N$ and $0.50N$ time steps. During the drifting time, the degrees of dominance represent the probability that an example of the old or the new concept will be presented to the system. For a detailed explanation of different types of drift, we recommend [9].

The data sets are composed of $2N$ examples and each example corresponds to 1 time step of the learning. The first N examples of the training sets were generated according to the old concept ($v_o(t) = 1$, $1 \leq t \leq N$), where $N = 1000$ for circle, sineV, sineH and line and $N = 500$ for plane and Boolean. The next $drifting\_time$ training examples ($N < t \leq N + drifting\_time$) were generated according to the degree of dominance functions, $v_n(t)$ and $v_o(t)$. The remaining examples were generated according to the new concept ($v_n(t) = 1$, $N + drifting\_time < t \leq 2N$).

The range of x or x_i was $[0, 1]$ for circle, line and plane; $[0, 10]$ for sineV; and $[0, 4\pi]$ for sineH. The range of y was $[0, 1]$ for circle and line, $[-10, 10]$ for sineV, $[0, 10]$ for sineH and $[0, 5]$ for plane. For plane and Boolean, the input attributes are normally distributed through the whole input space. For the other problems, the number of instances belonging to class 1 and 0 is always the same, having the effect of changing the unconditional probability distribution function when the drift occurs. Eight irrelevant attributes and 10% class noise were introduced in the plane data sets.

4.2 Real World Data Sets

The real world data sets used in the experiments with DDD are: electricity market [7], KDD Cup 1999 network intrusion detection data [33] and PAKDD 2009 credit card data.

The electricity data set is a real world data set from the Australian New South Wales Electricity Market. This data set was first described in [34]. In this electricity market, the prices are not fixed and may be affected by demand and supply. Besides, during the time period described in the data, the electricity market was expanded with the inclusion of adjacent areas, causing a dampening of the extreme prices. This data set contains 45,312 examples, dated from May 1996 to December 1998. Each example contains 4 input attributes (time stamp, day of the week and 2 electricity demand values) and the target class, which identifies the change of the price related to a moving average of the last 24 hours.

The KDD Cup 1999 data set is a network intrusion detection data set. The task of an intrusion detection system is to distinguish between intrusions/attacks and normal connections. The data set contains a wide variety of intrusions simulated in a military network environment. During the simulation, the local-area network was operated as if it were a true Air Force environment, but peppered with multiple attacks, so that attack is not a minority class. The data set contains 494,020 examples. Each example corresponds to a connection and contains 41 input attributes, such as the length of the connection, the type of protocol, the network service on the destination, etc. The target class identifies whether the connection is an attack or a normal connection.

The PAKDD 2009 data set comprises data collected from the private label credit card operation of a major Brazilian retail chain, along stable inflation condition. We used the modelling data, which contains 50,000 examples and corresponds to a 1 year period. Each example corresponds to a client and contains 27 input attributes, such as sex, age, marital status, profession, income, etc. The target class identifies if the client is a "good" or "bad" client. The class "bad" is a minority class and composes around 19.5% of the data.

5 THE EFFECT OF LOW/HIGH DIVERSITY ENSEMBLES ON THE PREQUENTIAL ACCURACY

This section presents an analysis of the prequential accuracy of the ensembles/strategies presented in table 1, aiming at answering the first three research questions explained in section 2. The prequential [13] accuracy is the average accuracy obtained by the prediction of each example to be learnt, before its learning, calculated in an online way. The rule used to obtain the prequential accuracy on time step t is presented in equation 1 [12]:

$$acc(t) = \begin{cases} acc_{ex}(t) \ , & \text{if } t = f \\ acc(t-1) + \frac{acc_{ex}(t) - acc(t-1)}{t - f + 1} \ , & \text{otherwise} \end{cases}$$

(1)

where acc_{ex} is 0 if the prediction of the current training example ex before its learning is wrong and 1 if it is correct; and f is the first time step used in the calculation.

In order to analyse the behaviour of the ensembles before and after the beginning of a drift, the prequential accuracy shown in the graphs is reset whenever a drift starts ($f \in \{1, N + 1\}$). The learning of each ensemble is repeated 30 times for each data set.

The online ensemble learning algorithm used in the experiments is the modified version of online bagging proposed in [9] and explained in section 3.2. As commented in that section, [9] shows that higher/lower λs produce ensembles with higher/lower average Q statistic (lower/higher diversity).

Section 5.1 presents the analysis itself.

5.1 Experimental Results and Analysis

The experiments used 25 lossless ITI online decision trees [35] as the base learners for each ensemble. The parameter λ_l for the $Poisson(\lambda)$ distribution of the low diversity ensembles is 1, which is the value used for the original online bagging [1]. The λ_h values for the high diversity ensembles were chosen in the following way:

1) Perform 5 preliminary executions using $\lambda_h = 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5$, giving a total of 35 executions for each data set.
2) Determine the prequential accuracy obtained by each execution at the time step $1.1N$. This time step represents the moment in which high diversity ensembles are more likely to outperform low diversity ensembles, according to [9].
3) Calculate the main effect of λ_h on the average prequential accuracy, considering all the data sets for each particular problem at the same time. For example, the main effect for the circle problem is the average among the online accuracies obtained by the 5 executions using all the 9 circle data sets.
4) For each problem, choose the λ_h which obtained the best main effect. These values are 0.05 for circle, sineH and plane, and 0.005 for line and sineV. For Boolean, both $\lambda_h = 0.1$ and $\lambda_h = 0.5$ obtained the same main effect. So, we chose $\lambda_h = 0.1$, as it obtained the best main effect at $1.5N$ and $2N$.

Figure 1 shows the prequential accuracy obtained for the circle problem for low and high severities and speeds. The figures for the other data sets were omitted due to space limitations. In order to answer the first research question, we analyse the prequential accuracy **after** the beginning of the drift, for each data set. We can observe that different ensembles obtain the best prequential accuracy depending on the type of drift.

For drifts with low severity and high speed (e.g., figure 1(a)), the best accuracy after the beginning of the drift is usually obtained by the old high diversity ensemble. For the Boolean problem, the old low diversity gets similar accuracy to the old high diversity ensemble. So, in general, **it is a good strategy to use the old high diversity ensemble for this type of drift**.

An intuitive explanation for the reason why old high diversity ensembles are helpful for low severity drifts is that, even though the new concept is not the same as the old concept, it is similar to the old concept. When a high level of diversity is enforced, the base learners are forced to classify the training examples very differently from each other. So, the ensemble learns a certain concept

only partly, being able to converge to the new concept by learning it with low diversity. At the same time, as the old concept was partly learnt, the old high diversity ensemble can use information learnt from the old concept to aid the learning of the new concept. An old low diversity ensemble would provide information from the old concept, but would have problems to converge to the new concept [9].

For drifts with high severity and high speed (e.g., figure 1(b)), the new low diversity ensemble usually obtains the best accuracy, even though that accuracy is similar to the old high diversity ensemble's in half of the cases (Boolean, sineV and line). For the Boolean problem, the old high diversity, old low diversity and new low diversity obtain similar accuracy. So, in general, **it is a good strategy to use the new low diversity ensemble for this type of drift**.

