An On-line Weighted Ensemble of Regressor Models to Handle Concept Drifts

Symone Gomes Soares*, Rui Araújo

Institute for Systems and Robotics, and Department of Electrical and Computer Engineering, University of Coimbra, Polo II, PT-3030-290 - Coimbra, Portugal.

Abstract

Many estimation, prediction, and learning applications have a dynamic nature. One of the most important challenges in machine learning is dealing with concept changes. Underlying changes may make the model designed on old data, inconsistent with new data. Also, algorithms usually specialize in one type of change. Other challenge is reusing previously acquired information in scenarios where changes may recur. This strategy improves the learning accuracy and reduces the processing time. Unfortunately, most existing learning algorithms to deal with changes are adapted on a batch basis. This process usually requires a long time, and such data may not reflect the current state of the system. However, even if the system is adapted on a sample basis, existing algorithms may adapt slowly to changes and cannot conciliate old and new information. This paper proposes an On-line Weighted Ensemble (OWE) of regressor models which is able to learn incrementally sample by sample in the presence of several types of changes and simultaneously retain old information in recurring scenarios. The key idea is to keep a moving window that slides when a new sample is available. The error of each model on the current window is determined using a boosting strategy that assigns small errors to the models that predict accurately the samples predicted poorly by the ensemble. To handle recurring and non-recurring changes, OWE uses a new assignment of models’ weights that takes into account the models’ errors on the past and current windows using a discounting factor that decreases or increases the contribution of old windows. In addition, OWE launches new models if the system’s accuracy is decreasing, and it can exclude inaccurate models over time. Experiments with artificial and industrial data reveal that in most cases OWE outperforms other state-of-the-art concept drift approaches.

Keywords: Concept drift; Ensemble learning; Learning in changing environments; Regression; Ensemble pruning strategies.

1. Introduction

On-line learning applications where the target concept may change over time pose serious problems. Underlying changes may make the model designed on old data, inconsistent with new data. This problem is known as concept drift. One of the most important challenges in machine learning is dealing with concept changes. Other challenge in on-line learning algorithms is to adapt and deal with changes without being informed about them, and make use of the past experiences in situations where old contexts may reappear (Widmer and Kubat, 1996; Bosnić et al., 2014). In industry, a large demand of algorithms for on-line prediction is observed, such algorithms are usually called Soft Sensors (Ibargüengoytia et al., 2013; Kadlec et al., 2011). They employ predictive models to provide on-line estimations of difficult-to-measure variables based on some easy-to-measure variables (Wu et al., 2009). Examples of applications are the groundwater level prediction in a coastal aquifer (Taurmina et al., 2012), the river flow discharge prediction in a reservoir (Cheng et al., 2005), and others (Kadlec and Gabrys, 2011; Chau, 2007). Unfortunately, industrial processes exhibit time-varying behavior. Causes for such a behavior are changes in the measuring devices, environment changes, and changes of process behavior or of some external process condition (Vergha et al., 2012). It is important to develop on-line adaptive methodologies that should be able to handle time varying behavior in prediction settings.

The main adaptive mechanisms to deal with concept drift can be classified as: instance selection (A1), instance weighting (A2), and ensemble learning (A3). Instance selection approaches obtain a set of relevant samples of the current concept. A common strategy is the moving window (MW), in which a window slides along the data and the newest samples are included to be taken into consideration into the model and the oldest samples are excluded from the model. One difficulty is the selection of the window’s size. In in-
stance weighting (IW), samples are weighted according to their age and/or relevance to the current concept. Recursive methods, where the models’ parameters are updated over time, belong to this category. The method most commonly used is the Recursive Partial Least Squares (RPLS) proposed in (Qin, 1998). In this work, a currently existing Partial Least Squares (PLS) model is merged with the new data using a forgetting factor that is employed to update the model. Recent RPLS models update the mean and variance of data using a MW approach to track the process changes (Ahmed et al., 2009). In general, the RPLS works well in cases where the process dynamics are well represented in the initial training data set. On the contrary, the model may not track the process dynamics occurring in the new data. Boosting is also an IW approach, where samples are re-weighted in order to emphasize those samples predicted incorrectly by the previous model (Drucker, 1997).

Ensemble learning has been proven itself as a valuable tool to handle concept drift scenarios. It combines a set of models in order to get a final prediction (Chao et al., 2014; Huang and Chau, 2008). Results indicate that ensemble learning improves the generalization capability and the overall system performance (Soares et al., 2012; D’Este et al., 2014). Adaptive ensembles can combine a subset of the following strategies (Polikar, 2012): adaptation of the models’ weights ($E_1$); adaptation of the models’ parameters ($E_3$); and/or add new models in the ensemble ($E_3$) according to each incoming sample or batch of samples. Adaptive ensembles can also be classified as batch-based or sample-based if they are adapted when a batch of data or a sample is available, respectively.

Most adaptive ensembles are batch-based and focus on classification tasks, for example the Learn++.NSE (Elwell and Polikar, 2011), which is inspired by the Boosting (Drucker, 1997). When a batch arrives, Learn++.NSE identifies the samples misclassified by the ensemble and obtains a penalty distribution. The distribution is used to assign errors to each model based on its contribution to the ensemble. The weight of each model is assigned using a weighted average of their errors on the current and old batches by a sigmoid function with two slope parameters. The parameters’ setting is not an easy task, since Learn++.NSE is sensitive to their values. Moreover, Learn++.NSE requires a long time for waiting a batch, and when it is available, it may not reflect the current concept.

On the contrary, sample-based ensembles offer faster adaptation capability. Examples are the Additive Expert (AddExp) (Kolter and Maloof, 2005) and the Incremental Local Learning Soft Sensing Algorithm (ILLSA) (Kadlec and Gabrys, 2011) algorithms. AddExp uses a loss bound to obtain the models’ errors, and models’ weights are adapted according to their actual losses and a decreasing factor, employed to reduce a model’s weight when it performs poorly. The ILLSA algorithm has two phases. On the training phase, a set of models is designed, where each model is trained with samples of a different concept contained on the training data; while on the on-line phase, for each incoming sample, the models’ weights are adapted using the posterior probability by a Bayesian framework. ILLSA works well when the process dynamics are well represented in the training data set. One drawback is that few models are designed if the training data contains few concepts.

This paper proposes an on-line weighted ensemble of regressor models (OWE) which is able to learn incrementally sample by sample in the presence of several types of changes and simultaneously retain old information in recurring scenarios. OWE employs several adaptive mechanisms to deal with several types of drifts. OWE is inspired by Learn++.NSE (Elwell and Polikar, 2011). But unlike Learn++.NSE, in the OWE, the ensemble is adapted on a sample basis, leading the system to faster recovery from changes and increasing the system accuracy. Additional and new strategies are proposed to increase the OWE’s accuracy. The experiments indicate that OWE outperforms Learn++.NSE in all tests.

The key idea is to keep a fixed MW slides along data when a new sample is available. Then, the error of each model on the current window is determined using a boosting strategy (Feely, 2000; Shrestha and Solomatine, 2006) that assigns small errors to the models that predict accurately the samples predicted poorly by the ensemble. To handle recurring and non-recurring changes, OWE uses a new method for assigning the models’ weights that takes into account the models’ errors on the past and recent windows using a discounting factor that decreases or increases the contribution of old windows. In addition, OWE launches new models if the system’s accuracy is decreasing, and it can remove inaccurate models for reducing memory and computational time. The method removes the model with the largest total error rate on the current and old windows. Experiments on artificial data sets and industrial data sets are detailed to evaluate, and demonstrate the performance and the effectiveness of the OWE over the state-of-the-art concept drift approaches.

The main contributions of this work are: (1) a new on-line weighted ensemble of regressor models with faster adaptation capability; (2) regression scope (while most on-line ensemble applications for handling changes is devoted to classification tasks); (3) systematic analysis of the related ensemble algorithms; (4) thorough analysis of the experimental results using both artificial data sets and industrial data sets, demonstrating faster adaptation capability and
accuracy of the OWE over the main state-of-the-art approaches; (5) implementation of a new Learn++.NSE algorithm for regression tasks.

The paper is organized as follows. Section 2 presents the theory of concept drift. Section 3 outlines the related works. In Section 4 the OWE algorithm is described. In Section 5 the results are presented and discussed. Finally, Section 6 presents the concluding remarks.

