Reacting to Concept Changes using a Committee of Experts

Ghazal Jaber[*,**], Antoine Cornuéjols[*]
Philippe Tarroux[**]

[*]AgroParisTech, département MMIP et INRA UMR-518
16 rue Claude Bernard
F-75231 Paris Cedex 5, France
ghazal.jaber, antoine.cornuejols@agroparistech.fr,
http://www.agroparistech.fr/mia/equipes:membres:page:ghazal
[**]Université de Paris-Sud, LIMSI, Bâtiment 508,
F-91405 Orsay Cedex, France
ghazal.jaber, philippe.tarroux@limsi.fr

Abstract. We present a general framework to deal with concept changes in online machine learning. Our approach relies on a committee of experts where each expert is trained on a different history size. The experts change constantly based on their performance which creates a dynamic committee that can adapt to a large variety of concept changes. The experiments, based on synthetic data, simulate abrupt and global concept changes. We test different methods to weight the experts and to combine their predictions. The experimental results show that our ensemble algorithm learns a concept change better than when a single expert learns each concept separately. Different types of experts as neural networks, SVMs and others may coexist in the committee in order to increase the diversity and improve the overall performance. We show how our algorithm is robust to the existence of potentially "bad" types of experts in the committee.

1 Introduction

Classical machine learning algorithms suppose that the training data are independent and identically distributed and that the concept they aim to learn is stationary i.e. does not change with time. These conditions are not met in general in online learning where the training data is received in a stream generated by a system with potentially evolving states. This system, sometimes referred to as a hidden context (Widmer and Kubat (1996)), may introduce changes in the training data, producing a "concept change".

Different types of concept changes exist in the litterature: A *concept drift* (Zliobaite (2009)) refers to the change of the statistical properties of the concept's target value (Cornuéjols (2010)). For example, the behavior of customers in shopping might evolve with time and thus the concept capturing this behavior evolves as well. The speed of the change can be gradual or sudden. A sudden drift is sometimes referred to as a *concept shift* (Widmer and Kubat (1996), Yang et al. (2006)). Another type of change, known as *virtual concept drift* (Syed et al.

(1999)), *pseudo-concept drift* (Ruping (2001)), *covariate shift* (Bickel et al. (2009)) or *sample selection bias* (Fan et al. (2005)) occurs when the distribution of the training data in the feature space changes with time. Even if the underlying concept remains the same, the model should be updated in order to cope with the current distribution. In Minku et al. (2010), any kind of concept change is a concept drift. Thus, the same denomination might not necessarly mean to the same thing in the community. However, according to Stanley (2003), it is not important from a practical point of view to categorize concept changes based on the source of change (change in the feature space distribution, the target values or both) since the model should be revisited anyway. Minku et al. (2010) divides changes into different types, according to different criteria (severity, speed, frequency, recurrence and predictability) creating mutually-exclusive and non-heterogeneous categories.

A main issue when dealing with concept drift is to find the optimal history size to learn the concept. When the concept is stable, a long history of training data may allow for more precise predictions. However, a long history can also be misleading if the concept is changing since it will contain old and expired data. In such case, a small history allows for a fast adaptation to the change. This is the basis of the well-known stability-plasticity dilemma.

Time sliding windows on the training data have been suggested to deal with concept drift. The window size can be fixed or adaptative. When the window size is fixed, selecting a "good" window size requires an apriori knowledge about the change in order to deal with the stability-plasticity dilemma. An algorithm that fixes the window size also assumes that the environment behaves the same way all the time, a condition that is not met in the real world. Adaptative window sizes were a solution to this problem. In Widmer and Kubat (1996), the window keeps growing when the concept is stable. When a change is detected, the window size is shrinked accordingly. Example weighting and selecting techniques have also been suggested. The training instances can be weighted depending on the instance age or performance regarding the current concept.

Several online ensembles methods have been proposed for tackling changing concepts. Ensemble methods have the advantage of holding diverse experts in the ensemble. According to Minku et al. (2010), diversity helps reduce the initial drop in accuracy that happens just after the change. When the concept is stable, however, low diversity in the ensemble gives more accurate results. In many ensemble approaches, experts are removed from the ensemble when their performance drops under a threshold and are then replaced by new experts with a small window size. Expulsing an expert from the ensemble can be the result of a concept change which makes its training window unadapted to the current situation. Thus, we don't need to explicitly detect a change and adapt the expert's window size: this can be done implicitly with a committee of experts.

In this paper, we suggest an online ensemble method that deals with concept changes. We solve the stability-plasticity dilemma by using a committee of experts where each expert is trained on a different history size. The committee members change constantly based on their prediction performance: the lowest performing members are removed from the committee and replaced with new members with a small history size. The history size of the remaining

members is incremented. Thus, we create a committee of experts whose history size evolves dynamically, which allows the adaptation to different types of changes.

The rest of this paper is organized as follows. In Section 2, we discuss related work on ensemble methods and introduce our online ensemble algorithm that deals with concept drift. The algorithm is explained in detail in Section 3. We conducted several experiments on artificial data sets to describe the behavior of the algorithm in practice. The results are reported in Section 4. Finally, in Section 5, we summarize and suggest future research directions.

2 State of the Art

Many ensemble algorithms have been proposed to deal with concept drift. The ensemble methods can be grouped as follows:

Data receipt: The data is either received and processed one instance at a time or in sequential batches. A main problem when learning on sequential batches is that a drift might occur inside a batch. An expert trained on this batch will learn misleading and sometimes contradicting information. Choosing the batch size might also be a problem when we have no apriori knowledge of the training data [1].