The reason for that is that, when a drift has high severity and high speed, it causes big changes very suddenly. In this case, the new concept has almost no similarities to the old concept. So, an ensemble which learnt the old concept either partly or fully will not be so helpful (and could be even harmful) for the accuracy on the new concept. A new ensemble learning from scratch is thus the best option.

For drifts with medium severity and high speed, the behaviour of the ensembles is similar to when the severity is high for sineH, circle, plane and line, although the difference between the old high diversity and the new low diversity ensemble tends to be smaller for sineH, circle and plane. The behaviour for Boolean and sineV tends to be more similar to when severity is low. So, drifts with medium severity sometimes have similar behaviour to low severity and sometimes to high severity drifts when the speed is high.

For drifts with low speed (e.g., figures 1(c) and 1(d)), either the old low or both the old ensembles present the best accuracy in the beginning of the drift, independent of the severity. So, considering only shortly after the beginning of a drift, the best strategy is to use the old low diversity ensemble for slow speed drifts. Longer after the drift, either the old high or both the old high and the new low diversity ensembles usually obtain the best accuracies. So, considering only longer after the drift, the best strategy is to use the old high diversity ensemble. If we consider both shortly and longer after the drift, **it is a good strategy to use the old high diversity ensemble**, as it is the most likely to have good accuracy during the whole period after the beginning of the drift.

For drifts with medium speed, the behaviour is similar to low speed, although the period of time in which the old ensembles have the best accuracies is reduced and the old low diversity ensemble rarely has the best accuracy by itself shortly after the beginning of the drift (it usually obtains similar accuracy to the old high diversity ensemble). When the severity is high, there are two cases (sineH and plane) in which the best accuracy is obtained by the new low diversity ensemble longer after

(a) Low Sev, High Sp   (b) High Sev, High Sp   (c) Low Sev, Low Sp   (d) High Sev, Low Sp
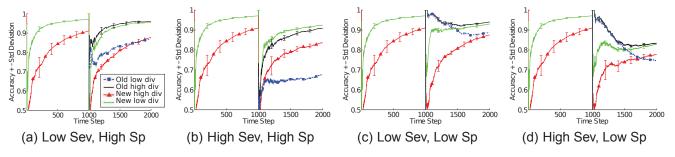
Fig. 1. Average prequential accuracy (equation 1) of the four ensembles analysed for the circle problem considering 30 runs using "perfect" drift detections. The accuracy is reset when the drift starts ($f \in \{1, 1001\}$). The new ensembles are created from scratch at the time steps 1 and 1001. The old ensembles correspond to the new ensembles before the beginning of the drift.

the drift. This behaviour approximates the behaviour obtained when the severity is high and the speed is high.

This analysis is a straight answer to the first research question and also allows us to answer the second and third questions: (2) Ensembles which learnt an old concept using high diversity can converge to a new concept if they start learning the new concept with low diversity. (3) When the drift has low severity and high speed or longer after the drift when the speed is low or medium, the high diversity ensembles learning with low diversity are the most accurate in most of the cases. Besides, when the speed is low or medium, shortly after the beginning of the drift, they are more accurate than the new ensembles and frequently have similar accuracy to the old low diversity ensembles. Even when the drift has medium or high severity and high speed, the old high diversity ensembles sometimes obtain similar accuracy to the new low diversity ensembles. So, in fact, it is a good strategy to use the old high diversity ensembles for most types of drift.

The analysis shows that the strategy of resetting the learning system as soon as a drift is detected, which is adopted by many approaches, such as [7], [12], [23], is not always ideal, as an ensemble which learnt the old concept can be helpful depending on the drift type.

6 DIVERSITY FOR DEALING WITH DRIFTS

This section proposes Diversity for Dealing with Drifts (DDD).[2] Section 6.1 describes DDD. Section 6.2 and 6.3 explain the experiments done to validate DDD and provide an answer to the last research question from section 2.

6.1 DDD's Description

DDD (algorithm 2) operates in 2 modes: prior to drift detection and after drift detection. We chose to use a drift detection method, instead of treating drifts implicitly, because it allows immediate treatment of drifts once they are detected. So, if the parameters of the drift detection method are tuned to detect drifts the earliest possible

2. An initial version of DDD can be found in [36].

and the approach is designed to be robust to false alarms, we can obtain fast adaptation to new concepts. The parameters used by DDD are summarized in table 2.

Algorithm 2 DDD

Inputs:

- multiplier constant W for the weight of the old low diversity ensemble;
- online ensemble learning algorithm $EnsembleLearning$;
- parameters for ensemble learning with low diversity p_l and high diversity p_h;
- drift detection method $DetectDrift$;
- parameters for drift detection method p_d;
- data stream D;

1: mode ← before drift
2: h_{nl} ← new ensemble /* new low diversity */
3: h_{nh} ← new ensemble /* new high diversity */
4: h_{ol} ← h_{oh} ← null /* old low and high diversity */
5: acc_{ol} ← acc_{oh} ← acc_{nl} ← acc_{nh} ← 0 /* accuracies */
6: std_{ol} ← std_{oh} ← std_{nl} ← std_{nh} ← 0 /* standard deviations */
7: **while** true **do**
8:    d ← next example from D
9:    **if** mode == before drift **then**
10:        prediction ← $h_{nl}(d)$
11:    **else**
12:        sum_{acc} ← $acc_{nl} + acc_{ol} * W + acc_{oh}$
13:        $w_{nl} = acc_{nl}/sum_{acc}$
14:        $w_{ol} = acc_{ol} * W/sum_{acc}$
15:        $w_{oh} = acc_{oh}/sum_{acc}$
16:        prediction ← WeightedMajority($h_{nl}(d)$, $h_{ol}(d)$, $h_{oh}(d)$, w_{nl}, w_{ol}, w_{oh})
17:        Update(acc_{nl}, std_{nl}, h_{nl}, d)
18:        Update(acc_{ol}, std_{ol}, h_{ol}, d)
19:        Update(acc_{oh}, std_{oh}, h_{oh}, d)
20:    **end if**
21:    drift ← DetectDrift(h_{nl}, d, p_d)
22:    **if** drift == true **then**
23:        **if** mode == before drift OR (mode == after drift AND $acc_{nl} > acc_{oh}$) **then**
24:            h_{ol} ← h_{nl}

```
25:        else
26:            h_ol ← h_oh
27:        end if
28:        h_oh ← h_nh
29:        h_nl ← new ensemble
30:        h_nh ← new ensemble
31:        acc_ol ← acc_oh ← acc_nl ← acc_nh ← 0
32:        std_ol ← std_oh ← std_nl ← std_nh ← 0
33:        mode ← after drift
34:    end if
35:    if mode == after drift then
36:        if acc_nl > acc_oh AND acc_nl > acc_ol then
37:            mode ← before drift
38:        else
39:            if acc_oh − std_oh > acc_nl + std_nl AND acc_oh −
                std_oh > acc_ol + std_ol then
40:                h_nl ← h_oh
41:                acc_nl ← acc_oh
42:                mode ← before drift
43:            end if
44:        end if
45:    end if
46:    EnsembleLearning(h_nl, d, p_l)
47:    EnsembleLearning(h_nh, d, p_h)
48:    if mode == after drift then
49:        EnsembleLearning(h_ol, d, p_l)
50:        EnsembleLearning(h_oh, d, p_l)
51:    end if
52:    if mode == before drift then
53:        Output h_nl, prediction
54:    else
55:        Output h_nl, h_ol, h_oh, w_nl, w_ol, w_oh, prediction
56:    end if
57: end while
```

Before a drift is detected, the learning system is composed of two ensembles: an ensemble with lower diversity (h_{nl}) and an ensemble with higher diversity (h_{nh}). Both ensembles are trained with incoming examples (lines 46 and 47), but only the low diversity ensemble is used for system predictions (line 10). The reason for not using the high diversity ensemble for predictions is that it is likely to be less accurate on the new concept being learnt than the low diversity ensemble [9]. DDD assumes that, if there is no convergence of the underlying distributions to a stable concept, new drift detections will occur, triggering the mode after drift detection. DDD then allows the use of the high diversity ensemble in the form of an old high diversity ensemble, as explained later in this section.

