A new on-line learning method for coping with recurring concepts: the ADACC system

Ghazal Jaber[1,2,3], Antoine Cornuéjols[1,2], and Philippe Tarroux[3]

[1] AgroParisTech, UMR 518 MIA, F-75005 Paris, France
[2] INRA, UMR 518 MIA, F-75005 Paris, France
[3] Université de Paris-Sud, LIMSI, Bâtiment 508, F-91405 Orsay Cedex, France
{ghazal.jaber,antoine.cornuejols}@agroparistech.fr
philippe.tarroux@limsi.fr

Abstract. When the environment changes, as is increasingly the case when considering unending streams and long-life learning tasks, it is necessary to rely on on-line learning with the capability to adapt to changing conditions a.k.a. concept drifts. Previous works have focused on means to detect changes and to adapt to them. Ensemble methods relying on committees of base learners have been among the most successful approaches. In this paper, we introduce a new second-order learning mechanism that is able to detect relevant states of the environment in order to recognize recurring contexts and act pro-actively to concepts changes. Empirical comparisons with existing methods on well-known data sets show the advantage of the proposed algorithm.

Keywords: online learning, ensemble methods, concept drift

1 Introduction

Recent years have witnessed the emergence of a whole new set of applications involving data streams made of pairs $(\mathbf{x}_t, y_t)$, where the "answer" or true label y_t is revealed (sometimes long) after the input $\mathbf{x}_t$. Additionally, it is often the case that the unknown target function evolves with time, something known as *concept drift*. These new needs have spurred a surge of research works geared towards the development of adaptive strategies [1]. Most of them operate either by passively tracking the evolving concept or by using an explicit detection mechanism of concept changes before launching an adaptation or relearning process.

However, better learning strategies may take advantage of the examination of the history of past concept in order to anticipate likely future changes or to recognize when a past concept recurs. We have developed ADACC (*Anticipative Dynamic Adaptation to Concept Change*), a system that uses this kind of second order learning to accelerate its adaptation to changing conditions in the environment. This is accomplished through an ensemble method that controls a pool of incremental learners. Our main innovation lies in the design of a method to select decisive concepts in the history of the system. This enables the recognition

of recurring concepts and opens at the same time the possibility of learning the underlying trends. The latter will be described elsewhere.

In the following, noticeable existing works on online learning in the context of concept changes are described in Section 2. ADACC is presented in Section 3 while Section 4 reports our empirical results and a comparison to the state of the art methods. Section 5 concludes with possible avenues for future work.

2 Relevant Works

In presence of concept drift, the central concern it to learn from data relevant to the regularities currently governing the world. Several directions have been proposed to dynamically adapt the memory of the past data. One is to use time sliding windows with a fixed or a varying size [2]. This necessitates either a priori knowledge about the dynamics of the environment, or a good heuristic that guesses when to shrink or expand the sliding window. Another approach relies on weighting past data with respect to their relevance to the current situation [3]. Again, this is all dependent on the design of an appropriate weighting mechanism. By contrast with these types of approaches that explicitly control the memory of past data, another set of techniques relies instead on the control of past hypotheses. Thus, ensemble-based approaches maintain a pool of on-line learners that each maintain their own memory of the past and compete for giving advice on the current query $\mathbf{x}_t$. By managing the population of these base-learners, and possibly their weights, based on their current prediction performance, one is implicitly controlling the use of past data [4].

A specially interesting case is the one of recurring concepts. This may happen for instance when the environment is subject to seasonal variations which repeats over time. This problem introduces a new challenge in that apparently obsolete data or learned concepts may well be relevant again in the future. It would thus be profitable to exploit this past knowledge as soon as it is appropriate rather than learn anew from the incoming data stream. This requires, however, that interesting memory traces be stored and that their relevance to the current situation be quickly recognized. In recent years, several proposals have been put forward to meet this challenge, particularly within the ensemble-based approach (see for instance [5–9]). They either work at the level of the examples themselves, by chunking them in some way, possibly organizing a hierarchy of chunks, or they work at the level of the learned concepts, trying to learn significant concepts in the stream of examples while avoiding redundancy.

