Pruning AdaBoost for Continuous Sensors Mining Applications

M. Rastgoo, G. Lemaitre, X. Rafael Palou, F. Miralles and **P. Casale** [1]

Abstract. In this work, pruning techniques for the AdaBoost classifier are evaluated specially aimed for a continuous learning framework in sensors mining applications. To assess the methods, three pruning schemes are evaluated using standard machine-learning benchmark datasets, simulated drifting datasets and real cases. Early results obtained show that pruning methodologies approach and sometimes out-perform the no-pruned version of the classifier, being at the same time more easily adaptable to the drift in the training distribution. Future works are planned in order to evaluate the approach in terms of time efficiency and extension to big-data analysis.

1 Introduction

As the number of sensors deployed every day in the real world increases, the ambition of mining these continuous data-streams becomes a crucial part in applications. In the recent years, data mining techniques started to be very popular in sensors mining tasks specially when related to learning from data streams [3] [10]. These techniques, stated upon the machine learning framework, are designed to generate a predictive model from a well sampled training dataset distribution. The model is further used to classify any future instance of data without the possibility to be updated if the value distribution of the data-stream changes. In other words, the paradigm provided by the typical machine learning setting is not suitable for continuous mining of data streams [5]. The AdaBoost learning function [2] allows a suitable framework for mining continuous streams [8]. Being an incremental ensemble of classifiers, this learning function is updated to *grow its knowledge* just adding new classifiers to the previous models. Nevertheless, when many subsequent batches of data are provided, Adaboost tends to create large ensembles that suffer of two main drawbacks: (i) increasing memory needed to store the decision model and (ii) over-fitting. *Pruning* techniques can be suited for reducing the dimension of the ensemble by selecting only specific models. The first attempt of pruning an AdaBoost classifiers was introduced by Margineantu and Dietterich [6] by mean of comparing five different methods, namely (i) *early stopping*, (ii) *KL divergence*, (iii) *Kappa statistics*, (iv) *Kappa error convex Hull* and (v) *Reduce error with back-fitting*. Hernanadez-Lobato et al. [4] used Genetic Algorithms to prune the AdaBoost ensemble, searching in the space of all possible subsets of classifiers created by AdaBoost. Zhang et al. [11] defined pruning as a quadratic integer programming problem with the aim to find a fixed size subset of k classifiers with minimum misclassification and maximum diversity. Nevertheless, those works are no suitable solutions for pruning AdaBoost in a continuous learning framework. In this paper, experiments on pruning methods for continuous data-streams mining are performed. The AdaBoost algorithm is trained on subsequent batches of incoming data followed by consecutive *pruning* steps. The advantage of this approach is twofold: (i) on the first hand, when new concepts are learned, pruning allows to maintain the ensemble in order to be the least memory consuming and (ii) on the other hand, pruning provides a first attempt to retain only the *significant information* acquired from previous knowledge. The reminder of this paper is organized as follows. In Section 1, the continuous learning framework, the AdaBoost algorithm and the used pruning methods are introduced and explained in details. In Section 3, validation protocols are described and, in Section 4, results are presented. Finally, Section 5 discusses the obtained results and concludes the paper.

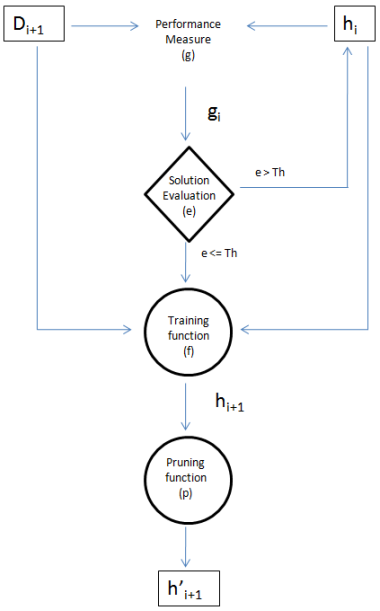


Figure 1. Continuous Learning Framework

2 Pruning AdaBoost in Continuous Learning

In a continuous learning framework, as shown in Fig. 1, new knowledge is acquired only when the current model does not fit anymore the incoming data-stream distribution [9]. This decision is performed by evaluating the current classifier using the function g as performance measure and evaluating the obtained performance e. When e is not *good enough*, the current model h_i is updated training the