2. The Concept Drift Problem

For analyzing the problem of concept drift (Kadlec and Gabrys, 2011; Klinkenberg, 2005), consider an on-line learning framework where samples arrive incrementally one by one, and each sample $s = (x, y)$ is composed by $r$ inputs grouped into an input vector, $x \in \mathbb{R}^{r \times 1}$, and one output $y \in \mathbb{R}$. Consider that the samples are grouped into several windows of equal size $m$:

$$
S_{w(1)}, \ldots, S_{w(m)}, S_{w(1)+1}, \ldots, S_{w(m)+1}, \ldots
$$

where $S_{w(i)}$ is the $i$-th sample of window $w$. For each window $w$, the data is independently and identically distributed according to a distribution $\mathcal{D}_w(x, y)$. If all the windows are distributed over the same distribution, the concept is considered stable and thus there is no concept drift. On the other hand, if two windows $a$ and $b$ have different data distributions, i.e. $\mathcal{D}_a(x, y) \neq \mathcal{D}_b(x, y)$, there is a concept drift. Learning algorithms to handle the concept drift problem should be able to predict the next data window (e.g. $t + 1$) using the old data windows (from 1 to $t$) or a subset of them.

Changes over time may occur in different forms. In the literature, drifts are classified with respect to their speed, cyclical nature, scope, etc (Minku et al., 2010; Elwell and Polikar, 2011; Zliobaite, 2009). The drift speed describes the rate by which old concepts are substituted by new concepts. An abrupt drift happens when an old concept is abruptly replaced by a new concept; while a gradual drift happens when an old concept is slowly substituted by a new concept. Gradual drifts are harder to identify since they result in small data shift and lower error prediction when compared to abrupt drifts.

Drifts can also be classified according to their cyclical nature. A recurring drift happens if a previously occurring concept recurs after some time; while a non-recurring drift happens if a previously occurring concept cannot recur over time. Recurring drifts may occur due to the cyclic nature of a system (e.g. due to the seasons of the year). Other drift classification is with respect to scope. A local drift affects only some regions of the instance space; while a global drift affects the whole instance space. In local drifts, changes depend on the location in the instance space; and therefore, a learning algorithm should detect such changes and adapt only those locations of the model that cover the influenced regions of the instance space (Ikonomovska, 2012).

3. Related Works

A learning algorithm that deals with changing environments can be characterized in several ways, such as classification or regression scope; an explicit or implicit algorithm (Minku and Yao, 2012; Elwell and Polikar, 2011); the types of drifts that the algorithm can deal with; the incorporated adaptive mechanisms (A1, A2, A3); the adaptive ensemble learning mechanisms (E1, E2, E3) (for the ensemble-based approaches); time step (if the algorithm learns one sample or a batch of samples at a time), etc. Table 1 lists the main ensemble-based approaches and related works.

An explicit drift algorithm constantly monitors the system to detect the starting time and the severity of drifts, allowing the algorithm to adapt to changes and continue the learning accordingly (Bosnić et al., 2014; Baena-García et al., 2006). In implicit drift algorithms, the system does not monitor the time instant when the drift occurs, and it constantly learns from the environment by constructing and organizing the knowledge. Approaches with implicit drift mechanisms usually associate to each model of the ensemble a weight based on its accuracy in the current concept. Additionally, some approaches dynamically add new models to the ensemble for learning new concepts and, remove old models that perform poorly (possibly trained with old concepts).

Table 1 reveals that most approaches employ adaptation of the models’ weights ($E1$) and addition of models ($E3$). Only AddExp and ILLSA adapt the models’ parameters ($E2$). This strategy may be useful for ensuring faster convergence of the ensemble to new concepts. However, in systems where the parameters of all the models are adjusted to the new concept, this adaptation may produce redundant models and affect the system’s performance in recurring drifts. Ensembles can employ an ensemble pruning strategy, which is based on pruning the ensemble by selecting a subset of models from the original set of models, and excluding those models that are either detrimental to the ensemble’s performance or contain redundant information, normally with the goal of limiting the number of models and/or improving the ensemble accuracy (Soares et al., 2013; Zhang and Chau, 2009).

Adaptive Classifiers-Ensemble system (ACE) creates a new model when the buffer of samples is full
or a drift is detected, where a drift mechanism monitors the confidence interval of the models’ accuracies (Nishida et al., 2005). For each new sample, the models’ weights are updated according to the models’ accuracies in that sample. ACE can remove the model with the lowest pruning weight when the number of models exceeds a threshold (Nishida and Yamauchi, 2004). Initially, when a model is created, its pruning weight \( w \) is set to 0. At each time step, if model \( j \) achieves larger accuracy on the new buffer when compared to its accuracies on the old buffers, then \( w_j \) is increased by 1. Otherwise, \( w_j \) is reduced by 1. The drawback is that the newest model may be easily excluded, since its pruning weight is set to 0.

ILLSA and AddExp are sample-based ensembles for regression. ILLSA does not use ensemble pruning strategies. AddExp is an ensemble for predicting a discrete set of classes, AddExp.D, or continuous values in the interval \([0, 1]\). AddExp.C. The models’ weights are dynamically updated according to their current losses and a decreasing factor \( \beta \). A new model is added when the ensemble’s loss is greater than a factor \( \tau \). The new model’s weight is set according to the sum of the remaining models’ loss and a factor \( \gamma \), employed to control the importance of the new model. A model can be removed when the number of models is greater than \( K \). Two pruning strategies are proposed: *oldest first*, where the oldest model is removed from the ensemble; and *weakest first*, where the model with the lowest weight is removed from the ensemble. AddExp does not reveal which samples should be taken for the initial training of a new component model.

Fast and Light Boosting (FLB) (Chu and Zaniolo, 2004), Incremental Boosting (IBOOST) (Grovoric and Vucetic, 2011), and Learn++NSE (Elwell and Polikar, 2011) are algorithms based on boosting. In the FLB, when a batch is available, the ensemble predicts it and a new model is included into the ensemble. Each sample from the batch receives a weight based on its prediction error, and a weighted training batch is obtained using the samples’ weights and is used to train the new model. Changes are detected using a statistical decision theory, and a new ensemble is created from scratch when a change is detected. This approach leads the system to bad performance in scenarios where drifts recur, since models trained on old drifts are excluded. IBoost is an ensemble that employs a stochastic gradient descent procedure to dynamically assign the models’ weights. It uses a MW to keep the most recent samples. At a pre-defined frequency \( p \), a model is added to the ensemble, if the ensemble misclassifies the newest sample. At this frequency, if the number of models exceeds a threshold, then the model with the lowest weight is excluded. At each time step, the models with weights less than zero are removed from the ensemble. Results in (Grovoric and Vucetic, 2011) show that IBoost has faster adaptation than batch-based ensembles. The Learn++.NSE algorithm suggests two pruning strategies when the number of models is greater than a threshold (Elwell and Polikar, 2009): the oldest model is excluded, or the model with the largest current error is excluded.

Boosting algorithms were firstly developed for solving binary classifications problems. Freund and Schapire (1997) proposed the first regression boosting algorithm called AdaBoost.R. The main idea is to map each regression sample into an infinite set of binary classification samples. Although it has theoretical proof of its convergence, the number of classification samples grows linearly in each iteration, hindering its practical application. Drucker (1997) proposed the AdaBoost.R2 algorithm, a modification of the AdaBoost.R, that has promising results. AdaBoost.R2 uses loss functions to convert regression loss into the domain of classification loss. Big Error Margin (BEM) boosting (Feely, 2000) is quite similar to AdaBoost.R2. However, BEM is less sensitive to noise and the system can handle weak learners with larger errors. In BEM, the absolute predictive error of a sample is compared to a pre-defined threshold, and the corresponding sample is demarcated as incorrect or correct. The BEM algorithm has a problem when the variability of the real values is very high. To overcome this drawback, in the OWE proposed in this pa-

<table>
<thead>
<tr>
<th>Approach</th>
<th>Scope</th>
<th>Types of drifts</th>
<th>Drift mech.</th>
<th>Adaptive mechanisms</th>
<th>Ensemble learn. mech.</th>
<th>Time step</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACE</td>
<td>classification</td>
<td>recurring drifts</td>
<td>explicit</td>
<td>A3</td>
<td>E1, E3</td>
<td>sample basis</td>
</tr>
<tr>
<td>AddExp</td>
<td>sample/regression</td>
<td>all the types</td>
<td>implicit</td>
<td>A2, A3</td>
<td>E1, E2, E3</td>
<td>batch basis</td>
</tr>
<tr>
<td>FLB</td>
<td>classification</td>
<td>gradual and abrupt drifts</td>
<td>explicit</td>
<td>A2, A3</td>
<td>E3</td>
<td>batch basis</td>
</tr>
<tr>
<td>IBoost</td>
<td>regression</td>
<td>all the types</td>
<td>implicit</td>
<td>A1, A2, A3</td>
<td>E1, E3</td>
<td>sample basis</td>
</tr>
<tr>
<td>ILLSA</td>
<td>regression</td>
<td>all the types</td>
<td>implicit</td>
<td>A3</td>
<td>E1, E3</td>
<td>sample basis</td>
</tr>
<tr>
<td>Learn++NSE</td>
<td>regression</td>
<td>all the types</td>
<td>implicit</td>
<td>A1, A2, A3</td>
<td>E1, E3</td>
<td>sample basis</td>
</tr>
<tr>
<td>OWE</td>
<td></td>
<td>all the types</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: The main ensemble-based approaches to deal with the concept drift problems.
per, a threshold is used to demarcate an incorrectly or correctly predicted sample based on its absolute relative predictive error, as in the AdaBoost.RT algorithm (Shrestha and Solomatine, 2006).