This paper presents a new technique which solves both the problem of detecting regularities (concepts) that are significant and new, and the problem of recognizing when to use past knowledge. It is supposed that the data stream is governed by piecewise stationary environments with concept shifts in between. We show how an ensemble-based method can be used to detect stable environments and how a memory of past concepts can be built which both avoids redundancies and permits a quicker recognition of relevant past concepts than a purely adaptive approach. Unlike other systems handling concept recurrence

[6, 7], our method does not recognize new concepts using a drift detection system. Thus, it avoids the difficulties inherent in having to detect gradual concept changes while still being robust to false alarms.

3 Concept Changes: Adaptive and Anticipative

The proposed technique, called ADACC, implements an anticipative mechanism, able in particular to exploit recurring concepts, build on top of an ensemble-based adaptive online learner. We first briefly describe the latter one.

3.1 Adapting to Concept Changes

The main idea is to maintain a pool of base learners $\{h_t^i\}_{1 \leq i \leq N}$, each of them adapting to the new input data, and to administer this pool or ensemble thanks to a strategy for inserting and deleting base learners. Sketchily:

- Each *base learner* in the pool continuously adapts with new incoming data until it is removed from the pool.
- Every τ time steps, the base learners are *evaluated* on a window of size τ_{eval}.
- Based on the results of this evaluation, the deletion procedure *chooses* a base learner to be removed.
- A new base learner is *created* and inserted in the pool. It is protected from possible deletion for a duration τ_{mat}.
- For each new incoming instance $\mathbf{x}_t$, the prediction $H(\mathbf{x}_t)$ results from a *combination* of the prediction of the individual base learners $h_t(\mathbf{x}_t)$.

Variations around this general framework lead to specific algorithms [4, 10–13]. For instance, after extensive testings, we converged on the following settings. The *evaluation* procedure counts the number of erroneous predictions on the last τ_{eval} time steps. The *deletion strategy* randomly selects one base learner from the worst half of the pool evaluated as above. The *global prediction* uses the prediction from the current best base learner (a vote is applied in case of ties). For simplicity $\tau = \tau_{mat}$. This simple method offers a good trade-off between keeping as much as possible relevant information about the past and be reactive when the underlying concept changes aka. the stability-plasticity dilemma.

3.2 Recognizing recurring concepts

Coping with recurring concepts implies, first, to be able to store memory traces of past relevant concepts and, second, to recognize which past concept is relevant again in order to exploit this knowledge.

In our approach, the memory lies entirely in the pool of base-learners. The question is then to use this pool in order to detect when a regularity deserves to be stored away and then to memorize this regularity for potential future use.

The technique we propose is based on the assumption that the base learners converge toward approximately the same, near optimal, concept, as measured

by their prediction performance, given a stationary environment and a sequence of examples of sufficient length. Assuming this, therefore, one way to detect that a stationary environment has settled is to check that the diversity of the base learners is low, under some threshold, while their prediction rate peaks.

Accordingly, we define a *stability index* that compounds a measure of diversity with an estimation of performance for the best half of the base learners in the pool[4]. We suggest the kappa statistics $\mathcal{K}$ [14] in order to compute *diversity*. This statistics measure evaluates the degree of agreement between the classification of a set of items by two classifiers[5]. In case of complete agreement, $\mathcal{K} = 1$. If there is no agreement other than what would be expected by chance, $\mathcal{K} = 0$. The *stability index* at time t is computed over the last τ_s received examples: $I_{stability} = agreement - error$ where *agreement* and *error* are computed over the best half of the current hypotheses in the pool.

$$agreement = \frac{\sum_{i=1}^{N/2} \sum_{\substack{j=1 \\ i \neq j}}^{N/2} \mathcal{K}_{h_t^i, h_t^j}}{\frac{N}{2} * (\frac{N}{2} - 1)} \tag{1}$$

and:

$$error = \frac{\sum_{j=0}^{\tau_s - 1} \sum_{i=1}^{N/2} err\big(h_t^i(\mathbf{x}_{t-j}), y_{t-j}\big)}{\tau_s * \frac{N}{2}} \tag{2}$$

where N is the size of the pool.

