Anticipative and Dynamic Adaptation to Concept Changes

Ghazal Jaber[1,2,3], Antoine Cornuéjols[1,2], 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

June 1, 2013

Abstract

Learning from data streams is emerging as an important application area. When the environment changes, as is increasingly the case when considering unending streams and long-life learning, 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 go one step further by introducing a second-order learning mechanism that is able to detect relevant states of the environment, to recognize recurring contexts and to *anticipate* likely concepts changes. Results of an empirical comparison with adaptive methods show that, for a very slight price in memory and computation load, the proposed algorithm always improves on, or at worst equals, the prediction performance of a mere adaptive approach.

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 $\mathbf{x}_t$. For instance, a set of customers can be submitted to adds or offers arriving on sequence, to which they answer with buying actions or not. The task for a forecaster is to predict the answer to each new incoming incentive $\mathbf{x}_t$, possibly in order to regulate the stocks, even before the outcome y_t is known. Because of varying economic conditions, or because of changes in the season or in the weather, the customers may modify their buying behavior. Sometimes it can even occur abruptly like when a big amendment in the economic policy is announced.

Learning from streams [Gam10] is usually treated using online machine learning techniques which differ from classical batch learning methods in three major points. First, streaming data are not stored or reprocessed, due to memory constraints. Secondly, the prediction model should give answers in an any time fashion, while updating itself with each received information from the stream. Finally, online learning does not presuppose that the training data be independent and identically distributed. It is ready to adapt to changing conditions. This is why, even though most works in recent years have dealt with the computational issues raised by the demand for a small constant learning time and near constant memory resources, a stream of research has also focused on evolving concepts in case of non-stationary environments, specially on *how to detect concept changes* and *how to best adapt* to them.

In this context, passive adaptation to concept changes may not be the best learning strategy. Indeed, a learner may profit from the information possibly conveyed by the very sequence of data. For instance, one can gain precious time and avoid costly incorrect predictions by being able to recognize a recurring situation or to anticipate the likely evolution to come along. This kind of second order learning is the object of the approach presented in this paper. The ADACC (*Anticipative Dynamic Adaptation to Concept Change*) method, that we suggest, deals both with the challenge of optimizing the stability-plasticity dilemma (keeping as much data as possible in order to get the best possible hypothesis while at the same time recognizing when data points become obsolete and potentially misleading) and with the anticipation and recognition of incoming concepts. This is accomplished through an ensemble method that controls a pool of incremental learners. The management of the pool of learners enables to naturally adapt to the dynamics of the concept changes with few parameters to set, while a learning mechanism managing the changes in the pool provides means for the anticipation of, and the quick adaptation to, the underlying modification of the context.

The rest of the paper is organized as follows. In Section 2, we discuss existing works on online learning in the context of concept changes. Section 3 presents our contribution, which is followed by the report of empirical results in Section 4. The last section presents conclusions and possible avenues for future work.

2 Relevant Works

Aside from the question of the severe constraints posed on the computational load of online learning (see [DH00] for a pioneering study and [Kir07] for more references), the main issue in online learning is connected with the problem of learning in the presence of a changing environment. These changes can come in two guises according to which of the input distribution $\mathbf{p}(\mathbf{x})$ or the conditional distribution $\mathbf{p}(y|\mathbf{x})$, is affected. If only $\mathbf{p}(\mathbf{x})$ changes, it is said that a *virtual drift* or *covariate shift* occurs. If the changes in the environment concern the concept itself $\mathbf{p}(y|\mathbf{x})$, a *concept drift* occurs, which can be gradual or abrupt. In an abrupt drift, the target concept gives suddenly way to a new one (see Figure 1). In a gradual drift, however, the new target concept takes over the old one over a period τ_{drift}. The most general type of changing environments involves changes both in $\mathbf{p}(\mathbf{x})$ and in $\mathbf{p}(y|\mathbf{x})$. When the environment varies, it is important to be able to detect the changes and to modify the prediction model accordingly in order to maintain as best as possible the prediction performance.

Usually, the *detection of change* is done by monitoring either variations in the distribution of the incoming data, or variations in the prediction's performance of the system. In both cases, it is necessary to set an alarm threshold and to decide upon an adequate duration of observation in order to trigger justified alarms only. This requires some knowledge of the dynamic of the environment.

Likewise, *adaptation to changes* meets a dilemma regarding the adequate length of the subsequence of the data stream that must be trusted for providing relevant information about the current target concept. The earliest techniques used sliding windows over the data stream [WK96]. The length of the window can be set beforehand, but more sophisticated approaches rely on some dynamic control of this length depending on the stability of the prediction performance. A second strategy uses a weighting scheme over past instances which is deemed to reflect the relevance of the data to the current context. Both these techniques, windowing and weighting, imply some a priori choices about threshold, decay factors and so on [Kli04]. A third strategy tries to avoid these choices by relying instead on an ensemble-based learning technique, akin to bagging or boosting. The idea is to let a committee of base learners incrementally learn over the stream of data, and to replace the worst ones at certain times [Sta03b]. Thus, the hope is that the committee automatically eliminates the no longer relevant base learners while keeping and improving the most promising ones. In addition, for each incoming instance, the prediction to be made can result from a combined vote from the base learners, hopefully leading to more accurate answers, as in classical ensemble learning methods.

