A Dynamic and On-line Ensemble Regression for Changing Environments

Symone G. Soares*, Rui Araújo

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

Abstract

On-line learning in environments and applications with time-varying behavior pose serious challenges. Changes may lead the learning model designed with old data, to become inconsistent with the new data, so that adaptation strategies are necessary. Unfortunately, most adaptation strategies are performed only on a batch basis, i.e. after accumulating certain number of samples. This process usually requires a long time, and thus such data may not reflect the current state of the system. However, even the learning system is adapted on a sample basis, most existing on-line learning algorithms adapt slowly to the abrupt changes. To overcome these drawbacks, a new dynamic and on-line ensemble regression (DOER) with fast adaptation capability for on-line prediction of variables given on a sample basis is proposed in this paper. DOER brings together desired properties which are not given by the previous works on on-line ensemble for regression: (1) on-line inclusion and removal of models to keep only the most accurate models with respect to the current state of the system; (2) dynamic adaptation of the models’ weights based on their on-line predictions on the recent samples; and (3) on-line adaptation of the models’ parameters. The accuracy of each model is obtained using a sliding window that is filled with the predictive errors of the most recent samples. Based on the model’s accuracies, weights are dynamically assigned, where accurate models are heavily weighted. When a new sample is available, all the models are retrained, and a new model may be included if the ensemble’s performance is not satisfactory. Inaccurate models can be removed for reducing the computational costs. Experiments on synthetic and real-world data sets are reported to evaluate the effectiveness of the DOER. Results show that DOER offers faster adaptation capability when compared to the state-of-the-art approaches.

Keywords: Ensemble learning, learning in changing environments, regression, concept drift.

1. Introduction

In many industrial applications, some important quality variables cannot be automatically measured at all, or can only be measured at high cost, inaccurately, sporadically or with high delays (Fortuna et al., 2006). An example is when important variables are determined only by laboratory (off-line) analysis, introducing delays and economic penalties. During the last decades, predictive models have become attractive tools. The first generation of predictive models, known as data-driven models, relied on the off-line modeling based on the recorded historical data (Kadlec et al., 2011). Popular data-driven models are the Partial Least Squares (PLS) (von Stosch et al., 2011), Neural Networks (Soares et al., 2011), and Support Vector Machines (SVM) (Liu et al., 2010).

Despite their popularity, there are critical issues for predictive model development. Process plants are rather dynamic, being very difficult for the models to react to the changes, and thus leading to a deterioration of the model prediction accuracy. The main reasons for such changes are the sensor drift and/or process drift. Sensor drift is a temporal shift of a sensor (which cannot be predicted or defined) due to aging or environment changes (Vergara et al., 2012). Process drift is related to the changes of process behavior or to some external process conditions over time (Schimunek et al., 2008). Sensor drift and process drift are difficult to detect and handle, since many other factors may be changing the process conditions in parallel. In machine learning, all these drifting problems are summarized under the term concept drift (Tsymbal, 2004).

To cope with all these effects, the development of predictive models with adaptive capability is necessary. Different strategies for on-line adaptation have been proposed in the literature. In the sliding window (SW) approach, as new samples become available, a window moves along the data and a new model is trained. Recently, Just-in-time Learning (JITL) has been raising much attention in modeling industrial systems with time-varying behavior (Jin et al., 2014). JITL designs a local model using similar samples of a testing sample. Once this sample is predicted, the local model is discarded. The main drawback of SW and JITL approaches is the high computational time to continuously train a new model from scratch whenever a new sample is available. Other strategies are the recursive methods such as the Recursive Least Squares (RLS) (Gjerkes et al., 2011), the Recursive PLS (RPLS) (Qin, 1998), or the On-line Sequential Extreme Learning Machine (OS-ELM) (Liang et al., 2006), where model’s parameters are recursively adapted over time. Recursive methods perform well in cases where the process dynamics are well represented in the training data set. However, these methods are not efficient to deal with new process dynam-
ics characteristics occurring in new samples, becoming difficult to adapt quickly to abrupt changes of the process.

Alternatively, the ensemble learning algorithm approach has been proven itself as a valuable system to deal with the concept drift problem. In ensemble learning, a set of models is trained and their outputs are combined to get a final prediction. Online ensemble approaches constitute a specific class of ensembles which can be adapted on-line (Farid et al., 2013). An online ensemble can be classified as sample-based or batch-based, when it learns on-line and incrementally on a sample basis, or when it learns from a set of samples, respectively. Batch-based ensembles usually require a long time to wait for a batch, and, when complete, such data may not reflect the current state of the process anymore. On the other hand, sample-based ensembles can assure faster adaptivity in changing environments, offering good performance in applications where variables are measured at low sampling rates. In the literature, there are few online ensemble methods for regression purposes (Kadlec and Gabrys, 2011; Lan et al., 2009), since most of them focus on classification tasks (Nishida et al., 2005; Nishida and Yamauuchi, 2007; Minku and Yao, 2012; Elwell and Polikar, 2009; Chu and Zaniolo, 2004).

Adaptive sample-based ensembles for regression inspired by SW have been adopted as predictive models for industrial processes. One of the first methods to introduce this concept is the Incremental Local Learning Soft Sensing Algorithm (ILLSA) (Kadlec and Gabrys, 2011). A window slides along the training data, and when a process change is detected by a t-test, a model is trained using the data window and added to the ensemble. Kaneko and Funatsu (2014) proposes an ensemble of on-line SVM models. On the on-line phase, a new model is added at a fixed frequency using the current data window. In (Lv et al., 2013) the training data is partitioned into different subsets using the Fuzzy C-means cluster algorithm. Then, each subset is employed to train a Least Squares SVM model. No adaptive mechanisms are employed to the ensemble. However, all these listed ensembles do not include and exclude models during on-line operation; but the on-line inclusion and exclusion of models can be an important factor for improving ensemble prediction performance.

The frequency of including a new model in the ensemble is an important issue in on-line ensembles. Ensemble applications usually add a new model only at a pre-defined fixed frequency (e.g. batch frequency) (Brzezinski and Stefanowski, 2014). However, results indicate that when new models are added at a sample frequency, the ensemble can adapt quickly to the changes and the system’s performance is improved significantly (Kolter and Maloof, 2005). Another important issue is the dynamic removal of models from the ensemble, since the used memory and computational resources may be increasing considerably; and some models may contain few information about the current state of the process (Soares et al., 2013). A dynamic ensemble of PLS models is proposed in (Soares and Araújo, 2015). On the on-line phase, new models are added when the ensemble’s performance is deteriorating, and the least contributing model of the ensemble may be removed using Boosting theory (Shrestha and Solomatine, 2006). The models’ weights are adapted, but the models are not retrained.

This paper proposes a new dynamic and on-line ensemble regression (DOER) approach with fast adaptation capability for on-line prediction of variables measured at low sampling frequency and on a sample basis in applications with time-varying behavior. DOER brings together desired properties which are not given by previous works on on-line ensemble for regression: (1) on-line inclusion and removal of models to keep only the most accurate models with respect to the current state of the system; (2) dynamic adaptation of the models’ weights based on their on-line predictions on the recent samples; and (3) on-line adaptation of the models’ parameters. DOER design an ensemble based on a SW approach. The accuracy of each model is obtained using a window that is filled with the predictive errors of the most recent samples. Based on the model’s accuracies, weights are dynamically assigned, where the accurate models are heavily weighted. When a new sample is available, all the models are retrained, and a new model may be added if the ensemble’s performance is not satisfactory. Over time, models can be removed for reducing the computational costs and assuring the ensemble’s performance, since inaccurate models may be decreasing the ensemble accuracy. Experiments on four synthetic data sets and six real-world industrial data sets are reported to evaluate the effectiveness of the DOER. Results show that DOER has high adaptation capability, and DOER is not only comparable to the state-of-the-art approaches, but in most cases, DOER has better accuracy when compared to them.

This work is organized as follows. Section 2 reports some related works. In Section 3, the structures of some on-line learning algorithms are detailed. Section 4 describes the DOER algorithm. The experiments are reported and analyzed in Section 5. Finally, Section 6 contains some concluding remarks.

2. Related Works

2.1. First Concepts

Consider an on-line learning scenario where samples from a data set are presented incrementally. Samples can be provided incrementally on a sample basis (i.e. one by one) or on a batch basis (i.e. data blocks). It is assumed that an on-line supervised learning algorithm is employed to predict on-line new samples. At each time step, the model can use an input or a set of inputs to predict the output(s). And after some time, or at some time steps, the corresponding real output(s) is(are) available and can be used as additional information for the learning.