It is then possible to draw a curve of the successive stability indexes (see Figure 1 bottom). For each time step when the curve overcomes a given threshold, the best base learner in the pool is considered as a candidate *snapshot* (a description of the current governing concept). This snapshot is compared with the list of already stored ones and is kept only insofar as it sufficiently differs from all of them. Here again the agreement statistics can be used to measure the difference between a candidate snapshot h_t^* and each stored snapshot $h_{t_k}^*$ on the last τ_s examples. In case the agreement is less than some predefined threshold θ_d, the current candidate snapshot h_t^* is added to the list $\mathcal{M}_{LT}$ (Long Term Memory) which represents past stationary states of the environment. These snapshots are evaluated in the same way as the base learners in the pool and compete for the prediction of y_t given the current $\mathbf{x}_t$. In this way, except for a moderate overhead, the prediction performance of the system is guaranteed to be at least as good as the one of the purely adaptive strategy. (See Algorithm 1)

4 Datasets and experiments

We carried out experiments on two artificial datasets (STAGGER and ELIST) and one real dataset (SPAM). These data sets are well-known benchmarks (see for instance [2, 5, 10, 12]).

[4] Taking into account the poorest base learners induces instabilities that lead to inferior performances

[5] Other agreement statistics should do as well.

Algorithm 1: Selection of snapshots by ADACC.

input : The stability threshold θ_I, the difference threshold θ_d, and the evaluation window τ_s

begin
  $E_0 \leftarrow \emptyset$; /* Ensemble of experts */
  $\mathcal{M}_{LT} \leftarrow \emptyset$; /* List of snapshots */
  $k \leftarrow 0$;
  for $t = 1$ **to** ∞ **do**
    /* Adaptation */
    $(\mathbf{x}_t, y_t)$ is the current training instance;
    $[E_t, \tilde{y}_t] \leftarrow AdaptationEnsemble(E_{t-1}, \mathbf{x}_t, y_t)$;
    /* Anticipation */
    $H = \left\{h_t^i\right\}_{i=1}^{N/2}$ is the best half of experts in E_t;
    $$agr = \frac{1}{\frac{N}{2} * (\frac{N}{2} - 1)} \sum_{i=1}^{N/2} \sum_{\substack{j=1 \\ i \neq j}}^{N/2} \mathcal{K}_{h_t^i, h_t^j};$$
    $$error = \frac{1}{\tau_s * \frac{N}{2}} \sum_{j=0}^{\tau_s - 1} \sum_{i=1}^{N/2} err\Big(h_t^i(\mathbf{x}_{t-j}), y_{t-j}\Big);$$
    $I_{stability} = agr - error$;
    /* Detect Stable Concept */
    if $I_{stability} \geq \theta_I$ **then**
      $h_t^* = snapshot(E_t)$;
      /* Detect New Concept */
      if $isEmpty(\mathcal{M}_{LT})$ **or** $\mathcal{K}_{C_j, h_t^*} \leq \theta_d, \forall j \in [1 \dots k]$ **then**
        $k = k + 1$;
        $C_k = h_t^*$;
        $\mathcal{M}_{LT} = add(\mathcal{M}_{LT}, C_k)$;
end

STAGGER [15] This data stream corresponds to three successive target concepts: $A \Leftrightarrow size = small \wedge color = red$, $B \Leftrightarrow color = green \wedge shape = circular$ and $C \Leftrightarrow size = medium \vee large$. Each concept governs the labeling of 10,000 training instances chosen uniformly from the instance space. To simulate recurring contexts, we concatenated the original sequence with a copy of it, creating a stream of size 60,000.