[1] Barcelona Digital Technology Center, Barcelona, Spain, email: plcasale@bdigital.org

learning function f with the new incoming data D_{i+1}. Incremental learning functions should be preferred. In this way, only the new incoming data will be used for both maintaining the previous knowledge acquired, not having to store historical data. The AdaBoost algorithm represents an incremental learning function able to properly meet these requirements. Nevertheless the classifiers created by AdaBoost grows linearly as many subsequent learning steps are performed. Here, the pruning function p allows to maintain the model computationally optimal. Aim of this work is evaluating between different pruning functions p in terms of classifier performance. In the following subsection, AdaBoost and the pruning methods are presented and explained in details.

2.1 AdaBoost

AdaBoost, short for Adaptive Boosting, is an ensemble learning algorithm that allows to obtain an high performance classifier by a linear combination of weak learners. Algorithm 1 shows the pseudocode for AdaBoost. The algorithm takes as input a training set $(\mathbf{x}_i, y_i)$ where $\mathbf{x}_i$ is a N-dimensional feature vector, and y_i are the class labels. After T rounds of training, T weak classifiers h_t and T weights α_t are combined to assemble the final strong classifier. Higher weights α_t are assigned to the best weak classifiers h_t. Instantiations of AdaBoost may differ due to the choice of the

Algorithm 1 AdaBoost Algorithm

Input:
- Training set of N samples $(\mathbf{x}_i, y_i)$, with $i = 1 \dots N$, $\mathbf{x}_i \in R^N$, $y_k \in Y = \{1, +1\}$;
- Weak learning algorithm *WeakLearn* ;
- Number of learning iteration T ;

Initialize $W_1(k) = 1/N, k = 1, \dots, N$;

for $t = 1, \dots, T$ **do**

1. Train *WeakLearn* using distribution W_t and get weak hypothesis h_t ;
2. Compute classification error $\epsilon_t = Pr_{k \sim W_t}[h_t(x_k) \neq y_k]$;
3. Compute $\alpha_t = \frac{1}{2} \ln(\frac{1-\epsilon_t}{\epsilon_t})$;
4. Update distribution:

$$W_{t+1}(k) = \frac{W_t(k) \exp(-\alpha_t y_k h_t(x_k))}{Z_t} ;$$

where Z_t is a normalization factor chosen so that W_{t+1} will be a proper distribution function.

end for

Output:
$H(x) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(x))$;

weak learning algorithm, defined as a learner performing slightly better than random guessing ($> 50\%$ right-classification). A variety of weak learners e.g., neural networks or decision trees can be used. Decision stumps are the most common weak classifiers used in AdaBoost. Decision stumps are one-level decision trees equivalent to a threshold that best splits the data. Each stump learner is characterized by three parameters: (i) the n^{th} dimension of the features set where the classifier is applied, (ii) the decision level, i.e., the *threshold* splitting the data in the n^{th} given dimension and (iii) the decision sign (-1 or $+1$) determining the inequality direction for the thresholding. For a given batch of data with a set of features of size n, at each iteration of AdaBoost the decision stump that minimizes the error ϵ_t in an n^{th} dimension of the training distribution is selected. The information provided by the final set of decision stumps selected by AdaBoost can be used for mining which are the significant features of the data-stream and, more important, which is the best split in the data.

2.2 Pruning methods

Three different pruning methods have been used and compared, namely (i) Reduce Error, (ii) Learner Weights Analysis and (iii) Pareto Analysis. The Reduced Error algorithm was used first in [6]. Being the original implementation not suitable from a continuous learning framework, an improved version is proposed in this work in order to speed-up the process. Pruning has also been performed using Learner Weights and Pareto Analysis methodologies, both of them able to provide a set of most discriminative learners from the whole ensemble. From the far of our knowledge, no previous application of those methodologies has been done in the tasks of pruning an AdaBoost ensemble.