A drift detection method based on monitoring the low diversity ensemble is used (line 21). This method can be any of the methods existing in the literature, for instance, the one explained in section 3.1. After a drift is detected, new low diversity and high diversity ensembles are created (lines 29 and 30). The ensembles corresponding to the low and high diversity ensembles

before the drift detection are kept and denominated old low and old high diversity ensembles (lines 24 and 28). The old high diversity ensemble starts to learn with low diversity (line 50) in order to improve its convergence to the new concept, as explained in section 5. Maintaining the old ensembles allows not only a better exploitation of diversity and the use of information learnt from the old concept to aid the learning of the new concept, but also helps the approach to be robust to false alarms.

Both the old and the new ensembles perform learning (lines 46-50) and the system predictions are determined by the weighted majority vote of the output of (1) the old high diversity, (2) the new low diversity and (3) the old low diversity ensemble (lines 12-16). The new high diversity ensemble is not considered because it is likely to have low accuracy on the new concept [9].

The weights are proportional to the prequential accuracy since the last drift detection until the previous time step (lines 13-15). The weight of the old low diversity ensemble is multiplied by a constant W (line 15), which allows controlling the trade-off between robustness to false alarms and accuracy in the presence of concept drifts, and all the weights are normalized. It is important to note that weighting the ensembles based on the accuracy after the drift detection is different from the weighting strategy adopted by the approaches in the literature which do not use a drift detection method. Those approaches use weights which are likely to represent more than one concept at the same time and need some time to start reflecting only the new concept.

During the mode after drift detection, the new low diversity ensemble is monitored by the drift detection method (line 21). If two consecutive drift detections happen and there is no shift back to the mode prior to drift detection between them, the old low diversity ensemble after the second drift detection can be either the same as the old high diversity learning with low diversity after the first drift detection or the ensemble corresponding to the new low diversity after the first drift detection, depending on which of them is the most accurate (lines 24 and 26). The reason for that is that, soon after the first drift detection, the new low diversity ensemble may be not accurate enough to become the old low diversity ensemble. This strategy also helps the approach to be more robust to false alarms.

All the four ensembles are maintained in the system until either the condition in line 36 or the condition in line 39 is satisfied. When one of these conditions is satisfied, the system returns to the mode prior to drift detection. The accuracies when considering whether the old high diversity ensemble is better than the others are reduced/summed to their standard deviations to avoid premature return to the mode prior to drift, as this ensemble is more likely to have higher accuracy than the new low diversity very soon after the drift, when this ensemble learnt just a few examples.

When returning to the mode prior to drift, either the old high diversity or the new low diversity ensemble

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING

9

becomes the low diversity ensemble used in the mode prior to drift detection, depending on which of them is the most accurate (lines 36-44). An additional parameter to control the maximum number of time steps in the mode after drift detection can be used to avoid a too long time maintaining four ensembles and is proposed as future work.

As we can see, DDD is designed to better exploit the advantages of diversity to deal with concept drift than the other approaches in the literature, by maintaining ensembles with different diversity levels, according to the experiments presented in section 5.

Besides, the approaches which use old classifiers as part of an ensemble in the literature, such as [25]–[27], do not adopt any strategy to improve their learning on the new concept. Nevertheless, as it was shown in [9], these classifiers are likely to have low accuracy on the new concept if no additional strategy is adopted to improve their learning. DDD is designed to use information learnt from the old concept in order to aid the learning of the new concept, by training an old high diversity ensemble with low diversity on the new concept. Such a strategy has shown to be successful in section 5.

Moreover, the approach is designed to be more robust to false alarms than approaches which reset the system when a drift is detected [7], [12], [23] and to have faster recovery from drifts than approaches which do not use a drift detection method [26], [27], as it maintains old ensembles after a drift detection, but takes immediate action to treat the drift when it is detected, instead of having to wait for the weights to start reflecting the new concept.

The approach is not yet prepared to take advantage of recurrent or predictable drifts. We propose the use of memories for dealing with these types of drift as future work.

A detailed analysis of time and memory occupied by DDD is not straightforward, as it depends on the implementation, base learner and source of diversity. However, it is easy to see that, if we have a sequential implementation, the complexity of each ensemble is $O(f(n))$ and the source of diversity does not influence this complexity, DDD would have, in the worst case, complexity $O(4f(n)) = O(f(n))$. So, DDD does not increase the complexity of the system in comparison to a single ensemble.

6.2 Experimental Objectives, Design and Measures Analysed

The objective of the experiments with DDD is to assist its analysis and to validate it, showing that it is an answer to the last research question presented in section 2, i.e., it obtains good accuracy both in the presence and absence of drifts. We also aim at identifying for which types of drift DDD works better and why it behaves in that way.

In order to do so, we analyse measures such as weights attributed by DDD to each ensemble, number of drift detections and prequential accuracy (equation 1, from section 5). In some cases, the false positive and negative rate is also analysed. DDD is compared to a low diversity ensemble with no drift handling abilities, EDDM [12] and DWM [26]. The diversity study using "perfect" drift detections presented in section 5.1 and an approach which would always choose to use the same ensemble (i.e., always choose the old high diversity ensemble, or always choose the old low diversity ensemble, etc) are also used in the analysis.

The prequential accuracy is calculated based on the predictions given to the current training example before the example is used for updating any component of the system. It is important to observe that the term *update* here refers not only to the learning of the current training example by the base learners, but also to the changes in weights associated to the base learners (in the case of DWM) and to the ensembles (in the case of DDD). The prequential accuracy is compared both visually, considering the graphs of the average prequential accuracy and standard deviation throughout the learning, and using T student statistical tests [37] at particular time steps, for the artificial data sets.

The T tests were performed as follows. For each artificial problem, T student statistical tests were done at the time steps $0.99N$, $1.1N$, $1.5N$ and $2N$. These time steps were chosen in order to analyse the behaviour soon after the drift, fairly longer after the drift and long after the drift, as in [9]. Bonferroni corrections considering all the combinations of severity, speed and time step were used. The overall significance level was 0.01, so that the significance level used for each individual comparison was $\alpha = 0.01/(3*3*4) = 0.000278$.

The drift detection method used by DDD was the same as the one used by EDDM, in order to provide a fair comparison. There is the possibility that there are other drift detection methods more accurate in the literature. However, the study presented in section 5 shows that the old ensembles are particularly useful right after the beginning of the drift. So, a comparison to an approach using a drift detection method which could detect drifts earlier would give more advantages to DDD.