ELIST [5] This is a stream of email messages from different topics that are sequentially labeled as *interesting* or *junk* by a user. The stream contains 1,500 examples with 913 attributes (boolean bag-of-words representation). Two contexts succeed each other. In one, the user is only interested in messages related to medicine. In the other, the user's interest switches to space and baseball. The stream is the sequence C_1, C_2, C_1, C_2, C_1 where C_1 and C_2 are sequences of 300 email messages, labeled according to the first and second context respectively.

SPAM This dataset [5] consists of 9,324 instances drawn from the email messages of the Spam Assassin Collection using the boolean bag-of-words representation with 500 attributes. As mentioned in [5], the characteristics of the spam messages gradually change as time passes. ELIST and SPAM are available in arff format at `http://mlkd.csd.auth.gr/concept_drift.html`.

4.1 Experiments

The ADACC system was evaluated against the following methods:

Simple incremental classifier (SIC): a single incremental classifier.
Moving window (MW): incrementally learns over the last w instances.
Weighted examples (WE): larger weights are assigned to recent examples in order to gradually forget the outdated information.
Dynamic Weighted Majority (DWM) [10]: an ensemble method that does not use a drift detection system. Each classifier is initially assigned a weight of one. If a classifier misclassifies an instance, its weight is multiplied by a factor $\rho < 1$. A classifier is removed if its weight falls below a threshold θ. A new classifier is added when the ensemble misclassifies an instance. The ensemble size is thus variable. The frequency of updating weights, removing and adding classifiers, is controlled by a parameter p. The decision of the ensemble is obtained by weighted majority voting.
Leveraging Bagging (LBAG) [16]: a version of online bagging that uses the ADWIN method [17] to detect concept drifts. The instances are weighted according to a Poisson(λ) distribution with $\lambda > 1$ in order to achieve more diversity in the generated weight values. When a concept drift is detected, the worst classifier is replaced with a new one. ADWIN's parameter is a confidence bound γ.
Early Drift Detection Method (EDDM) [18]: a classifier with a drift detection system. EDDM monitors the distance between consecutive classification errors d and defines two distance levels (warning and drift) with respective thresholds α, β ($\beta < \alpha$). If $d < \alpha$, the examples are stored in anticipation for change. If $d < \beta$, the classifier is reset and trained on the examples stored since the warning level.
Conceptual Clustering and Prediction (CCP) [5] : an ensemble method that uses clustering and is able to handle recurring concepts. Each batch of m instances is mapped into a conceptual vector (descriptor). The vector is either assigned to an existing cluster according to a distance threshold θ or a new cluster is created. In the latter case, a classifier trained on the batch is assigned to the cluster. In the former case, the classifier of the corresponding cluster is updated with the batch instances. The classifier of the new or existing cluster is then used to classify the next m instances. The number of clusters cannot be larger than c_{max} otherwise the new item is incorporated into the nearest cluster.
Dynamic Adaptation to Concept Changes (DACC) the mere adaptive side of the ADACC approach, as presented in Section 3.1.

4.2 Experimental Setup

Incremental Naive Bayes classifiers were used as base learners, because they naturally learn incrementally and are often used in studies of on-line learning.

LBAG and EDDM algorithms are available in the MOA (*Massive Online Analysis*) API[6]. We implemented all remaining algorithms on top of MOA, except for CCP and DWM whose results reported below are taken from the work of Katakis et al. in [5]. The parameters of DWM were set to $\rho = 0.5$, $\theta = 0.01$, $p = 1$ and those of CCP were set to $b = 50$, $c_{max} = 10$ and $\theta = 4$ for ELIST and $\theta = 2.5$ for SPAM. Both DWM and CCP were not evaluated on STAGGER in [5] and thus no results are reported on this dataset.

