Solving Nonstationary Classification Problems with Coupled Support Vector Machines

Guillermo L. Grinblat, Lucas C. Uzal, H. Alejandro Ceccatto, and Pablo M. Granitto

***Abstract*— Many learning problems may vary slowly over time: in particular, some critical real-world applications. When facing this problem, it is desirable that the learning method could find the correct input–output function and also detect the change in the concept and adapt to it. We introduce the time-adaptive support vector machine (TA-SVM), which is a new method for generating adaptive classifiers, capable of learning concepts that change with time. The basic idea of TA-SVM is to use a sequence of classifiers, each one appropriate for a small time window but, in contrast to other proposals, learning all the hyperplanes in a global way. We show that the addition of a new term in the cost function of the set of SVMs (that penalizes the diversity between consecutive classifiers) produces a coupling of the sequence that allows TA-SVM to learn as a single adaptive classifier. We evaluate different aspects of the method using appropriate drifting problems. In particular, we analyze the regularizing effect of changing the number of classifiers in the sequence or adapting the strength of the coupling. A comparison with other methods in several problems, including the well-known STAGGER dataset and the real-world electricity pricing domain, shows the good performance of TA-SVM in all tested situations.**

***Index Terms*— Adaptive methods, drifting concepts, support vector machine.**

I. Introduction

In MANY real-world applications, pattern recognition problems may vary slowly over time. For example, weather conditions under which meteorological alerts should be raised are seasonal, or the state of a critical mechanical system that should trigger an alarm could change with the wear of the machine. In most cases, the underlying causes and characteristics of these slow changes are not evident from the data under analysis. Under such circumstances, it is desirable for the pattern recognition method to be able to learn related but distinct input–output functions at different epochs and, in particular, to have the flexibility to do it in a continuous way, profiting from the slow-drift property and thereby harnessing information from the entire historical database.

In the next section, we review some previous works on this topic, which is sometimes called "drifting concepts" [1]–[4]. In this context, some authors distinguish sudden or instantaneous drift from gradual change [5], [6]. As Stanley [5] points out, the two problems present very different challenges. Algorithms appropriate for sudden concept changes [7]–[13] should be fast in detecting the change and react to it in an appropriate way. In gradual drift [4], [14]–[18], on the other hand, there is no need for a rapid reaction and the interesting problem is how to use efficiently the information from the full dataset. Our method focus on this latter kind of problems, and in particular on situations with scarce data, but also works efficiently for problems with a sudden change, as we will show later.

Most previous approaches to handle concept drift rely on the use of "local" classifiers, each one fitted or adapted to a particular temporal window of a given length [2], [7]–[9], [19], [20]. As we discuss in Section II, the methods differ in how they select the length of the window, or in how they weigh the selected samples, or even in how they use the set of classifiers (some methods keep several classifiers in an ensemble, others use only the classifier corresponding to the current window). Here we present a new approach to this problem, the use of a sequence of classifiers that vary following the concept change, but which are all fitted in a global way. To build the sequence of classifiers, we selected one of the most powerful methods nowadays, the support vector machine (SVM) [21], [22], which we adapted accordingly. As in most previous methods, each SVM in the sequence is trained using data points from only one of a set of consecutive nonoverlapping time windows. The novelty of our method is that the sequence is not independent. We solve all the SVMs at the same time, using a coupling term that force time neighbors to be similar to each other. In our method, the interval of validity of each classifier can be as small as needed to follow the change in the concept but with reduced overfitting because the classifiers are trained to minimize a global measure of the error instead of adjusting them locally.

In a previous work [23], we introduced a limited version of this method and showed its potential using an artificial drifting problem. In this paper, we describe an extended version of our algorithm that can use fewer classifiers than points in the dataset,[1] producing more robust and efficient solutions. Based on the ideas of [4], we evaluate the new method in three different settings for drifting concepts: *estimation*, *prediction*, and *extrapolation*.

Manuscript received February 16, 2010; revised July 18, 2010; accepted September 22, 2010. Date of publication November 9, 2010; date of current version January 4, 2011. This work was supported in part by the ANPCyT under Grant PICT-2006 643 and Grant 2226.

The authors are with the CIFASIS - French Argentine International Center for Information and Systems Sciences, UPCAM (France) / UNR-CONICET (Argentina), Rosario S2000EZP, Argentina (e-mail: grinblat@cifasis-conicet.gov.ar; uzal@cifasis-conicet.gov.ar; ceccatto@cifasis-conicet.gov.ar; granitto@cifasis-conicet.gov.ar).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2010.2083684

[1]The previous version was limited to use one SVM for each point in the training sequence.

In the estimation task, we train a sequence of classifiers using a given dataset, and then we test the sequence of classifiers in a new dataset equivalent to the training one (involving the same time span of the training set). This estimation task is appropriate, for example, to the analysis of a slow drifting problem with a cyclic behavior [24]. In this case, it is important to model accurately all the time span of the dataset, not only the last section. Our method is particularly aimed at this task, in which one can use not only information from past records but also from records corresponding to a future time.

For the prediction and extrapolation tasks, we train a sequence of classifiers on a section of a dataset and then test it on the following section. In the prediction task, we evaluate each sequence using only the next point in the dataset. In the more difficult extrapolation task, we use a subset of data points including several steps ahead in the future. In this case, we need to extrapolate the position of the decision boundary, for which we use a simple linear technique. The objective in these two tasks is to make short-term predictions on a system that is evolving in a completely unknown way. In both cases, the evaluation puts emphasis in the performance of the last classifier(s) in the sequence. We discuss the three settings again in Section IV.