Thirty repetitions were done for each data set and approach. The ensemble learning algorithm used by DDD and EDDM was the (modified) online bagging, as in section 5. The drift detection method used by DDD in the experiments was the one explained in section 3.1.

The analysis of DDD and its conclusions are based on the following:

1) EDDM always uses new learners created from scratch. Nevertheless, resetting the system upon drift detections is not always the best choice. DWM allows us to use old classifiers, but does not use any strategy to help the convergence of these classifiers to the new concept. So, it cannot use information from the old concept to learn the new concept faster, as DDD does. At least in the situations in which new learners created from scratch or old

learners which attempted to learn the old concept well are not the best choice, DDD will perform better if it manages to identify these situations and adjust the weights of the ensembles properly. This is independent of the base learner.

2) The diversity study presented in section 5 shows that such situations exist and that an old high diversity ensemble, which is used by DDD, is benefic for several different types of drift. In section 6.3.1, we show using artificial data and decision trees that DDD is able to identify such situations and adjust the ensemble weights properly, being accurate in the presence and absence of drifts. We also identify the drifts for which DDD works better and explain why. Section 6.3.2 analyses the number of time steps in which DDD maintains four ensembles. Section 6.3.3 shows that DDD is robust to false alarms and explains the influence of the parameter W. Additional experiments using naive bayes and multi-layer perceptrons on real world problems (section 6.3.4) further confirm the analysis done with the artificial data.

6.3 Experimental Results and Analysis

6.3.1 Experiments With Artificial Data

The sets of parameters considered for the experiments are shown in table 4. The parameters λ_h were chosen to be the same as in section 5.1. The parameter γ (β for EDDM) was chosen so as to provide early drift detections. This parameter was the same for DDD and EDDM, providing a fair comparison.

Preliminary experiments with 5 runs for each data set show that α does not influence much EDDM's accuracy. Even though an increase in α is associated to increases in the total number of time steps in warning level, the system is usually in warning state for very few consecutive time steps before the drift level is triggered, even when very large α is adopted. That number is not enough to significantly affect the accuracy. So, we decided to use $\alpha = 0.99$.

Similarly to λ_h, the parameter p of DWM was chosen considering the best average accuracy at the time step $1.1N$, using the default values of $\rho = 0.5$ and $\theta = 0.01$. The average was calculated using 5 runs for all the data sets of a particular problem at the same time, as for the choice of λ_h. After selecting p, a fine tuning was done for DWM, again based on 5 preliminary runs, by selecting the parameter ρ which provides the best main effect on the prequential accuracy at the time step 1.1N when using the best p. The execution time using $\rho = 0.7$ and 0.9 became extremely high, especially for Plane and Circle. The reason for that is that a combination of a low p with a high ρ causes DWM to include new base learners very frequently, whereas the weights associated to each base learner reduce slowly. So, we did not consider these values for these problems.

The base learners used in the experiments were lossless ITI online decision trees [35] and both DDD and EDDM used ensemble size of 25 ITIs. DWM automatically selects the ensemble size.

Figure 2 shows the prequential accuracy and figure 3 shows the weights attributed by DDD for some representative data sets. Graphs for other data sets were omitted due to space restrictions.

We first compare DDD's prequential accuracy to EDDM's. During the first concept, DDD is equivalent to EDDM if there are no false alarms. Otherwise, DDD has better accuracy than EDDM. This behaviour is expected, as EDDM resets the system when there is a false alarm, having to re-learn the current concept. DDD, on the other hand, can make use of the old ensembles by increasing their weights. Figure 3 shows that indeed DDD increases the weights of the old ensembles when there are false alarms (the average number of drift detections is shown in brackets in figure 2).

We concentrate now on comparing DDD to EDDM in terms of accuracy after the average time step of the first drift detection done during the second concept. The experiments show that DDD presents better accuracy than EDDM mainly for the drifts known from the diversity study (section 5) to benefit from the old ensembles (drifts with low severity or low speed). Figures 2(a), 2(b), 2(e), 2(f), 2(g) show examples of this behaviour.

In the case of low severity and high speed drifts, the best ensemble to be used according to the study presented in section 5 is the old high diversity, especially during the very beginning of the learning of the new concept, when the new low diversity is still inaccurate to the new concept. DDD gives considerable weight to the the old high diversity ensemble, as shown in figures 3(a) and 3(b). Even though it is not always the highest weight, it allows the approach to get better accuracy than EDDM. When there are false alarms, there are successful sudden increases on the use of the old low diversity ensemble, as can be observed in the same figures.
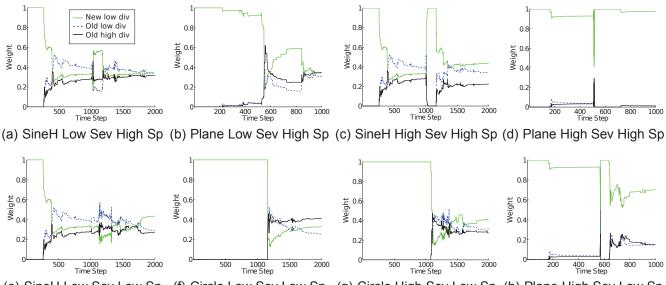
However, the non-perfect (late) drift detections sometimes make the use of the old high diversity ensemble less beneficial, making DDD get similar accuracy to EDDM, instead of better accuracy. Let's observe, for example, figure 4(a). Even though the study presented in section 5 shows that when there are perfect drift detections, the best ensemble for this type of drift would be the old high diversity, the accuracy of an approach which always chooses this ensemble becomes similar to the new low diversity ensemble's during some moments when the drift detection method is used. DDD obtains similar accuracy to EDDM exactly during these moments (figure 2(b)) and just becomes better again because of the false alarms.

In the case of low speed drifts, the best ensembles to be used according to the study presented in section 5 are the old ensembles (especially the old low diversity) soon after the beginning of the drift. DDD manages to attain better accuracy than EDDM (e.g., figures 2(e), 2(f)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING

11

TABLE 4
Parameters Choice for Artificial Data. $W = 1$, $\lambda_l = 1$ and $\theta = 0.01$ were fixed.

Data Set	Values For Preliminary Experiments					Chosen Values
	λ_h (DDD)	γ, β (DDD, EDDM)	α (EDDM)	ρ (DWM)	p (DWM)	
Circle	{0.0005,	{0.75,	{0.96,	{0.001,	{1,	$\lambda_h = 0.05$, $\gamma = \beta = 0.95$, $\alpha = 0.99$, $\rho = 0.5$, $p = 1$
SineV	0.001,	0.85,	0.97,	0.01,	10,	$\lambda_h = 0.005$, $\gamma = \beta = 0.95$, $\alpha = 0.99$, $\rho = 0.5$, $p = 1$
SineH	0.005,	0.95}	0.98,	0.1,	20}	$\lambda_h = 0.05$, $\gamma = \beta = 0.95$, $\alpha = 0.99$, $\rho = 0.5$, $p = 10$
Line	0.01,		0.99,	0.3,		$\lambda_h = 0.005$, $\gamma = \beta = 0.95$, $\alpha = 0.99$, $\rho = 0.5$, $p = 1$
Plane	0.05,		1.1,	0.5,		$\lambda_h = 0.05$, $\gamma = \beta = 0.95$, $\alpha = 0.99$, $\rho = 0.5$, $p = 5$
Bool	0.1, 0.5}		1.2, 1.3}	0.7, 0.9}		$\lambda_h = 0.1$, $\gamma = \beta = 0.95$, $\alpha = 0.99$, $\rho = 0.5$, $p = 10$