We evaluated MW with three different window sizes: 50, 100 and 200 (retaining the best one: 100). WE was tested with the weighting formula $w(n) = w(n-1)+n^2$ where $w(n)$ is the weight of the n-th example. The threshold values of EDDM are automatically set by MOA.

In LBAG, we tuned the parameters λ and γ experimentally converging on $\lambda = 20, \gamma = 0.002$ for ELIST and STAGGER and on $\lambda = 20, \gamma = 0.01$ for SPAM. Error correcting codes can be used for LBAG to add more diversity in the ensemble, but this does not improve the accuracy and is thus not used.

The parameters of ADACC, our *anticipative meta-learning* approach, were set to $\theta_I = 0.8$, $\theta_d = 0.7$ and $\tau_s = 100$ in all experiments. To reduce the computation cost, the anticipation mechanism is called every $p = 100$ time steps (lines 8 to 17 of Algorithm 1). Finally, the parameters of the *adaptive learning* mechanism, DACC, were optimized experimentally and $\tau_{eval} = \tau_{mat} = 20$ were used for all datasets. The ensemble size was fixed to 20 for DACC and LBAG.

4.3 Results

Table 1 reports the results averaged over 10 runs showing the accuracy, precision, recall and the run time in CPU seconds. All experiments were executed on an Intel Core i5 CPU at 2.4 GHz with 4.0 GB of RAM. The execution time of DWM and CCP are not given since they were tested in [5] on a different machine.

In all cases, ADACC yields the best accuracy. Figure 1 (top) shows a moving average of the accuracy of DACC, ADACC, LBAG and EDDM over sliding windows of 1,000 instances. DACC adapts faster to concept drifts than EDDM and LBAG, probably because of the frequent removal and addition of classifiers (every 20 time steps) which makes it ready to any upcoming change. However, despite the very good performance of DACC, ADACC still tops it by recognizing recurring concepts starting at time step 30,000. This comes at the expense of the execution time which is multiplied by a factor of 1.5 to 3. The amount of computation can be reduced by increasing the value of p, the period separating two calls of the meta-learning mechanism. In STAGGER, the run time of ADACC is reduced to 2.8 CPU seconds when $p = 1,000$ time steps (instead of 100). Note that the classification performance is not hurt as long as p is small enough to take snapshots of all encountered concepts (i.e. $p < 10,000$ for STAGGER).

[6] http://sourceforge.net/projects/moa-datastream/

Table 1. The accuracy, precision, and recall (in %) along with the execution time (in CPU seconds) of the different approaches on the ELIST, SPAM and STAGGER datasets using Naive Bayes classifiers as base learners.

	ELIST				**SPAM**				**STAGGER**			
Algorithm	Acc.	Precis.	Recall	Time	Acc.	Precis.	Recall	Time	Acc.	Precis.	Recall	Time
SIC	54.2	50.7	69.3	0.53	90.7	94.2	93.2	1.27	64.5	62.4	89.5	0.35
MW	74.7	70.6	78.4	0.48	90.7	90.6	97.5	1.28	98.8	98.4	99.3	0.39
WE	66.9	64.9	63.9	0.53	92.8	95.2	95.0	1.28	78.5	77.6	85.9	0.4
DWM	43.8	47	42.5	-	91.8	84.8	83.1	-	-	-	-	-
CCP	77.5	79.7	77.6	-	92.3	85.7	83.9	-	-	-	-	-
EDDM	75.6	72.9	75.9	1.15	90.8	92.0	95.9	1.68	99.7	99.73	99.8	0.6
LBAG	58.5	54.4	68.3	15.0	91.8	95.6	93.3	14.60	89.9	85.7	98.1	1.51
DACC	76.2	73.8	75.9	9.52	94.7	95.1	97.8	11.9	99.9	99.9	99.9	1.06
ADACC	**77.5**	**75.2**	**77.2**	13.6	**94.9**	**95.6**	**97.6**	18.92	**99.9**	**99.9**	**99.9**	3.22