This paper is organized as follows. In Section II, we discuss previous works on concept drift. In Section III, we introduce our solution to the problem and show illustrative examples, leaving mathematical details to Appendix A. We also discuss the relation with similar solution in other areas. Then, in Section IV, we present empirical results and comparisons with similar methods using artificial and real-world datasets and, finally, Section V closes the work with some remarks and conclusions.

II. Previous Work

Drifting concepts are specific classification problems in which the labels of examples or the shape of the decision boundaries change over time. In particular, we are interested in problems that change slowly over time. In a recent work, Kolter *et al.* [25] include a lengthy review of the state of the art in the field, starting from the early work of Schlimmer and Granger Jr. [26]. According to this, in this section we limit ourselves to briefly describe most of the previous methods and refer the interested reader to [25] for details and further references.

There are three main approaches to concept drift in the literature: sample selection, sample weighting, and ensemble methods [5], [6].

As we stated before, the most common solution to the drifting concepts problem is to use a temporal window of a given length, also called a sliding window (SW) and to build a different classifier (or adapt a previous one) for each window [2], [7]–[9], [19], [20]. Some authors prefer to use the equivalent idea of uniform or stationary "batches" [20], [27], [28]. If the window is too big, the response time needed by the algorithm to follow the changes is excessive. On the contrary, when the window is too small, the algorithm adapts quickly to any drift in the data, but it is also more sensitive to noise and loses accuracy because it must learn the input–output relationship from only a few examples. As a potential solution, many algorithms include an adaptive window size. One of the first to do that was FLORA2 [8]. Klinkenberg and Renz [19] presented an algorithm that modifies the number of stationary batches in the dataset by monitoring the accuracy, recall, and precision of the method. They applied it to the problem of detecting relevant documents in a series. Klinkenberg and Joachims [2] used SVMs to find the optimal time interval. The method adjusts SVMs with various window sizes, calculates the corresponding $\xi\alpha - estimator$ [29] using the last batch, and keeps the window size that minimizes that quantity.

Castillo *et al.* [27] and also Lanquillon [20] use statistical quality control to determine whether there is a concept change in a given batch. When this happens, a new classifier is constructed from scratch using only the data points considered to belong to the new context. Koychev *et al.* [30] also used a (different) statistical test to determine whether there was a change in concept in the last batch. In an interesting series of papers, Alippi and Roveri [12], [13], [31] developed another test for concept change and an adaptive classifier based on nearest neighbors.

In a work focused on recurrent systems, Koychev [32] proposed to use a relatively small time window to learn the current context, then to select the past episodes that show a high predictive accuracy, and finally to train again the classifier using the original and the newly selected data points. In a similar way, Maloof *et al.* [33] introduced a method for the selection of examples in a partial memory learning system. They select some extreme examples and add them to the current ones to model the actual concept description. FLORA3 [8] also takes recurrence into account.

In general, all these methods select an appropriate subset of the original dataset to train independent classifiers that are, each one, accurate at the corresponding time. In most cases, the selected subset is taken from the most recent examples. As a representative of sample selection methods in the comparisons we present in Section IV, we selected to use a simple SW, but with the length of the window optimized using independent validation sets.

In an early work, Koychev [34] proposed to decrease the importance of old examples in the classifier simply by giving each data point a relative weight that decreases with time. The method, called gradual forgetting (GF), can be viewed as a softening of the sliding windows (SW) strategy, which gives "hard" (0/1) weights to (older/newer) examples. The author suggests using a simple linearly decreasing function for the relative weight. Klinkenberg [28] used an exponentially decaying function to weight older samples. The GF method is simple and easy to implement, and usually gives better results than SW. We also included (linear) GF in the comparisons in Section IV as a representative of sample weighting methods.

Several authors have discussed the use of ensemble methods for drifting concepts. The streaming ensemble algorithm [10] fits an independent classifier to each batch of data, which are combined into a fixed-size ensemble using a heuristic replacement strategy. Wang *et al.* [35] used a similar strategy, weighting each member of the ensemble according to its accuracy on the current batch. Gao *et al.* [36] applied ensembles to the task of classifying data streams and Hashemi *et al.* [37] used evolving one-versus-all multiclass classifiers for the same problem. Polikar and co-workers [24], [38], [39] developed an ensemble-based framework for different nonstationary problems. Recently, Kolter and Maloof [25], [40] introduced an improved ensemble method based on a previous work of Littlestone *et al.* [41]. Their dynamic weighted majority algorithm (DWM) dynamically creates and removes weighted experts in response to changes in performance. DWM uses four mechanisms to cope with concept drift: it trains, weights, or removes learners based on their individual performance, and also adds new experts based on the global performance of the ensemble. The authors produced an extensive evaluation and concluded that DWM outperformed most of the other learners considered in their work. DWM is also included in the evaluation in Section IV as the representative of ensemble methods.

Finally, out of the scope of our work, we mention that the drifting problem has also been addressed from a computational learning theory point of view [4], [14], [16], [17], where some guarantees and theoretical bounds regarding the learning of sequences of functions were established.