(a) SineH Low Sev High Sp (1.13, 2.23) >=>> (b) Plane Low Sev High Sp (0.17, 1.47) =>=> (c) SineH High Sev High Sp (1.13, 2.17) >=>> (d) Plane High Sev High Sp (0.10, 1.03) =<<<

(e) SineH Low Sev Low Sp (1.13, 2.43) >=>> (f) Circle Low Sev Low Sp (none, 1.26) ==>> (g) Circle High Sev Low Sp (none, 3.13) =>>> (h) Plane High Sev Low Sp (0.10, 2.09) ====

Fig. 2. Average prequential accuracy (equation 1) of DDD, EDDM, DWM and an ensemble without drift handling, considering 30 runs. The accuracy is reset when the drift begins ($f \in \{1, N + 1\}$). The vertical black bars represent the average time step in which a drift is detected at each concept. The numbers in brackets are the average numbers of drift detections per concept. The results of the comparisons aided by T tests at the time steps $0.99N$, $1.1N$, $1.5N$ and $2N$ are also shown: ">" means that DDD attained better accuracy than EDDM, "<" means worse and "=" means similar.

(a) SineH Low Sev High Sp (b) Plane Low Sev High Sp (c) SineH High Sev High Sp (d) Plane High Sev High Sp

(e) SineH Low Sev Low Sp (f) Circle Low Sev Low Sp (g) Circle High Sev Low Sp (h) Plane High Sev Low Sp

Fig. 3. Average weight attributed by DDD to each ensemble, considering 30 runs.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

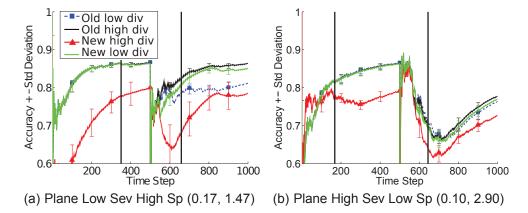IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING

12

Fig. 4. Average prequential accuracy (equation 1) obtained by an approach which always chooses the same ensemble for prediction, considering 30 runs. The accuracy is reset when the drift begins ($f \in \{1, 501\}$). The vertical black bars represent the average time step in which a drift was detected at each concept. The numbers in brackets are the average numbers of drift detections per concept.

and 2(g)) because it successfully gives high weight for these ensembles right after the drift detections and keeps the weight of the old low diversity ensemble high when there are false alarms, as shown in figures 3(e), 3(f) and 3(g).

The non-perfect (especially the late) drift detections also reduce the usefulness of the old ensembles for the low speed drifts, sometimes making DDD attain similar accuracy to EDDM. An example of this situation is shown by figures 2(h), 3(h) and 4(b). As we can see in figure 4(b), the accuracy of an approach which always chooses the old high diversity is similar to an approach which always chooses the new low diversity because of the non-perfect drift detections. For only one problem, when the speed was low and severity high, the late drift detections made DDD attain worse accuracy than EDDM.

When the drifts present high severity and high speed, the accuracy of DDD was expected to be similar (not better) to EDDM, as the new low diversity is usually the most accurate and EDDM's strategy is equivalent to always choosing this ensemble. However, the experiments show that DDD sometimes presents similar, sometimes worse (figure 2(d)) and sometimes better (figure 2(c)) accuracy than EDDM.

The reason for the worse accuracy is the inaccuracy of the initial weights given to the new low diversity soon after the drift detection, as this ensemble learnt too few examples. If the initial weights take some time to become more accurate, as shown in figure 3(d), DDD needs some time for its prequential accuracy to eventually recover and become similar to EDDM's, as shown by the accuracy's increasing tendency in figure 2(d). If the accuracy of the old ensembles decreases very fast in relation to the time taken by the new low diversity ensemble to improve its accuracy, DDD manages to attain similar accuracy to EDDM since soon after the drift detection. Besides, DDD can attain better accuracy than EDDM even for this type of drift due to the false alarms (figure

2(c) and 3(c)).

The accuracy of DDD for medium severity or speed drifts was never worse than EDDM's and is explained similarly to the other drifts.

We shall now analyse the number of win/draw/loss of DDD in comparison to EDDM at the time steps analysed through T tests after the average time step of the first drift detection during the second concept. It is important to note that, when there are false alarms, DDD can get better accuracy than EDDM before this time step. Considering the total number of win/draw/loss independent of the drift type, DDD obtains better accuracy in 45% of the cases, similar in 48% of the cases and worse in only 7% of the cases. So, it is a good strategy to use DDD when the drift type is unknown, as it obtains either better or similar accuracy and only rarely obtains worse accuracy.

Considering the totals per severity, DDD has more wins (67%) in comparison to draws (33%) or losses (0%) when severity is low. When severity is medium, DDD is similar in most of the cases (68%), being sometimes better (32%) and never worse (0%). When severity is high, DDD is usually similar (42%) or better (38%) than EDDM, but in some cases it is worse (20%). If we consider the totals per speed, the approach has more wins (61%) in comparison to draws (34%) or losses (5%) when speed is low. When the speed is medium, the number of draws is higher (64%, against 36% for wins and 0% for losses). When speed is high, the number of draws and wins is more similar (47% and 39%, respectively), but there are some more losses (14%) than for the other speeds. This behaviour is understandable, as, according to section 5, the old ensembles are more helpful when the severity or the speed is low.

DDD has usually higher accuracy than DWM, both in the presence and absence of drifts (e.g., figures 2(a) to 2(h)). Before the drift, DDD is almost always better than DWM. Considering the total number of win/draw/loss independent of the drift type for the time steps $1.1N$,

$1.5N$ and $2N$, DDD obtains better accuracy than DWM in 59% of the cases, similar in 25% of the cases and worse in 15% of the cases. As we can see in the figures, DDD usually presents faster recovery from drifts.

A probable reason for the lower accuracy of DWM during stable concepts is that it adds new classifiers when it gives misclassifications, independent of how accurate the ensemble is to the current concept. The new classifiers are initially inaccurate and, even though the old classifiers compensate somewhat their misclassifications, the accuracy of the ensemble as a whole is reduced. A probable reason for the lower accuracy of DWM in the presence of drifts is that its weights take some time steps to start reflecting the new concept, causing slow recovery from drifts. So, DWM is usually better than an ensemble without drift handling, but worse than DDD. In a few cases, when drifts that do not affect much the accuracy of old ensembles are detected, DWM obtained better accuracy than DDD. A detailed analysis of DWM is outside the scope of this paper.

In a very few occasions, not only DDD, but also EDDM and DWM get worse accuracy than an ensemble without drift handling during a few moments soon after the beginning of the drift when the drift is not fast (e.g., 2(e) and 2(h)). That happens because, in the beginning of the drift, ensembles which learnt the old concept are expected to be among the highest accuracies while the old concept is still dominant over the new concept. Nevertheless, as the number of time steps increases and the old concept becomes less dominant, the accuracy of an ensemble without drift handling is highly affected and reduced.