4. On-line Weighted Ensemble of Regressor Models (OWE)

This section details the new On-line Weighted Ensemble of Regressor Models (OWE) algorithm. OWE incorporates all the main adaptive mechanisms (A1, A2, A3) to deal with the problem of concept drift. OWE employs the common assumption that the most recent data provides the best and most relevant representation of the current concept and near-future concept; and only this input data should be kept (Brzeziński and Stefanowski, 2014; Klinkenberg, 2005) (but the ensemble model keeps information about other past concepts). For this purpose, a fixed MW is used to keep the most recent set of samples (A1). These samples are employed to obtain the ensemble’s accuracy based on the error predictions, and to train a new model. Additionally, OWE also incorporates instance weighting (A2) mechanisms based on boosting (Feely, 2000; Shrestha and Solomatine, 2006). That is, a weighted distribution of the ensemble’s error on the current window is obtained, and then the error of each model is calculated based on its contribution to the ensemble. This contribution is seen as the ability of a model to predict accurately the samples poorly predicted by the ensemble. Moreover, OWE is an ensemble learning (A3) algorithm that takes into account that the data exhibits time-varying behavior. The main adaptive ensemble mechanisms for dealing with concept drift are: (E1) adaptation of the models’ weights (with respect to their contributions on the recent and old windows); (E3) dynamic inclusion of models when the ensemble’s performance is degrading; and removal of models over time (more details in Table 1). Adaptation of the models’ parameters (E2) is not employed, since it usually results in redundant models and inaccuracy in recurring drifts.

Steps of the OWE algorithm are detailed in Algorithm 1. The inputs of the algorithm are (Step 1): a data set \( D = \{(x_i, y_i) \mid x_i \in \mathbb{R}^{\times r}, y_i \in \mathbb{R}, i = 1, \ldots, M\} \), where \( x_i \) is a vector of \( r \) inputs, \( y_i \) is the output and \((x_i, y_i)\) is a sample; the window’s size, \( m \); a factor for demarcating correct and incorrect predictions, \( \theta \); a factor to control the inclusion of a new model, \( \alpha \); a discount factor, \( \kappa \); where \( 0 < \theta, \alpha < 1 \) and \( 0 \leq \kappa < 1 \); a generic supervised learning algorithm for regression, Weak Learner; a pruning activation factor, \( \rho \); and the maximum number of models, \( B \) (enforced if and only if (iff) \( \rho \) is activated). In Step 2, the initialization of some variables is done. Variable \( k \) denotes the number of models in the ensemble, \( f_k \) denotes the most recently designed model, and \( t \) is the time step. \( D' \) is a data window (of size \( m \)) produced at time \( t \).

The penalty distribution (or window data weights), \( \mathcal{D} = \{d_1, \ldots, d_m\} \), holds the weights of each sample on the current data window, where each weight is based on the ensemble prediction accuracy on a sample. \( \mathcal{D} \) is firstly initialized to be an uniform distribution \([1/m, \ldots, 1/m] \). The penalty distribution is not employed during the model training as instance weighting, e.g. for resampling the training dataset. It is considered that all the training samples have the same weight/contribution during the training process. Step 3 is repeated when a new sample from data set \( D \) becomes available. The data window \( D' \) is firstly filled with the first \( m \) samples of \( D \), if \( t \) is not greater than \( m \) (Step 3a(ii)). Otherwise, the window slides along \( D \) (Step 3a(i)). In Step 3b, the algorithm is directed to create the first model if \( t \) is equal to \( m \). In Step 3c, the ensemble \( F(\cdot) \) is used to predict the new sample. The ensemble’s output is obtained by a weighted sum of the models’ outputs using a logarithm function. Step 3d obtains the absolute relative error of the samples predicted by the ensemble using \( D' \). \( \text{ARE}_D^F = \{\text{ARE}_1^F, \ldots, \text{ARE}_m^F\} \). Each error of a sample \( \text{ARE}_t^F \) (where \( i = s, \ldots, t \)) is obtained using the real output value \( y_i \) and the predicted output of the ensemble \( F(x_i) \), where \((x_i, y_i) \in D' \).

In Step 3e, samples of \( D' \) are demarcated as incorrectly or correctly predicted by the ensemble using a threshold \( \theta \). The objective is to count the total number of samples incorrectly predicted by the ensemble for obtaining the new penalty distribution. OWE employs concepts similar to the BEM boosting (Feely, 2000) and to the AdaBoost.RT (Shrestha and Solomatine, 2006) boosting regression algorithms. The strategy works as follows: if an error \( \text{ARE}_i^F \) (of a sample \( i \in D' \) predicted by a component model, \( \mu = 1, \ldots, k \), or the ensemble, \( \mu = F \); Steps 3e, 3(g,ii), 3i) is greater than \( \theta \), then sample \( i \) is demarcated as incorrect, otherwise as correct. As in the AdaBoost.RT and BEM algorithms, OWE is sensitive to the setting of \( \theta \). The methods perform well when \( \theta \) is between 0 and 0.4.

Step 3f determines the values of variables \( \text{upFactor} \) and \( \text{downFactor} \) using \text{totalSamples} \. The variable \( \text{upFactor} \) increases the weights of the samples predicted incorrectly by the ensemble and \( \text{downFactor} \) decreases the weights of the samples predicted correctly by the ensemble. In Step 3g, the values of \( \mathcal{D} \) are updated. First, in Step 3g(ii), distribution \( \mathcal{D} \) is reinitialized to be uniform. In Step 3g(iii), each weight \( d_i \) of a sample \( i \) is obtained. The main idea is to assign larger weights to the samples predicted incorrectly by the ensemble and small weights to the samples predicted correctly by the ensemble.

In Step 3h, a new model is launched to the ensemble if the absolute relative error of the ensemble on the
Algorithm 1 On-line Weighted Ensemble of Regressor Models (OEW)

1. **Inputs:** a data set $D = \{(x_i, y_i)\}_{i=1}^m$; window’s size, $m$; $\theta$; for demarcating correct and incorrect predictions; $\alpha$, factor to include a new model; $k$, discounting factor; a supervised learning algorithm, Weak Learner; pruning activation, $\rho$; maximum number of models, $B$ (enforced if $\rho$ is activated); 

2. **Initialization:** Set $s = 1$; $k = 0$; $D^0 \leftarrow \emptyset$; the distribution $\mathcal{D} = [d_1, \ldots, d_m]^T$, where $d_i = 1/m$, for $i = 1, \ldots, m$; 

3. for $t = 1, \ldots, M$:
   
   (a) if ($t > m$)
      
      i. then slide the window: $s \leftarrow s + 1$; $D^t = D^{s-1} + (x_s, y_s) - (x_{s-m}, y_{s-m})$;
      
      ii. else fill the window: $D^t = D^{s-1} + (x_s, y_s)$;
   
   (b) if ($t = m$) then go to Step 3(h); else if ($t < m$) then go to Step 3;
   
   (c) Predict $y_t$ as: $y_t^e = F(x_t) = \left( \sum_{j=1}^m \log \left(1/\hat{E}_j \right) f_j(x_t) \right) / \sum_{j=1}^m \log \left(1/\hat{E}_j \right)$;
   
   (d) Obtain the error of the ensemble on each sample of $D^t$: $\text{ARE}_i^t = \| (F(x_i) - y_i)/y_i \|$, for $i = s, \ldots, t$;
   
   (e) Count the number of samples incorrectly predicted by the ensemble: $\text{totalSamples} = \sum_{i=1}^t [\text{ARE}_i^t > \theta]$;
   
   (f) Calculate $\text{upFactor}$, and $\text{downFactor}$ as: $\text{upFactor} = m/\text{totalSamples}$, and $\text{downFactor} = 1/\text{upFactor}$;
   