III. TA-SVM

Let us assume that we have a dataset $[(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)]$, where each pair $(\mathbf{x}_i, y_i)$ was obtained at time i (that is, they are time ordered), $\mathbf{x}_i$ is a vector in a given vector space, $y_i = \pm 1$, and that the relation between $\mathbf{x}$ and y has a slow change in time. Our strategy to cope with this problem is to divide the dataset into m consecutive nonoverlapping time windows tw_ν (with $\nu = 1, \ldots, m$ and $m \leq n$), and to create a coupled sequence of m (static) classifiers, each one being optimal in the corresponding time window. As we are assuming that the concept has a slow evolution, we expect that the classifiers will have the same property. According to this, we seek for a sequence of good classifiers in which time neighbors are similar to each other. The best solution to our problem should be a compromise between (individual) optimality and (neighbor) similarity. If we can define a simple distance measure $\mathrm{d}(c_\nu, c_\mu)$ to quantify the diversity between two neighbor classifiers c_ν and c_μ, the base idea of our method is to minimize a two-term cost function

$$\min \frac{1}{m}\sum_{\mu=1}^{m} Err_\mu^2 + \gamma \sum_{\mu=1}^{m-1} \mathrm{d}(c_\mu, c_{\mu+1}) \qquad (1)$$

where the first term is the average of the usual cost function for each of the m classifiers and the second evaluates the total difference among the sequence of discriminant functions. The free parameter γ regulates the compromise between both terms, as in any regularized fitting. In principle, this method can be used with any classifier, if an appropriate distance measure can be defined. In this formulation, we use linear SVMs as classifiers (as, usually, we can use Kernels to produce nonlinear predictors if needed). Therefore, we look for a sequence of m pairs $(\mathbf{w}, b)$, each one defining a high-margin hyperplane h_ν given by $\mathbf{w}_\nu \mathbf{x} + b_\nu = 0$, where $\mathbf{x}$ belongs to the dataset's vector space.

We use a simple quadratic distance measure to quantify the diversity between hyperplanes

$$\mathrm{d}(h_\nu, h_\mu) = ||\mathbf{w}_\nu - \mathbf{w}_\mu||^2 + (b_\nu - b_\mu)^2.$$

Applying this measure to (1), we can introduce a new cost function for the full sequence of SVMs

$$\min \frac{1}{m}\sum_{\mu=1}^{m} ||\mathbf{w}_\mu||^2 + C\sum_{i=1}^{n}\xi_i + \frac{\gamma}{m-1}\sum_{\mu=1}^{m-1}\mathrm{d}(h_\mu, h_{\mu+1}) \quad (2)$$

subject to

$$\begin{aligned} \xi_i &\geq 0 \\ y_i(\mathbf{w}_{\mu(i)}\mathbf{x}_i + b_{\mu(i)}) - 1 + \xi_i &\geq 0 \end{aligned}$$

where $i = 1, \ldots, n$ and $\mu(i)$ indicates the time window including point $\mathbf{x}_i$. The first two terms in (2) correspond to the usual margin and error penalization terms in SVM [42], but for a complete set of classifiers, each one trained on a different time window. It is easy to see that the solution of this two-term problem gives the same sequence of SVMs that can be obtained by solving each SVM individually (if we use the same C for all SVMs). The last term in (2) corresponds to the new diversity penalization. The inclusion of this term couples the sequence, making each SVM dependent of all the others. The free parameter γ regulates the relative cost of the new term. Low γ values will almost decouple the sequence of classifiers, allowing for increased flexibility. High γ values, on the other side, will produce a sequence of almost similar SVMs.

In this formulation, we have only considered the case in which data points arrive at regular time intervals. The more general case of nonconstant intervals (including missing data or data coming in bursts) can be addressed with simple extensions, for example, by giving different relative weights to the distances considered in the second term of (1), or by assigning different amounts of points to each hyperplane (see Appendix A).

It is interesting to see that this formulation is valid even for time windows including only one point ($m = n$), because the coupling introduced by the new penalization term breaks with the indetermination of having only one point to define a hyperplane.

As we show in detail in Appendix A, by deriving the corresponding dual (as usual in SVM methods) we can rephrase the problem in (2) as

$$\max_\alpha -\frac{1}{2}\alpha^T R\alpha + \sum_i \alpha_i$$

subject to

$$\begin{aligned} 0 \leq \alpha_i &\leq C \\ \sum \alpha_i y_i &= 0 \end{aligned}$$

where α_i are the Lagrange multipliers and R is a matrix with Kernel properties. The solution to this maximization problem is a coupled set of SVMs that evolve in time, which we call time-adaptive SVMs (TA-SVMs).

The time complexity of a TA-SVM is similar to that of the plain SVM. It can be analyzed in two stages, the kernel computation and the solution of the optimization problem. The new kernel can be computed in $O(n^2)$, as we show in Appendix B. What is left after this is solving a conventional SVM optimization problem, which is also $O(n^2)$ using for example sequential minimal optimization (SMO) [43].

Overall, TA-SVM, in its basic form, has the same scaling problems as plain SVM. For the estimation task this is not critical, as one usually needs to solve the problem only once. But basic TA-SVM does not work by making updates of previous solutions, it always looks for a new global solution. Then, for the prediction and extrapolation tasks our method requires to solve an $O({n_t}^2)$ problem each time a new batch arrives, where n_t is the total number of instances at time t.

A. Connections with Other Areas

There are connections between TA-SVM and methods from other areas, so it is interesting to discuss them at this point.