2.2.1 Reduce Error (RE)

In this algorithm, the first step is performed in order to initialize the pruning distribution W_t and to select the weak classifier h_t from the ensemble H which minimizes the classification error ϵ_t on W_t distribution. This classifier is added to the pruned ensemble P, a weight α_t is assigned to it and W_{t+1} distribution is also updated as in AdaBoost routine. Then, iteratively, each remaining classifier h_t is individually added to the ensemble P and the classification error ϵ_t of this new ensemble is evaluated on the pruning set using W_{t+1} distribution. In order to select the best classifier, the classifier h_t combined with P minimizing the classification error ϵ_t is definitely added to P, a weight α_t is assigned to it and W_{t+2} distribution is also updated as in AdaBoost routine. The routine stops when the number of classifiers in the sub-ensemble P reaches a ppre-specified size. The two main changes with respect the original RE algorithm are the following. In the original version, a final back-fitting approach is performed only after the selection of each weak classifier while in our approach selection is done at each step. In addition, each weak classifier is added to the pruned ensemble P only after being re-weighted. This procedure ensures better classification results than the original RE formulation.

2.2.2 Learner Weights Analysis (WA)

From the distributions of the weights α_t in the ensemble, weak learners were selected based on the following assumptions: (i) weak learners with higher ensemble weight α_t are the best weak learners of the ensemble and (ii) an ensemble is better when more diversified the classifiers forming it are. The technique works as follow. AdaBoost is applied on the batch of data to obtain an ensemble of T classifiers. Then, a matrix M is built, by grouping the ensemble weights α_t of each decision stump classifier using their dimension parameter. M is of size $n \times D$ where n is the number of element for each of the D dimensions. In order to select the *best* classifiers, M is first sorted formerly by row and subsequently by column, always in a descendant order. M is transformed into a vector V by concatenating all its columns. Finally t classifiers corresponding to the t first weights of V, with $t << T$, are selected. The value of t determines the pruning percentage.

2.2.3 Pareto Analysis (PA)

PA is based on the assumption that few key actions will produce significant overall effects. Applied to ensemble learning, this technique implies that only few key weak classifiers will have an high impact on the overall performance of the ensemble. *PA* proposed a statistical

point of view in order to select these key classifiers. This technique is used to estimate effectiveness of each feature dimension, and accordingly selects the classifiers from feature dimensions with high impact. The effectiveness could be adjusted using a threshold. First, the features are grouped based on the total number of ensemble weight which are considers as outliers in each dimension. The outliers could be found with reference to first and third quartile (Q_1, Q_3), and inter quartile range (IQR). Values above $Q_3 + 1.5 \times (IQR)$ are considered as outliers in each case. The frequency distribution of these outliers is sorted in descendant order and the cumulative distribution is computed. Then, the features dimensions are selected based on a threshold level corresponding to the number of classifiers to keep. All dimensions with lower cumulative percentage than the threshold (i.e. desired percentage of maximum cumulative value) are taken into account. From the selected feature dimensions, the maximum weights are used to highlight the learners. The technique can be perceived as a principle dimension selection, where the dimensions considered as more important are selected.

3 Validation Protocol

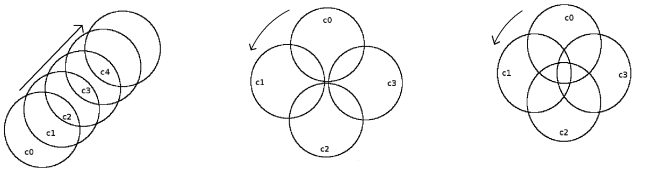
Three typologies of experiments have been performed in order to validate the effectiveness of the pruning methods on both static and drifting distributions. A cross-validation approach has been used for validating the methods. At each step of the cross-validation procees, the dataset has been randomly divided into three sub-sets, training (50%), pruning (40%) and testing(10%) sets. In the following sections the validation protocols adopted for each topology of experiment are described. Under the model described in Fig. 1, a proper threshold Th has been chosen in order to train the model always on the new incoming data.