Data pre-processing: The data is either used as is or it can be pre-processed by changing the data distribution using sampling and/or weighting methods.

Experts' learning extent: In most cases, when the data is received in batches, an expert learns on a block of data and its learning stops at this point. In other scenarios, however, experts don't stop learning: they constantly update their knowledge with newly observed training data.

Experts deletion: In some approaches, an expert is deleted from the ensemble when its performance drops under a predefined threshold. In some other approaches, whenever the experts in the ensemble are evaluated, the worst expert is removed. In such case, and if the data is processed one instance at a time, good experts might be removed if the expert didn't get the chance to observe enough training data or if the data is noisy. To avoid random and unmeaningful deletion operations, approaches may set a maturity age that prohibits the deletion of an expert if the expert is not mature [2]. Other approaches set a frequency parameter for evaluation, creation & deletion of experts. This deletion problem is not encountered when the data is processed one block at a time since the block size is normally large enough to learn a stable expert.

Ensemble's final prediction: The ensemble classifies a test instance using the experts' predictions and weights. In case of unweighted ensembles, the weights are set to the same value for all the experts. In case of weighted ensembles, each expert is assigned a weight that reflects its predictive performance. Different methods have been proposed for weights computing: most

1. If, for instance, we know that what we learn depends on the season of the year then we may set the batch size such that it covers each season separately.

2. The maturity age corresponds to the minimum number of training examples that the expert should learn in order to be considered as a mature expert.

methods evaluate an expert based on its classification performance on recently observed data. Other methods use local accuracy to weight an expert: the expert's classification performance is evaluated on the neighborhood of the test instance and the weight is set accordingly. The experts predictions are then integrated to give the ensemble's final prediction. Classical integration methods include: majority voting, weighted majority voting, selecting the prediction of the expert with the highest weight, using a roulette-wheel selection on the experts weights etc...

Next, we present five online ensemble approaches that have been proposed to deal with concept drift:

The **KBS-stream algorithm** (Scholz and Klinkenberg (2005)) is a boosting-like method that trains a classifier ensemble from data streams. In Scholz and Klinkenberg (2005), the data are received in batches. With each received batch, either a new classifier is added or the latest added classifier is updated. Before learning, the data distribution in the recent batch is changed using sample weighting and sampling strategy: the data is passed through the existing classifiers and the distribution is modified such that the last classifier learns something different than the remaining ones in the ensemble. The data is assumed to be independent and identically distributed in each batch and a drift is allowed to occur between two consecutive batches but never inside a batch. These conditions are not met in many real-world online problems.

The **ASHT-bagging algorithm** (Bifet et al. (2009)) is a variant of bagging designed to tackle with non-stationary concepts. The approach builds an ensemble of Hoeffding Trees (Domingos and Hulten (2000)) with different tree sizes: small trees to adapt more quickly to changes and large trees which are better for stationary concepts. In this approach, the data is received one instance at a time. A tree in the ensemble is updated with each newly observed training data. When a tree reaches its maximum size, it is pruned, even for stationary concepts. The diversity of the ensemble improves bagging and allows the adaptation to concept drifts.

The **DWM (Dynamic Weighted Majority) algorithm** (Kolter and Maloof (2007)) receives and processes the data one instance at a time. Each classifier in the ensemble is initially assigned a weight of one. The weight is decreased if the classifier misclassifies an instance. This ensemble method copes with concept drift by dynamically adding and removing classifiers: a new classifier is added if the ensemble misclassifies a test sample and a classifier is removed if its weight is lower than a predefined threshold. This strategy makes the ensemble size variable. As for the ensemble final prediction, it corresponds to the class label with the highest accumulated weight.

In the **dynamic integration of classifiers**, Tsymbal et al. (2008) proposes an ensemble integration technique to handle concept drift. In this approach, the data is received as a sequence of fixed-size batches. The data is then divided into overlapping or non-overlapping blocks. With each data block, a new expert is trained and added to the ensemble. When the ensemble size reaches its maximum size, the *replace the loser* pruning strategy is used: the worst expert in the ensemble is removed and replaced by the new one, trained on the most recent data block. In dynamic integration of classifiers, each classifier is given a weight proportionnal to its local accuracy with regard to the instance tested and the best classifier is selected or the classifiers are integrated using weighted voting. They show that dynamic integration gives better accuracy than the most commonly used integration techniques, specially in the case of local drifts.

The **committee of decision trees** in Stanley (2003) generates a weighted committee of decision trees that votes on the current classification. Each expert votes for a newly received test instance then it modifies its knowledge when the instance's label becomes available. The voting performance of each expert is evaluated. If some experts see their voting performance drop under a predefined threshold, the worst classifier is removed and replaced by a new classifier trained on the last observed training instance [3]. This strategy allows the experts to affine their knowledge when the concept is stable. When the concept changes, experts that learnt on old and misleading data will encounter a drop in their performance and will be replaced.