In online learning [41], [44], [45], the objective is to learn the correct input–output relationship as fast as possible from data that arrive sequentially, one instance at a time. Algorithms for online learning mostly work by making updates from the solution at the previous step. Many concept drift methods, including for example ensemble methods or the FLORA series described before, also update their classifiers as a function of the last batch. Some of these methods can use very short batches or even learn from one point at a time, making them closer to online learning strategies. The main difference between the two settings is that online learning does not necessarily track concept changes, and therefore no efforts are made to forget out-of-date information in most cases. In particular, algorithms for sudden drift are more related to online learning, given their common objective of fast convergence to the optimal solution.

In a recent work, [46] introduced an online learning algorithm for kernel methods, the passive-aggressive (PA) algorithm. At each step, PA modifies its solution trying to give the right label to the newly arrived instance while keeping the solution similar to the previous step by using a coupling term, similar to our formulation (2). The general idea of keeping a tradeoff between local accuracy and smoothness of the time evolution of the solution is similar to ours, but with two main differences. One is determined by the different objectives of the methods. TA-SVM is able to use information from both past and future times (when available), because it fits (offline) the full sequence of SVMs at the same time, while PA looks for the best current solution using only past information. This becomes particularly relevant for the estimation task, as defined before. The second difference is that TA-SVM looks actively for a maximum margin solution, while PA does not update its solution if the margin of new points is greater than a given value, because of its passive nature. Other authors have also used the idea of constraining the possible updates of the current solution [47], [48]. For example, [49] projects the update to a convex set, which allows establishing shifting bounds on the total loss for some general additive regression algorithms.

TA-SVM is in fact more related to the area of multiple task learning (MTL) [50], [51]. MTL algorithms are designed to learn several related tasks at the same time, profiting from the relevant information available in the different tasks. Our method makes use of the same idea. In TA-SVM, each of the related tasks is the classification problem corresponding to a given time window. As we are assuming a gradual drift, time-neighbor tasks should be similar to each other. Recently, [52] introduced a framework for MTL with kernel methods. In their formulation, the relation between the different tasks can be regulated using an appropriate homogeneous quadratic function. TA-SVM can be viewed as a particular case of this formulation, using the first and third terms of (2) as the regularizer.

As we mentioned in the introduction, we use three different tasks to evaluate TA-SVM, namely estimation, prediction, and extrapolation. The estimation task is more related to the MTL problem, giving the same importance to all problems, while the prediction and extrapolation tasks are more tied to the online learning problem and its emphasis on the current hypothesis.

B. Illustrative Example

As a first example of the potential of TA-SVM, we apply it to the artificial sliding Gaussians dataset. This is a two-class problem, in which each class is sampled from a Gaussian distribution. Both classes drift together, following a sinusoidal trajectory on a 2-D input space. We generated $n = 500$ points according to

$$\mathbf{x}_i = \left\{\frac{2i\pi}{500} - \pi + 0.2y_i + \varepsilon_1, \sin\left(\frac{2i\pi}{500} - \pi + 0.2y_i\right) + \varepsilon_2\right\}$$

where $i = 1, \ldots, 500$, $\varepsilon_{1,2}$ are sampled from a normal distribution with zero mean and $\sigma = 0.1$, and y_i is a balanced random sequence of ± 1. Fig. 1 shows a realization of the dataset at three different times.

We used the first 450 points as training set, and generated in each case a second realization of 450 points to use as validation set, in order to select the optimal values of γ, C, and the length l of the window used by SW. Fig. 2 shows the sequence of hyperplanes obtained with TA-SVM ($m = n$) and SW-SVM. In this latter case, for each point $\mathbf{x}_i$ we trained an SVM using a specific time window of length $2l + 1$ centered on that point (when this is not possible, here and in all other experiments we used l points from one side and all the available points from the other). To improve the readability of the figure, we show only one in every ten consecutive SVMs. It is evident that the coupled solution of TA-SVM produces a more regular less noisy sequence of classifiers than the use of independent optimal SWs.

C. Dependence on γ

As a second demonstrative example, we evaluated the dependence on γ of TA-SVM solutions. In this case, we used the rotating hyperplane dataset, which is a set of 500 points sampled from a uniform distribution in a d-dimensional hypercube $[-1, 1]^d$ [35], [53], [54]. The decision boundary for the two classes is a slowly rotating hyperplane (passing through

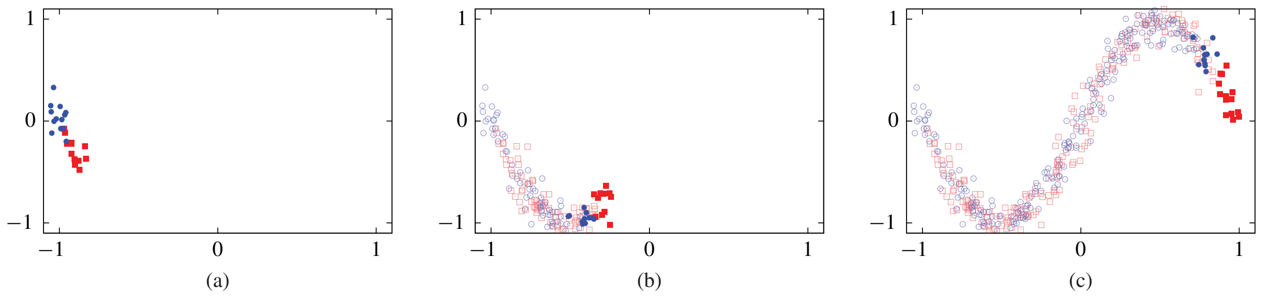


Fig. 1. Sliding Gaussians dataset at (a) $t = 25$, (b) $t = 175$, and (c) $t = 475$ time units. In each figure, the last 25 generated points are filled.