In summary, the experiments in this section show that DDD gets usually similar or better accuracy than EDDM and usually better accuracy than DWM both in the presence and absence of drifts. DDD also usually gets better accuracy than an ensemble without drift handling in the presence of drifts and similar accuracy in the absence of drifts.

6.3.2 Time Steps Maintaining Four Ensembles

In this section, we compare the time and memory occupied by DDD to EDDM and DWM indirectly, by considering the number of ensembles maintained in a sequential implementation using λ as the source of diversity and decision trees as the base learners. The high diversity ensembles have faster training and occupy less memory, as they are trained with much less examples (on average, λ times the total number of examples). So, we will compare the number of time steps in which DDD requires four ensembles to the number of time steps in which EDDM requires two ensembles to be maintained.

The experiments presented in section 6.3.1 show that DDD required the maintenance of four ensembles during, on average, 4.11 times more time steps than EDDM required two ensembles. Considering the total number of time steps of the learning, DDD is likely to use, on average, 1.22 times more time and memory than EDDM.

DWM always maintains one ensemble with variable size and this size was, on average, 0.45 times the size of the ensembles maintained by DDD and EDDM. However, DWM is likely to create/delete ensembles with a high rate when the accuracy is not very high, increasing its execution time.

6.3.3 Robustness to False Alarms and the Impact of W

We performed additional experiments by forcing false alarms on particular time steps during the first concept of the artificial data sets corresponding to low severity drifts, instead of using a drift detection method.

When varying W, the experiments show that this parameter allows tuning the trade-off between robustness to false alarms and accuracy in the presence of real drifts. The graphs are omitted due to space limitations.

Higher W ($W = 3$) makes DDD more robust to false alarms, achieving very similar accuracy to an approach with no drift handling, which is considered the best one in this case. $W = 3$ makes DDD less accurate in the presence of real drifts, but still more accurate than an ensemble without drift handling in the presence of drifts.

Lower W ($W = 1$) makes DDD less robust to false alarms, but still with considerably good robustness and more robust than EDDM, besides being more accurate in the presence of real drifts. So, unless we are expecting to have many false alarms and few real drifts, it is a good strategy to use $W = 1$.

6.3.4 Experiments With Real World Data

The experiments using real world data were repeated using 2 different types of base learners: multi-layer perceptions (MLPs) and naive bayes (NB). The MLPs contained 10 hidden nodes each and were trained using backpropagation with 1 epoch (online backpropagation [38], [39]), learning rate 0.01 and momentum 0.01. These base learners were chosen because they are faster than ITIs when the data set is very large. Both DDD and EDDM used ensemble size of 100.

The parameters used in the experiments are shown in table 5. All the parameters were chosen so as to generate the most accurate accuracy curves, based on 5 runs. The first parameters chosen were γ and p, which usually have bigger influence in the accuracy. The 5 runs to choose them used the default values of $\lambda_h = 0.005$ and $\rho = 0.5$. After that, a fine tuning was done by selecting the parameters λ_h and ρ. For electricity, preliminary experiments show that the drift detection method does not provide enough detections. So, instead of using the drift detection method, we forced drift detections at every $FA = \{5, 25, 45\}$ time steps. The only exception was EDDM using NB. In this case, $\beta = 1.15$ provided better results.

Each real world data set used in the experiments has different features. So, in this section, we analyse the behaviour of DDD according to each real world data set separately, in combination with a more detailed analysis of the features of the data. For each data set, the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING

14

TABLE 5
Parameters Choice for Real World Data. $W = 1$, $\lambda_l = 1$ and $\theta = 0.01$ were fixed.

Data Set	Values For Preliminary Experiments					Chosen Values	
	λ_h (DDD)	γ, β (DDD, EDDM)	α (EDDM)	ρ (DWM)	p (DWM)	MLP	NB
Electricity	{0.0005, 0.005, 0.1}	{0.75, 0.95, 1.15}	{0.80, 0.99, 1.20}	{0.3, 0.5, 0.7}	{1, 10, 50}	$\lambda_h = 0.005$, $FA = 5$, $\rho = 0.3$, $p = 1$	$\lambda_h = 0.1$, $FA = 45$, $\beta = 1.15$, $\alpha = 1.20$, $\rho = 0.5$, $p = 1$
PAKDD						$\lambda_h = 0.005$, $\gamma = \beta = 0.75$ $\alpha = 0.80$, $\rho = 0.5$, $p = 50$	$\lambda_h = 0.005$, $\gamma = \beta = 0.75$ $\alpha = 0.80$, $\rho = 0.5$, $p = 50$
KDD						$\lambda_h = 0.005$, $\gamma = \beta = 0.95$ $\alpha = 0.99$, $\rho = 0.3$, $p = 1$	$\lambda_h = 0.005$, $\gamma = 1.15$, $\beta = 0.95$ $\alpha = 0.99$, $\rho = 0.5$, $p = 1$

(a) Electricity using MLPs   (b) PAKDD using MLPs   (c) KDD using MLPs   (d) KDD using NB
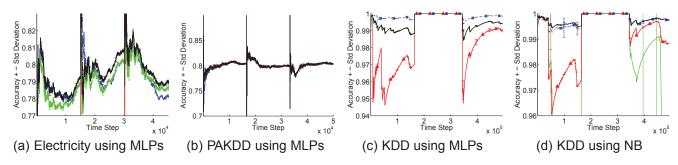
Fig. 5. Average prequential accuracy (equation 1) reset at every third of the learning, using MLPs/NB as base learners.

prior probability of class 1 at the time step t estimated according to the sample, $P(1)(t)$, is given by:

$$P(1)(t) = \frac{\sum_{i=t}^{t+w_{size}-1} y(i)}{w_{size}}$$

where $y(i)$ is the target class (1 or 0) of the training example presented at the time step i, and w_{size} is the size of the window of examples considered for calculating the average.

The first data set analysed is electricity. In this data set, the prior probability of an increase in price calculated considering the previous 1440 examples (1 month of observations) varies smoothly during the whole learning period. These variations possibly represent several continuous drifts, to which DDD is expected to have a good behaviour. Figure 5(a) shows the accuracy obtained for this data set using MLPs. As we can see, DDD is able to outperform DWM and EDDM in terms of accuracy. DDD was able to attain even higher accuracy in comparison to DWM and EDDM when using NB. The graph was omitted due to space limitations.

The second data set analysed is PAKDD. In this data set, the probability of a fraudulent customer considering the previous 2000 examples has almost no variation during the whole learning period. So, this data set is likely to contain no drifts. In this case, DDD is expected to obtain at least similar accuracy to EDDM and DWM. If there are false alarms, DDD is expected to outperform EDDM. The experiments show that all the approaches manage to attain similar accuracy for this problem when using MLPs (figure 5(b)). A similar behaviour happens when using NB. In particular, the drift detection method performed almost no drift detections – average of 0.37 drift detections during the whole learning when using MLPs and of 3.60 when using NB. So, EDDM did not

have problems with false alarms. Experiments using a parameters setup which causes more false alarms show that DDD is able to maintain the same accuracy as DWM in that condition, whereas EDDM has reduced accuracy.