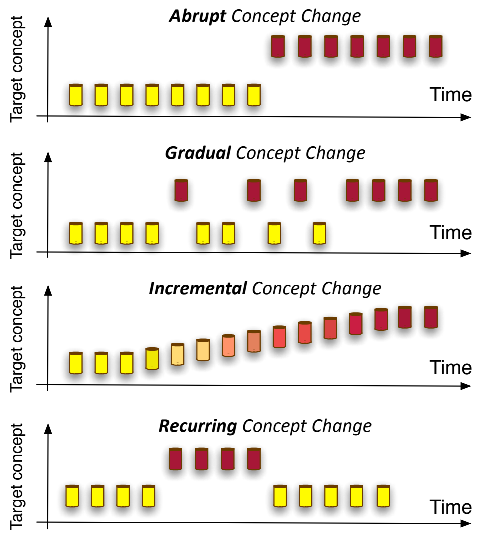
The committee based strategy still entails some choices regarding the base learners, the size of the ensemble or committee, the deletion mechanism, the introduction and initialization of new learners in the ensemble, and the decision voting process. Recent years have seen several proposals among these lines showing promising results on a variety of online learning tasks [BGdCÁF$^+$06, TPCP08, BHP$^+$09]. A limit of these algorithms, however, is that they passively wait for the changes to occur and then try to follow them as best as possible rather than proactively predict what is likely to happen.

Very few works have confronted the anticipation of concept changes. Among them, the PreDet [BSK08] algorithm uses decision trees as classifiers and anticipates future trees by predicting for each decision node the evaluation measure of each attribute, this value being used to determine which attribute will split the node. In the case of a leaf node, it predicts its class label distribution. Future changes are predicted using a linear regression model trained on a fixed size history. Another prediction system, RePro [YWZ06], stores the observed concepts in a Markov chain. RePro assumes that the same concepts repeat over time. When a change is detected, the Markov chain is used to predict the most likely concept to come.

The method we present below aims at benefiting from the knowledge of the past to predict future concept changes. Contrary to other systems, it can be used with any base learner that produced parameterized hypotheses (e.g. neural networks) and is not limited to decision trees. Furthermore, it does not necessitate that evolution changes repeat over time to make anticipation, unlike Markov chains.

3 Concept Changes: Adaptive and Anticipative

In online learning, the learning protocol is as follows. At each time step t, an input $\mathbf{x}_t$ is received, the learning algorithm is asked to make a prediction $\hat{y}_t$ over its label, and then only is revealed the true label y_t. The algorithm can then adapt its current view of the world, noted here H_t, and be ready for the next input $\mathbf{x}_{t+1}$ to come.



Abrupt Concept Change
Gradual Concept Change
Incremental Concept Change
Recurring Concept Change
Target concept
Time

Figure 1: Four different types of concept changes.

The overall goal is to minimize the loss over time : $\frac{1}{T}\sum_{t=1}^{T} \ell(H_t(\mathbf{x}_t, y_t))$ with $T \rightarrow \infty$. Usually, the loss function is simply the $0-1$ loss function which counts the number of mistakes.

In the most general case, the target concept, which decides the label for the next input $\mathbf{x}_t$, can change arbitrarily over time, and even be manipulated by an adversary. Then, the only known theory is the online learning theory which characterizes the *regret* that can be achieved by a learning algorithm with respect to the best available "expert" in a given pool of experts [CBL06]. If, however, more benign assumptions can be made about the dynamics of the environment, such as it evolves gradually or incrementally or it evolves with sudden changes but interspersed with stationary states (see Figure 1), then it becomes legitimate to try to assess the future performance of a classifier on the base of (a partial view of) its past history.

In this paper, we work on anticipating future concepts that evolve in a predictable way, and on the recognition of recurring concepts, whether the change between consecutive concepts happens gradually or suddenly.

In the following, we first introduce the adaptive architecture of the learning system we take as our basis. Then, we present the anticipating mechanism that builds upon it.

3.1 Adapting to Concept Changes

Being able to recognize changes in the environment and to anticipate them requires some kind of second-order or meta learning. The learning system needs to be able to analyze and reflect upon its past experiences and to decide what is the best course of action or decision given the past. For this meta-learning to take place, the adaptive strategies based on ensemble of base classifiers are well suited because they naturally manage a set of potential models of the changing environment[1], . At any time, the current ensemble of base classifiers represents a kind of memory of the past, and offers the opportunity for second order learning. Furthermore, these ensemble methods generally adapt gracefully to a variety of changing conditions in the environment and do not require fine tuning of their few parameters.