Figure 1 (bottom) shows the stability index on the STAGGER dataset and highlights when snapshots are stored. All data points above $\theta_I = 0.8$ are candidate snapshots (a total of 575) but only 5 are kept as relevant in the list $\mathcal{M}_{LT}$. The unstable behavior of the stability index from time step 1 to 10,000 reflects the difficulty of learning the first concept. Three different snapshots of the first concept are stored during this period, capturing different subspaces. Only one additional snapshot is stored for the first concept when it reappears (time steps 30,000 to 40,000) confirming that redundant snapshots are avoided by ADACC. The second concept is learnt more easily and only two distinct snapshots are taken. Learning the third concept corresponds to the highest stability index, suggesting a rather easy learning task. Only one representative snapshot of the third concept is stored at the end of the experiment. Regarding the other streams, a total of 8 snapshots were stored for SPAM and 4 for ELIST.

In our experiments, the threshold values for stability and conceptual equivalence were fixed. We varied their values on the STAGGER dataset to study their effect on the number of candidate snapshots, the number of stored snapshots and the classification performance of ADACC. The results are shown in Table 2. Smaller stability thresholds increase the number of candidate snapshots and thus of the stored ones but very much less significantly. Only 5 additional snapshots are stored when the threshold switches from 0.9 to 0.5. When varying the concept equivalence threshold, the number of candidates doesn't change since it is only related to the stability threshold value. The stored snapshots however evolve. Larger threshold values entice larger numbers of stored snapshots. Remarkably, changing both threshold values may impact the accuracy of the anticipative mechanism (ADACC) but never to the extent of being worse than the mere adaptive scheme (DACC).

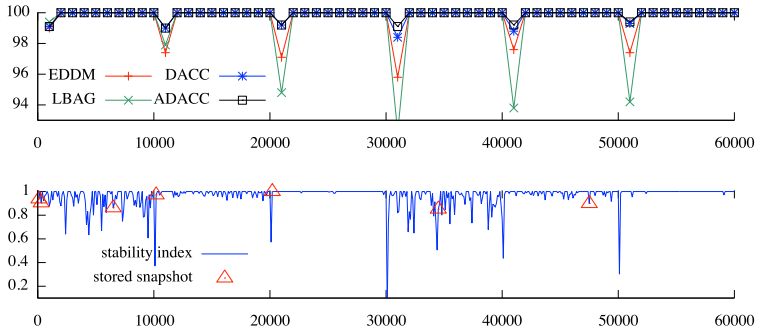

Fig. 1. (Top) The classification accuracy of ADACC, DACC, LBAG and EDDM on STAGGER, averaged over sliding windows of size 1,000. (Bottom) The evolution of the stability index on STAGGER.

Table 2. *Left*: The effect of the conceptual equivalence threshold on ADACC with $\theta_I = 0.8$. *Right*: The effect of the stability index threshold on ADACC with $\theta_d = 0.7$.

θ_d	# candidates	# stored	Accuracy
0.5	575	4	99.918
0.6	575	5	99.915
0.7	575	7	99.916
0.8	575	9	99.918
0.9	575	12	99.918

θ_I	# candidates	# stored	Accuracy
0.5	596	9	99.908
0.6	594	9	99.908
0.7	587	9	99.91
0.8	575	7	99.916
0.9	540	4	99.916

5 Conclusions and Future Work

A meta-learning mechanism that deals with recurrent concepts in the context of online machine learning has been presented. The main contribution of ADACC, which explains the good performances obtained on the benchmark datasets, lies in the use of a stability measure that monitors the pool of base learners *and* the long-term memory of past useful concepts. Snapshots of the relevant states of the world are stored and re-used them when old contexts reappear. This mechanism is completely embedded in the natural functioning of the ensemble method. It relies on few parameters that do not need to be finely tuned.