Over time, the target concept may change. Here the term concept refers to a target function of a problem that the learning system is trying to model along a period of time. Concept drifts can be classified according to their speed, cyclical nature, scope, etc (Tszymal, 2004). The drift speed describes the rate at which a new concept replaces an old concept. An abrupt drift occurs when an old concept is replaced abruptly by a new concept. A gradual drift occurs when an old concept is replaced gradually by a new concept. Gradual drifts are harder to identify since they result in small data shifts and lower degradation of the model’s performance. Regarding the cyclical nature,
a drift is recurring if a previous concept reappears after some time, while a drift is non-recurring if a previous concept cannot reappear after some time. Drifts can also be classified according to their scope (Ikonomovska, 2012). Global drifts reappear after some time. Drifts can also be classified according to whether a drift is recurring or not, and according to regions of the instance space, while local drifts affect only some regions of the instance space. In this last case, the model should be adapted only in those affected regions of the instance space.

2.2. Adaptive Approaches in Changing Environments

Algorithms to deal with concept drift can be classified as explicit or implicit. Explicit algorithms employ a drift detection strategy to detect the starting time and severity of a drift. Early Drift Detection Method (EDDM) is an example of an explicit algorithm to deal with changes. EDDM measures the distance (interval of time) between two classification errors. It considers that, if the distance increases, then the system is improving its predictions. Otherwise, if the distance decreases, EDDM assumes that the system is learning a new concept, and so a drift is detected. In this case, the system is reset and a new model is trained using a recent set of samples stored since an early drift warning instant is detected. Other examples of drift detection approaches are the t-test (Kadlec and Gabrys, 2011) and the Page-Hinkley test (Ikonomovska, 2012).

Implicit algorithms do not perform techniques to detect the starting time of a drift. They constantly learn from the environment, adjusting and constructing the knowledge without explicitly detecting drifts. The main approaches are instance selection, instance weighting, and ensemble learning (Kadlec, et al., 2011; Tsymbal, 2004). In instance selection, a set of relevant samples of the actual concept are selected to build or adapt the model. A common technique is the SW. The window can slide on a sample basis or on a batch basis. An important issue in SW is the selection of the window’s size. Small windows can provide faster adaptability, but in more stable phases they can affect the model’s performance; while large windows can be more stable but they cannot react faster to the changes. To overcome these issues, methods to adapt the window’s size can be performed (Bifet and Gavalda, 2007).

In instance weighting, samples are weighted according to their age and/or relevance to the current concept, e.g. recursive methods. Recursive methods usually involve down-weighting of the old samples’ contribution using a forgetting factor $\rho$ (Qin, 1998). The forgetting factor indicates the strength of the adaptation. Its value should be flexible so that adaptation can overcome faster and slower changes. In (Gjerkes et al., 2011) it is proposed a RLS algorithm, where the value of $\rho$ is adjusted according to the model’s prediction error. If the error is small, it is assumed that the estimation is correct and that the process is not changing, so that $\rho$ is increased; while if the error is large, then $\rho$ is decreased to allow a quick adaptation of the model.

This paper focuses on ensemble learning algorithms. Table 1 lists the main on-line ensembles existing in the literature. An ensemble to deal with concept drift can have the following characteristics: (i) adapt the models’ weights; (ii) adapt the models’ parameters; and/or (iii) add new models or exclude models (Polikar, 2012). Additionally, other approaches recreate a new ensemble from scratch when a drift is detected (iv) (Minku and Yao, 2012; Chu and Zaniolo, 2004).

Currently, most approaches perform strategy (iii). The removal of models can be performed using an ensemble pruning strategy. In ensemble pruning, a subset of relevant models from the original set of models is selected, and those models that do not contribute to the ensemble’s performance are removed. The exclusion of a model from the ensemble can occur when the number of models exceeds a threshold (Nishida and Yamauchi, 2007; Elwell and Polikar, 2009; Chu and Zaniolo, 2004; Kolter and Maloof, 2005); at a fixed frequency; when a model’s parameter reaches a value (Grbovic and Vucetic, 2011); and/or when the memory usage exceeds a threshold (Brzezinski and Stefanowski, 2014). Another decision to be taken concerns as to which model should be removed from the ensemble. The excluded model can be the oldest model (Elwell and Polikar, 2009; Chu and Zaniolo, 2004; Kolter and Maloof, 2005) or the model with the worst performance (Nishida and Yamauchi, 2007; Elwell and Polikar, 2009; Kolter and Maloof, 2005; Grbovic and Vucetic, 2011). In changing environments, identifying the best ensemble pruning strategy is not an easy task. In recurring drifts, there is a risk of removing a model that may be important in the future. Therefore, weakest first strategy should be preferred over oldest first strategy.

Batch-based ensembles tend to be more stable, in the sense that even if a batch contains an outlier, the system may perform well. Examples are the Learn++.NSE (Elwell and Polikar, 2011) and the Fast and Light Boosting (FLB) (Chu and Zaniolo, 2004). On both cases, when a new batch is available, the ensembles are employed to predict it. Then, each sample from the batch receives a weight proportional to its prediction error and, a weighted training batch is obtained using the samples’ weights. The objective is to train a new model using the weighted training batch. In the FLB, if a change is detected using statistical decision theory, a new ensemble is created from scratch for fast adaptation of the system to the current concept. This approach may lead the system to a poor performance in scenarios where concepts can recur, since models trained on old concepts are removed. FLB obtains the ensemble’s output using average of the models’ outputs. On the other hand, Learn++.NSE obtains the ensemble’s output using a weighted average of the models’ outputs, where each model’s weight is calculated using a weighted average of its prediction errors on the old and current batches.

On-line Bagging (OB) is a sample-based ensemble inspired by the batch Bagging algorithm (Oza and Russell, 2001). Given a training data set $D$ with $m$ samples, OB creates a set of $B$ base models, each one trained with a different training data set $(D_1, \ldots, D_m)$ of size $m$ obtained from $D$ by bootstrap. Bootstrap is a sampling technique to randomly sample with replacement from an original training data set. When $m$ tends to infinite, a data set $D_i$ ($i = 1, \ldots, B$) can contain $K$ copies of a sample from $D$, so that OB assumes that the distribution of $K$ tends to a Poisson distribution. Specifically, during the on-line phase of OB, when a new sample is available, it is presented $K$ times for retraining each base model, where $K \sim \text{Poisson}(\lambda)$ and $\lambda = 1$. OB uses simple average for combining the models’ outputs and
no ensemble pruning strategy is applied.

Incremental Boosting (IBOost) (Grbovic and Vucetic, 2011), ILLSA (Kadlec and Gabrys, 2011) and Online Accuracy Updated Ensemble (OAUE) (Brzezinski and Stefanowski, 2014) are ensembles inspired on the SW concept. IBOost is a sample-based ensemble that keeps a SW with the most recent samples. At a fixed frequency, a model may be added to the ensemble if the ensemble’s prediction on the newest sample is incorrect. IBOost has faster adaptation when compared to the batch-based ensembles. ILLSA is an ensemble of RPLS models. On the on-line phase, the models’ expertise is updated in each input space are obtained using a weighted two-dimensional Parzen Window method. Then models’ weights on the new sample are calculated using the posterior probability by a Bayesian framework. ILLSA does not prune dynamically the ensemble. OAUE and Adaptive Classifiers-Ensemble system (ACE) are ensembles for classification tasks that employ a hybridization of strategies of sample-based and batch-based ensembles. Namely, the models’ parameters are updated whenever a new sample is available. However, a model is added or removed from the ensemble only after accumulating m samples.

In the OAUE, the error of each model $f_j$, $MSE_j^t$ at time $t$, is estimated by calculating the Mean Square Error (MSE) using the most recent m samples. Then the weight of each model $f_j$ is obtained as: $w_j = MSE_j^t / MSE_j^t + MSE_j^t + \epsilon$, where $\epsilon$ is a small positive value and $MSE_j^t$ is an error prediction threshold used as reference to the weighting strategy, that is, models with MSE values greater than $MSE_j^t$ value receive lower weights. $MSE_j^t$ is obtained using the most recent m samples by $MSE_j^t = \sum_{j=1}^{l} P(\omega_j) (1 - p(\omega_j))^2$, where $P(\omega_j)$ is the a priori probability (or percentage) of a sample belonging to class $\omega_j$, for a J-class problem. At time instants defined by fixed period of m samples, a candidate model is trained with the most recent m samples. Afterwards, the candidate model is incrementally trained with the next m samples, and then it is finally added to the ensemble and a new candidate model is trained with the new m most recent samples. If the number of models raises above a threshold, the weakest model is substituted. In our experiments (Section 5) with the OAUE algorithm, the models’ weights are obtained as $1/(MSE_j^t + \epsilon)$ to convert OAUE for regression tasks.