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 deleting strategy and an insertion one. The main principles of these ensemble methods are the following:

- 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 based 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 based learners $h_t(\mathbf{x}_t)$.

Individual variations around this general framework lead to specific algorithms [Sta03a, KM07]. For instance, in our studies, and after extensive testings, we converged on the following settings. The *evaluation* procedure simply counts the number of erroneous predictions on the last τ_{eval} time steps. The *deletion strategy* randomly select one base learner from the worst half of the pool evaluated as above. The *global prediction* merely uses the prediction from the current best base learner (randomly chosen in case of ties). The deletion and prediction strategies could easily be modified. They were decided upon because their performances topped the ones of other strategies like deleting all base learners with a poor performance or predicting using votes of the base learners.

This simple method offers a good trade-off on the plasticity-stability dilemma and leads to fast adaptation when the underlying concept changes. It is not the center of this paper.

[1] In this paper we employ indifferently the terms "base classifiers", "base learners", "experts" and "hypotheses".

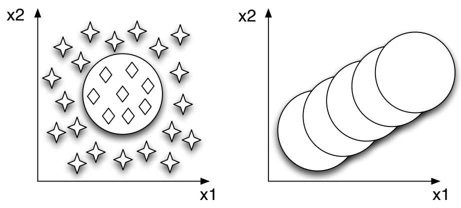


Figure 2: *Left*, a concept represented by a circle. The examples are classified into one of two classes: inside, outside the circle. *Right*, the circle moving with time, creating concept drifts.

3.2 Anticipating Concept Changes

In an evolving environment, two aspects should be considered when anticipating future concept changes: recurrence and predictability. *Recurrence* means that the same concepts (or close approximations) might reappear with time, either in a cyclic manner (e.g. seasonal variations) or in an irregular manner (e.g. inflation rate, market mood). *Predictability* means that the concept evolves in a predictable manner, but without necessarily repeating over time. Here, "predictable" refers to an underlying prediction system, that is a learning system that, taking as input information about the past history of the concept evolution, is able to predict its (near) future. This, of course, implies that the past history has to be captured and represented. We explain below how we do it with a system that determines relevant *snapshots* of the evolution.

As an illustration, consider a concept corresponding to a circle in a 2-dimensional input space. The instances are labeled into two classes depending on whether they lie inside or outside the circle (see Figure 2). Suppose now that the radius is fixed while the circle's center moves with a constant speed, creating a concept drift. It might be the case that a learning system be able to learn from such a sequence of concept changes and predicts the likely future concepts. For instance, we did it with Ellman's neural networks [Wil93]. Note however that in such a case, the same circle doesn't repeat over time and there is no recurrence of past concept thus precluding the use of Markov chains or of any method using frequency measures.

In addition to being illustrative of a simple scenario, this example shows also why an approach commonly thought about in sequence learning, that of Markov chains, is here helpless. Learning a Markov chain requires indeed that the concepts be encountered repeatedly in order to reliably estimate transition matrices. When concepts do not recur, this is impossible.

In the following, we first describe a mechanism for the *recognition of the models of the world* before presenting the *second-order learning mechanism* that works on the models of the world identified as significant in order to predict future models. Algorithm 1 offers a formal view of the steps and methods described below.

Recognition of significant models of the world

In this step, we aim at recognizing the different concepts encountered during the data streaming. Our goal is to take a snapshot (copy) of the underlying stable concept via the learned models or hypotheses in the adaptive ensemble. A main challenge here is to decide when to take a snapshot, or equivalently to decide when the ensemble reflects the underlying concept. We should ensure that enough training examples have been learnt from the current concept before the snapshot is taken.

One approach has been proposed in the ADWIN system [BG07]. It supposes a priori that m training instances are necessary for the system to stabilize after an abrupt concept change. Consequently, a snapshot is taken m time steps after a change is detected. We did not retain this method because of its many limitations. It has difficulties to detect gradual concept changes and the snapshots then taken have often low relevance. Furthermore, the choice of the value for the parameter m is difficult if one does not know in advance the properties of the incoming concept changes.

In this work, we aimed to find an implicit way to detect periods of stability, and this via the evolving hypothesis in the adaptive ensemble. When the environment has been in a stable state for a sufficient time, the best hypotheses in the pool of base learners should converge toward the same, and near optimal, predictive performance. Therefore, their diversity, measured as below, should be low, while their error rate should decrease toward the best achievable performance or close to it. This suggests to take into account both diversity of the best hypotheses in the pool and their error rate as an index of the stability of the environment. In our study, we use the kappa statistics $\mathcal{K}$ [Car96] in order to compute *diversity*. This statistics measure evaluates the degree of agreement between the classification of a set of items by two classifiers[2]. 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: $I_{stability} = agreement - performance$ where *agreement* and *performance* are computed over the best half of the current hypotheses in the pool, and are defined as:

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

[2] Other agreement statistics should do as well.

and:

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

where N is the size of the pool.