Our algorithm is inspired by the work of Stanley (2003). The main difference is that instead of adding one expert at a time, we add a set of different experts. This allows the increase of the diversity in the committee by using different types of experts. Thus, we can add several models as decision trees, SVMs, neural networks and others and we can also use different structures of the same model as neural networks with different activation functions or layers, decision trees with different sizes, etc... In addition, in our approach, we don't set a predefined threshold for expert deletion. The worst and mature experts are deleted all the time. We show in the experiments that this deletion strategy doesn't affect the committee's predictive performance even when the concept is stable. We test different ways of evaluating and weighting the experts; we also explore different ways of integrating their predictions. When a training instance is to be processed, the experts are evaluated twice: first before learning, when asked to classify the instance, the experts are evaluated for the committee's final prediction, and secondly after the true label of the instance is revealed, the experts are re-evaluated and their weights are used this time to delete the worsts among them. The weighting strategy for both evaluation steps can be different.

3 The Proposed Algorithm

In an online learning scenario, the training data is received in a stream and a training instance is processed once, on arrival. A training example is represented by a pair $(\mathbf{x}, y)$, where $\mathbf{x} \in R^p$ is a vector in a p-dimensional feature space and y is the desired output or target. In a classification problem, y is a discrete value whereas in a regression problem y is a real value. The desired output y can also be represented as a d-dimensional vector, in a more general formulation. A concept represents the distribution of the problem $p(\mathbf{x}, y)$ (Minku et al. (2010)). This distribution might evolve with time creating a concept change.

The concept can be a decision tree, a neural network, an SVM or any other model that can capture $p(\mathbf{x}, y)$. In this paper, the concept is an ensemble of classifiers that predict an instance's target or label. The ensemble is required to adapt rapidly to a concept change and to predict the target y of a test sample $\mathbf{x}$ at anytime. The main parts of the ensemble method are presented next. The pseudo-code is shown in Algorithm 1.

The pool of predictors. In our algorithm a *pool of predictors* is a set of experts with possibly different prediction models (ex: SVM, neural networks, decision trees), different model structures (ex: different activation functions or number of layers in a neural network) and/or

3. The approach uses incremental learning where the training data is learnt in a sequence i.e. one instance at a time.

different training algorithms. The pool of predictors has a size of n_{pool}. The predictors can be used for regression (Jaber et al. (2011)) or classification purposes. In this paper, a predictor predicts the label of an example in a classification task.

The committee of predictors. Our committee consists of pools of predictors. Instead of adding one predictor to the committee at a time, we add a pool of predictors, which allows the existence of various types of prediction models and structures in the committee. We define the maximum number of pools of predictors in the committee as max_{pool}.

The committee before reaching its maximum size. The committee is initially empty. At each time step, a new training data $e = (\mathbf{x}, y)$ is received where $\mathbf{x}$ is the example and y the corresponding label. A new pool of predictor trained on $e = (\mathbf{x}, y)$ is added to the committee. The predictors that are already in the committee are also trained incrementally on the new training data $e = (\mathbf{x}, y)$. For instance, at time step 1, when the first example e_1 is received, a pool b_1 trained only on $\{e_1\}$ is added to the committee. At time step 2, a new pool b_2 is added to the committee, trained on $\{e_2\}$ and the old pool b_1 is trained on $\{e_2\}$ also. Thus, b_1 is trained on $\{e_1, e_2\}$ and b_2 is trained on $\{e_2\}$ only. This operation is repeated until the committee reaches its maximum size. At time step t, the pool b_j is trained on $\{e_j, ..., e_t\}$, where $j \in [1, max_{pool}]$ and $j <= t$.

The committee after reaching its maximum size. The committee reaches its maximum size at time step $t = max_{pool}$. From this time on, the committee is updated by removing the worst predictors and adding new ones. The updating operation however cannot be executed unless all the predictors in the committee are mature i.e. they all have been trained on a minimum number of training data. When a deletion operation is to be executed, each predictor is evaluated and a weight is assigned accordindly. The larger the weight, the better the predictor. The predictors with the lowest weights are removed and replaced by a new pool trained on the last observed example. In order to keep the committee size fixed, the worst $pool_{size}$ predictors are removed from the committee. In this paper, we don't set a specific function to compute the weights; we leave it as a general function. We will explore several methods for weights computing in the experiments section. These weights, used to remove the worst predictors, will be refered to as "deletion weights" in the remainder of this paper.

The committee's final prediction. The committee predicts the label $\tilde{y}$ of an incoming example $\mathbf{x}$. The label of $\mathbf{x}$ is first unknown to the committee. It is just after the comittee predicts its label $\tilde{y}$ that the real label y is revealed. Since each predictor gives its own prediction of the label, how does the committee constructs a final prediction $\tilde{y}$? The "selection weights" are computed to evaluated each predictor in the committee. Then, according to the selection weights, the predictions are combined using an integration function (ex: voting, weighted voting, the prediction of the best predictor etc...). The result of the integration function gives the committee's final prediction. It is important to notice that an unmature predictor is not taken into account in the final prediction, unless all the predictors in the committee are still unmature. In such case, all the predictors share the same weight.