3.1 UCI Datasets Repository

Five datasets from the UCI repository [1] have been used for evaluating the effectiveness of the pruning methods. In this validation step, the KL divergence method as originally proposed in [6], has been added in order to have a baseline comparison. The datasets considered are Australian, Breast, Diabetes, Heart and Pima. The mean number of instances in the datasets is around 700, except Heart having 270 instance. The aim of the experiment is to analyse the results by pruning at 90% an initial ensemble. The average error rate for each technique was computed using a modified version of ten fold cross-validation able to consider the pruning sets into the evaluation process, with the percentage previously outlined. AdaBoost algorithm was used to create an ensemble of hundred weak classifiers. Then, each pruning method was performed in order to create a pruned sub-ensemble containing only ten classifiers.

3.2 Simulated Drifting Datasets

The second set of experiments has been focused on testing the pruning methods in a continuous learning framework. These have been performed using three sets of simulated data-streams that include drifting. The datasets are generated using the software provided by [7]. Figure 2 shows the three different settings for each experiment. Four linear drifts have been considered for the first dataset and three circular drifts have been created for the remaining two datasets. The ensemble was incrementally grown using all the drifted distributions. The experiments performed using the simulated datasets are described in the following.



(a) Linear Drift (b) Circular Drift (c) Circular Narrow Drift

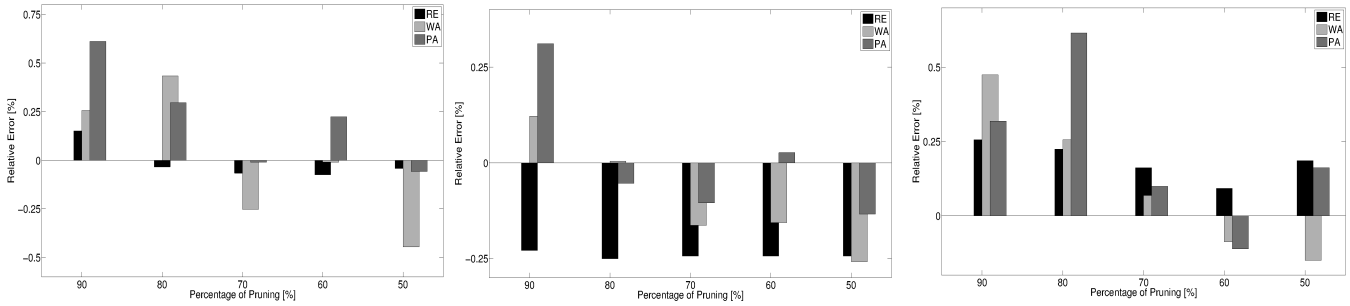
Figure 2. Artificial data with drifts

Exp. 1: In the first experiment, it is assumed that data distribution is subject to the change due to different drifts and the ensemble is incrementally grown over the drifted batches of incoming data with the main aim to classify the current batch of information. After the training, the pruning and the testing are applied on a different samplings of the same drifted batch. The experiment is repeated five times following a 5-fold cross-validation paradigm.
Exp. 2: The aim of the second experiment is to evaluate the potential of pruning in classifying both previous and current information. With the training kept as in the previous experiment, at each step i the ensemble is pruned and tested on pruning and testing sets of the joint distribution $C_0 \cup \ldots \cup C_i$. The experiment has been performed on five different runs, following a 5-fold cross-validation paradigm.

3.3 Real World Datasets

Three real-world datasets have been used in order to evaluate the proposed methodology on a real world scenario. The datasets considered are described in the following.