Each point in the stability index curve, over some predefined threshold θ_I, is suggestive of a stable environment and is therefore a privileged moment to take a *snapshot* of the current best hypothesis, the one that seems to best represent the state of the world.

One must however be careful not to store consecutive hypotheses that correspond to the same underlying state of the environment. Here again, the agreement statistics, in our case the kappa statistics, can be used to measure the agreement between a candidate snapshot h_t^* and the preceding one $h_{t_k}^*$. For this purpose, a set of n unlabeled data $U = \{\mathbf{x}_i\}_{i=1}^n$ is generated at random using a Gaussian distribution around the last τ_s examples, and the predictions of h_t^* are then compared with the predictions of the last stored snapshot $h_{t_k}^*$. If the estimated agreement is less than some predefined threshold θ_d, the current candidate snapshot h_t^* is considered different enough from $h_{t_k}^*$ and is therefore added to the list $\mathcal{M}_{LT}$ of snapshots representing past stationary states of the environment.

The long term memory and second order learning

The list of past snapshots $\mathcal{M}_{LT} = \{C_1, C_2, \ldots, C_k\}$, ordered according to the snapshots' time appearance, is the basis of the second order learning mechanism. It serves two purposes. *First*, it provides a sequence of successive models of the environment that can be used by a learning algorithm in order to predict the most likely future state in the series. *Second*, it stores a memory of past successful models of the world, models that should be repeatedly tested against current data in case a recurring concept can be recognized.

As previously mentionned, the anticipation mechanism deals with both: predictability and recurrence. In our experiments, we used Elman's recurrent neural networks as the *predictability* mechanism because of their generality and their good performance reported for learning tasks similar to our's [Wil93]. A network is trained on the pairs of consecutive concepts in $\mathcal{M}_{LT}$: $\{(C_1, C_2), (C_2, C_3), \ldots, (C_{k-1}, C_k)\}$ in order to predict the next likely snapshot $\tilde{C}_{k+1}$. To simplify the discussion, a snapshot C is represented as a vector of parameters of dimension n.

$$C = [c_1, c_2, ..., c_n] \quad (3)$$

For instance, if the concept is learnt by a neural network, the snapshot can be represented as a vector of the network weight values. We call the pair consisting of two consecutive snapshots $\delta_i = (C_i, C_{i+1})$ *a change sample*. After recognizing k stable concepts, the $k-1$ change samples $(\delta_1, ..., \delta_{k-1})$ form the set of training examples learnt by the Elman network in order to predict the next snapshot $\tilde{C}_{k+1}$. The predicted snapshot is temporally added to the list of snapshots $\mathcal{M}_{LT}$. It is replaced by the next snapshot C_{k+1} when this one is acquired.

Therefore, the list $\mathcal{M}_{LT}$ contains snapshots representing past stationary states of the environment, which can be useful in case of recurring contexts, in addition to the next predicted concept according to the Elman's Network, which can be useful in case of a predictable sequence of concept changes. Each snapshot in the list is then evaluated according to the evaluation strategy used by the adaptive ensemble to evaluate its base learners in the pool. A snapshot is used for prediction if its evaluation record is the best among all

Algorithm 1: Selection of snapshots by ADACC.

```
begin
    $E_0 \leftarrow \emptyset$;            /* Ensemble of experts */
    $\mathcal{M}_{LT} \leftarrow \emptyset$;          /* List of snapshots */
    $k \leftarrow 1$;
    for $t = 1$ to $\infty$ 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                        */
        /* ------------------                  */
        $H = \{h_t^i\}_{i=1}^{N/2}$ is the best half of experts in $E_t$;
        $agr = \frac{1}{\frac{N}{2} * (\frac{N}{2} - 1)} \sum_{i=1}^{N/2} \sum_{j=1, i \neq j}^{N/2} \mathcal{K}_{h_t^i, h_t^j}$;
        $perf = \frac{1}{\tau_s * \frac{N}{2}} \sum_{j=0}^{\tau_s - 1} \sum_{i=1}^{N/2} err\big(h_t^i(\mathbf{x}_{t-j}), y_{t-j}\big)$;
        $I_{stability} = agr - perf$;
        /* Detect Stable Concept               */
        if $I_{stability} \geq \theta_I$ then
            $h_t^* = snapshot(E_t)$;
            /* Detect New Concept              */
            if $isEmpty(\mathcal{M}_{LT})$ then
                $C_k = h_t^*$;
                $\mathcal{M}_{LT} = add(\mathcal{M}_{LT}, C_k)$;
            else if $\mathcal{K}_{C_k, h_t^*} \leq \theta_d$ then
                $k = k + 1$;
                $C_k = h_t^*$;
                $\mathcal{M}_{LT} = replace(\mathcal{M}_{LT}, \tilde{C}_k, C_k)$;
                $\tilde{C}_{k+1} = predictNextConcept(\mathcal{M}_{LT})$;
                $\mathcal{M}_{LT} = add(\mathcal{M}_{LT}, \tilde{C}_{k+1})$;
end
```

candidate hypotheses from both the pool of base learners and $\mathcal{M}_{LT}$.