Algorithm 1 The Concept Change Adaptation Algorithm

Require: $\mathbf{x}$ is an incoming example with no corresponding label received yet; n_{pool} is the size of the pool of predictors; max_{pool} is the maximum number of pools in the committee; P is the committee and max_P is the committee's maximum size: $max_P = n_{pool} * max_{pool}$.
Ensure: $\tilde{y}$ is the committee's prediction of $\mathbf{x}$'s label

{PREDICTION}
$H \leftarrow \phi$ is the set of predictions
for $k = 1 \rightarrow$ size(P) **do**
  $p_k \in P$ is the k^{th} predictor
  $\tilde{y}_{p_k}$ is the prediction of p_k on $\mathbf{x}$'s label
  $H \leftarrow H \cup \tilde{y}_{p_k}$
end for

$WSEL \leftarrow \phi$ are the selection weights
for $k = 1 \rightarrow$ size(P) **do**
  $p_k \in P$ is the k^{th} predictor
  $wsel_{p_k} = evals\_fct(p_k)$ is p_k's selection weight
  $WSEL \leftarrow WSEL \cup wsel_{p_k}$
end for

$\tilde{y} = integ\_fct(WSEL, H)$ is the committee's final prediction
Read the real label y of $\mathbf{x}$

{LEARNING}
$WDEL \leftarrow \phi$ are the deletion weights
for $k = 1 \rightarrow$ size(P) **do**
  $p_k \in P$ is the k^{th} predictor
  $wdel_{p_k} = evald\_fct(p_k)$ is p_k's deletion weight
  $WDEL \leftarrow WDEL \cup wdel_{p_k}$
end for

if size(P) $>= max_P$ **and**
for all $p \in P$, p's age$>=$ maturity_age **then**
  for $k = 1 \rightarrow n\_pool$ **do**
    $p \in P$ is the predictor with the lowest deletion weight
    $P \leftarrow P \setminus p$
  end for
end if

if size(P) $< max_P$ **then**
  b is the pool of predictors; the age of each $p \in b$ is equal to 0
  $P \leftarrow P \cup b$
end if

for $k = 1 \rightarrow$ size(P) **do**
  $p_k \in P$ is the k^{th} predictor
  train p_k on $(\mathbf{x}, y)$
  increment p_k's age
end for

4 Experiments

We conducted several experiments to examine the behavior of the algorithm. In Section 4.1, we check if the algorithm gives more importance to plasticity over stability i.e. if it was designed to react well on changing concepts only or if it can perform well on stable concepts also. In Section 4.2, we observe the effect of "bad predictors" in the pool of committee on the predictive performance. Finally, in Section 4.3, we compare different ways of weighting the experts and integrating their predictions.

4.1 Experiment 1: *the stability-plasticity dilemma*

In this set of experiments, we examine the behavior of the committee when the concept is stable and when a sudden and severe drift occurs i.e. when the old concept is directly replaced by a completely new one. The deletion strategy should allow a fast adaptation to a concept change but we should ensure that deleting experts constantly doesn't affect the committee's predictive performance when the concept is stable. We also show that our committee learns a concept drift better than a single expert learns each concept individually.

Experimental Setup In the experiments, the target concept C is a hyperplane. A hyperplane in a d-dimensional space is represented by the equation:

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

We simulate one drift event and thus we create two different concepts C_1 and C_2: The first concept C_1 is learnt from a data stream of 200 training examples $(\mathbf{x}, y_1)$ where $\mathbf{x} \in [0,1]^d$ is a randomly generated vector of dimension d and y is set as follows:

$$y = \begin{cases} 1 & \text{if } \sum_{i=1}^{d-1} w_i x_i - w_0 > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The value of w_0 is set to:

$$w_0 = 1/2 \sum_{i=1}^{d-1} w_i \tag{3}$$

so that nearly half of the y's are positive and the other half is negative and $d = 2$. At time step 201, C_1 is replaced by another concept C_2, represented by another sequence of 200 training examples $(\mathbf{x}, y_2)$ where $\mathbf{x}$ is the same as in C_1 and y_2 is the opposite class of y_1. Thus, we flip the classes from C_1 to C_2. Since all the input space is affected by the drift, this is called a *severe drift* (Minku et al. (2010)) or a *global drift* (Tsymbal et al. (2008)). The second concept replaces completely the first one at time step 201 causing what is called a *sudden drift* (Minku et al. (2010)).

In order to evaluate the predictive performance, we compute the *online error* which is calculated as follows: at time step t, the label of a training instance $\mathbf{x}_t$ is predicted before the instance is learnt. The prediction error is calculated and the average prediction error from time step 1 to t is updated. The average error is known as the online error (Minku et al. (2010)).

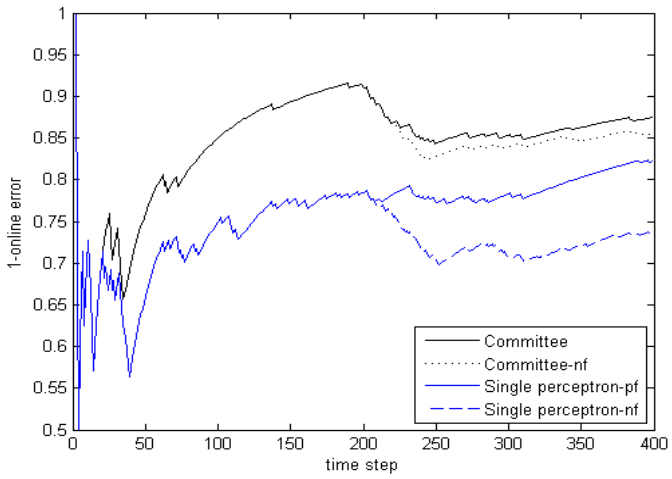

FIG. 1 – *The online predictive performance with the four experiments described in Section 4.1; nf and pf refer to "no forgetting" and "perfect forgetting", repectively*

This procedure is also called the progressive evaluation (Tsymbal et al. (2008)) or the interleaved test-then-train approach (Bifet et al. (2009)). In our experiments, the prediction error is the absolute value between the predicted value and the real value. The online predictive performance or accuracy (acc) and the online error (err) are related by the following equation: $acc = 1 - err$.