- The *Sensor Stream*(SS) dataset [12] contains sensors information (temperature, humidity, light and sensor voltage) collected from fifty-four sensors deployed at Intel Berkeley Research Lab. The whole stream contains consecutive information over two months (2 219 803 instances). The experiment aims to infer the illuminance state based on the measurements provided by each sensor. Illuminance higher than 200 *lux* are considered as class 1 otherwise considered as class -1. Every fifteen days, a new batch of data is collected which leads to three drifts considering the changes in the lab environment due to weather, humidity and office work. The experiment was performed using 4-fold cross-validation paradigm.
- *Power Supply*(PS) [12] is the second dataset used. The dataset contains hourly power supply consumptions of the Italian electricity company. The stream contains three year power supply records from 1995 to 1998 (29 928 instances). The experiment aims to predict the day state morning (1) - night (-1)) based on the raw consumption value. The drifting in this stream is mainly derived by some features such as the season, weather, hours of a day and the day of the week. The data were split in three batches representing one drift for each year. The experiment was performed using 3-fold cross-validation paradigm.
- *Elec 2*(E2) is the third dataset used. This dataset containing 27 549 instances is composed of seven drifts, each representing a week day. The drifts are due to changes of power consumptions over the weekdays. The experiment was performed using 7-fold cross-validation paradigm.



(a) Results obtained on Linear Drift in Exp.1 (b) Results obtained on Circular Drift in Exp.1 (c) Results obtained on Circular Narrow Drift in Exp.1

Figure 3. Results obtained on simulated drifting datasets for Exp.1

As in the Exp. 2 on simulated data, AdaBoost is trained for each drift on the training set of current data. The pruning function is applied on a pruning set which contains samples of previous and new batches of data.

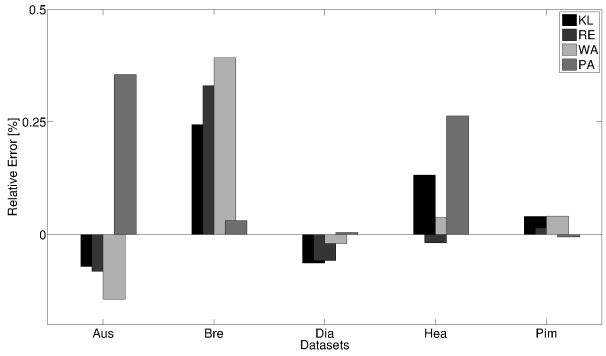


Figure 4. Performance of pruning methods on UCI datasets with %90 pruning

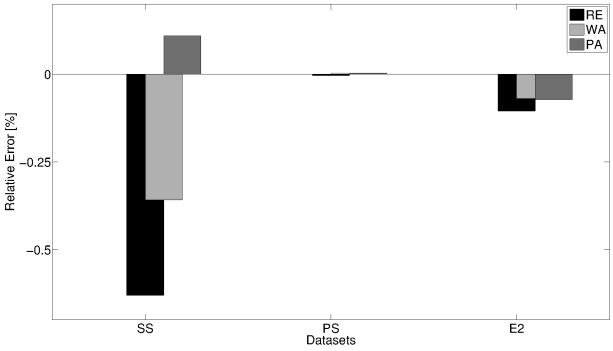


Figure 5. Performance of pruning methods on real world datasets

4 Experimental Results

In this section, results obtained on the experiments described in the previous section are reported. Misclassification error has been chosen as performance measure. In particular, the pruning methods has been evaluated using the relative error (ϵ_{rel}) with respect to the error provided by the no-pruned version of AdaBoost, computed as shown in Eq. 1. Hence, methods with negative relative errors are performing better than the reference model.

$$\epsilon_{rel} = -1 \cdot \frac{\epsilon_{no\ pruned} - \epsilon_{pruned}}{\epsilon_{no\ pruned}} \quad (1)$$