While the pool of candidate base learners is managed according to the policy outlined in Section 3.1 and is therefore of a finite constant size, the list $\mathcal{M}_{LT}$ of snapshots may a priori increase forever if new hypotheses are continually retained as worthy of storage. Fortunately, it is possible to keep this size under control by recognizing that the two roles of $\mathcal{M}_{LT}$: anticipation and memory for recurring concepts, ask for two different memory management systems. Indeed, since Elman's networks are incremental learners, they do not need to keep the past history of snapshots at all. Regarding the memory for recurring concepts, it can be kept constant using various heuristics. One is to delete the oldest or the least reccuring snapshots from the memory. Another one, more sophisticated, would be to store prototype snapshots instead of the original ones, using a hierarchical clustering technique. Because of the limited length of the sequences in our experiments, we did not rely on such memory management schemes.

4 Empirical Results

The aim of the experiments was threefold. First, to test the performance of *the snapshot mechanism* and specially its ability to detect both abrupt and gradual concept changes and store the relevant target concepts with no, or limited, redundancy. Second, to examine the gain, if any, brought by *the anticipation scheme* compared to the mere adaptation mechanism. Obviously, this depends on the ability to anticipate the next state of the environment, and therefore on the underlying structure (if any) of the sequence of changes [BBDK00]. Thirdly, to test the mechanism for *the recognition of recurring concepts* and the gain it can bring.

In the worst case, where it is not possible to anticipate the next concept and when no recurring concept arises, the prediction performance of the system should fall back to the performance of its adaptation mechanism. Indeed in these cases, no snapshot in $\mathcal{M}_{LT}$ does outperform the best base learners and the resulting behavior is the one of the adaptive system alone.

4.1 Experiments and Datasets

We conducted experiments on artificial and real data sets. The artificial sets were used to simulate *recurrent* and *predictable concept changes* while controlling the timing of the change, its speed (abrupt, or more or less gradual) and its severity (amount of change) [MWY10]. The real data set comes from video sequences taken with a mobile robot wandering in and out of rooms in a laboratory, creating *recurring contexts*.

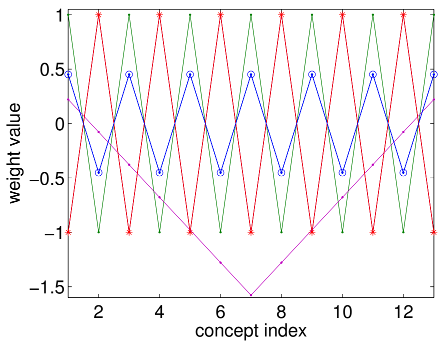


Figure 3: A typical evolution in the weight values of the hyperplane used in the artificial datasets. Here the target concept involves 4 different weights that vary from a concept to the next one.

In the *artificial data sets*, the input space $\mathcal{X}$ is d-dimensional and the target concept is a linear decision boundary (a hyperplane) described by the relation $y(\mathbf{x}) = sign(\sum_{i=1}^{d} w_i x_i + w_0)$. The experiments were carried out on streams with 7,150 time steps and hence data points.

In each stream, 12 concept changes were simulated by changing the weights $\{w_i\}_{i=0}^{d}$ of the target hyperplane. The first 7 concepts evolved through the successive addition or substraction of constant values (differing according to the experiments) to the weights. The idea was to look at the capacity of the anticipative mechanism to identify this regularity and therefore to predict likely future concepts. The last 6 concepts were recurring concepts, that is concepts already encountered in the past data stream (see Figure 3).

Three artificial data streams were generated, each involved a different level of severity in its concept changes: *low* or *medium* or *high* respectively involving changes in 1, 5 and 9 parameters out of the 11 that define the target concept, with respectively approximately 3%, 60% and 84% of the input space changing class between successive concepts. In each stream, the changes happened either suddenly or gradually, in a linear manner, between successive target concepts.

The transition between consecutive concepts took from 0 to 200 time steps and changes would start happening every 400 to 700 time steps. We did not observe any effect of the dimension up to more than one hundred and therefore only report results for the 10-dimensional case. The base learners in the adaptation ensemble were perceptrons with 10 input units and one output unit, involving 11 weights (10 + 1 for the bias). The Elman's networks took as input the 11 weights of a snapshot and gave as output the 11 weights of the next predicted snapshot.