   (g) Update and normalize the distribution $\mathcal{D}$:
      
      i. Set $d_i = 1/m$, for $i = s, \ldots, t$;
      
      ii. if ($\text{ARE}_t^t > \theta$), then $d_i \leftarrow d_i \times \text{upFactor}$, else $d_i \leftarrow d_i \times \text{downFactor}$, for $i = s, \ldots, t$;
      
      iii. $d^{(\text{new})} \leftarrow d_i/\sum_{i=s}^t d_i$, for $i = s, \ldots, t$; $d_i \leftarrow d^{(\text{new})}$, for $i = s, \ldots, t$;
   
   (h) Check if a new model should be included to the ensemble:
      
      if ($\| (F(x_i) - y_i)/y_i \| > \alpha$)
      
      i. then Set $k \leftarrow k + 1$; set $\tau_k = 0$; Call Weak Learner and obtain a new model $f_k(x)$ trained with $D^t$; include $f_k$ to the ensemble;
      
      (i) Evaluate the models using $D^t$ and obtain their current error rates using $\mathcal{D}$, for $j = 1, \ldots, k$:
      
      $\text{ARE}_j^t = \| (f_j(x_i) - y_i)/y_i \|$, for $i = s, \ldots, t$; $\tau_j \leftarrow \tau_{j+1} + 1$; and $e_j^t = \sum_{i=s}^t d_i [\text{ARE}_i^t > \theta]$;
      
   (j) Determine the total error rate of each model on the past and recent windows using the decreasing factor $\kappa$:
      
      $\hat{E}_j = \left( \sum_{j=1}^m \kappa^{\tau_{j+1} - \tau_{j}} \text{ARE}_j^t \right) / \sum_{j=1}^m \kappa^{\tau_{j+1} - \tau_{j}}$, for $j = 1, \ldots, k$;
      
   (k) Check if a model should be removed:
      
      if ($\rho = 1$) and ($k > B$), then
      
      i. Exclude the model $f_j$ where $j = \text{argmax}_{j=1,\ldots,k} \hat{E}_j$, and set $k \leftarrow k - 1$; and
      
      ii. Renumber models: $\sigma_q \leftarrow \sigma_{q+1}$, for $\sigma_q \in \{f_{\tau_q}, \tau_q, e_{\tau_q}, \hat{E}_q\}$, $q = j, \ldots, k$, $p = 1, \ldots, \tau_q$;

4. end for

newest sample is greater than $\alpha$. The new model is trained using all samples of $D^t$. Then, all the models are evaluated based on their predictions of $D^t$ and their contributions to the ensemble (using $\mathcal{D}$) in the Step 3i. In summary, for each model $j$, it is obtained the absolute relative error $\text{ARE}_j^t$ for each sample $(x_i, y_i) \in D^t$. The current error rate $e_j^t$ is calculated using the sum of the weights (from $\mathcal{D}$) of the samples demarcated as incorrectly predicted by model $j$. Therefore, since the sum of the elements of $\mathcal{D}$ is 1, $e_j^t$ can assume values between 0 and 1. As mentioned before, large errors are given to the models that predict poorly the samples predicted incorrectly by the ensemble; and small errors are given to the models that predict correctly the samples predicted incorrectly by the ensemble. In this Step, the variable $\tau_j$ for each model $j$ is incremented. It holds the total number of windows where model $j$ has been evaluated.

The total error rate $\hat{E}_j$ of each model $j$ is calculated using a discounting factor that weights the model’s errors on the past and recent windows (Step 3j). The discounting factor weights recent errors more heavily than old errors for obtaining $\hat{E}_j$ as:

$$
\hat{E}_j = \left( \sum_{p=1}^{\tau_j} \kappa^{\tau_{j+1} - \tau_p} e_j^p \right) / \sum_{p=1}^{\tau_j} \kappa^{\tau_{j+1} - \tau_p},
$$

where $j$ is a model of the ensemble; $\kappa$ is the discounting factor with $0 < \kappa < 1$; and $e_j^p$ denotes the error of model $j$ on a window $p$. Equation 1 can be rewritten as:

$$
\hat{E}_j = \frac{1}{\sum_{p=1}^{\tau_j} \kappa^{\tau_{j+1} - \tau_p}} \left( e_j^{\tau_{j+1}} + \ldots + \kappa^{\tau_{j+1} - 1} e_j e_j^{\tau_{j+1} - 1} + \kappa^\theta e_j^\theta \right).
$$

(2)
The sequence of weights $\kappa^{T_i-p}$ is decreasing with the oldness of the window, $\tau_j - p$, so that $\kappa^{T_j-1} \leq \ldots \leq \kappa^1 < \kappa^0$, as can be observed in the Figures 1 and 2. Also, smaller values of $\kappa$ imply that lower weights are given to the errors on the old windows. This case works well in non-recurring drifts, since more importance is given to the current scenario/concept. Larger values of $\kappa$ imply that larger weights are given to the errors on the old windows. This case performs well in scenarios with recurring drifts, since more importance is given to the errors on the old windows. The proposed approach employs concepts of the discounted Mean Square Forecast Error (MSFE) method (Shen et al., 2008). It uses discounting factors to obtain weights of each forecast in an ensemble system. This strategy is simple and easy to apply, and it is simple to tune the discount factor value. Other strategy is Weighted Majority Algorithm (WMA) which combines a set of models using a weighted majority vote of the models’ predictions (Littlestone and Warmuth, 1994). When a model incorrectly classifies a sample, then WMA decreases its weight by a constant. Therefore, the weight is discounted only when it performs poorly on the newest sample. AddExp also employs the same concept. Learn++.NSE employs a sigmoid function with two parameters to decrease or increase the contribution of the old batches. However, the parameters’ setting is not an easy task, and Learn++.NSE is sensitive to their values.

In Step 3k, if the pruning factor $\rho$ is activated by the user, then a model is removed if the number of models is greater than the threshold $B$. The pruning strategy removes the model with the largest total error rate $\hat{e}_j$. Note that a new model $f_k$ created at sample-iteration $t$ is never removed by the pruning strategy at that same iteration $t$.

5. Experimental Results

Experimental results are detailed in this section to compare OWE to the state-of-the-art approaches. The approaches are evaluated on different scenarios using two artificial data sets and two industrial data sets. The use of artificial data sets allows the control of relevant parameters and to empirically evaluate the algorithms in several types of changes. There is a lack of artificial data sets to simulate changing environments for regression tasks. In this paper, it is used the hyper-plane data set proposed by (Kolter and Maloof, 2005), a benchmark for evaluating algorithms that deal with concept drifts for both classification and regression tasks; and the drifting Friedman’s function proposed by (Ikonomovska, 2012), a recent data set created for evaluating regression algorithms in changing environments. The real-world data sets enable us to evaluate the merit of the proposed approach in real-world problems, and compare it to the most recent works in real-life problems. However, it may not be possible to precisely state when drifts occur or if there is any drift at some specific time instant. This paper uses two well-known industrial data sets widely employed to evaluate algorithms for dynamic system modeling: the catalyst activation data set, which has slowly changing process dynamics due to the catalyst decay over the period of one year (Kadlec and Gabrys, 2011; Ni et al., 2012); and the Fluidized Catalytic Cracking Unit (FCCU) data set, a benchmark for evaluating dynamic systems (Liu et al., 2009; Yan et al., 2004). Other public data sets for soft sensor modeling are the debutizer column and the sulfur recovery unit proposed by (Fortuna et al., 2006). However, they are rarely employed to evaluate systems with time-varying behavior. The tests have been performed on the Matlab environment, running on a PC equipped with an Intel Core i7-2600 3.4GHz process of 4 cores and 8GB of RAM.

5.1. Experimental Setup: Data Set Description

Drifting Friedman’s function. The Friedman’s function is a well-known function for producing benchmark data (Friedman, 1991). It has linear and non-linear relations between inputs variables and output variable. The function contains 5 input variables, $x^1, \ldots, x^5$, and 1 output variable, $y_i$, and is given by:

$$y_i = 10\sin(\pi x^1_i x^2_i) + 20(x^3_i - 0.5)^2 + 10x^4_i + 5x^5_i + \epsilon,$$  \hspace{1cm} (3)

where $\epsilon \sim N(0, 1)$ is a zero-mean, unit-variance Gaussian random variable. The input space is enlarged by including other 5 input variables $x^6, \ldots, x^{10}$ that are not relevant for predicting $y$. The input variables are
uniformly distributed over the interval of \([0, 1]\). To create drifting scenarios, 3 drifting data sets using the original Friedman’s function were produced according to (Ikonomovska, 2012), each one with 2000 samples. The first data set, **local and abrupt drift** data set (Friedman-LA), introduces changes in 2 different regions of the input space (local drift) using 3 points of abrupt changes. The second data set, **global recurring abrupt drift** data set (Friedman-GRA), has global, abrupt, and recurring drifts introduced in 2 drift points of the data. The third data set, **global non-recurring gradual drift** data set (Friedman-GnRG), contains 2 points at which gradual concept changes are beginning to be introduced. It is generated by slowly introducing samples which belong to a different function in contrast to the initial samples. The complete description of these drifting data can be found in (Ikonomovska, 2012).