Additive Expert (AddExp) is the most popular on-line ensemble for regression. It applies a loss bound to measure the models’ performances, and weights are adapted according to their current losses and a decreasing factor, $\beta$, used to decrease a model’s weight when it predicts incorrectly. The output values must be set in the interval $[0, 1]$. Unlike AddExp, DOER can work out of this interval. In AddExp, a new model is included when the total ensemble’s loss is greater than a factor $\tau$. The new model’s weight is set according to a factor $\gamma$. A model can be removed when the number of models is greater than a threshold. Two pruning strategies are proposed: oldest first or weakest first. AddExp does not reveal which samples should be taken for training a new model. In our experiments, we consider AddExp adapted on a sample basis, where a new model is trained using the last m samples and the weakest first pruning strategy is used.

### 3. Main structures of on-line learning algorithms

This section describes the scheme of some types of on-line learning algorithms. Below, the impact of these approaches will be experimentally tested and their results will support the design of the DOER algorithm. Consider a data set $D = \{(x_i, y_i)\}^T_{i=1}$ divided into batches $(D_1, D_2, \ldots, D_M)$ of size m; an on-line supervised learner.

Algorithm 1 A generic on-line batch-based algorithm using a single model

1. **Input:** a data set $D = \{(x_i, y_i)\}^T_{i=1}$ divided into batches $(D_1, D_2, \ldots, D_M)$ of size m; an on-line supervised learner;
2. $f \leftarrow$ obtain a model trained with $D_1$; set $b = 2$;
3. **while** $b \leq M$ **do**:
   a. obtain the output prediction of $f$ using $D_b$;
   b. retrain the model $f$ using $D_b$;
   c. $b \leftarrow b + 1$;
4. **end while**

**Table 1: The main on-line ensembles to deal with changing environments.**

<table>
<thead>
<tr>
<th>Approach/Reference</th>
<th>Scope</th>
<th>Types of drifts</th>
<th>Drift mechanism</th>
<th>Ensemble learn. mech.</th>
<th>Time step</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACE (Nishida et al., 2005, Nishida and Yamanchi, 2007)</td>
<td>classification</td>
<td>mainly recurring drifts</td>
<td>explicit/implicit</td>
<td>(i), (ii), (iii)</td>
<td>sample basis</td>
</tr>
<tr>
<td>AddExp (Kolter and Maloof, 2005)</td>
<td>classification/regression</td>
<td>all the types</td>
<td>explicit/implicit</td>
<td>(i), (ii), (iii)</td>
<td>sample/batch bases</td>
</tr>
<tr>
<td>Fast and Light Boosting (FLB) (Chu and Zaniolo, 2004)</td>
<td>classification</td>
<td>gradual and abrupt drifts</td>
<td>explicit/implicit</td>
<td>(ii), (iii)</td>
<td>batch basis</td>
</tr>
<tr>
<td>IBoost (Grbovic and Vucetic, 2011)</td>
<td>classification</td>
<td>all the types</td>
<td>explicit/implicit</td>
<td>(i), (ii), (iii)</td>
<td>batch basis</td>
</tr>
<tr>
<td>ILLSA (Kadlec and Gabrys, 2011)</td>
<td>classification</td>
<td>all the types</td>
<td>implicit</td>
<td>(i), (ii)</td>
<td>sample basis</td>
</tr>
<tr>
<td>Learn**/<em>NSE</em>* (Elwell and Polikar, 2011, 2009)</td>
<td>classification</td>
<td>all the types</td>
<td>implicit</td>
<td>(i), (ii)</td>
<td>sample basis</td>
</tr>
<tr>
<td>OAUE (Brzezinski and Stefanowski, 2014)</td>
<td>classification</td>
<td>all the types</td>
<td>implicit</td>
<td>(i), (ii)</td>
<td>sample basis</td>
</tr>
<tr>
<td>OB (Oza and Russell, 2001)</td>
<td>classification</td>
<td>all the types</td>
<td>implicit</td>
<td>(i)</td>
<td>sample basis</td>
</tr>
</tbody>
</table>

* No reference about the types of drifts in which the approach can deal with.
Algorithm 2 A generic on-line sample-based algorithm using a single model

1. **Input:** a data set $D = \{(x_i, y_i)\}_{i=1}^m$; number of samples for the initial training phase, $m$; an on-line supervised learner;
2. **Initialization:** set the training data as $D_{train} = \{(x_i, y_i)\}_{i=1}^m \subset D$;
3. $f \leftarrow$ obtain a model trained with $D_{train}$; set $t = m + 1$;
4. **while** $t \leq T$ **do**:
    (a) obtain the output prediction of $f$ using $x_i$;
    (b) retrain $f$ using $(x_i, y_i)$;
    (c) $t \leftarrow t + 1$;
5. **end while**

Algorithm 3 A generic on-line sliding window (SW) algorithm using a single model

1. **Input:** a data set $D = \{(x_i, y_i)\}_{i=1}^m$; window’s size, $m$; a supervised learner;
2. **Initialization:** set $t = m$ and the window as $D' = \{(x_i, y_i)\}_{i=1}^m \subset D$;
3. $f \leftarrow$ obtain a model trained with $D'$; set $t = m + 1$;
4. **while** $t \leq T$ **do**:
    (a) slide the window: $D' = D'^{-1} + (x_i, y_i) - (x_{t-m}, y_{t-m})$;
    (b) obtain the output prediction of $f$ using $x_i$;
    (c) replace $f$ with a new model trained with $D'$;
    (d) $t \leftarrow t + 1$;
5. **end while**

A batch and/or sample bases. The parameters are the number of hidden neurons, the activation function and the number of samples to be used on the initial training phase. On the on-line phase, once a new sample or a new batch is available, it is employed for re-training and then it can be discarded. Results indicate that OS-ELM provides better generalization capability and faster speed when compared to the other on-line learning algorithms which use a single model. OS-ELM can also operate using a SW of fixed size $m$, as in Algorithm 3. In this way, when a new sample is available, an old model trained on the old data window is replaced by a new model trained on the current data window. Algorithm 3 usually outperforms Algorithms 1 and 2, since, in the SW approach the model is trained only with the most recent set of samples. However, Algorithm 3 is more computationally expensive, since a new model must be trained at each step.

Lan et al. (Lan et al., 2009) proposed the sample-based EOS-ELM ensemble, an ensemble of OS-ELM models, which can provide better performance and more stability when compared to the original OS-ELM. Initially, EOS-ELM creates a set of models, all trained using the same activation function and number of hidden neurons, and then the models’ outputs are combined by average. On the on-line phase, when a sample is available, EOS-ELM re-trains all the models. One important issue in ensemble learning is the diversity between the models (Soares et al., 2013; Abellán and Mantas, 2014). EOS-ELM can possibly have inferior performance when compared to the other ensembles, since all the models are trained on the same data and they have the same architectural structure. One alternative is the OB which manipulates the training samples so that each model of the ensemble can be retrained on different samples, increasing the diversity degree between the models.

Other algorithms are the on-line batch-based ensemble and the on-line sample-based ensemble using a SW, detailed in Algorithms 4 and 5, respectively. They depend on the number of models $B$, and $e()$ a generic model error function to measure the accuracy of the ensemble and the models on a data. For
example, for the OAEU, the models’ errors are obtained using the MSE between the predicted and real outputs on the current window. Algorithm 5 depends on a factor to add a new model on the ensemble. Namely, a model is included to the ensemble when the ensemble’s prediction error on a new sample is greater than a pre-defined factor. In contrast, in Algorithm 4, a new model is added when a new batch is available. On both the algorithms, a model is replaced by a new model, if the number of models exceeds B. One important issue on on-line ensemble algorithms is the initial weight of a new model. The weighting strategy should not set an initial weight according to the performance on the training data set, avoiding that a possible overfitting during the training phase degrades the ensemble’s accuracy (Caruana et al., 2000). For example, AddExp employs a factor for decreasing a new model’s weight.