4.1 UCI Datasets Repository

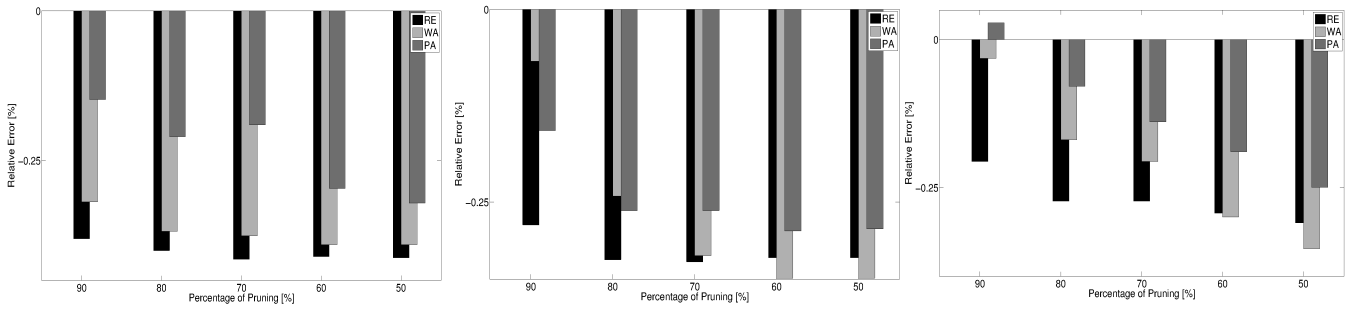
In Fig. 4 the results obtained on the five UCI datasets are reported. RE is the method performing better than the others, being better than the reference in Aus, Dia and Hea datasets, and slightly worst than the reference in Pim. Similar behavior is obtained by WA. Pruning performs always bad on Bre, where the best result is provided by PA.

4.2 Simulated Drifting Datasets

Results obtained on simulated drifting datasets with Exp. 1 are reported in Fig. 3. RE is the best pruning method for linear and circular drifting datasets, as previous experiments suggest. In both linear and circular drifting, WA performs better than PA. Non of the pruning methods work better than the no-pruned version for high percentage of pruning. Nevertheless, WA works slightly better than no-pruned Adaboost when the percentage of pruning is almost 50%. As it may be expected, the performance of the pruned ensemble generally get worse as the percentage of pruning increases. Nevertheless, RE is able to maintain its performance constant over the pruning percentage in the circular dataset and almost constant in the narrow pruning dataset. For Exp. 2, results obtained on simulated drifting datasets are reported in Fig. 6. In this setting, all the pruned ensemble behave better than their correspondent no-pruned classifiers. As all previous experiment suggest, RE is the best method, followed by WA. Also in this case, although the performance of the methods decreases as the percentage of pruning increases, RE remains almost constant regardless of the percentage. It should be also noted that the AdaBoost performance in this experiment is rather bad, reaching a global error up to 40%. The pruning methods improve this performance until reaching an error of 25%.

4.3 Real World Datasets

Results obtained on the real world dataset are shown in Fig. 5. Results obtained with the PS datasets are shown in Fig. 7. RE confirms to be the best pruning method, followed by WA. For SS and E2 datasets, WA and PA provide the same performance. It should be noted that RE performs better than the no-pruned version for all the experiments.



(a) Results obtained on Linear Drift in Exp.2 (b) Results obtained on Circular Drift in Exp.2 (c) Results obtained on Circular Narrow Drift in Exp.2