We compared the predictive performance using four approaches:

a **Committee**: The evolving committee described in Section 3. The pool of predictors contains a single perceptron and the maximum number of pools is set to 20 ($n_{pool} = 1, max_{pool} = 20$). Both deletion & selection weights are computed as the inverse of the online error on the recent data [4]. The expert with the largest selection weight is chosen for the final prediction and the expert with the lowest deletion weight is removed [5]. The maturity age is set to 20 [6].

b **Committee-nf**: The committee is the same as the previous one. However, we don't remove any classifier: all the committee members increase their history size as new training data is observed.

c **Single Perceptron-pf**: A single perceptron that learns the current concept. It erases its memory -by resetting its weights to 0- when the drift occurs [7]. This simulates a perfect forgetting of the old concept C_1 when it is replaced by the new concept C_2.

d **Single Perceptron-nf**: A single perceptron that learns the current concept without erasing its memory. It keeps learning as new training data becomes available.

4. We compute the error on the last 20 predictions.
5. Since we have one expert in the pool, only one expert is removed at deletion time.
6. The value 20 has been chosen randomly.
7. The perceptrons' initial weights are fixed to 0 in all the experiments for comparison purposes.

Results & Analysis The results are shown in figure 1. We notice that:

- When the concept is stable from time step 1 to 200, our committee is either comparable or better than the other approaches (b), (c) and (d). Thus, the deletion strategy doesn't affect the committee's predictive performance when the concept is stable. In general, when the concept is stable, the committee keeps the predictors with a large history size and expulses the predictor(s) with the smallest history size since they will perform poorly. The expulsed predictors will be replaced by new ones, trained on a small history size which will cause their expulse again from the committee and so on. Meanwhile, the remaing predictors increase their history size which allows learning the stable concept.
- In experiment (b), the online learning algorithm of the perceptron makes the perceptron able to forget C_1 when enough training data of C_2 has been observed. We see however, that the committee adapts faster to the change in experiment (a) than in (b). This suggests that the fast adaptation in experiment (a) is related to the approach forgetting strategy -realized by removing bad experts- and not to the perceptron's forgetting strategy embedded in its learning algorithm. The advantage of the deletion strategy is emphasized when using a lossless learning algorithm which keeps the memory of all previously learnt data. Lossless online algorithms are available for decision trees, Naive Bayes models and others (Oza and Russell (2001), Utgoff et al. (1997)).
- The committees of experts in experiments (a) & (b) outperform the single classifiers in experiments (c) & (d). By comparing the results of experiments (a) & (c), we also notice that our committee learns a concept drift better than the single perceptron learns each concept individually. Erasing the memory of the perceptron (experiment (c)), when the drift occurs, gives better predictive results than keeping the memory of the previously learnt concept (experiment (d)) since it takes time for the perceptron to forget its acquired knowledge.

4.2 Experiment 2: *the pool of predictors*

The pool of predictors allows the use of different training models, algorithms and structures in the committee. While this has the advantage of adding diversity, it might hurt the committee performance when badly chosen experts have been added to the pool. In this set of experiments, we observe the behavior of the committee when "bad" experts exist in the pool of predictors. We show how our approach increases the percentage of the "good" experts in the committee while decreasing the percentage of the "bad" experts. The predictive performance is not affected during this process.

Experimental Setup The same settings described in Section 4.1 were used (training data, maturity age, selection and deletion weights). However, we varied the pool of predictors in the following two experiments:

a The pool of predictors contains a single perceptron and the maximum number of pools is set to 20. Thus, the committee maximum size is equal to 20 ($n_{pool} = 1, max_{pool} = 20, max_P = 20$).

b The pool of predictors contains two perceptrons: The first one will be normally trained on its training data. We will refer to this perceptron as a "normal perceptron". The second

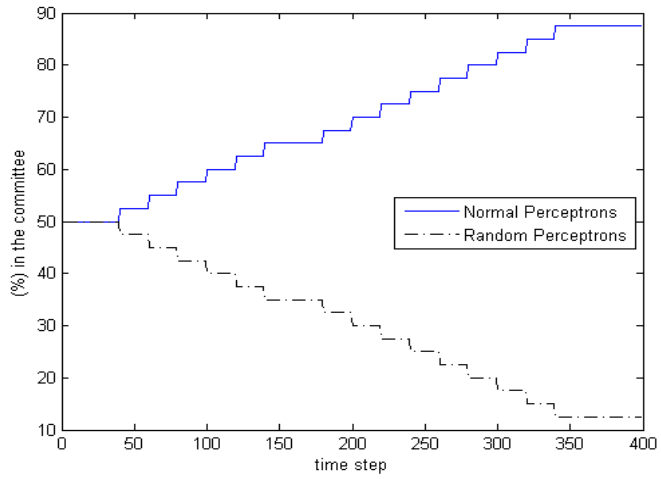

FIG. 2 – *The percentage of normal and random perceptrons in the committee as described in experiment (b) of Section 4.2.*

one represents a randomly generated hyperplane which classifies randomly half of the examples as positive and the other half as negative. This perceptron -referred to as a "random perceptron"- stays as is in the committee until it is removed.
The maximum number of pools is also set to 20. Thus, the committee maximum size is equal to 40 ($n_{pool} = 2, max_{pool} = 20, max_P = 40$).