Learn++.NSE and FLB use the scheme presented in the Algorithm 4. However, they do not apply Step 5c, i.e. no re-training of models is applied. AddExp and DOER employ a scheme similar to Algorithm 5, a robust solution when compared to Algorithm 4, since Algorithm 5 evaluates the models and ensemble on every new sample. Additionally, Algorithm 5 can add new models in a high frequency when compared to the batch-based ensemble, avoiding the on-line ensemble’s degradation. DOER weights models using the function \(e()\) on \(D^t\) as the OAUE approach, while AddExp uses the function \(e()\) on \((x_t, y_t)\). The first strategy is more robust to outliers. OAUE performs Algorithm 5 with some modifications, since the models are added only when a new batch is available and a candidate model is trained incrementally.

### 4. Dynamic and On-line Ensemble Regression (DOER)

This section presents the DOER algorithm. DOER offers together the following strategies or characteristics which are not given by previous works: (1) on-line ensemble learning, most ensembles are developed off-line and do not take into account that the process or data may exhibit time-varying behaviors; (2) regression scope, as there is a lack of on-line regression ensembles; (3) sample-based ensemble which offers higher accuracy and faster adaptivity when compared to batch-based ensembles; (4) adaptation of the models’ weights (i), DOER incorporates dynamic adaptation of the models’ weights based on the models’ accuracies on the most recent samples, DOER assigns high weights to the most accurate models, allowing that inaccurate models do not degrade the ensemble’s performance; (5) adaptation of the models’ parameters (ii), leading the system to a faster adaptation in changing environments; (6) dynamic inclusion and removal of models (iii), new models are launched to the ensemble and models that do not contribute to the ensemble are excluded; the (8) pruning strategy removes the model with the worst accuracy on the most recent samples, therefore old and accurate models can be kept; DOER can work with (9) unnormalized data; and (10) overfitting control, the model evaluation does not consider the performance on the training phase.

#### Algorithm 6 Dynamic and On-line Ensemble Regression (DOER) algorithm

1. **Input:** a data set \(D = \{(x_t, y_t) | x_t \in \mathbb{R}^{input}, y_t \in \mathbb{R}, t = 1, \ldots, T\}; \) window’s size, \(m\); an on-line supervised learner; \(\alpha\), factor to add a new model; maximum number of models, \(B\);
2. **Initialization:** set \(E \leftarrow \emptyset, t = m, k = 1\) and the current window \(D^t = \{(x_t, y_t) | t \leq m \in D\}; \)
3. \(f_k \leftarrow \) obtain a new model trained with \(D^t\); set \(\text{life}_k = 0, \text{MSE}_k^t = 0; w_k = 1\) and \(E \leftarrow E \cup f_k; \)
4. **while** \(t \leq T\) **do:**
   (a) slide the window: \(t \leftarrow t + 1; D^t = D^{t-1} + (x_t, y_t) - (x_{t-m}, y_{t-m}); \)
   (b) predict \(y_t: y^*_t = F(x_t) = \left(\sum_{j=1}^{k} w_j f_j(x_t)\right) / \sum_{j=1}^{k} w_j; \)
   (c) for all models \(f_j \in E, \) obtain the prediction error \(e_j^t\) on \(x_t\) as \(e_j^t = (y_t - f_j(x_t))^2; \) and set \(\text{life}_j \leftarrow \text{life}_j + 1; \)
   (d) obtain \(\text{MSE}_j^t\) for each model \(f_j \in E\) using Eq. (2).
   (e) weight all models from \(E\) using Eqs. (3) and (4).
   (f) re-train all models from \(E\) using \((x_t, y_t);\)
   (g) if \(|F(x_t) - y_t| / \text{MSE}_t^t > \alpha\)
      i. \(f_0 \leftarrow \) obtain a new model trained with \(D^t\); set \(\text{life}_0 = 0, \text{MSE}_0^t = 0; \) and \(w_0 = 1;\)
      ii. if \(k < B\)
         A. **then** include \(f_0\) to \(E; \) set \(k \leftarrow k + 1, f_k \leftarrow f_0,\) and \(E \leftarrow E \cup f_k; \)
         B. else replace model \(f_j\) by \(f_0,\) where \(j = \arg\max_{i=1, \ldots, k} (\text{MSE}_i^t): f_j \leftarrow f_0;\)
5. **end while**

#### 4.1. DOER Description

DOER builds a dynamic sample-based ensemble of weighted models based on the SW approach. A data window of fixed size is maintained, and when a new sample is available, it is included into the window, and the oldest sample is removed from the window. The main idea is to add a new model trained with the data window when the ensemble’s performance is not satisfactory on the newest sample of the window. The proposed on-line ensemble regression method is presented in Algorithm 6.

The algorithm starts by defining some inputs at the Step 1: a data set \(D = \{(x_t, y_t) | x_t \in \mathbb{R}^{input}, y_t \in \mathbb{R}, t = 1, \ldots, T\}; \) where \((x_t, y_t)\) is the sample given at time \(t, x_t\) is a vector of \(r\) input variables, and \(y_t\) is the output variable; the window’s size, \(m;\) a generic on-line supervised learner; a factor to control the inclusion of a new model, \(\alpha;\) and the maximum number of models, \(B.\)

In Step 2, some variables are set: \(E\) denotes the set of models; \(k\) is the number of models; and \(D_t\) corresponds to the current data window of size \(m;\) at time \(t, \) \(D_t\) initially receives the first \(m\) samples from \(D. \) Step 3 creates the first model from the ensemble. It is trained with the initial data window.

On the on-line phase of the algorithm (from Step 4 to Step 5), for each new incoming sample, the window slides along
the data (Step 4a). This operation excludes the oldest sample, \((x_{t-m}, y_{t-m})\) from the window and includes the newest sample, \((x_t, y_t)\), in the window. The prediction \(y_t^f\) of a new input sample \(x_t\) is obtained using a weighted sum of the models' outputs (Step 4b).

The error of each model \(f_j\) from the ensemble \(E\) \((j = 1, \ldots, k)\) on the newest sample \((x_t, y_t)\) is calculated as (Step 4c):

\[
    e^f_t = (y_t - f_j(x_t))^2.
\]

In this step, the variable life\(_j\) is also incremented. It denotes the total number of on-line evaluations performed with a model \(f_j\). After this, the current error of each model on the current data window, \(\text{MSE}^f_t\), is obtained as (Step 4d):

\[
    \text{MSE}^f_t = \begin{cases} 
        0, & \text{if } \text{life}_j = 0, \\
        \frac{\text{life}_{j-1} \cdot \text{MSE}^f_{t-1} + 1}{\text{life}_j} \cdot e^f_t, & \text{if } 1 \leq \text{life}_j \leq m, \\
        \text{MSE}^f_{t-1} + \frac{e^f_t}{m}, & \text{if } \text{life}_j > m. 
    \end{cases}
\]

This approach (2) is similar to one proposed by OAUE (Brzezinski and Stefanowski, 2014). The objective is to estimate the average of the predictive error of a model \(f_j\) on the last \(m\) samples using MSE. Eq. 2 works like an adaptive MSE on the current window, where the error on the recent sample is included to \(\text{MSE}^f_t\), and the error on the old sample is excluded from \(\text{MSE}^f_t\). A new model initially receives \(\text{MSE}^f_1\) equal to 0. As this model performs on-line predictions and \(\text{life}_j\) is incremented, the window of errors is also enlarged up to a maximum width \(m\), so that new errors \(e^f_t\) are considered to compute \(\text{MSE}^f_t\). Note that DOER discards the predictive error during the training phase. This prevents that a possible overfitting degrades the ensemble's accuracy. The impact of \(m\) will be discussed in the next Section.

Step 4e dynamically assigns the current weight \(w_j\) of each model \(f_j\) according to its error on the window, \(\text{MSE}^f_t\), as:

\[
    w_j = \exp\left(-\frac{\text{MSE}^f_t - \text{med}(\text{MSE}')}visão \text{med}(\text{MSE}') \right), \quad \text{where} \quad \text{MSE}' = [\text{MSE}_1', \ldots, \text{MSE}_k'],
\]

where \(\text{med}(\text{MSE}')\) is the median value of the models' errors, \(\text{MSE}'\). Eq. (3) transforms the weights in such a way that a model \(f_j\) with \(\text{MSE}^f_j\) around the median value receives a weight equal to 1, while models with \(\text{MSE}^f_j\) lower than the median have their weights exponentially increased, and models with \(\text{MSE}^f_j\) larger than the median have their weights exponentially decreased. This strategy allows that models with low accuracies do not impact negatively the ensemble’s performance. On the other hand, more “credit” is given to the models that have high accuracy. In Step 4f, all the models are re-trained, keeping the models updated on the current scenario.