Figure 6. Results obtained on simulated drifting datasets for Exp. 2

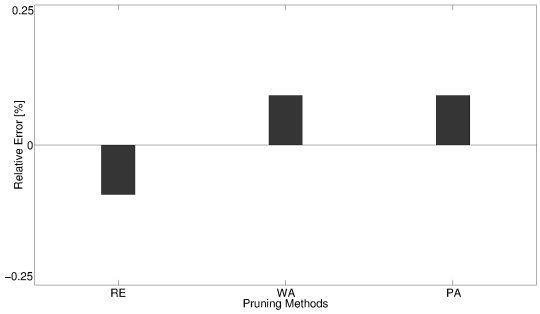


Figure 7. Performance of pruning methods on the PS dataset

5 Discussions and Conclusions

In this work, experiments have been carried out in order to evaluate the potential of different pruning methods and their performance in the framework of continuous learning. The *Reduced Error* method is the most consistent method followed by *Learner Weight Analysis*. The use of *Pareto Analysis* does not seem to be justified during the experiment. Nevertheless, one of the important characteristic of this method consists in the capability of defining automatically the number of classifiers of the pruned ensemble. PA may be automatized by thresholding the performance. Early results show that this *automatic* version performs better than the original method in most of the cases. Experiments on simulated datasets in case of Exp 1 show that pruning methods are more efficient over wider drifted distribution rather than narrow drifted distribution. Due to the nature of the narrow circular dataset, drift stages have more common area and since in this experiment, current stage has more effect for pruning, compare to previous stage, the pruning performances are slightly lower. At the same time, Exp 2 show that pruning methods perform better than the original classifier when the whole drifting distribution is presented. Based on Fig. 6, pruning ensemble through the incremental learning, definitely improves the final results. Finally, results obtained by experiments on real datasets prove that pruning through the continuous learning process provides very close or better results than AdaBoost. As future works, an evaluation of the method efficiency in terms of computational complexity will be considered since this parameter has a great importance in a continuous learning framework. For this main motivation, the reduced error method had been modified in our research in order to be conceptually capable to run following time efficiency guidelines and methods based on genetic algorithm and semi-definite programming have been not used for comparison. Finally, a study on the extension of the proposed methods towards a big-data approach is planned to be done. This research shows that pruning by selecting the weak classifiers from different pools of subsampled data may improve the final ensemble in terms of accuracy, diversity and adaptation ability to drift. The employed procedures in this work can be easily adapted for large datasets and continuous learning environment with the high quantity of incoming data.

6 Acknowledgments

This work is supported by the Information and Communication Technologies Collaborative Project action BrainAble within the Seventh Framework of the European Commission, project number ICT-2010-247447.

REFERENCES

[1] D.J. Newman A. Asuncion. UCI machine learning repository, 2007.
[2] Y. Freund and R. Schapire, 'A short introduction to boosting', *J. Japan. Soc. for Artif. Intel.*, **14**(5), 771–780, (1999).
[3] J. Gama and M. Gaber (Eds), *Learning from Data Streams – Processing techniques in Sensor Networks*, Springer, 2007.
[4] Daniel Hernández-Lobato, José Miguel Hernández-Lobato, Rubén Ruiz-Torrubiano, and Ángel Valle, 'Pruning adaptive boosting ensembles by means of a genetic algorithm', in *IDEAL*, pp. 322–329, (2006).
[5] Ludmila I. Kuncheva, 'Classifier ensembles for changing environments', in *In Multiple Classifier Systems*, pp. 1–15. Springer, (2004).
[6] Dragos D. Margineantu and Thomas G. Dietterich. Pruning adaptive boosting, 1997.
[7] Leandro L. Minku, Allan P. White, and Xin Yao, 'The impact of diversity on online ensemble learning in the presence of concept drift', *IEEE Trans. on Knowl. and Data Eng.*, **22**(5), 730–742, (May 2010).
[8] Martin Scholz and Ralf Klinkenberg, 'Boosting classifiers for drifting concepts', *Intelligent Data Analysis (IDA), Special Issue on Knowledge Discovery from Data Streams*, **2006**, 2007, (2006).
[9] Jan Tozicka, Michael Rovatsos, Michal Pechoucek, and Stepan Urban, 'Malef 58: Framework for distributed machine learning and data mining', *Int. J. Intell. Inf. Database Syst.*, **2**(1), 6–24, (February 2008).
[10] Haixun Wang, Wei Fan, Philip S. Yu, and Jiawei Han, 'Mining concept-drifting data streams using ensemble classifiers', in *Proceedings of the ninth ACM SIGKDD*, KDD '03, pp. 226–235, New York, NY, USA, (2003). ACM.
[11] Yi Zhang, Samuel Burer, and W. Nick Street, 'Ensemble pruning via semi-definite programming', *Journal of Machine Learning Research*, **7**, 1315–1338, (2006).
[12] X. Zhu. Stream data mining repository, 2010.