**Drifting hyperplane data set.** It is a well-known drifting data set used to evaluate algorithms that deal with concept drift (Minku et al., 2010; Shaker and Hüllermeier, 2012). It contains noise, and gradual and non-recurring drifts, and is similar to the one proposed in (Kolter and Maloof, 2005) (AddExp). The whole data set consists of 10 inputs with uniform distribution over the interval of \([0, 1]\) and 2000 samples \((M = 2000)\). The data set contains 4 concepts, where each concept holds 500 samples. The outputs of the data set are given by:

- **Concept 1:** \(y_i = (x_i^1 + x_i^2 + x_i^3)/3\), for \(i = 1, \ldots, M\);
- **Concept 2:** \(y_i = (x_i^2 + x_i^3 + x_i^4)/3\), for \(i = (M/4 + 1), \ldots, M/2\);
- **Concept 3:** \(y_i = (x_i^4 + x_i^5 + x_i^6)/3\), for \(i = (M/2 + 1), \ldots, 3M/4\);
- **Concept 4:** \(y_i = (x_i^7 + x_i^8 + x_i^9)/3\), for \(i = (3M/4 + 1), \ldots, M\).

A random variance noise uniformly distributed in the interval of \([-0.1, 0.1]\) is injected to each output sample \(y_i\) (for \(i = 1, \ldots, M\)). The value of \(y_i\) is clipped to 0 or 1 if its value is less than 0 or greater than 1, respectively.

**Catalyst activation data set.** The data set is a benchmark for evaluating adaptive soft sensing algorithms (Kadlec and Gabrys, 2011). It describes a polymerization reactor which consists of 1000 tubes filled with catalyst, used to oxidize a gaseous feed. The reaction speed has a nonlinear relation to the temperature of the coolant which is used to cool the reactor. The plant is equipped with measurements to log all the variations of the feed and the operating conditions of the process. Moreover, measurements are done to show the

\[ x_i^n \] denotes the value of the input \(n\) of a sample \(i\), where \(x_i \in \mathbb{R}^{r \times n}\), \(r\) is the number of inputs and \(n = 1, \ldots, r\).
In all the pruned algorithms, the maximum number of models in the ensemble is set to 20. This choice was considered the best suitable for all the ensembles, since the maximum number of models usually varies between 15 and 30 (Elwell and Polikar, 2009; Kolter and Maloof, 2005; Nishida and Yamauchi, 2007), and the use of more models linearly increases the processing time of the experiments.

Either PLS (Qin, 1998) or RPLS (Ahmed et al., 2009) are used as base models in the methods studied in the experiments. The PLS is able to deal with large dimensional co-linear data by projecting the input and outputs into a new space. The PLS projects the scaled and mean-centered input and output data into separate latent variables. This paper uses the Statistically Inspired Modification of the Partial Least Squares (SIMPLS) algorithm (de Jong, 1993) available in the PLS Toolbox (Wise et al., 2003). The SIMPLS calculates the PLS components faster and more accurately when compared to the other PLS algorithms. The optimal number of latents is determined by 10-fold cross-validation using the sum of squared prediction errors (PRESS) between the real output and the predicted output.

The following structure is performed in all the algorithms (except ILLSA). Consider a data set $D = \{(x_i, y_i)\}_{i=1}^M$ with $M$ samples. The first model is designed using the first $m$ samples from $D$. While the other ($M - m$) samples of $D$ are grouped to form the on-line data to simulate an on-line scenario. Each method is evaluated using the Mean Square Error (MSE), which is calculated using the predicted outputs and the real outputs from the on-line data. MSE is a widely used metric to evaluate models. It provides a quadratic loss function that penalizes larger errors. MSE has the disadvantage of heavily weighting outliers. But as the data sets are free of outliers, this paper considers MSE as a metric to evaluate models.

The RPLS is implemented by updating recursively the mean and variance data, where the oldest sample is excluded and the newest sample is included into the model simultaneously (Ahmed et al., 2009). This method can be seen as a MW approach, since the model is always trained using a fixed number of the most recent samples. The RPLS allows the adaptation to the new events and the partial retention of the process history. Here, the RPLS starts by calculating the mean and variance of the initial training data. Then, the data are mean-centered and variance-scaled to design a model using the SIMPLS. When a new sample is available, the model is employed to predict it. Then, the mean and variance data are updated using the new sample. Next, the new sample is included to, and the oldest sample is removed from, the previous training data set. The data are scaled and the SIMPLS is repeated to design a new model.

The models of the AddExp.C are implemented using RPLS models, each one trained using the most recent $m$ samples. The parameters are set based on the pilot studies from (Kolter and Maloof, 2005): $\beta = 0.5$, $\gamma = 0.1$, and $\tau = 0.05$. In our experiment, the ensemble is adapted on a sample basis. If a new model should be included at time step $t$, its training data is obtained as: $D^t = \{(x_i, y_i)\}_{i=m+1}^{m+D} \subset D$. As the AddExp.C requires the output data to be normalized to the interval of $[0, 1]$, the outputs of all the data sets for all the methods are normalized to this interval.

As the Learn++.NSE is an algorithm for classification tasks, a new scheme is proposed to adapt it for regression tasks. Learn++.NSE was implemented using a boosting regression algorithm, the AdaBoost.RT (Shrestha and Solomatine, 2006). Each time step of the Learn++.NSE consists of a batch of samples (it can be viewed as the window’s size). Therefore, the batches are considered to have size $m$. The Weak Learner is the SIMPLS. Parameters of the Learn++.NSE are set according to the authors’ suggestions (Shrestha and Solomatine, 2006): $a = 0.5$ and $b = 10$.

The ILLSA is implemented according to the works (Kadlec and Gabrys, 2011; Miranda, 2012). The ILLSA is simulated by dividing the data set $D$ into two data sets: 30% of $D$ is used as a training data set (the initial samples of $D$), for building the pool of RPLS models; and 70% of $D$ is used as an online data set (the ending samples of $D$). Therefore, the on-line data set of ILLSA is different from the other methods. Here, even if only one concept is detected on the training data set to train one model, two models are designed to assure that an ensemble is designed. ILLSA does not have an ensemble pruning strategy. The size of the initial window, $n_{\text{init}}$, is set as $m$. In each experiment, the optimal values of the kernel size ($\sigma$) and the kernel size for the adaptation masks ($\sigma_{\text{adapt}}$) are chosen by 10-fold cross-validation (using the training data) using values in the range of $[10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}]$.

The OWE and pruned OWE are implemented according to Algorithm 1, where the SIMPLS is used as a Weak Learner. Based on pilot studies, $\theta$ is set to 0.05 for the OWE, pruned OWE and Learn++.NSE; since in our tests, large values of $\theta$ produce unstable systems. For the pruned OWE, $\rho$ is set to 1 and $B$ is set to 20; and for the OWE, $\rho$ is set to 0. For each experiment, the result is obtained by averaging 20 independent runs.

### 5.3. Analysis of OWE Parameters

The parameters’ setting is discussed in this section. In on-line ensembles, the frequency of adding new models may impact the ensemble’s performance, and the discounting factor $\kappa$ can be tuned according to the
data characteristics. Based on pilot studies, tests of the OWE algorithm are conducted by varying \( \alpha \) from 0.01 to 0.1 in steps of 0.01 (\( \alpha \) lower than 0.01 produces a very large number of models and increases the computational time, while \( \alpha \) greater than 0.1 may produce inaccurate ensembles); varying \( \kappa \) from 0 to 0.95 (in steps of 0.05) and also using \( \kappa = 0.99 \) (its values range over all the interval of \( 0 \leq \kappa < 1 \) for analyzing its behavior); and varying \( m \) in the following ranges:

- \( m \in [20, 30, 40, 50, 60] \), for the artificial data sets;
- \( m \in \{10, 15, 20, 25, 30\} \), for the industrial data sets.

These ranges of \( m \) are sufficient to analyze the behaviors with the other parameters.