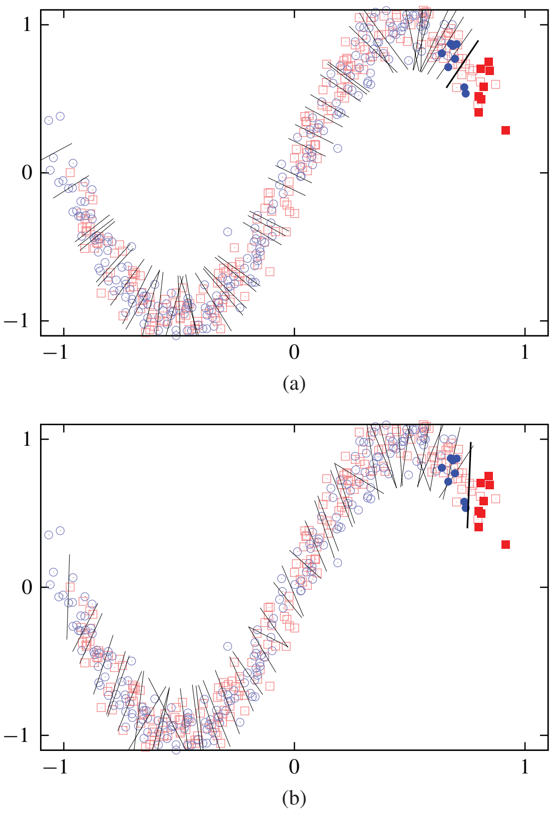


Fig. 2. Sequence of hyperplanes obtained as a solution of the sliding Gaussians dataset with (a) TA-SVM and (b) SW-SVM.

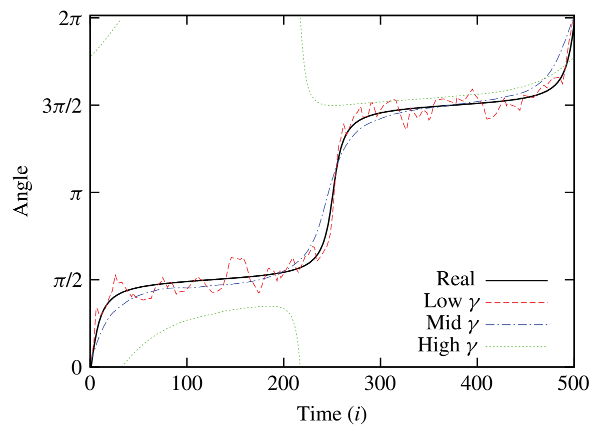


Fig. 3. TA-SVM solutions for the rotating hyperplane problem using different γ values.

the origin). The direction of the hyperplane is defined by its normal vector $\mathbf{v}$, which in this experiment follows the law:

$$v_1(i) = \frac{\cos(2\pi i/500)}{10}$$
$$v_2(i) = \sin(2\pi i/500)$$
$$v_{3,\ldots,d}(i) = 0.$$

Point $\mathbf{x}_i$ has class $y_i = sign(\mathbf{x}_i \mathbf{v}(i))$.

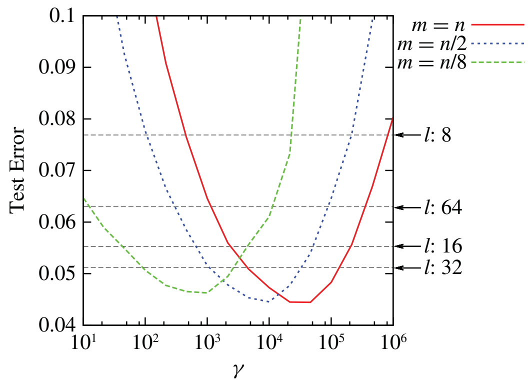
In this first experiment with this dataset, we used $d = 2$, $m = n$, and three different γ values that show the typical responses of our method. The C parameter was set to 1 in all cases, because we checked that the solutions are almost independent of C in this problem.

In Fig. 3, we plot the real angle between $\mathbf{v}$ and the first axis, as a function of time. We also show the solutions found by TA-SVM for three values of γ. Using a low γ ($\sim 10^2$, dashed line), the TA-SVM solution is too flexible, following particularities of the training set. For an adequate mid-γ value ($\sim 10^5$, dash-dotted line), there is an optimal solution, with a balance between local accuracy and global flexibility. Last, for a high γ ($\sim 10^8$, dotted line), the change over time of the hyperplane is highly penalized, therefore it remains almost constant and similar to the solution that can be found by a classical SVM. For this particular dataset, as $\mathbf{v}$ does a complete turn, both classes are almost uniformly distributed and the optimal static solution is a null vector. The soft and erratic trajectory of TA-SVM corresponds to the angle of a nearly null vector in this case.