Step 4g evaluates if a new model should be included to the ensemble. The criterion includes a new model when the absolute relative error of the ensemble on the newest sample is greater than \(q\). The new model \(f_0\) is trained using the samples from the current data window, \(\mathbf{D}'\), where its weight is initially set as 1. Therefore, it receives the same weight as a model with error around the median error, \(\text{med}(\text{MSE}')\). This criterion smooths the contribution of a new model at the time \(t+1\), period in which this model will be evaluated on-line for the first time.

In Step 4gii, if the number of models of the ensemble \((k)\) is lower than \(B\), then the value of \(k\) is incremented by 1 and the new model \(f_0\) is attributed to \(f_k\). Otherwise, if \(k\) is greater than \(B\), then \(f_0\) replaces the least accurate model \(f_j\) of the ensemble. The criterion substitutes the model \(f_j\) with the highest error \(\text{MSE}^f_j\). Therefore, a new model created at iteration \(t\) is never excluded by the pruning strategy at the same time \(t\).

5. Experimental Results

In this section, experiments are performed with the DOER and the results are detailed and compared to state-of-the-art approaches. Two synthetic data sets and six real-world data sets are employed to evaluate the algorithms’ effectiveness on different changing scenarios. The tests have been performed on the Matlab environment, running on a PC equipped with an Intel Core i7-2600 3.4 GHz processor of 4 cores and 8 GB of RAM.

5.1. Experimental Setup: Data Set Description

The synthetic data sets are the hyperplane data (Kolter and Maloof, 2005), a benchmark to evaluate approaches that deal with changing environments, and the modified Friedman’s function proposed by Ikonomovska et al. (Ikonomovska, 2012). The hyperplane data set contains involving noise, gradual drift and non-recurring drift (Kolter and Maloof, 2005) (Ad-DeExp). It consists of 10 input variables with uniform distribution over the interval of \([0,1]\), 1 output variable within the interval of \([0,1]\), and \(T\) samples. The data set has 4 concepts, \([C_1; C_2; C_3; C_4]\), where each concept holds \(T/4\) samples. The output of each concept is given by:

- concept \(C_1\): \(y_t = (x_t^1 + x_t^2 + x_t^3)/3\), for \(t = 1, \ldots, T\);
- concept \(C_2\): \(y_t = (x_t^2 + x_t^3 + x_t^4)/3\), for \(t = (T+1), \ldots, T\);
- concept \(C_3\): \(y_t = (x_t^3 + x_t^4 + x_t^5)/3\), for \(t = (T+1), \ldots, 2T\);
- concept \(C_4\): \(y_t = (x_t^4 + x_t^5 + x_t^6)/3\), for \(t = (2T+1), \ldots, T\),

where \(T\) varies in each experiment. A random variate noise uniformly distributed over the interval of \([-0.1,0.1]\) is added to all the output samples. The value of the output is clipped to 0 or 1 if its value is less than 0 or greater than 1, respectively.

The Friedman’s function is a benchmark for generating data (Friedman, 1991). It employs linear and non-linear relations between input and output variables. The original function has 5 input variables, and 1 output variable \(y_1\) which is given by:

\[
    y_t = 10\sin(\pi x_t^1 x_t^2) + 20(x_t^3 - 0.5)^2 + 10x_t^4 + 5x_t^5 + \epsilon, \quad (5)
\]

where \(\epsilon \sim \mathcal{N}(0,1)\) is a zero-mean and unit-variance Gaussian random variable. The input space is extended by adding

---

\(^1\)\(y_t^n\) denotes the value of an input variable \(n\) of a sample \(t\), where \(x_t \in \mathbb{R}^{c<1}\), \(r\) is the number of input variables, and \(n = 1, \ldots, r\).
other 5 input variables $x_5, \ldots, x^{10}$ that are not relevant for predicting $y$. The 10 input variables are uniformly distributed over the interval of [0, 1]. To simulate different changing scenarios, 3 data sets with 2000 samples were generated using the Friedman’s function. The first, the local and abrupt drift data set (Friedman-LA), introduces changes in two different regions of the input space (local drift) using 3 points of abrupt changes. The second, the global recurring abrupt drift data set (Friedman-GRA), simulates global, abrupt, and recurring drifts using 2 drift points. The third, the global non-recurring gradual drift data set (Friedman-GnRG), contains 2 points of gradual changes. It is produced by introducing gradually samples which belong to a different functional dependency in contrast to the original function. The complete description of these data sets can be found in (Ikonomovska, 2012).

Six real-world data sets are considered, as listed in Table 2. They aim to predict important variables in industrial applications. Most industrial processes exhibit some kind of time-varying behavior, and so these data sets are crucial to evaluate the proposed methodologies. The cement kiln data set was obtained in a real-world environment of a cement plant, where the input variable (e.g. temperatures, pressures, concentrations, etc.) samples were recorded with a sampling interval of $T = 1[\text{min}]$, while the output variable samples were obtained with different sampling intervals using a laboratory automation system. Details and descriptions of the other data sets can be found on their corresponding references (Fortuna et al., 2006; Kadlec and Gabrys, 2011; Grbič et al., 2013). Pre-processing strategies were applied to exclude outliers and select input variables in every single data set. For the input and output variables, outliers were removed or replaced by the median using the Hampel identifier (Lin et al., 2007). Input variables highly correlated to the output variable were selected (Fortuna et al., 2006).

5.2. Experimental Setup: Approach Setup and Description

Experiments are performed by comparing DOER to four on-line strategies using the single model OS-ELM algorithm; and five on-line ensemble algorithms (EOS-ELM, AddExp, On-line Bagging (OB), Learn++NSE, and OAUE). The single models are designed using the algorithms presented in Section 3: OS-ELM sample-based (OS-ELMx), as in Alg. 2; OS-ELM batch-based (OS-ELMb), as in Alg. 1; OS-ELM sample-based using a SW (OS-ELMx-SW), as Alg. 3; and OS-ELM batch-based using a SW (OS-ELMb-SW), a modified version of the Alg. 3, where samples are given in batches.

The following structure is employed to evaluate all the approaches. Consider a data set $D = \{(x_i, y_i)\}_{i=1}^T$ with $T$ samples. The initial single model, or the first model of the ensemble, or the pool of models (depending on the approach) is created using the first $m$ samples from $D$, while the other $(T - m)$ samples are grouped to form the on-line data to simulate an on-line scenario. For each approach, its performance (accuracy) is evaluated using mean and standard deviation of the MSE between real and predicted outputs of the on-line data in 20 trials.

The OS-ELM is the base model for all the approaches. For each model, sigmoid is the activation function. The number of neurons in the hidden layer, $N$, is selected by varying it in the interval of [1, 20]. The value of $N$ with best performance on 10-fold cross-validation using the training data set or the current data window was selected for each model. This interval is adjusted to $[1, m]$, if the number of training samples $m$ is lower than 20. On-line data scaling of the input and output variables is applied using zero-mean and unit-variance, where on the online phase, the mean and standard deviation of each variable is recursively adapted as new samples are available (Galicia et al., 2012). The AddExp requires that the output samples are normalized to the [0, 1] interval. Therefore, the outputs of all data sets were firstly normalized to this interval for facilitating the comparison of AddExp with the other methods.

For the on-line ensembles, the maximum number of models is 15, i.e. $B = 15$. This value was chosen to reduce the processing time and memory, since there is not a considerable improvement of the ensembles’ performances when $B$ further increases. In the EOS-ELM, $N$ is set the same for all models, as described by (Lan et al., 2009). The value of $N$ is selected according to 10-fold cross-validation performance on the training data set in 20 trials. While for the other approaches, for each base model, 1 trial was considered to selected the best $N$ for each model. OB was implemented according to the structure in (Oza and Russell, 2001).

The parameters of the AddExp are set based on studies from (Kolter and Maloof, 2005): $\beta = 0.5$, $\gamma = 0.1$, and $\tau = 0.05$ (their descriptions are presented in the Section 2). In our experiment, the first model is created using the first $m$ samples from $D$. If a new model should be included to the ensemble at the time $t$, it is trained using the last $m$ samples from $D$ as: $D' = \{(x_{i}, y_{i})\}_{t-m+1}^{t-1} \subset D$.