Nevertheless, the class representing a fraudulent customer (class 1) is a minority class. So, it is important to observe the rates of false positives fpr and negatives fnr, which are calculated as follows:
$fpr(t) = num_{fp}(t)/num_n(t)$ and
$fnr(t) = num_{fn}(t)/num_p(t)$,
where $num_{fp}(t)$ and $num_{fn}(t)$ are the total number of false positives and negatives until the time step t; and $num_n(t)$ and $num_p(t)$ are the total number of examples with true class zero and one until the time step t.

In PAKDD, it is important to obtain a low false negative rate, in order to avoid fraud. When attempting to maximize accuracy, the false positive rate becomes very low, but the false negative rate becomes very high for all the approaches analysed. So, the parameters which obtain the best false negative rate are different from the parameters which obtain the best accuracy.

Besides, DDD can be easily adapted for dealing with minority classes by getting inspiration from the skewed (imbalanced) data sets literature [40]. Increasing and decreasing diversity based on the parameter λ of the $Poisson$ distribution is directly related to sampling techniques. A $\lambda < 1$ can cause similar effect to under-sampling, whereas a $\lambda > 1$ can cause similar effect to over-sampling.

So, experiments were done using $\lambda_l = \lambda_h = 2$ for the minority class, $\lambda_h = 0.005$ for the majority class, $\gamma = \beta = 1.15$, $\alpha = 1.20$, $\rho = 0.3$, $p = 1$ both when using MLPs and NB; $\lambda_l = 0.4$ for the majority class when using MLPs; and $\lambda_l = 0.1$ for the majority class when using NB. As we can see in figure 6, DDD obtains the best

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

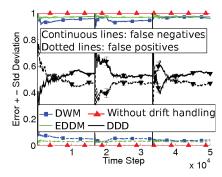IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING

15

Fig. 6. Average false positive and negative error rates for PAKDD, reset at every third of the learning using MLPs as base learners and considering 30 runs.

false negative rate when using MLPs. It also obtains the best rate when using NB. Additional work with minority classes is proposed as future work.

The last data set analysed is KDD. In this problem, the probability of an intrusion considering the previous 2000 examples has several jumps from 1 to 0 and vice versa during the first and last third of the learning. So, there are probably several severe and fast drifts which reoccur with a high rate during these periods. Even though DDD (and EDDM) is prepared for dealing with severe and fast drifts, it is not prepared for dealing with recurrent concepts yet. DWM does not have specific features to deal with recurrent drifts either, but it can obtain good accuracy if these drifts are close enough to each other so that the weights of the base learners do not decay enough for them to be eliminated.

Figures 5(c) and 5(d) show that DDD obtains worse accuracy than DWM during the first and last thirds of the learning when using MLPs, but similar (or slightly better) when using NB. The experiments show that, if the drifts are very close to each other and there are no false alarms, DDD can make use of the learning of the old concept through the old ensembles when the concept reoccurs. This is what happens when using NB, as the weight given to the old low diversity ensemble presents peaks during the learning and the number of drift detections is consistent with the changes in the estimated prior probability of attack. However, false alarms cause DDD to lose the ensembles which learnt the old concept (the old ensembles are replaced), being unable to use them when this concept reoccurs. This is what happens when using MLPs, as the number of drift detections was more than twice the number of detections when using NB, probably representing false alarms.

In summary, the experiments in this section reaffirm the analyses done in the previous sections: for a database likely to contain several continuous drifts, DDD attained better accuracy than EDDM and DWM. For a database likely to contain no drifts, DDD performed similarly to other approaches. EDDM would perform worse if there were false alarms. For a database which may contain very severe and fast drifts which reoccur with a high

rate, DDD performed similarly to DWM when it could make use of the ensembles which learnt the old concept, but performed worse when these ensembles were lost.

7 CONCLUSIONS

This paper presents an analysis of low and high diversity ensembles combined with different strategies to deal with concept drift and proposes a new approach (DDD) to handle drifts.

The analysis shows that different diversity levels obtain the best prequential accuracy depending on the type of drift. It also shows that it is possible to use information learnt from the old concept in order to aid the learning of the new concept, by training ensembles which learnt the old concept with high diversity, using low diversity on the new concept. Such ensembles are able to outperform new ensembles created from scratch after the beginning of the drift, especially when the drift has low severity and high speed, and soon after the beginning of medium or low speed drifts.

DDD maintains ensembles with different diversity levels, exploiting the advantages of diversity to handle drifts and using information from the old concept to aid the learning of the new concept. It has better accuracy than EDDM mainly when the drifts have low severity or low speed, due to the use of ensembles with different diversity levels. DDD has also considerably good robustness to false alarms. When they occur, its accuracy is better than EDDM's also during stable concepts due to the use of old ensembles. Besides, DDD's accuracy is almost always higher than DWM's, both during stable concept and after drifts. So, DDD is accurate both in the presence and in the absence of drifts.

Future work includes experiments using a parameter to control the maximum number of time steps maintaining four ensembles, further investigation of the performance on skewed data sets and extension of DDD to better deal with recurrent and predictable drifts.

REFERENCES

[1] N. C. Oza and S. Russell, "Experimental comparisons of online and batch versions of bagging and boosting," in *Proc. of ACM SIGKDD*, San Francisco, 2001, pp. 359–364.

[2] A. Fern and R. Givan, "Online ensemble learning: An empirical study," *Machine Learning*, vol. 53, pp. 71–109, 2003.

[3] R. Polikar, L. Udpa, S. S. Udpa, and V. Honavar, "Learn++: An incremental learning algorithm for supervised neural networks," *IEEE Transactions on Systems, Man, and Cybernetics - Part C*, vol. 31, no. 4, pp. 497–508, 2001.

[4] F. L. Minku, H. Inoue, and X. Yao, "Negative correlation in incremental learning," *Natural Computing Journal - Special Issue on Nature-inspired Learning and Adaptive Systems*, vol. 8, no. 2, pp. 289–320, 2009.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING

16

[5] H. Abdulsalam, D. B. Skillicorn, and P. Martin, "Streaming random forests," in *Proc. of IDEAS*, Banff, Canada, 2007, pp. 225–232.

[6] A. Narasimhamurthy and L. I. Kuncheva, "A framework for generating data to simulate changing environments," in *Proc. of the 25th IASTED AIA*, Innsbruck, Austria, 2007, pp. 384–389.

[7] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Proc. of the 7th Brazilian Symposium on Artificial Intelligence (SBIA'04) - Lecture Notes in Computer Science*, vol. 3171. São Luiz do Maranhão, Brazil: Springer, 2004, pp. 286–295.

[8] J. Gao, W. Fan, and J. Han, "On appropriate assumptions to mine data streams: Analysis and practice," in *Proc. of IEEE ICDM*, Omaha, NE, 2007, pp. 143–152.

[9] F. L. Minku, A. White, and X. Yao, "The impact of diversity on on-line ensemble learning in the presence of concept drift," *IEEE TKDE*, vol. 22, pp. 730–742, 2010. [Online]. Available: http://dx.doi.org/10.1109/TKDE.2009.156

[10] N. Littlestone and M. K. Warmuth, "The weighted majority algorithm," *Information and Computation*, vol. 108, pp. 212–261, 1994.

[11] N. Kasabov, *Evolving Connectionist Systems*. Great Britain: Springer, 2003.