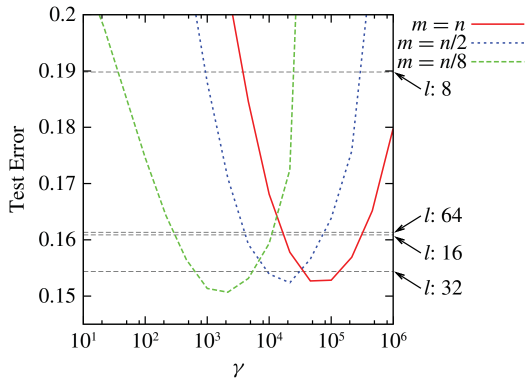
D. Dependence on m

In the previous example, we used the maximum flexibility of TA-SVM, $m = n$, and regularized it by optimizing the value of γ. TA-SVM has other simple way to control its complexity, i.e., to use a shorter sequence of SVMs ($m < n$), with the added advantage of a reduced computational burden.

In Fig. 4, we show the evaluation of this possibility. Again, we used 100 realizations of the rotating hyperplane dataset, with $d = 2$. We considered three settings: one classifier for each training data ($m = n$, full line), one classifier for every two data points ($m = n/2$, dotted line) and one for every eight points ($m = n/8$, dashed line). In (a), we show the corresponding results as a function of γ. The first observable result is that the optimal values of γ decrease when using fewer classifiers. This is easy to explain considering that shorter sequences are naturally less flexible (simply because there are fewer hyperplanes and hence fewer



(a)



(b)

Fig. 4. Test errors for the rotating hyperplane problem as a function of γ. (a) Results using $d = 2$ and different values of m. (b) Same as before, but for a noisy dataset with 10% flipped labels.

adjustable parameters) and thus require a lower diversity penalization. A second observation is that the best result is obtained with $m = n$, and that in this case the use of fewer SVMs produces a small decrease in performance. This result is a consequence of using a noiseless dataset, as will become clear when analyzing (b) of the same figure. Horizontal lines represent the error rates produced by SW-SVM with different lengths (l). It is interesting to note that in all cases there is a wide range of γ values for which TA-SVM outperforms the results of the optimal l.

We repeated the full experiment using a noisy dataset, in which 10% of the labels were randomly switched. We show the corresponding results in Fig. 4(b). Qualitatively, the results are similar to the noiseless dataset. TA-SVM results are better than SW-SVM over a wide range of γ values. The only difference is that in this noisy case the best performance is obtained using eight points per hyperplane ($m = n/8$, dashed line). In this case, the higher flexibility of the $m = n$ models allow them to learn some noisy characteristics of the datasets, and the problem cannot be avoided using a stronger coupling. Using more points per hyperplane allows TA-SVM to filter some noise locally, at each SVM, thereby improving the performance of the sequence.

E. STAGGER

As a last example, we applied TA-SVM to the most used benchmark in drifting concepts methods [8], [30], [32]–[34], i.e., the STAGGER dataset [26]. The dataset has three categorical inputs, each one taking three possible values. The dataset has 120 training instances, and the concept changes abruptly two times, one every 40 instances. This is a particularly challenging problem for TA-SVM, because there are only sudden drifts of the concept in this case. With this dataset, we are demonstrating the capabilities of our method in the most unfavorable situation.

We first generated the training sequence of 120 data points, each time sampling with repetition from the set of 27 possible instances, and labeling each point with the right concept for each time step. As in [8], we generated a similar test sequence but with 100 points at each one of the 120 time steps. We also generated a third sequence with 100 points at each time step to use as a validation sequence.[2]

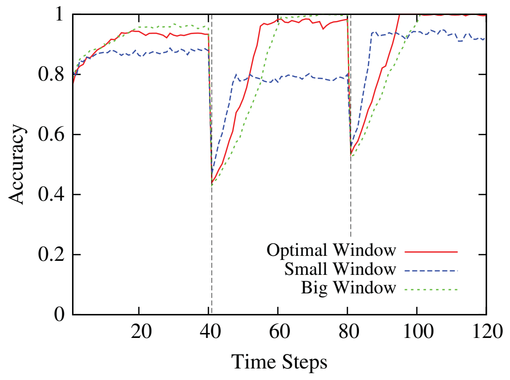
At each time step i, we trained a sequence of classifiers using $\mathbf{x}_{1,\ldots,i}$, and used the last classifier in the sequence to predict point $\mathbf{x}_i$ in each one of the 100 test points (i.e., the prediction setting). For both methods, we used one SVM per training instance ($n = m$) and the independent validation set to select the optimal γ and l values for the full sequence, as in previous datasets. Again, we used a fixed C value in this noiseless dataset, because we verified that both methods are nearly independent of C also in this case. The full experiment was repeated 100 times.

In Fig. 5, we show the average accuracy of both methods as a function of time. In both (a) and (b), the two vertical dashed lines correspond to the concept changes. In (a), we show the performance of independent SWs for three different lengths. For a short window, there is a quick response to changes, but there is also a lack of information about the concept. On the contrary, for the biggest window the adapting times are bigger, but the final performance is better. The optimal length compromises between both situations. In (b), we compare the optimal settings for both methods. TA-SVM is equivalent to or better than independent SW-SVM even in this (most unfavorable) dataset.