Learn++NSE (Elwell and Polikar, 2011, 2009) and OAUE (Brzezinski and Stefanowski, 2014) are ensembles for classification tasks. Here we present modified versions of them to adapt for regression. Learn++NSE was implemented using an boosting regression algorithm, the AdaBoost.RT (Shrestha and Solomatine, 2006). Parameters of the Learn++NSE and AdaBoost.RT are set according to the suggestions of their authors. For example, the factor $\phi$ for demarcating incorrect and correct predictions in the AdaBoost.RT is set to 0.2. Additionally, the parameters $a$ and $b$ are set to 0.5 and 10, respectively. They define the slope and the halfway crossing point of a sigmoid function, a function to compute the models’ weights in the Learn++NSE. The model with the highest current error is removed from the ensemble when the number of models exceeds $B$. The OAUE was adapted and implemented according to the description in the Sections 2 and 3. The DOER algorithm was implemented according to the structure proposed in Section 4. The values of $m$ and $a$ can vary on each experiment and analysis.

5.3. Analysis of DOER parameters

The parameter setting is discussed in this section, since in SW approaches the window’s size is a factor that may influence the system’s accuracy; and in on-line ensemble systems that add dynamically new models, the frequency of including models may also impact the ensemble’s performance.
Figures 1(a)-1(e) indicate that the real data sets contain concepts of large concept drift. For example, for data sets where the rate of concept drift is large, DOER has better accuracy when $\alpha$ is set to a small value for including new models with high frequency so that the ensemble is adapted quickly to the new changes. On the contrary, for data sets with low rate of changes and that require low adaptation capability, $\alpha$ should be set to a large value for adding models in a low frequency. In the experiments below, the hyperplane data set has 2000 samples ($T = 2000$); and $\alpha$ is set to 0.10 for the synthetic data sets, and $\alpha$ is set to 0.04 for the real-world data sets.

5.4. Comparing DOER to other approaches

Tables 3, 4 and 5 report the experimental results based on the MSE of all the approaches using different values of $m$.

For the synthetic data sets, in general, the performances of all the methods improve when $m$ increases. For the SW approaches, as the OS-ELMs-SW and the OS-ELMs-$\omega$-SW, the ideal value of $m$ should correspond to the length of each concept in the data, since the model is trained with samples that maximize the relevance of each concept. As the synthetic data sets contain concepts of large time durations, the SW approaches have high performance for large windows. In other approaches, large windows offer low adaptation capability to the system. This strategy is important to data sets where concepts are replaced more slowly by new concepts, as the synthetic data sets. This can be observed in the Learn++NSE, where the ensemble’s accuracy improves when $m$ increases, since the ensemble is only adapted when a batch is available. Other important remark for the synthetic data sets is that in general the on-line single models have low accuracies when compared to the on-line ensembles, demonstrating that an ensemble is usually more accurate than any single model.

DOER outperforms the other approaches on the data sets with non-recurring concepts, i.e. the hyperplane and Friedman-GnRG data sets. On the other hand, the DOER has inferior accuracy when compared to OAUE and AddExp in the Friedman-LA data set; and the DOER has comparable accuracy to the AdExp method in the Friedman-GRA data set, a data set with recurring concepts. Suppose a recurring drift data set $[C_1; C_2; C_1]$
with 2 concepts, where the algorithm is learning data close to the 2\textsuperscript{nd} point of change, that is, \textit{C}_1 is an old recurring concept and \textit{C}_2 is the previous concept learned by the algorithm. In this scenario, DOER has adapted to the previous concept \textit{C}_2 and the ensemble contains only information about \textit{C}_2, so that DOER takes some time to reintroduce the old concept \textit{C}_1 to the ensemble. On the contrary, OAUE and AddExp adapt more slowly to previous concept \textit{C}_2 exposing that they may contain information about the old recurring concept \textit{C}_1. For example, in the OAUE, a candidate model included to the ensemble contains information/samples about the new batch (most recent samples) and the old batch (the previous batch of samples). On AddExp, the new models have low weights when compared to the old models so that old models have higher contribution to the ensemble, and consequently, the ensemble may still perform well on the old concepts. The experiments using the synthetic data sets seem to support that, in terms of accuracy, DOER is not only comparable to the other approaches of the state-of-the-art, but in most cases DOER has higher performance.

In the real-world data sets, different characteristics are noticed when \textit{m} increases or decreases. In the OS-ELMs, \textit{m} is the number of samples to be used on the training phase. Therefore, when \textit{m} increases the model performs better, and consequently its on-line performance is improved, as can be seen in the powder detergent data set, where for \textit{m} = 20 the MSE is 7.09\times10\textsuperscript{-3}, and for \textit{m} = 60 the MSE is 6.48\times10\textsuperscript{-3}. On the other hand, in the OS-ELMb, \textit{m} has impact not only in the number of samples to be employed on the training, but \textit{m} also influences the
Table 3: Average and standard deviation of the errors obtained on 20 trials of the on-line algorithms using the synthetic data sets. The error corresponds to the MSE samples employed to train a new model. Therefore, data sets

11

<table>
<thead>
<tr>
<th>Method/Data set</th>
<th>Window’s size</th>
<th>All the window’s sizes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(m = 20)</td>
<td>(m = 30)</td>
</tr>
<tr>
<td></td>
<td>(m = 40)</td>
<td>(m = 50)</td>
</tr>
<tr>
<td></td>
<td>(m = 60)</td>
<td></td>
</tr>
<tr>
<td><strong>Friedman-HA data set</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OS-ELM</td>
<td>23.00 (0.28)</td>
<td>21.39 (0.86)</td>
</tr>
<tr>
<td>OS-ELMw</td>
<td>22.42 (1.75)</td>
<td>21.36 (1.20)</td>
</tr>
<tr>
<td>OS-ELMw-SW</td>
<td>23.81 (0.93)</td>
<td>19.47 (0.38)</td>
</tr>
<tr>
<td>OS-ELMw-SW</td>
<td>24.46 (1.02)</td>
<td>16.00 (1.11)</td>
</tr>
<tr>
<td>OB</td>
<td>19.10 (0.37)</td>
<td>19.24 (0.24)</td>
</tr>
<tr>
<td>Learn**-NSE</td>
<td>16.54 (0.88)</td>
<td>11.38 (0.47)</td>
</tr>
<tr>
<td>OAUE</td>
<td>19.70 (0.17)</td>
<td>19.55 (0.12)</td>
</tr>
<tr>
<td>AddExp</td>
<td>7.26 (0.24)</td>
<td>6.73 (0.15)</td>
</tr>
<tr>
<td>DOER</td>
<td>5.89 (0.12)</td>
<td>5.52 (0.07)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Friedman-GRA data set</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OS-ELM</td>
<td>12.81 (4.53)</td>
<td>9.55 (3.14)</td>
</tr>
<tr>
<td>OS-ELMw</td>
<td>11.93 (3.65)</td>
<td>8.48 (1.54)</td>
</tr>
<tr>
<td>OS-ELMw-SW</td>
<td>20.90 (1.47)</td>
<td>15.60 (0.56)</td>
</tr>
<tr>
<td>OS-ELMw-SW</td>
<td>9.61 (1.32)</td>
<td>6.88 (0.35)</td>
</tr>
<tr>
<td>OB</td>
<td>8.45 (0.67)</td>
<td>6.85 (0.23)</td>
</tr>
<tr>
<td>Learn**-NSE</td>
<td>12.77 (0.60)</td>
<td>10.01 (0.39)</td>
</tr>
<tr>
<td>OAUE</td>
<td>6.55 (0.17)</td>
<td>6.51 (0.15)</td>
</tr>
<tr>
<td>AddExp</td>
<td>7.16 (0.22)</td>
<td>6.84 (0.13)</td>
</tr>
<tr>
<td>DOER</td>
<td>7.28 (0.13)</td>
<td>6.89 (0.06)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Friedman-GnRG data set</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OS-ELM</td>
<td>18.87 (4.20)</td>
<td>18.00 (4.84)</td>
</tr>
<tr>
<td>OS-ELMw</td>
<td>22.57 (5.92)</td>
<td>17.93 (7.07)</td>
</tr>
<tr>
<td>OS-ELMw-SW</td>
<td>36.56 (1.80)</td>
<td>26.83 (1.69)</td>
</tr>
<tr>
<td>OS-ELMw-SW</td>
<td>15.75 (5.28)</td>
<td>13.15 (1.25)</td>
</tr>
<tr>
<td>OB</td>
<td>14.79 (0.91)</td>
<td>13.19 (0.78)</td>
</tr>
<tr>
<td>Learn**-NSE</td>
<td>28.43 (2.7)</td>
<td>21.06 (1.01)</td>
</tr>
<tr>
<td>OAUE</td>
<td>12.46 (0.36)</td>
<td>12.21 (0.25)</td>
</tr>
<tr>
<td>AddExp</td>
<td>12.43 (0.20)</td>
<td>11.76 (0.17)</td>
</tr>
<tr>
<td>DOER</td>
<td>12.72 (0.20)</td>
<td>11.68 (0.12)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>The values have been multiplied by 10</strong>.<strong>3</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1 The values have been multiplied by $10^3$.