It has been observed that for the hyperplane data set and the Friedman-GnRG data set, the OWE’s performance improves when \( \alpha \) increases. While for the other artificial data sets, OWE has almost constant accuracy when \( \alpha \) varies. For the industrial data sets, the accuracy of OWE slightly improves when \( \alpha \) decreases. From the performed analysis it is seen that the most adequate \( \alpha \) depends on the characteristics of each data set, i.e. in data sets that require faster adaptation capability, \( \alpha \) should be set to a low value for including new models at a higher frequency. For data sets with small amounts of concept changes and that require low adaptation, \( \alpha \) should be set to a large value for adding new models at a low frequency. In the tests below, \( \alpha \) is set to 0.10 for the artificial data sets, and to 0.04 for the industrial data sets.

The MSE results of the OWE when \( \kappa \) varies using all the data sets are shown in Figure 3. The tests show that for the hyperplane data set and Friedman-GnRG data set, OWE performs well when \( \kappa \) assumes small values; And in this case, the MSE increases substantially when \( \kappa \) is 0.99. This happens because both data sets have non-recurring drifts and more importance should be given to the current concept. Therefore, the ensemble has better performance when the total error rate of each model is assigned by decreasing the contribution of the old window errors, and consequently, increasing the contribution of the recent window errors. In contrast, on the Friedman-GRA data set, which is a recurring drift data set, OWE improves significantly its performance when \( \kappa \) is large, since the total error of the models takes more into account the errors on the old windows. For the Friedman-LA data set, OWE has almost constant accuracy when \( \kappa \) varies. In the catalyst activation data set, it is observed that OWE performs well when \( \kappa \) is low; while in the FCCU data set, OWE oscillates its performance when \( \kappa \) varies. In the tests below, \( \kappa \) is set to 0.2 for all the data sets, except for the Friedman-GRA where \( \kappa \) is set to 0.99.

5.4. Experimental Results Using Artificial Data Sets

**Results.** In this section, results of the algorithms using the artificial data sets are detailed and analyzed. Results of the ILLSA can be hidden in some figures due to its large errors when compared to the other methods, making difficult the analysis of the experiments. The first test aims to determine the impact of the window’s size on the algorithms’ accuracies. Figure 4 shows the algorithms’ errors when the window’s sizes \( m \) varies from 10 to 150 (in steps of 5). This range was chosen in pilot tests that indicated that, for all the approaches, the accuracies do not significantly improve when \( m \) is greater than 150 (see Figure 4). As can be observed, \( m \) is a factor that may influence the algorithms’ accuracies. The results indicate different behaviors when \( m \) varies. But in most cases, the approaches’ accuracies increase when \( m \) is large. Figures 4b and 4c show that all the algorithms’ errors do not significantly reduce after \( m = 100 \). Table 2 shows the average and standard deviation of the MSE results of all the approaches on all values of \( m \); bold font is used to indicate the best result in each data set. The MSE results reveal that OWE and pruned OWE have the best accuracies in most cases. The exception is in the hyperplane data set, where AddExp.C outperforms \((6.34 \times 10^{-3})\) OWE \((13.35 \times 10^{-3})\) and pruned OWE \((6.95 \times 10^{-3})\) on average. In the Friedman-LA data set, OWE and pruned OWE have achieved the lowest errors \((6.46 \times 10^{-7} \text{ and } 7.11 \times 10^{-3})\) respectively; while other approaches have MSE greater than \(8 \times 10^{-3}\) on average. In the Friedman-GRA data set, OWE and pruned OWE also outperform \((11.78 \times 10^{-3} \text{ and } 11.81 \times 10^{-3})\) respectively) AddExp.C, pruned AddExp.C, RPLS, Learn++NSE and pruned Learn++NSE \((13.08 \times 10^{-3}, 13.13 \times 10^{-3}, 13.67 \times 10^{-3}, 13.87 \times 10^{-3}\) and \(13.91 \times 10^{-3}\) respectively) on average. In the Friedman-GnRG data set, pruned OWE has the lowest error \((10.86 \times 10^{-3})\), while other methods have errors greater than \(12 \times 10^{-3}\). It is observed that OWE and pruned OWE have the lowest standard deviation of the MSE when compared to the state-of-the-art approaches.

A test is applied to show the impact of the ensemble size (maximum number of models) on the algorithms’ accuracies. Figure 5 shows the MSE errors of the pruned ensembles when the maximum number of models is increased (\( m \) is set to 40). For the drifting Friedman data sets, the Learn++NSE and the AddExp.C have constant accuracies when the maximum number of models increases; while the OWE tends to reduce the error when the maximum number of models increases. However, for the hyperplane data set, the OWE performs better when the maximum number of models is small. Table 3 shows some details of all the approaches using the artificial data sets. The values are calculated by obtaining the average and standard deviation of the running (computation) time and the num-
and pruned Learn

Figure 3: OWE’s accuracy using different values of the discounting factor ($\kappa$).

Table 2: Average and standard deviation of the MSE$^*$ by varying the value of $m^\dagger$ using the artificial data sets.

<table>
<thead>
<tr>
<th>Data set/Approach</th>
<th>Hyperplane</th>
<th>Fried.-LA</th>
<th>Fried.-GRA</th>
<th>Fried.-GnRG</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPLS</td>
<td>6.62 (2.46)</td>
<td>8.46 (2.53)</td>
<td>13.67 (4.24)</td>
<td>12.37 (3.72)</td>
</tr>
<tr>
<td>ILLSA</td>
<td></td>
<td></td>
<td>38.13 (9.90)</td>
<td></td>
</tr>
<tr>
<td>AddExp.C</td>
<td>6.34 (1.80)</td>
<td>8.07 (1.97)</td>
<td>13.08 (3.34)</td>
<td>11.87 (3.00)</td>
</tr>
<tr>
<td>Pruned AddExp.C</td>
<td>6.37 (1.87)</td>
<td>8.12 (2.03)</td>
<td>13.13 (3.42)</td>
<td>11.91 (3.04)</td>
</tr>
<tr>
<td>Learn$^{++}$.NSE</td>
<td>8.72 (1.63)</td>
<td>8.18 (1.94)</td>
<td>13.87 (3.46)</td>
<td>12.29 (3.15)</td>
</tr>
<tr>
<td>Pruned Learn$^{++}$.NSE</td>
<td>8.75 (2.13)</td>
<td>8.25 (2.13)</td>
<td>13.91 (3.73)</td>
<td>12.35 (3.35)</td>
</tr>
<tr>
<td>OWE</td>
<td>13.35 (1.06)</td>
<td>6.46 (0.28)</td>
<td>11.78 (0.37)</td>
<td>12.04 (0.19)</td>
</tr>
<tr>
<td>Pruned OWE</td>
<td>6.95 (0.96)</td>
<td>7.11 (0.82)</td>
<td>11.81 (1.17)</td>
<td>10.86 (1.28)</td>
</tr>
</tbody>
</table>

$^*$ the values have been multiplied by $10^3$;
$^\dagger$ $m$ is varied from 10 to 150 (in steps of 5).

ber of models using different values of $m$ for each algorithm. Learn$^{++}$.NSE and pruned Learn$^{++}$.NSE have

the lowest running time among all the methods (including the single model RPLS). OWE and AddExp.C
produce larger number of models when compared to ILSA and Learn++ .NSE. It can be observed that the ILSA produces less models than the other methods.

**Discussion.** ILSA has achieved the largest error when compared to the other approaches. This is because, in ILSA, each model is trained with a different concept of the training data set; And, if the training data contains few concepts (for example, the Friedman-GRA data set, which contains one concept in the training data set), few models are built and they may be insufficient to deal with the dynamics of the data during the on-line phase. Regarding Learn++ .NSE and RPLS, they rarely outperform OWE and AddExp.C algorithms. In the Learn++ .NSE, the ensemble is adapted only when a new batch is available, requiring a long period of time for system adaptation. In contrast, sample-based ensembles (such as the OWE and the AddExp.C) have faster adaptation capability, since the ensembles are adapted on a sample basis. The RPLS is also adapted on a sample basis, but it employs few strategies to deal with concept drifts: the only strategy is to recursively include each new sample into the model at each time.

AddExp.C has good prediction performance when compared to the Learn++ .NSE and RPLS algorithms; and to the OWE for some values of \( m \) (mainly in the hyperplane data set). AddExp.C outperforms RPLS since AddExp.C is a dynamic ensemble of models, while RPLS is composed of only one model. This shows that an ensemble is generally more accurate than any single model. AddExp.C outperforms Learn++ .NSE, because in the AddExp.C, the ensemble is adapted when a sample is available (rather than on a batch basis). AddExp.C has worse performance when

![Figure 4: Results of the approaches in the artificial data sets using different window’s sizes (m).](image)

**Table 3:** Average and standard deviation of the n. of models and running time (minutes) of the approaches in the artificial data sets by varying the value of \( m \).