[12] M. Baena-García, J. Del Campo-Ávila, R. Fidalgo, and A. Bifet, "Early drift detection method," in *Proc. of the 4th ECML PKDD International Workshop on Knowledge Discovery From Data Streams (IWKDDS'06)*, Berlin, Germany, 2006, pp. 77–86.

[13] A. Dawid and V. Vovk, "Prequential probability: Principles and properties," *Bernoulli*, vol. 5, no. 1, pp. 125–162, 1999.

[14] W. Street and Y. Kim, "A streaming ensemble algorithm (SEA) for large-scale classification," in *Proc. of ACM KDD*, 2001, pp. 377–382.

[15] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proc. of ACM KDD*. New York: ACM Press, 2003, pp. 226–235.

[16] F. Chu and C. Zaniolo, "Fast and light boosting for adaptive mining of data streams," in *Proc. of PAKDD'04*, Sydney, 2004, pp. 282–292.

[17] M. Scholz and R. Klinkenberg, "An ensemble classifier for drifting concepts," in *Proc. of the Second International Workshop on Knowledge Discovery from Data Streams*, Porto, 2005, pp. 53–64.

[18] ——, "Boosting classifiers for drifting concepts," *IDA - Special Issue on Knowledge Discovery From Data Streams*, vol. 11, no. 1, pp. 3–28, 2007.

[19] S. Ramamurthy and R. Bhatnagar, "Tracking recurrent concept drift in streaming data using ensemble classifiers," in *Proc. of ICMLA'07*, Cincinnati, Ohio, 2007, pp. 404–409.

[20] J. Gao, W. Fan, J. Han, and P. Yu, "A general framework for mining concept-drifting data streams with skewed distributions," in *Proc. of SIAM ICDM*, Minneapolis, Minnesota, 2007.

[21] H. He and S. Chen, "IMORL: Incremental multiple-object recognition and localization," *IEEE TNN*, vol. 19, no. 10, pp. 1727–1738, 2008.

[22] K. Nishida and K. Yamauchi, "Adaptive classifiers-ensemble system for tracking concept drift," in *Proceedings of the Sixth International Conference on Machine Learning and Cybernetics (ICMLC'07)*, Honk Kong, 2007, pp. 3607–3612.

[23] ——, "Detecting concept drift using statistical testing," in *Proceedings of the Tenth International Conference on Discovery Science (DS'07) - Lecture Notes in Artificial Intelligence*, vol. 3316, Sendai, Japan, 2007, pp. 264–269.

[24] K. Nishida, "Learning and detecting concept drift," Ph.D. dissertation, Hokkaido University, Japan, 2008. [Online]. Available: http://lis2.huie.hokudai.ac.jp/~knishida/paper/nishida2008-dissertation.pdf

[25] K. O. Stanley, "Learning concept drift with a commitee of decision trees," Department of Computer Sciences, University of Texas, Austin, USA, Tech. Rep. AI-TR-03-302, 2003.

[26] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: An ensemble method for drifting concepts," *Journal of Machine Learning Research*, vol. 8, pp. 2755–2790, 2007.

[27] ——, "Using additive expert ensembles to cope with concept drift," in *Proc. of ICML'05*, Bonn, Germany, 2005, pp. 449–456.

[28] K. Tumer and J. Ghosh, "Error correlation and error reduction in ensemble classifiers," *Connection Science*, vol. 8, no. 3, pp. 385–404, 1996.

[29] L. I. Kuncheva and C. J. Whitaker, "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy," *Machine Learning*, vol. 51, pp. 181–207, 2003.

[30] E. K. Tang, P. N. Sunganthan, and X. Yao, "An analysis of diversity measures," *Machine Learning*, vol. 65, pp. 247–271, 2006.

[31] G. Yule, "On the association of attributes in statistics," *Phil. Trans.*, vol. A, 194, pp. 257–319, 1900.

[32] J. Schlimmer and R. Granger, "Beyond incremental processing: Tracking concept drift," in *Proceedings of the 5th AAAI*, Philadelphia, USA, 1986, pp. 502–507.

[33] "The UCI KDD archive," 1999. [Online]. Available: http://mlr.cs.umass.edu/ml/databases/kddcup99/kddcup99.html

[34] M. Harries, "Splice-2 comparative evaluation: Electricity pricing," Artificial Intelligence Group, School of Computer Science and Engineering, The University of New South Wales, Sidney, Tech. Rep. UNSW-CSE-TR-9905, 1999.

[35] P. Utgoff, N. Berkman, and J. Clouse, "Decision tree induction based on efficient tree restructuring," *Machine Learning*, vol. 29, no. 1, pp. 5–44, 1997.

[36] F. L. Minku and X. Yao, "Using diversity to handle concept drift in on-line learning," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN09)*, Atlanta, 2009, pp. 2125–2132.

[37] I. H. Witten and E. Frank, *Data Mining - Practical Machine Learning Tools and Techniques with Java Implementations*. San Francisco: Morgan Kaufmann Publishers, 2000.

[38] N. C. Oza and S. Russell, "Online bagging and boosting," in *Proc. of IEEE International Conference on Systems, Man and Cybernetics*, vol. 3. New Jersey: Institute for Electrical and Electronics Engineers, 2005, pp. 2340–2345.

[39] F. L. Minku and X. Yao, "On-line bagging negative correlation learning," in *Proc. of IJCNN08*, Hong Kong, 2008, pp. 1375–1382.

[40] S. Kotsiantis, D. Kanellopoulos, and P. Pintelas, "Handling imbalanced datasets: A review," *GESTS International Transactions on Computer Science and Engineering*, vol. 30, no. 1, pp. 25–36, 2006.

Leandro L. Minku received the BSc and MSc degrees in Computer Science from the Federal University of Paraná, Brazil, in 2003, and from the Federal University of Pernambuco, Brazil, in 2006, respectively. He is currently working towards the Ph.D. degree in Computer Science at the University of Birmingham, UK. His research interests include online learning, concept drift, neural network ensembles and evolutionary computation. Mr. Minku was the recipient of the Overseas Research Students Award (ORSAS) from the British Government (2006) for 3 years and of several Brazilian Council for Scientific and Technological Development (CNPq) scholarships (2006, 2004, 2002 and 2001). (Minku 63341860)

Xin Yao (M91-SM96-F03) received the BSc degree from the University of Science and Technology of China (USTC), Hefei, Anhui, in 1982, the MSc degree from the North China Institute of Computing Technology, Beijing, in 1985, and the PhD degree from USTC in 1990. He worked as an associate lecturer, lecturer, senior lecturer and associate professor in China and later on in Australia. Currently, he is a professor at the University of Birmingham (UK), a visiting chair professor at the USTC and the director of the Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA). He was the editor-in-chief of the IEEE Transactions on Evolutionary Computation (2003-2008), an associate editor or editorial board member of 12 other journals, and the editor of the World Scientific Book Series on Advances in Natural Computation. His major research interests include several topics under machine learning and data mining. Prof. Yao was awarded the President's Award for Outstanding Thesis by the Chinese Academy of Sciences for his PhD work on simulated annealing and evolutionary algorithms in 1989. He also won the 2001 IEEE Donald G. Fink Prize Paper Award for his work on evolutionary artificial neural networks. He is a fellow of the IEEE.