The *real data set* was issued from the COLD database of the Saarbrücken laboratory [PC09], a benchmark for vision-based localization systems. It contains sequences of images recorded by a mobile robot under different variations of illumination and weather: sunny, cloudy and night. We worked on the dataset captured in sunny conditions. The images were labeled into one of four classes: *corridor*, *one-person office*, *printer area* and *classroom*, and the total length of the data sequence was 753. The robot visited the rooms in the following order: *corridor*, *bathroom*, *corridor*, *one-person office*, *corridor*, *printer* and *corridor*. It stayed in the same room between 45 and 284 time steps. Images were first pre-processed into a 128-dimensional space using the Self-Organizing Map described in [GDTF11]. In the experiments, we used decision trees (as implemented in Matlab) as base learners in the adaptation ensemble.

One important goal of the experiments was to compare the performances achieved with the combined anticipative and adaptive mechanism, with the ones of a purely adaptive mechanism.

The *anticipative meta-learning* system itself involves three parameters that all pertain to the detection of relevant snapshots. They are the *stability threshold* θ_i, the *decision threshold* θ_d and the *duration for the evaluation* of candidate snapshots τ_s. They were set respectively to $\theta_i = 0.9$ and $\theta_d = 0.8$ while $\tau_s = 100$ was chosen for artificial data streams and $\tau_s = 25$ for robotics in order to cope with a faster dynamics. In the base version with no sophisticated management of the snapshot list $\mathcal{M}_{LT}$, there are no additional parameters.

The remaining parameters concern the adaptive mechanism, and we tried to optimize these in order to not unfairly attribute gains to the anticipative process. The parameters for the ensemble method for *adaptive online learning* include the size of the pool N, the maturity age τ_{mat} and the evaluation size τ_{eval}. After extensive experiments, they were set as follows.

The pool comprised $N = 20$ base learners for the artificial data sets ($N = 15$ for the robotics data). In order to be compared, base learners were evaluated on the most recent $\tau_{eval} = 20$ data points (time steps) ($\tau_{eval} = 15$ for the robot). The duration for maturity τ_{mat} was equally set to 20 time steps ($\tau_{mat} = 10$ for the robot).

4.2 Evaluation Measures and Methodology

In the experiments, we evaluated *the snapshots* stored by the system with respect to the known target concepts. Ideally, there would be one snapshot exactly for each encountered state during the data stream. For instance, in the top of Figure 4, candidate snapshots are indicated with small squares and the retained ones appear as red (or black) squares.

We also evaluated the *gain in prediction errors* resulting from the use of the anticipation mechanism over the use of the adaptation scheme alone. Likewise, we measured the gain (if any) due to the recognition of a recurring concept. The gain is simply the number of errors of prediction that were avoided with respect to the use of the adaptive strategy only (see Table 1 below).

Finally, the graphs (see Figure 4, the bottom three graphs) report at each time step the current *online predictive performance*, i.e. the mean number of instances correctly classified so far. In order to better visualize the gain after each concept change, the performances of the adaptation mechanism and of the adaptation + anticipation mechanisms were reset to 0.5 (the chance prediction rate). One can then observe, for instance, that the gains due to the anticipation mechanism start only to show after the fourth concept change, which is unsurprising.

4.3 Results

Table 1 sums up the experimental results, averaged over 10 experiments. The table shows the mean predictive error of the adaptive learning strategy, and the gain of using the anticipation mechanism, in both predictability and recurrence. The gain is measured as the difference between the number of prediction errors made by the adaptive ensemble and the number of prediction errors made by the anticipation mechanism. In the artificial data streams, we highlight the gain brought by the *predictability* of the first 7 concept changes, and the gain brought by the last 6 *recurring* concepts.

Figure 4 illustrates the mechanism for the selection of snapshots on one data stream and it shows the evolutions of the prediction performance over 10 repeated experiments according to the severity of the concept changes.

Detection of concept changes and selection of snapshots

As can be seen in Figure 4, the value of the stability index closely mirrors the concept changes. As soon as the appearance of a new concept is detected by the system and the corresponding candidate snapshot sufficiently differs from the previously stored ones, it is stored away in $\mathcal{M}_{LT}$. That policy enables the fast detection of novel target concept. In our experiments, 100 % of all new concepts (260 altogether) that were introduced in the data streams with *high* and *medium severity* levels triggered the storage of a new snapshot. There was no redundancy (no more than one snapshot per concept) in the artificial data streams. However, in the data stream with *low severity* level, since the consecutive concepts are quite similar, fewer snapshots than concepts were retained. Some snapshot's redundancy ap-

Stream		**Adaptation**			**Anticipation**	**Total gain**		**Due to prediction**		**Due to recurrence**	
name	size	base learner	mean err.	std-dev	predictor	mean	std-dev	mean	std-dev	mean	std-dev
10-D Low	7,150	perceptron	**107.2**	7.7	Elman net	**1.9**	1.7	0.0	0.0	1.9	1.7
10-D Med.	7,150	perceptron	**784.4**	32.2	Elman net	**317.7**	25.4	70.8	9.7	246.9	21.2
10-D High	7,150	perceptron	**937.4**	54.4	Elman net	**393.9**	46.5	120	18.1	273.9	34.3
Robot	753	decis. tree	**43.0**	2.6	-	**9.0**	1.9	-	-	9.0	1.9

Table 1: Summary of the experiments and the measured gains in prediction errors wrt. an adaptive only strategy.