<table>
<thead>
<tr>
<th>Data set / Approach</th>
<th>Hyperplane</th>
<th>Friedman-LA</th>
<th>Friedman-GRA</th>
<th>Friedman-GnRG</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>n. of models</td>
<td>run. time</td>
<td>n. of models</td>
<td>run. time</td>
</tr>
<tr>
<td>RPLS</td>
<td>1.00 (0.00)</td>
<td>0.36 (0.04)</td>
<td>1.00 (0.00)</td>
<td>0.36 (0.02)</td>
</tr>
<tr>
<td>ILSA</td>
<td>3.34 (1.78)</td>
<td>0.13 (0.10)</td>
<td>2.86 (2.03)</td>
<td>0.09 (0.08)</td>
</tr>
<tr>
<td>AddExp.C</td>
<td>1005.83 (87.81)</td>
<td>7.28 (0.48)</td>
<td>1023.48 (92.85)</td>
<td>6.90 (0.38)</td>
</tr>
<tr>
<td>Pruned AddExp.C</td>
<td>20.00 (0.00)</td>
<td>0.81 (0.03)</td>
<td>20.00 (0.00)</td>
<td>0.81 (0.01)</td>
</tr>
<tr>
<td>Learn++ .NSE</td>
<td>41.00 (41.68)</td>
<td>0.07 (0.12)</td>
<td>41.00 (41.68)</td>
<td>0.06 (0.07)</td>
</tr>
<tr>
<td>Pruned Learn++ .NSE</td>
<td>18.45 (2.47)</td>
<td>0.04 (0.02)</td>
<td>18.45 (2.47)</td>
<td>0.04 (0.02)</td>
</tr>
<tr>
<td>OWE</td>
<td>1201.90 (22.67)</td>
<td>3.52 (0.19)</td>
<td>792.48 (40.08)</td>
<td>2.52 (0.15)</td>
</tr>
<tr>
<td>Pruned OWE</td>
<td>20.00 (0.00)</td>
<td>0.61 (0.02)</td>
<td>20.00 (0.00)</td>
<td>0.52 (0.01)</td>
</tr>
</tbody>
</table>

\( m \) is varied from 10 to 150 (in steps of 5).
Experimental results of the algorithms using industrial data sets are shown in this section. The algorithms' accuracies when the window’s sizes vary from 10 to 20 (in steps of 1) are shown in the Figure 6. This range was chosen in pilot tests that indicated that, for all the approaches, the accuracies do not significantly improve when \( m \) is greater than 20. The test indicates that the OWE has the lowest error when compared to the other methods in most values of \( m \). For the FCCU data set, most approaches achieve smaller errors for large \( m \). Table 4 shows the average and standard deviation of the MSE results of all the algorithms for all values of \( m \), where bold font indicates the best result in each data set. The MSE results indicate that OWE and pruned OWE have the best results in most data sets. In the catalyst activation data set, OWE (0.97×10\(^{-3}\)) slightly outperforms the RPLS (1×10\(^{-3}\)) on average; while other approaches have MSE greater than 1×10\(^{-3}\). In the gasoline concentration prediction (FCCU data set), OWE, pruned OWE and AddExp.C have achieved the lowest errors on average, i.e. 2.48×10\(^{-3}\), 2.65×10\(^{-3}\) and 2.92×10\(^{-3}\), respectively; while most of the other approaches have errors larger than 30×10\(^{-3}\). In the LDO concentration estimation (FCCU data set), pruned OWE and OWE have the lowest errors, i.e. 33.23×10\(^{-3}\) and 33.41×10\(^{-3}\), respectively; and other methods have

of mechanisms. For example, OWE keeps a set of diverse models trained with different parts of the data so that when an old concept recurs, old models can be re-activated and the system performs well. Additionally, in the pruned OWE, the pruning strategy removes the model with the worst performance on the old and current windows. This strategy is important to assure the ensemble’s performance in recurring scenarios, since it reduces the probability of excluding good models that belong to old concepts. OWE dynamically launches new models if the ensemble’s performance is poor on the newest sample. Furthermore, the models’ weights are obtained by taking into account their accuracies on the recent and past windows.

Common behaviors are observed in the experiments. For example, pruned ensembles outperform ensembles without pruning strategy in the non-recurring drifts; and ensembles without pruning strategy outperform pruned ensembles in the recurring drifts. In recurring drifts, since an old concept recurs, the pruning strategy may exclude models trained on an old concept. On the other hand, in non-recurring drifts, the pruning strategy can be seen as a way to remove redundant models and keep the most accurate set of models that maximize the performance on the current concept.

5.5. Experimental Results Using Industrial Data Sets

Results. Experimental results of the algorithms using industrial data sets are presented in this section. The algorithms’ accuracies when the window’s sizes vary from 10 to 20 (in steps of 1) are shown in the Figure 6. This range was chosen in pilot tests that indicated that, for all the approaches, the accuracies do not significantly improve when \( m \) is greater than 20. The test indicates that the OWE has the lowest error when compared to the other methods in most values of \( m \). For the FCCU data set, most approaches achieve smaller errors for large \( m \). Table 4 shows the average and standard deviation of the MSE results of all the algorithms for all values of \( m \), where bold font indicates the best result in each data set. The MSE results indicate that OWE and pruned OWE have the best results in most data sets. In the catalyst activation data set, OWE (0.97×10\(^{-3}\)) slightly outperforms the RPLS (1×10\(^{-3}\)) on average; while other approaches have MSE greater than 1×10\(^{-3}\). In the gasoline concentration prediction (FCCU data set), OWE, pruned OWE and AddExp.C have achieved the lowest errors on average, i.e. 2.48×10\(^{-3}\), 2.65×10\(^{-3}\) and 2.92×10\(^{-3}\), respectively; while most of the other approaches have errors larger than 30×10\(^{-3}\). In the LDO concentration estimation (FCCU data set), pruned OWE and OWE have the lowest errors, i.e. 33.23×10\(^{-3}\) and 33.41×10\(^{-3}\), respectively; and other methods have

Figure 5: Pruned ensembles’ errors using the artificial data sets when the maximum number of models varies (for \( m = 40 \)).

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Error with ( m ) varies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hyperplane</td>
<td>0.12×10(^{-3})</td>
</tr>
<tr>
<td>Friedman-LA</td>
<td>0.13×10(^{-3})</td>
</tr>
<tr>
<td>Friedman-GRA</td>
<td>0.14×10(^{-3})</td>
</tr>
<tr>
<td>Friedman-GnRG</td>
<td>0.15×10(^{-3})</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Error with ( m ) varies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hyperplane</td>
<td>0.12×10(^{-3})</td>
</tr>
<tr>
<td>Friedman-LA</td>
<td>0.13×10(^{-3})</td>
</tr>
<tr>
<td>Friedman-GRA</td>
<td>0.14×10(^{-3})</td>
</tr>
<tr>
<td>Friedman-GnRG</td>
<td>0.15×10(^{-3})</td>
</tr>
</tbody>
</table>
It can be observed that ILLSA and the (i.e. 37.08 \times 10^{-3}), followed by OWE (i.e. 37.39 \times 10^{-3}). Figures from 7 to 10 show the estimated outputs of each algorithm on its best window’s size $m$ as evaluated by the MSE.

Table 4: Average and standard deviation of the MSE* by varying the value of $m$ using the industrial data sets.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Catalyst activ.</th>
<th>FCCU</th>
<th>FCCU</th>
<th>FCCU</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Approach</td>
<td>(gasoline conc.)</td>
<td>(LDO conc.)</td>
<td>(LPG conc.)</td>
</tr>
<tr>
<td>RPLS</td>
<td>1.00 (0.23)</td>
<td>35.33 (12.13)</td>
<td>38.16 (4.91)</td>
<td>47.43 (14.54)</td>
</tr>
<tr>
<td>ILLSA</td>
<td>1.71 (1.66)</td>
<td>80.68 (12.07)</td>
<td>91.28 (52.93)</td>
<td>198.27 (0.00)</td>
</tr>
<tr>
<td>AddExp.C</td>
<td>1.19 (0.24)</td>
<td>28.92 (3.23)</td>
<td>35.86 (6.20)</td>
<td>37.08 (2.37)</td>
</tr>
<tr>
<td>Pruned AddExp.C</td>
<td>1.19 (0.24)</td>
<td>29.35 (3.48)</td>
<td>35.22 (5.42)</td>
<td>37.63 (2.72)</td>
</tr>
<tr>
<td>Learn**.NSE</td>
<td>4.44 (1.27)</td>
<td>47.23 (55.49)</td>
<td>44.02 (16.35)</td>
<td>54.86 (21.47)</td>
</tr>
<tr>
<td>Pruned Learn**.NSE</td>
<td>4.50 (1.54)</td>
<td>46.42 (35.50)</td>
<td>42.79 (17.46)</td>
<td>54.03 (21.95)</td>
</tr>
<tr>
<td>OWE</td>
<td>0.97 (0.21)</td>
<td>24.48 (3.66)</td>
<td>33.41 (2.77)</td>
<td>37.39 (3.28)</td>
</tr>
<tr>
<td>Pruned OWE</td>
<td>1.03 (0.29)</td>
<td>26.05 (5.25)</td>
<td>33.23 (2.80)</td>
<td>38.12 (3.69)</td>
</tr>
</tbody>
</table>

The values have been multiplied by $10^3$; $m$ is varied from 10 to 20 (in steps of 1).