Results & Analysis We show in figure 2 the percentage of normal and random perceptrons in the committee for experiment (b). We notice the following:

- During time steps 1 to 20, there are 50% of each type of perceptrons in the committee. Since the committee has not reached its maximum size yet, pools are still added during this period.
- At time step 20, the lastly added pool requires 20 time steps to be mature. Thus, from time step 21 to 40, no deletion is performed on the committee members which keeps the percentage of each type of perceptrons fixed.
- The percentage of normal perceptrons increases with time while the percentage of random perceptrons decreases. This is explained by the fact that the number of random perceptrons deleted from the committee is higher than the added ones. For instance, in each of 15 deletion operations, the worst two experts -deleted from the committee- were random perceptrons and were replaced by a pool which contains one normal perceptron and one random perceptron. Thus, after these 15 deletion operations, 30 random perceptrons were deleted and 15 random perceptrons were added.

After deleting members from the committee and adding a new pool, 20 steps are required for the lastly added pool to be mature. Since deleting an expert cannot be performed unless all the

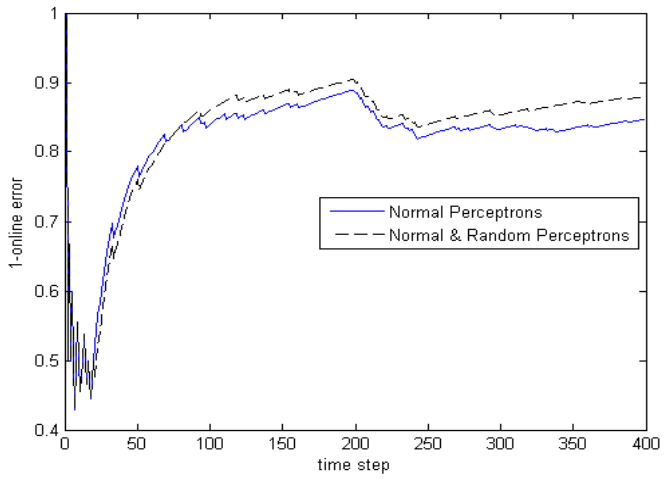

FIG. 3 – *The online predictive performance with the two experiments described in Section 4.2*

experts in the committee are mature, 20 steps are required between two consecutive deletion operations. This causes the step effect in figure 2.

In figure 3, we compare the predictive performance with experiments (a) and (b). We can see that adding random predictors didn't decrease the online error even when the committee contained a large number of random predictors (50%). In fact, the accuracy is improved. This is explained by the fact that the committee in experiment (b) is twice as big as the one in (a). The existence of clearly "bad predictors" in experiment (b) will cause their removal from the committee. However with only "good predictors" as in experiment (a), the relatively least performing predictors are removed even if they are globally good.

The results of the experiments suggest that if some predictors in the pool were badly selected (ex: unadapted model or structure to the current learning problem), there is not need to worry about a drop in the predictive performance [8]. The algorithm will manage to remove the worst predictors and keep the better ones. Even if bad experts remain in the committee, they won't be selected for the committee's final prediction. It is also important to note that if a "bad" model in the pool becomes "good" at a specific time, it will get a second chance to populate the committee since it is always included in each newly added pool.

4.3 Experiment 3: *the combination of experts' prediction*

In this set of experiments, we test two different ways of computing the *selection weights* and different ways of *combining* them for the final prediction.

8. Of course, if all the experts are badly selected in the pool, the approach won't solve the problem.

Selection Weights The selection weights are computed as follows:

- **Recent Window method (RW)**: As in the previous experiments, the weight of the expert is computed as the inverse of its online error on the recent training data. The weight value ranges from 0 to ∞.
- **Neighborhood Window method (NW)**: As suggested in Tsymbal et al. (2008), the weight of the expert is computed by evaluating its prediction performance on the neighborhood of the test data. The weight value ranges from -1 (when the expert misclassifies all the examples in the neighborhood) to +1 (when the expert correctly classifies all the examples in the neighborhood). The selection weight $wsel_{p_k}(\mathbf{x})$ of expert p_k given a test sample $\mathbf{x}$ is computed as follows:

$$wsel_{p_k}(\mathbf{x}) = \sum_{j=1}^{k} \big(\sigma(\mathbf{x}, \mathbf{x_j}).mr_{p_k}(\mathbf{x_j})\big) / \sum_{j=1}^{k} \sigma(\mathbf{x}, \mathbf{x_j}) \tag{4}$$

$$mr_{p_k} = \begin{cases} 1 & \text{if } \tilde{y}_{j,p_k} = y_j \\ -1 & \text{if } \tilde{y}_{j,p_k} \neq y_j \end{cases} \tag{5}$$

$$\sigma(\mathbf{x}, \mathbf{x_j}) = \frac{1}{||\mathbf{x} - \mathbf{x_j}||} \tag{6}$$

In the above equations, k is the size of the neighborhood, $\mathbf{x_j}$ is the j-th nearest neighbour from $\mathbf{x}$, y_j is $\mathbf{x_j}$'s real label and $\tilde{y}_{j,p_k}$ is $\mathbf{x_j}$'s label as predicted by the expert p_k.