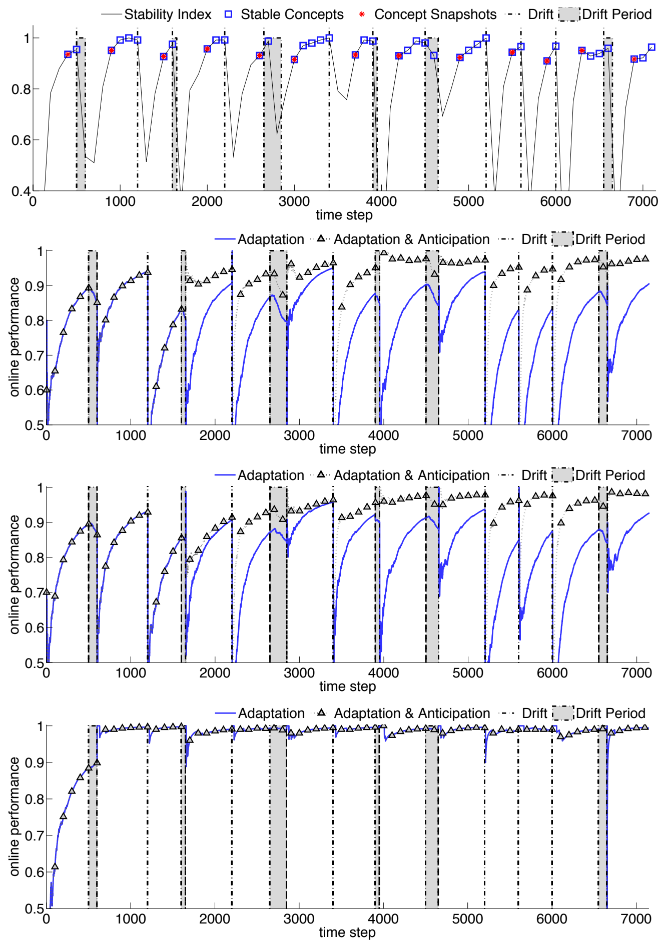


Figure 4: Curves for 10-dimensional artificial data streams. The *top* plot shows the evolution of the stability index in case of the medium severity concept changes. We show the time steps where candidate snapshots are considered (small squares), and when they are retained (red (or black) squares). A candidate snapshot must have a stability index larger than θ_i, and in order to be retained, it must differ from the last retained snapshot, according to a decision threshold θ_d. The three other plots show the online predictive performance using the adaptive learning strategy (continuous, blue, line) and with the second order learning taking place. They are averaged over 10 repeated experiments. The beginning/end of concept changes are indicated as vertical dotted lines. In case of gradual concept changes, the transition period between consecutive concepts is colored in gray. The online predictive performance is reset once a transition is complete. Among the three bottom plots, the *top* one corresponds to high severity concept change, the *middle* one to medium severity, and the *bottom* one to low severity.

peared for the robotic data because of the variation within each concept (e.g. In an *office* the robot's camera points to different parts of the room which may induce several snapshots).

Second order learning

For concept changes of *low severity* (Figure 4, bottom), the adaptive strategy is able to follow the variation of the environment as soon as enough candidate hypotheses are good enough, which happens at the end of the first concept (circa 400 time steps). Therefore, the anticipation strategy does not bring an advantage there. The situation is significantly altered, however, when the concept changes are of *medium* or *high severity*.

In our experiments, even though the concept changes occur at varying dates and with varying speed, the anticipation mechanism is able to predict relevant foreseeable target concepts that, in turn, are quickly recognized as the best for labeling the incoming examples. This brings significant gains in the online performance starting already after the 3rd (resp. 4th) change of concept for the high (resp. medium) severity context, and the gain increases thereafter with each new concept change.

Table 1 shows that the gain in the number or labeling errors attains more than 318/784 $\approx$ 40% for concept changes of medium severity, and approximately 42% in the case of high severity. These gains are impressive in face of a difficult learning task. It is unlikely that they could be obtained without a second order learning mechanism working over the adaptive one.

Table 1 distinguishes furthermore between the gain due to the predictability and the gain due to the fast recognition of a recurring concept. Predictability brings significant gain in the medium and high severity settings for the artificial data sets. In the case of the robotics data, the gain is totally due to the fast recognition of recurring concepts which outperforms the anticipation mechanism.

As expected, there is never a negative gain. As noted earlier, because the ensemble methods is based on a continual competition between base learners from the adaptive mechanism and base learners from the anticipative one, second order learning can never be detrimental to the overall prediction performance as compared to the adaptive only policy.