time interval of the model adaptation. So that, when $m$ is small, the time interval is small, and consequently the model will be adapted at a high frequency. In this case, for the OS-ELM/s, low values of $m$ offer higher adaptation and performance. This can be seen in the debutanizer column data sets, where when $m = 60$ the MSE is $13.85 \times 10^{-3}$, and when $m = 10$ the MSE is $11.90 \times 10^{-3}$. In the OS-ELMs-SW, $m$ holds the number of samples employed to train a new model. Therefore, data sets with less concepts operate better when $m$ is large (large window), while data sets with more concepts perform well with small windows, assuring faster adaptation. The OS-ELM/SW and Learn**-NSE are batch-based algorithms, and $m$ has impact on the time interval of the system adaptation, as in the OS-ELM. Therefore, low values of $m$ guarantee faster adaptation to changes. For example, for the Learn**-NSE using the thermal oxidizer data set, when $m = 20$ the MSE is $12.14 \times 10^{-3}$, and when $m = 60$ the MSE is $24.15 \times 10^{-3}$. In EOS-ELM and OB, $m$ works as in OS-ELMs, i.e. $m$ refers to the size of the initial training data set. Therefore, these algorithms improve their performances when $m$ increases. As shown in the cement kiln data set for the EOS-ELM, when for $m = 10$ the MSE is $22.88 \times 10^{-3}$, and for $m = 30$ the MSE is $19.13 \times 10^{-3}$. In the AddExp, $m$ only refers to the number of samples to be used to train a new model. The results show that variations of $m$ have small impact on the AddExp’s performance, as indicated in the standard deviation of the last column of the tables. In the OAUE, $m$ is applied to measure the models’ performances, to define the number of samples to train a new model ($2 \times m$), and to define the time interval for including a new model in the ensemble. Possibly, $m$ affects mainly the time interval for including a new model, since, in general, approaches that add new models at a high frequency (DOER and AddExp) have a
higher accuracy when compared to other approaches. This is because, in such high frequency approaches such as DOER and AddExp, ensembles contain mainly models trained on the samples of the most recent concept. The results show that when \( m \) decreases, the OAUE model’s performance increases. In the DOER, \( m \) is related to the number of samples to be used for training a new model and to measure the models’ performances. The tests indicated that \( m \) has mainly impact in the measurement of the models’ performances, revealing that more importance should be given to the models that perform better on the most (very) recent samples of the data. In this way, in general, when \( m \) is low, DOER achieves high accuracy. As can be seen, \( m \) plays an important role on all the approaches.

In the real-world data sets, the OS-ELMs-SW has a remark-
able improvement of performance when compared to the other on-line approaches. OS-ELMs-SW has high accuracy in scenarios where the most recent samples are sufficient to describe the system. However, the OS-ELMs-SW requires high computational time when compared to the other algorithms, since in the OS-ELMs-SW, a new model is trained when a new sample is available. The OS-ELMb-SW has low computation time, because a new model is trained only when a batch is available. However, the OS-ELMb-SW has larger error than the OS-ELMs-SW, since in the OS-ELMb-SW, the system is adapted on a batch basis, taking more time to adapt to the new concepts.

The experiments indicated that the presented industrial applications require faster adaptation capability. Therefore, on-line ensembles with few adaptive mechanisms have worse performances, as for example the EOS-ELM and OB which employ few adaptive ensemble mechanisms, since only the models' parameters (retraining of models) are adapted. On both the methods, the ensembles react slowly to changes. The OB slightly outperforms the EOS-ELM. In the OB, each new sample can be presented more times for retraining each model in comparison to the EOS-ELM, so that the OB can adapt faster.

Learn++.NSE employs more adaptive ensemble mechanisms when compared to EOS-ELM and OB. However, the Learn++.NSE’s performance is generally worse when compared to them, since the ensemble is adapted on a batch basis. Additionally, Learn++.NSE does not perform retraining of the models. In contrast to Learn++.NSE, OAUE retracts all the models when a sample is available. However, OAUE adds new models to the ensemble in a low frequency (batch frequency) when compared to AddExp and DOER; and in general, the tests show that AddExp and DOER outperform OAUE. AddExp employs the same adaptive ensemble mechanisms ((i), (ii), and (iii)) as the DOER. However, the AddExp’s performance is inferior in the Tables 4 and 5. As described before, in the AddExp, new models trained on the new concepts take more time to have their weights significantly increased. In scenarios that require faster adaptation to the new concepts, this strategy may fail. In contrast to AddExp, DOER assigns high weights to the new and accurate models if they have low errors on the recent samples.

Table 6 and Table 7 show the processing time (in seconds) of all the approaches in all on-line samples (AOS) and per on-line sample (POS) when $m = 60$ using the synthetic data sets.

<table>
<thead>
<tr>
<th>Method/Data set</th>
<th>AOS</th>
<th>POS</th>
<th>AOS</th>
<th>POS</th>
<th>AOS</th>
<th>POS</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS-ELMs</td>
<td>1.7e-04</td>
<td>4.7e-04</td>
<td>1.7e-04</td>
<td>4.7e-04</td>
<td>1.8e-04</td>
<td>9.0e-04</td>
</tr>
<tr>
<td>OS-ELMb</td>
<td>1.5e-04</td>
<td>4.4e-04</td>
<td>1.5e-04</td>
<td>4.4e-04</td>
<td>1.1e-04</td>
<td>5.5e-04</td>
</tr>
<tr>
<td>OS-ELMs-SW</td>
<td>2.2e-04</td>
<td>6.1e-04</td>
<td>2.2e-04</td>
<td>6.1e-04</td>
<td>2.2e-04</td>
<td>6.1e-04</td>
</tr>
<tr>
<td>OS-ELMb-SW</td>
<td>7.7e-04</td>
<td>8.7e-04</td>
<td>7.7e-04</td>
<td>8.7e-04</td>
<td>7.7e-04</td>
<td>8.7e-04</td>
</tr>
<tr>
<td>EOS-ELM</td>
<td>4.1e-02</td>
<td>2.3e-02</td>
<td>4.1e-02</td>
<td>2.3e-02</td>
<td>4.1e-02</td>
<td>2.3e-02</td>
</tr>
<tr>
<td>OB</td>
<td>4.3e-02</td>
<td>2.6e-02</td>
<td>4.3e-02</td>
<td>2.6e-02</td>
<td>4.3e-02</td>
<td>2.6e-02</td>
</tr>
<tr>
<td>Learn++.NSE</td>
<td>3.6e-02</td>
<td>1.8e-02</td>
<td>3.6e-02</td>
<td>1.8e-02</td>
<td>3.6e-02</td>
<td>1.8e-02</td>
</tr>
<tr>
<td>OAUE</td>
<td>7.9e-02</td>
<td>3.8e-02</td>
<td>7.9e-02</td>
<td>3.8e-02</td>
<td>7.9e-02</td>
<td>3.8e-02</td>
</tr>
<tr>
<td>AddExp</td>
<td>5.5e-01</td>
<td>3.7e-01</td>
<td>5.5e-01</td>
<td>3.7e-01</td>
<td>5.5e-01</td>
<td>3.7e-01</td>
</tr>
<tr>
<td>DOER</td>
<td>7.7e-01</td>
<td>4.7e-01</td>
<td>7.7e-01</td>
<td>4.7e-01</td>
<td>7.7e-01</td>
<td>4.7e-01</td>
</tr>
</tbody>
</table>

The results reveal that “8.7e-04” stands for 8.7 × 10^{-4}. The results reveal that sample-based windowing techniques (OS-ELMs-SW, DOER, and AddExp) are more time consuming when compared to the sample-based techniques that do not use a window (EOS-ELM, OB, etc). This is because, sample-based windowing techniques require that more models are trained. Therefore, in general,
DOER is more time consuming than other ensembles, since more models are designed. Nevertheless, in most cases, DOER gives better accuracy than other methods. For the real-world data sets, all the approaches can perform on-line prediction for each sample in less than 1 second.