Figure 11 shows the performance of the pruned ensembles when $B$ increases ($m$ is set to 20). In general, it is observed that the error tends to decrease when $B$ increases. Table 5 shows interesting details of all the approaches. The test shows that OWE produces more models than the other methods. However, in most cases, OWE has smaller running time when compared to AddExp.C. For the catalyst activation data set, ILLSA has produced a larger number of models when compared to the artificial data sets, because this data set contains more concept changes in the training data set.

**Discussion.** It can be observed that ILLSA and the Learn**.NSE are more sensitive to the value of $m$. The RPLS and ILLSA have achieved better accuracy in the catalyst activation data set when compared to their performances in the artificial data sets. Possibly, in the catalyst activation data set, the dynamics of the process are well represented for designing the set of models in the ILLSA; and for training the model in the RPLS. The AddExp.C and the OWE algorithms outperform the Learn**.NSE approach. As mentioned before, Learn**.NSE uses batch learning and it takes longer time to adapt to the changes. In most cases, the tests indicate that the ensembles without pruning usually have equal or superior performance when compared to their pruned versions. Therefore, this result reveals that a larger number of models may lead to a better ensemble’s accuracy. This performance may be related to the diversity among the models or because the data sets have recurring behavior and, consequently old models are necessary to be re-activated during predictions.

6. Conclusions

This paper proposes a new ensemble to deal with the concept drift: the on-line weighted ensemble of regressor models (OWE). The proposed ensemble is able to handle several types of drifts, including abrupt and gradual drifts, global and local drifts, and recurring and non-recurring drifts. The ensemble employs several adaptive strategies for avoiding system’s degradation in changing environments.

Experimental results of the OWE and other methods (RPLS, ILLSA, AddExp.C, and Learn**.NSE) using artificial data sets and industrial data sets have shown that, in general, OWE achieves better accuracy than other state-of-the-art methods, and in some cases, OWE has comparable accuracy to the other state-of-the-art approaches. RPLS assumes that samples that fall outside the moving window are irrelevant for the learning, and such method does not have capability to use the old acquired data, since the oldest samples are discarded. Other methods able to conciliate previous data and current data (e.g. Learn**.NSE) may perform poorly since a long time is required for system adaptation.

In this paper, the tests show notable behaviors. Results show that, in most cases, ensemble learning outperforms learning using only one single model. The tests also show that OWE has capability to deal with the concept drifts. Our analysis reveals that the frequency of including a new model to the ensemble ($\alpha$), the contribution of old windows over new windows ($\kappa$), and the maximum number of models are important issues in on-line ensembles that deal with changing environments.

Other important issues are shown in this paper. In ensemble learning, the re-training of all models on the same data can produce very similar models. In this specific case, the ensemble loses information about the old scenarios, leading the ensemble to a poor accuracy in scenarios where an old concept can recur. In recurring drifts, ensembles without pruning strategies are usually more accurate than pruned ensembles. Since
Table 5: Average and standard deviation of the n. of models and running time (minutes) of the approaches in the industrial data sets by varying the value of $m^\ast$.

<table>
<thead>
<tr>
<th>Data set / Approach</th>
<th>Catalyst activation</th>
<th>FCCU (gasoline concentration)</th>
<th>FCCU (LDO concentration)</th>
<th>FCCU (LPG concentration)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>n. of models</td>
<td>run. time</td>
<td>n. of models</td>
<td>run. time</td>
</tr>
<tr>
<td>RPLS</td>
<td>1.00 (0.00)</td>
<td>0.182 (0.047)</td>
<td>1.00 (0.00)</td>
<td>0.019 (0.005)</td>
</tr>
<tr>
<td>ILLSA</td>
<td>31.45 (15.40)</td>
<td>0.578 (0.234)</td>
<td>2.00 (0.00)</td>
<td>0.005 (0.002)</td>
</tr>
<tr>
<td>AddExp.C</td>
<td>44.64 (3.59)</td>
<td>0.205 (0.053)</td>
<td>73.72 (5.46)</td>
<td>0.065 (0.026)</td>
</tr>
<tr>
<td>Pruned AddExp.C</td>
<td>20.00 (0.00)</td>
<td>0.163 (0.048)</td>
<td>20.00 (0.00)</td>
<td>0.053 (0.014)</td>
</tr>
<tr>
<td>Learn*+NSE</td>
<td>44.91 (10.38)</td>
<td>0.070 (0.018)</td>
<td>6.82 (1.72)</td>
<td>0.005 (0.002)</td>
</tr>
<tr>
<td>Pruned Learn*+NSE</td>
<td>20.00 (0.00)</td>
<td>0.084 (0.013)</td>
<td>6.82 (1.72)</td>
<td>0.005 (0.002)</td>
</tr>
<tr>
<td>OWE</td>
<td>192.64 (17.87)</td>
<td>0.322 (0.090)</td>
<td>80.27 (3.61)</td>
<td>0.054 (0.014)</td>
</tr>
<tr>
<td>Pruned OWE</td>
<td>20.00 (0.00)</td>
<td>0.192 (0.052)</td>
<td>20.00 (0.00)</td>
<td>0.048 (0.014)</td>
</tr>
</tbody>
</table>

$^\ast m$ is varied from 10 to 20 (in steps of 1).

in recurring drifts, old concepts can recur, the pruning strategy may remove important models trained on these old concepts. OWE monitors the models’ performances on the current and old windows so that when an old concept recurs, old and accurate models can be re-activated. Despite the attractive characteristics of the OWE, its accuracy is related to the windows’ size and the $\kappa$ value. To cover these limitations, as a future work, we would to propose a variable window size that adapts according to the process dynamics (Khediri et al., 2011); and an adaptive setting of $\alpha$ that is automatically adjusted according to the change of characteristics, e.g. when a change occurs, $\alpha$ would be set to a low value to include new models in a high frequency. Moreover, as a future work, the authors would like to introduce other pruning strategies and methods to dynamically adjust other OWE’s parameters over time.

Acknowledgments

Symone Gomes Soares is supported by the Fundação para a Ciência e a Tecnologia (FCT) under the Grant SFRH/BD/68515/2010.

This work was supported by Project SCIAD “Self-Learning Industrial Control Systems Through Process Data” (reference: SCIAD/2011/21531) co-financed by QREN, in the framework of the “Mais Centro - Regional Operational Program of the Centro”, and by the European Union through the European Regional Development Fund (ERDF).

Rui Araújo acknowledges the support of FAPEG (“Fundação de Amparo à Pesquisa de Goiás”, in Portuguese). The authors are grateful to Dr. Petr Kadlec and Dr. Yi Liu for providing their data sets.

References


Chao, M., Xin, S. Z., Min, L. S., June 2014. Neural network ensembles based on copula methods and distributed multobjective central force optimization algorithm. Engineering Applications of Artificial Intelligence 32 (0), 203–212.


Elwell, R., Polikar, R., October 2011. Incremental learning of concept drift in nonstationary environments. IEEE Transactions on Neural Networks 22 (10), 1517–1531.


Secaucus, NJ, USA.
Minku, L. L., White, A. P., Yao, X., May 2010. The impact of diver-
sity on online ensemble learning in the presence of concept drift. IEEE Transactions on Knowledge and Data Engineering 22 (5), 730–742.


Figure 8: The predicted outputs of all the algorithms using FCCU data set (gasoline concentration).

Figure 9: The predicted outputs of all the algorithms using FCCU data set (LDO concentration).


Figure 10: The predicted outputs of all the algorithms using FCCU data set (LPG concentration).

Proc. of the 4th Int. Conf. on Genetic and Evolutionary Computation. ACM, pp. 681–688.

Figure 11: Pruned ensembles’ errors using the industrial data sets when the maximum number of models varies.

based on support vector machine and bayesian model selection. Computers & Chemical Engineering 28 (8), 1489–1498.