5 Conclusions and Future Work

The ability to make predictions when data arrives continuously in stream, possibly from a non stationary environment, is becoming increasingly important. Significant research works in recent years have brought new techniques to cope with these learning conditions. In this paper, we presented a general framework to endow adaptive online learning systems based on an ensemble approach with second order learning capacity.

Our method provides means (i) to identify significant stationary states of the world, (ii) to make anticipation about likely future states, and (iii) to recognize recurring concepts if they ever arise. Few parameters are involved in the second-order learning scheme and they do not need to be finely tuned.

The empirical evaluation explored various conditions for evolving data streams. It showed that as soon as the concept changes are significant (medium or high severity), second order learning yields substantial gains in prediction performance over a mere adaptation policy. Furthermore, second order learning can only improve and never deteriorate the prediction performance, at a small cost in memory and computation.

In the future, we plan to carry out experiments with very long streams of data ($\sim 10^5$ time steps) in order to test possible strategies for the management of the memory of snapshots. We will also extend the anticipation mechanism to non parametric representations of targets.

Acknowledgments

Two Grants from the French National Research Agency (ANR): "Holyrisk" (ANR "Blanc-2009" NT09_439612) and "Coclico" (ANR "MN-2012") supported in part this work.

References

[BBDK00] P.L. Bartlett, S. Ben-David, and S.R. Kulkarni. Learning changing concepts by exploiting the structure of change. *Machine Learning*, 41(2):153–174, 2000.

[BG07] A. Bifet and R. Gavalda. Learning from time-changing data with adaptive windowing. In *SIAM International Conference on Data Mining*, pages 443–448, 2007.

[BGdCÁF+06] M. Baena-García, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavaldà, and R. Morales-Bueno. Early drift detection method. *Fourth International Workshop on Knowledge Discovery from Data Streams*, 2006.

[BHP+09] A. Bifet, G. Holmes, B. Pfahringer, R.B. Kirkby, and R. Gavaldà. New ensemble methods for evolving data streams. 2009.

[BSK08] M. Bottcher, M. Spott, and R. Kruse. Predicting future decision trees from evolving data. In *Data Mining ICDM'08. Eighth IEEE International Conference on*, pages 33–42, 2008.

[Car96] J. Carletta. Assessing agreement on classification tasks: the kappa statistic. *Computational linguistics*, 22(2):249–254, 1996.

[CBL06] N. Cesa-Bianchi and G. Lugosi. *Prediction, learning, and games*. Cambridge University Press, 2006.

[DH00] P. Domingos and G. Hulten. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 71–80. ACM, 2000.

[Gam10] J. Gama. *Knowledge discovery from data streams*. Citeseer, 2010.

[GDTF11] H. Guillaume, M. Dubois, P. Tarroux, and E. Frenoux. Temporal bag-of-words: A generative model for visual place recognition using temporal integration. In *Proceedings of the International Conference on Computer Vision Theory and Applications*, 2011.

[Kir07] R.B. Kirkby. *Improving hoeffding trees*. PhD thesis, The University of Waikato, 2007.

[Kli04] R. Klinkenberg. Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis*, 8(3):281–300, 2004.

[KM07] J. Z Kolter and M. A Maloof. Dynamic weighted majority: An ensemble method for drifting concepts. *The Journal of Machine Learning Research*, 8:2755–2790, 2007.

[MWY10] L.L. Minku, A.P. White, and X. Yao. The impact of diversity on online ensemble learning in the presence of concept drift. *Knowledge and Data Engineering, IEEE Transactions on*, 22(5):730–742, 2010.

[PC09] A. Pronobis and B. Caputo. Cold: Cosy localiza- tion database. In *The International Journal of Robotics Research, 28(5)*, 2009.

[Sta03a] K. O Stanley. Learning concept drift with a committee of decision trees. *UT-AI-TR-03-302, Department of Computer Sciences, University of Texas at Austin, USA*, 2003.

[Sta03b] K.O. Stanley. Learning concept drift with a committee of decision trees. *Informe técnico: UT-AI-TR-03-302, Department of Computer Sciences, University of Texas at Austin, USA*, 2003.

[TPCP08] A. Tsymbal, M. Pechenizkiy, P. Cunningham, and S. Puuronen. Dynamic integration of classifiers for handling concept drift. *Information Fusion*, 9(1):56–68, 2008.

[Wil93] W.H. Wilson. A comparison of alternatives for recurrent networks. In *Proceedings of the sixth Australian conference on Neural Networks (ACNN'93)*, pages 189–192, 1993.

[WK96] G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. *Machine learning*, 23(1):69–101, 1996.

[YWZ06] Y. Yang, X. Wu, and X. Zhu. Mining in anticipation for concept change: Proactive-reactive prediction in data streams. *Data mining and knowledge discovery*, 13(3):261–289, 2006.