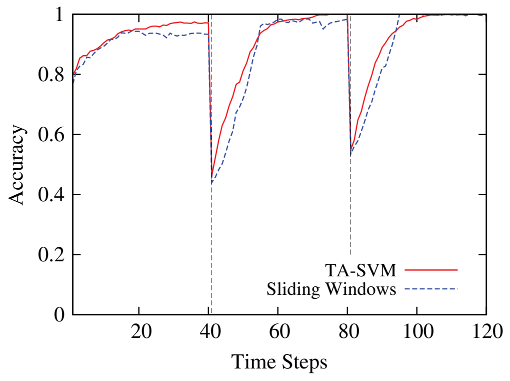
IV. Empirical Evaluation

In this section, we compare the new method with other state-of-the-art strategies for drifting problems under the three settings discussed in the introduction: prediction, estimation, and extrapolation. We use the same artificial datasets and the STAGGER concepts introduced in the previous section, and the real-world electricity pricing dataset [55]. Unless stated otherwise, we report the mean classification error, with its standard deviation, over 100 independent realization of the training sets. In all cases, we use independent validation sets to optimize the internal parameters of the methods under evaluation. We always use the same realizations of the training, validation, and test sets for all the methods.

[2]The use of a validation sequence is not typical in the previous literature on the STAGGER dataset. The setting we use in this example is therefore not comparable with previous works. We use it as a demonstration of the capabilities of TA-SVM in its optimal setting. In Section IV, we use this dataset to evaluate TA-SVM in the standard setting.



(a)



(b)

Fig. 5. Results for the STAGGER dataset. (a) Behavior of independent SVMs using overlapping sliding time windows. (b) Direct comparison of optimal settings of both methods.

A. Prediction

In the prediction task, we are given a subset of data points spanning some period of time and our goal is the prediction of the next arriving data point (which should have no drift from the last one). To evaluate the different methods in this case, we use the same settings as for the STAGGER dataset in the previous section. We compare TA-SVM with other three methods that we described in Section II: SW, GF, and DWM, in all cases using linear SVMs as classifiers.

In a first evaluation we use the rotating hyperplane dataset, but with a uniform rotation in this case: $\mathbf{v_i} = (\cos(2\pi i/500), \sin(2\pi i/500))$. In all cases, we generated 100 training sets with 500 points, and for each time i and each training set we generated an independent validation and test sequence with 100 points. We use different values of d in order to evaluate the performance of the methods in high-dimensional spaces. For each training set (and all the corresponding validation and test sets), we applied a random rotation of the original space, to produce a dataset in which all the variables are relevant to the concept. We use one classifier per data point (which is the most flexible setup) for the four methods. For TA-SVM, this means that we set $m = n$ (which could be not optimal, as we showed in Fig. 4). In the case of SW-SVM and GF-SVM, for each point $\mathbf{x}_i$ we fit an SVM using a specific time window of length $2l + 1$ centered on that point. It is worth mentioning that in this case consecutive time windows are almost completely overlapped. For DWM-SVM, this is the default setting where the ensemble is updated at each time step.

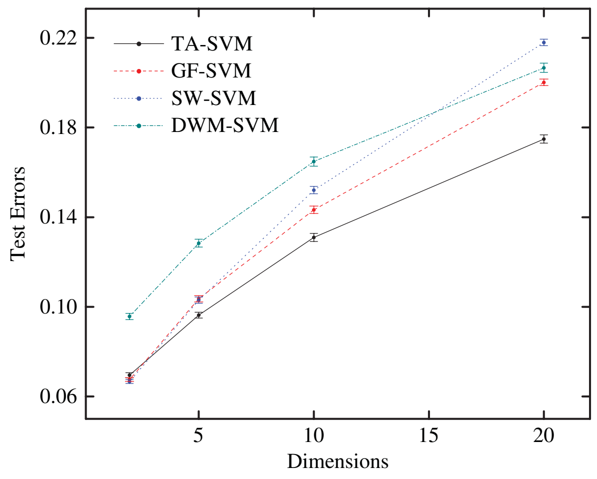
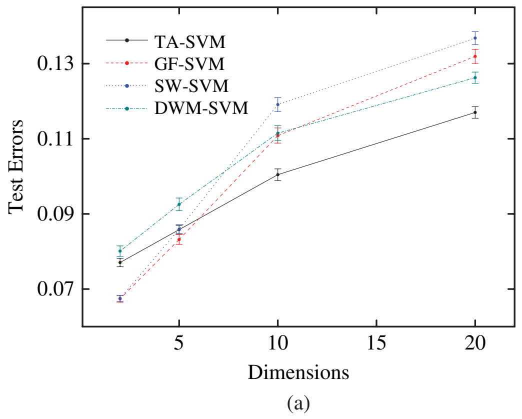


Fig. 6. Prediction test errors as a function of the dataset dimension for the rotating hyperplane problem.

In Fig. 6, we compare the performance of all methods as a function of the number of dimensions in the dataset. SW and GF are the best methods for $d = 2$, probably because of some small overfitting due to the nonoptimal setting of TA-SVM. On the other hand, they are clearly more affected by the increase in the number of dimensions. This can be explained considering that, even if SVMs are known to work well in high-dimensional spaces, there are always more chances of producing solutions with bad generalization in this case. SW and GF can use bigger time windows (that is more training points) to increase their performance, but this has the added cost of allowing a bigger concept change in the considered window. TA-SVM can deal more efficiently with this problem, because it searches for a global solution of the problem, sharing information among all classifiers as a consequence of the coupling. DWM starts with a relatively low performance, but it is also less affected by the “curse of dimensionality” than SW and GF, probably because it also shares information among the sequence of classifiers, as we discussed before.