5.5. Discussion

In summary, the results in the previous subsections reaffirm the superiority of DOER over the other methodologies in both the real scenarios and the artificial scenarios. SW algorithms using a single algorithm assume that samples that fall outside the window are irrelevant, and such algorithms do not have capability to handle the previously acquired data, since old data are discarded. On the other hand, algorithms able to retain the previously acquired knowledge (e.g. EOS-ELM and OB) have difficulty to adapt quickly to changes, since the old data are still relevant to the learned model. But even if the algorithm is able to conciliate previous data and current data, some algorithms have slow adaptation capability, because a long time is required to introduce new knowledge to the system.

The results in this paper revealed interesting characteristics of the proposed DOER method. DOER attains an error lower than the best single model, the OS-ELM. The results also indicate that DOER has capability to deal with changing environments. And they also indicate that the correct setting of the factor of including a new model to the ensemble, \( \alpha \), is an important issue to control the system’s adaptation capability. For example, in scenarios with less concept changes, \( \alpha \) should be set to approximately 0.01; while in scenarios that require fast adaptation (as in the cement kiln data set or in the polymerization reactor data set), \( \alpha \) should be set to 0.04, since low values of \( \alpha \) do not improve significantly the DOER’s performance; while in other scenarios (e.g. SRU data set), the DOER’s performance does not improve when \( \alpha \) varies. It has also been observed that the accuracy of each approach is related to the setting of \( m \). Therefore, \( m \) should be set using some knowledge about the data or using a value proportional to the data set’s size.

In contrast to other approaches, DOER does not perform any down-weighting mechanism of the old models. Additionally, a recently created model can have a contribution similar to the other accurate models from the ensemble, if they predict well the recent samples. Therefore, old and new models can have the same contribution to the ensemble if they predict accurately the recent set of samples. It has been observed that this characteristic is important to increase the ensemble’s accuracy. Moreover, the DOER’s weighting strategy decreases exponentially the contribution of models that perform poorly on the current window, not allowing that they affect the ensemble’s accuracy.

6. Conclusions

This paper proposed a new dynamic and on-line ensemble regression (DOER) method for on-line prediction of variables in changing environments, with application, for example, for predicting variables measured with significant delay, as in soft sensing applications. The main contribution of this work is the proposal of an on-line ensemble for regression that incorporates three different levels of adaptation (dynamic inclusion and removal of models, models’ weights adaptation, and models’ parameters adaptation), which enable to maintain the system’s performance in changing environments. DOER was shown to have higher accuracy when compared to state-of-the-art approaches in scenarios that require faster adaptation capability, and with non-recurring drifts. According to the simulation results of soft sensing applications, the proposed method can deliver more accurate predictions of the key variables in industrial processes when compared to the traditional SW approach using a single model, commonly used in soft sensing applications. Therefore, the proposed method can be designed for practical use in industrial applications, reducing the time and maintenance costs of traditional systems (e.g. laboratory measurement systems).

Overall, the proposed method has limited capability for predicting local and abrupt drift data sets, and gradual recurring data sets. This may happen because the proposed method loses information about the past scenarios. Moreover, the window’s size setting may have an important role in some data sets, e.g. the debutanizer column data set. Therefore, in these cases, it is important to apply extra experiments using the proposed method to define the window’s size.

Thus, as a future work, the authors propose a variable window’s size that adapts according to the process dynamics and characteristics (Ni et al., 2014). Moreover, it seems to be interesting to propose an adaptive setting of \( \alpha \). In this way, \( \alpha \)

<table>
<thead>
<tr>
<th>Method/Data set</th>
<th>polymerization</th>
<th>cement kiln</th>
<th>thermal oxidizer</th>
<th>powder detergent</th>
<th>debutanizer column</th>
<th>SRU (output 1)</th>
<th>SRU (output 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AOS / POS</td>
<td>AOS / POS</td>
<td>AOS / POS</td>
<td>AOS / POS</td>
<td>AOS / POS</td>
<td>AOS / POS</td>
<td>AOS / POS</td>
</tr>
<tr>
<td>OS-ELMs</td>
<td>0.6 / 9.5e-04</td>
<td>0.6 / 9.5e-04</td>
<td>1.4 / 6.8e-04</td>
<td>1.3 / 6.5e-04</td>
<td>1.2 / 6.7e-04</td>
<td>3.9 / 5.6e-04</td>
<td>4.0 / 5.8e-04</td>
</tr>
<tr>
<td>OS-ELMb</td>
<td>0.6 / 9.0e-04</td>
<td>0.6 / 8.2e-04</td>
<td>1.1 / 5.4e-04</td>
<td>1.1 / 5.5e-04</td>
<td>1.1 / 5.9e-04</td>
<td>3.1 / 4.6e-04</td>
<td>3.1 / 4.6e-04</td>
</tr>
<tr>
<td>OS-ELMs-SW</td>
<td>0.6 / 8.6e-01</td>
<td>0.6 / 8.6e-01</td>
<td>1.1 / 5.4e-04</td>
<td>1.1 / 5.5e-04</td>
<td>1.1 / 5.9e-04</td>
<td>3.1 / 4.6e-04</td>
<td>3.1 / 4.6e-04</td>
</tr>
<tr>
<td>OSS-ELM</td>
<td>2.1 / 3.4e-02</td>
<td>2.2 / 3.4e-02</td>
<td>4.3 / 2.1e-02</td>
<td>3.7 / 1.9e-02</td>
<td>3.8 / 2.1e-02</td>
<td>116.6 / 1.7e-02</td>
<td>123.2 / 1.8e-02</td>
</tr>
<tr>
<td>RSS-ELM</td>
<td>9.8 / 1.4e-02</td>
<td>12.0 / 1.8e-02</td>
<td>3.3 / 1.6e-02</td>
<td>4.2 / 2.2e-02</td>
<td>4.0 / 2.3e-02</td>
<td>126.4 / 1.8e-02</td>
<td>131.2 / 1.9e-02</td>
</tr>
<tr>
<td>Learn++ NSE</td>
<td>14.4 / 2.3e-02</td>
<td>18.5 / 2.8e-02</td>
<td>6.5 / 3.2e-02</td>
<td>5.7 / 1.3e-02</td>
<td>5.9 / 3.1e-02</td>
<td>215.4 / 3.1e-02</td>
<td>205.5 / 3.0e-02</td>
</tr>
<tr>
<td>OAUE</td>
<td>15.0 / 2.4e-02</td>
<td>37.3 / 5.6e-02</td>
<td>137.8 / 6.8e-02</td>
<td>132.5 / 6.9e-02</td>
<td>124.4 / 6.9e-02</td>
<td>705.3 / 1.0e-01</td>
<td>666.7 / 9.9e-02</td>
</tr>
<tr>
<td>AddExp</td>
<td>95.5 / 1.5e-01</td>
<td>242.8 / 3.6e-01</td>
<td>635.1 / 3.1e-01</td>
<td>429.1 / 2.2e-01</td>
<td>545.2 / 3.0e-01</td>
<td>200.0 / 2.9e-02</td>
<td>752.8 / 1.1e-01</td>
</tr>
<tr>
<td>DOER</td>
<td>140.0 / 2.3e-01</td>
<td>294.2 / 4.4e-01</td>
<td>896.4 / 4.4e-01</td>
<td>597.2 / 3.1e-01</td>
<td>599.1 / 4.4e-01</td>
<td>912.3 / 1.3e-01</td>
<td>810.1 / 1.1e-01</td>
</tr>
</tbody>
</table>

Table 7: Processing time (seconds) of all the approaches in all on-line samples (AOS) and per on-line sample (POS) when \( m = 30 \) using the real data sets.
should be set to a high value when a change occurs, and to a low value when no changes are detected. Additionally, future efforts can also be devoted to apply fast dynamic optimization strategies for on-line selection of the best subset of models from the ensemble (Mangat and Vig, 2014).

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).

The authors acknowledge the support of FCT project PEst-C/EEI/UI0048/2014.

The authors are grateful to the Nanyang Technological University for sharing the implementation of the OS-ELM algorithm; and to Dr. Petr Kadlec, M. Eng. Ratko Grbić and Dr. Luigi Fortuna for sharing their data sets.

References


Kaneko, H., Funatsu, K., October 2014. Adaptive soft sensor based on on-line support vector regression and bayesian ensemble learning for various states in chemical plants. Chemometrics and Intelligent Laboratory Systems 137 (0), 57–66.


Soares, S., Antunes, C. H., Araújo, R., December 2013. Comparison of a genetic algorithm and simulated annealing for automatic neural network en-
semble development. Neurocomputing 121 (0), 498–511.