It is important to note that we are testing the *selection weights* which are used to evaluate the experts for the final prediction. Not to be confused with the *deletion weights* which are used to remove the worst experts (see Algorithm 1). The deletion weights are computed the same way in all the experiments using the RW method [9].

Combination Rules After computing the weights, we tested different ways of combining the experts prediction:

- **V**: simple vote
- **WV**: weighted vote
- **WVD**: weighted vote after suppressing the predictions of the worst experts i.e. the experts with the performances that fall into the lower half of the performance interval.
- **S**: the prediction of the best expert is selected i.e. the expert with the largest weight.

Experimental Setup We conducted three experiments using the same training data as in Section 4.1 and 4.2. The experiments have the following settings:

a The selection weights are computed using the NW method: the size of the neighborhood k is set to 5 and the nearest neighbours are taken from the last 20 training samples. The pool of predictors contains one perceptron and the maximum number of pools is set to 20 i.e. $n_{pool} = 1, max_{pool} = 20, max_P = 20$. The perceptrons' weights are initialized to 0.

9. Using the same deletion weights allows one to evaluate the committee predictive performance based on the selection weights only.

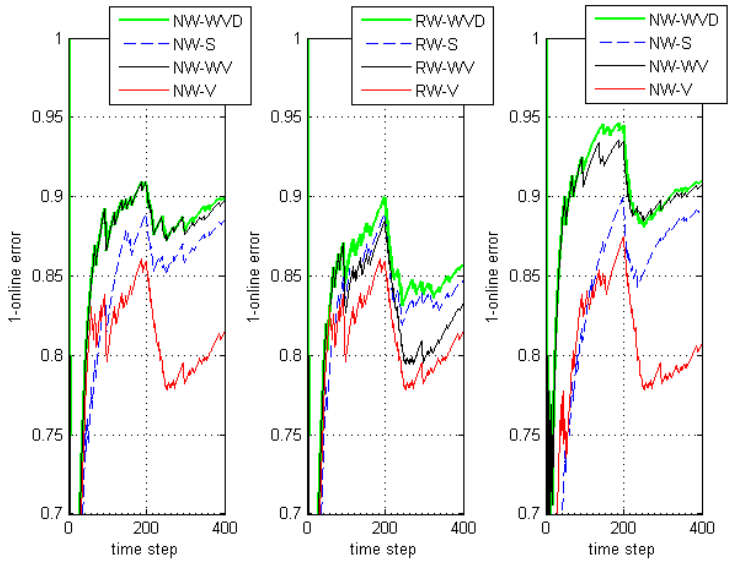

FIG. 4 – *From left to right: the online predictive error with the experiments* (a),(b) *and* (c) *described in Section 4.3, respectively. In each figure, we plot the online predictive performance with several selection & combination methods. The items in the legend are read as: weight computation method-combination method.*

b The selection weights are computed using the RW method: the online error is computed on the last 20 training data. The pool of predictors and the other settings are the same as in experiment (a).

c The selection weights are computed using the NW method with the same settings as in experiment (a). However, we use a different pool of predictors which contains two types of perceptrons: a normal perceptron and a random perceptron. The maximum number of pools is set to 10 i.e. $n_{pool} = 1, max_{pool} = 20, max_P = 20$. The perceptrons' weights are initialized randomly and their bias w_0 is computed as in equation 3.

Results & Analysis The results of the experiments are shown in figure 4. We notice the following:

– When comparing the combination methods in terms of prediction accuracy in the experiments (a) and (b), we notice that the simple voting (V) gives the highest online error. This is expected since it doesn't take into account the goodness/badness of the predictor. WVD always gives the best results while WV shows a different behavior when used with RW and NW weights. With the RW weights, we can see that shortly after the drift, the WV accuracy falls. This is explained by the fact that when a committee contains a relatively large number of bad experts, their weights will sum up and will be high enough to win the vote. This effect is not observed when WV is used with the NW weights because

bad experts have negative weights which will decrease the influence of their vote. Since WV dimishes the influence of the bad experts and WVD suppressed the votes of the bad experts, WVD and WV give very close results when used with the NW weights.
– When comparing NW and RW in experiment (a) and (b), we notice that NW gives a better prediction accuracy than RW when learning a stable concept. It also reacts and recovers faster from the concept drift. However, NW requires more computational time since, for each test instance $\mathbf{x}$, NW computes the predictive performance of each expert in the neighborhood of $\mathbf{x}$.
– In experiment (c), we added random predictors to emphasize the differences between the combination methods. We notice the unstable behavior of the S combination method: the gap increases between S and the best combination methods when random predictors exist in the committee. By comparing the results of experiment (c) with (a), we see an improved accuracy when learning the first concept C_1 in experiment (c). Initializing the perceptrons with different initial weights adds diversity to the committee which increases the overall predictive performance [10].

5 Conclusion

In this paper, we presented an approach that is able to deal with concept changes, using a dynamic and diverse committee of experts where each expert predicts the label of an incoming test sample. The diversity has an impact on the predictive performance of the committee: it improves the classification accuracy when the concept is stable and lowers the test error when the concept change happens (Minku et al. (2010)). In our approach, the committee is *diverse* since it can hold various types of experts in the committee. The experts can be different prediction models (neural networks, SVMs ...) with potentially different structures (number of layers in a neural network, the maximum size of a decision tree ...) and are also trained on different training data. Initializing the experts differently also increases the diversity in the committee. The committee is also *dynamic* since it constantly updates itself by removing the lowest performing experts and adding new experts with a small history size. As for the remaining experts which were not expelled from the committee, their history size is increased. This strategy avoids finding explicitly an appropriate window size in order to solve the stability-plasticity dilemma. Since the approach processes one instance at a time, there is no need for storage or reprocessing (unless NW method is used to weight the classifiers which requires to keep a small window of recent data for local accuracy evaluation).