We conducted experiments on real and artificial benchmark datasets and compared our method with various online learning systems. The empirical results show that combining the meta-learning mechanism with an ensemble method that adapts rapidly to drifting concepts (in comparison to other systems) can bring improvement in the classification performance, outperforming all compared systems.

In the future, we will explore ways to keep constant the size of the long term memory of the memorized snapshots. One such promising avenue is to store prototypes of snapshots instead of the original ones, using a hierarchical

clustering technique. We also plan to reduce the execution time of ADACC by computing the stability index using an agreement measure whose computational cost is less than quadratic in the number of base learners.

References

1. Gama, J.: Knowledge discovery from data streams. In: Citeseer (2010)
2. Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. In: Machine learning, 23(1), pp. 69–101 (1996)
3. Klinkenberg, R.: Learning drifting concepts: Example selection vs. example weighting. In: Intelligent Data Analysis, 8(3), pp. 281–300 (2004)
4. Stanley, K. O.: Learning concept drift with a committee of decision trees. In: Informe técnico: UT-AI-TR-03-302, Department of Computer Sciences, University of Texas at Austin, USA (2003)
5. Katakis, I., Tsoumakas, G., Vlahavas, I.: Tracking recurring contexts using ensemble classifiers: an application to email filtering. In: Knowledge and Information Systems, 22(3), pp. 371–391 (2010)
6. Yang, Y., Wu, X., Zhu., X.: Mining in anticipation for concept change: Proactive-reactive prediction in data streams. In: Data mining and knowledge discovery, 13(3), pp. 261–289 (2006)
7. Alippi, C., Boracchi, G., Roveri, M.: Just-In-Time Classifiers for Recurrent Concepts. In: IEEE Transactions on Neural Networks and Learning Systems, to be published (2013)
8. Gama, J., Kosina, P.: Tracking recurring concepts with meta-learners. In: Progress in Artificial Intelligence, pp. 423–434. Springer Berlin Heidelberg (2009)
9. Gomes, J. B., Sousa, P. A., Menasalvas, E.: Tracking recurrent concepts using context. In: Intelligent Data Analysis, 16(5), pp. 803–825 (2012).
10. Kolter, J. Z., Maloof, M.A.: Dynamic weighted majority: A new ensemble method for tracking concept drift. In: Data Mining, 2003. ICDM 2003. Third IEEE International Conference, pp. 123-130 (2013)
11. Tsymbal, A., Pechenizkiy, M., Cunningham, P., Puuronen, S.: Dynamic integration of classifiers for handling concept drift. In: Information Fusion, 9(1), (2008)
12. Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., Gavaldà, R.: New ensemble methods for evolving data streams. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM (2009)
13. Scholz, M., Klinkenberg, R.: An ensemble classifier for drifting concepts. In: Proceedings of the Second International Workshop on Knowledge Discovery in Data Streams. Porto, Portugal (2005)
14. Carletta, J.: Assessing agreement on classification tasks: the kappa statistic. In: Computational linguistics, 22(2), pp. 249–254 (1996)
15. Schlimmer, J. C., Granger Jr, R. H.: Incremental learning from noisy data. In: Machine learning, 1(3), pp. 317–354 (1986)
16. Bifet, A., Holmes, G., Pfahringer, B.: Leveraging bagging for evolving data streams. In: Machine Learning and Knowledge Discovery in Databases, pp. 135–150. Springer Berlin Heidelberg (2010)
17. Bifet, A.: Adaptive learning and mining for data streams and frequent patterns. In: ACM SIGKDD Explorations Newsletter, 11(1), pp. 55–56 (2009)
18. Baena-Garca, M., del Campo-vila, J., Fidalgo, R., Bifet, A., Gavald, R., Morales-Bueno, R.: Early drift detection method. In: Fourth International Workshop on Knowledge Discovery from Data Streams (2006)