The experimental results showed that our ensemble algorithm learns a concept change better than when a single expert learns each concept separately. Thus, our algorithm gives importance to plasticity by adapting rapidly to a concept change and it also gives importance to stability by learning a stable concept as a single online expert would do. In the experiments, we show how the algorithm is robust to the existence of potentially "bad" types of experts in the committee: the algorithm manages to remove the "bad" experts and increases the number of "good" experts in the committee without hurting the overall predictive performance during the process. Finally, we tested different methods to weight and combine experts. We noticed

10. In the experiments (a) and (b), the diversity comes from predictors trained on different training samples.

that even in a global drift scenario, evaluating an expert based on its local accuracy reflects better the expert's predictive performance on a test sample, than with its global accuracy.

For future work, we plan to test our algorithm on different types of changes and compare the results with already existing ensemble methods that deal with concept change. We also find it interesting to study the advantages of combining a pro-active approach which predicts how a concept will change in the near future, with a reactive approach which adapts passively to the current situation.

References

Bickel, S., M. Brückner, and T. Scheffer (2009). Discriminative learning under covariate shift. *The Journal of Machine Learning Research 10*, 2137–2155.

Bifet, A., G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà (2009). New ensemble methods for evolving data streams. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge discovery and data mining*, pp. 139–148.

Cornuéjols, A. (2010). On-line learning: where are we so far? *Ubiquitous knowledge discovery*, 129–147.

Domingos, P. and G. Hulten (2000). Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD International Conference on Knowledge discovery and data mining*, pp. 71–80.

Fan, W., I. Davidson, B. Zadrozny, and P. S. Yu (2005). An improved categorization of classifier's sensitivity on sample selection bias.

Jaber, G., A. Cornuéjols, and P. Tarroux (2011). Predicting concept changes using a committee of experts. In *ICONIP'11 International Conference on Neural Information Processing*, Shanghai, China, pp. 580–588.

Kolter, J. Z. and M. A. Maloof (2007). Dynamic weighted majority: An ensemble method for drifting concepts. *The Journal of Machine Learning Research 8*, 2755–2790.

Minku, L. L., A. P. White, and X. Yao (2010). The impact of diversity on online ensemble learning in the presence of concept drift. *IEEE Transactions on Knowledge and Data Engineering 22*(5), 730–742.

Oza, N. C. and S. Russell (2001). Online bagging and boosting. In *Artificial Intelligence and Statistics 2001*.

Ruping, S. (2001). Incremental learning with support vector machines. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, San Jose, CA , USA, pp. 641–642.

Scholz, M. and R. Klinkenberg (2005). An ensemble classifier for drifting concepts. In *Proceedings of the Second International Workshop on Knowledge Discovery in Data Streams*, pp. 53–64.

Stanley, K. O. (2003). Learning concept drift with a committee of decision trees. *UT-AI-TR-03-302, Department of Computer Sciences, University of Texas at Austin, USA*.

Syed, N. A., H. Liu, and K. K. Sung (1999). Handling concept drifts in incremental learning with support vector machines. In *Proceedings of the fifth ACM SIGKDD International*

Conference on Knowledge Discovery and Data Mining, pp. 272–276.

Tsymbal, A., M. Pechenizkiy, P. Cunningham, and S. Puuronen (2008). Dynamic integration of classifiers for handling concept drift. *Information Fusion 9*(1), 56–68.

Utgoff, P. E., N. C. Berkman, and J. A. Clouse (1997). Decision tree induction based on efficient tree restructuring. *Machine Learning 29*(1), 5–44.

Widmer, G. and M. Kubat (1996). Learning in the presence of concept drift and hidden contexts. *Machine learning 23*(1), 69–101.

Yang, Y., X. Wu, and X. Zhu (2006). Mining in anticipation for concept change: Proactive-reactive prediction in data streams. *Data mining and knowledge discovery 13*(3), 261–289.

Zliobaite, I. (2009). Learning under concept drift: an overview. Technical report, Vilnius University, Faculty of Mathematics and Informatics.

Résumé

Nous présentons une méthode d'ensemble capable de s'adapter aux changements d'un concept dans le cadre de l'apprentissage artificiel en ligne. Notre approche s'appuie sur un comité d'experts où chaque expert est formé sur des données différentes. Les experts évoluent constamment en fonction de leur performance ce qui crée un comité dynamique capable de s'adapter à une grande variété de changements de concept. Les expériences, basées sur des données artificielles, simulent un changement brusque de concept. Nous testons différentes méthodes pour évaluer les experts et combiner leurs prédictions. Les résultats montrent que notre méthode d'ensemble apprend un changement de concept mieux que quand un seul expert apprend chaque concept séparément. Différents types d'experts comme les réseaux de neurones, les arbres de décision et d'autres peuvent coexister au sein du comité afin d'augmenter la diversité et d'améliorer la performance globale. L'algorithme reste toutefois robuste à l'existence potentielle de "mauvais" types d'experts dans le comité.