We repeated the evaluation using the rotating Gaussians dataset. This dataset is very similar to the previous one. The main difference is that the classes are sampled not from uniform distributions at each side of the hyperplane but from normal distributions centered at $+\mathbf{v}_i/2$ and $-\mathbf{v}_i/2$, both with $\sigma = 0.3$. The optimal solution of this problem is more difficult for SVMs because it has some class overlapping and fewer points on the decision boundary. We consider two scenarios here. In the first one, all other settings of the problem are equal to the previous case. The corresponding results are shown in Fig. 7(a). Qualitatively, the behavior of all four methods is the same as in the previous dataset. There is a bigger gap for low-dimensional problems and also a bigger decay of the performance of SW and GF for high-dimensional datasets. In the second scenario, we considered a faster drift of the classes, including two full turns of the Gaussians around the origin at the same time. In this case, we generated sequences of 500 points using $\mathbf{v_i} = (\cos(2\pi i/250), \sin(2\pi i/250))$.



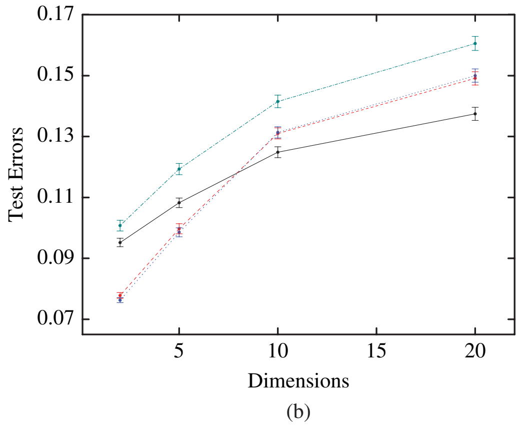


Fig. 7. Prediction test errors as a function of the dataset dimension for the rotating Gaussians problem (a) for a dataset including a full turn of the Gaussians and (b) for the same problem, but including two full turns of the classes, i.e., a faster drift.

In Fig. 7(b), we show the corresponding results. It is easy to see that there is a general increase in error levels, associated with the faster drifting of the classes. Again, SW and GF are better than the other methods for low-dimensional datasets. TA-SVM still outperforms the other methods for high-dimensional datasets, but DWM-SVM does not work well in any situation in this case.

We also evaluated the STAGGER dataset in the prediction task. In this case, we followed as much as possible the settings used in previous works with this dataset. According to this, we do not use an external validation sequence, we only use the training and test sequences described in the previous section. We replace the external validation with an internal fourfold cross validation using the training sequence available at each time step in order to set the optimal values of the free parameters of all the methods. In Fig. 8, we show the average prediction accuracy as a function of time. TA-SVM shows the fastest response to concept drift, but after that DWM-SVM shows a better convergence to the optimal decision. The bias to a continuous drift in TA-SVM reflects in a slower convergence rate in this case. In Table I, we show the same results averaged over time. Overall, TA-SVM shows a very good performance in this problem involving sudden concept drift. DWM-SVM slightly outperformed the new method, but most of the difference is in the first concept, before any concept drift.

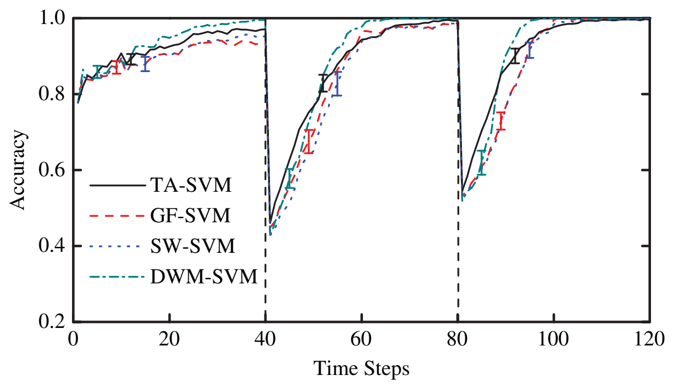


Fig. 8. Average prediction accuracy of the methods tested in this paper on the STAGGER dataset, as a function on time. Error bars show the 95% confidence interval.

TABLE I
PREDICTION ACCURACY ON THE STAGGER DATASET, AVERAGED OVER REALIZATIONS AND TIME, FOR THE METHODS TESTED IN THIS PAPER. IN PARENTHESIS WE SHOW THE STANDARD DEVIATION OF THE MEAN ACCURACIES

Method	Accuracy (%)
SW-SVM	87.14 (0.12)
GF-SVM	87.67 (0.12)
TA-SVM	90.00 (0.10)
DWM-SVM	90.83 (0.12)

B. Estimation

As we stated in the introduction, in the estimation task we train a sequence of classifiers using a given dataset and then we test the complete sequence of classifiers in an independent test set involving the same time span of the training set. The objective is to evaluate all the classifiers at the same time, not only the last one as in the previous task. In this case, for each training set of 500 points we generated 100 equivalent validation and test sets. We optimized all internal parameters using the validation sets and then we evaluated the resulting classifiers on the test sets. We assume that we know the time step i at which each point in the test set was measured, so we can use the corresponding classifier from the trained sequence. For the SW and GF methods we use symmetric time windows of length $2l + 1$, centered at time i. We do not use DWM in this case because it was not designed for the estimation task. In DWM, the classifier corresponding to time i can only use information from the points in its past. The other three methods can see the full training set, which would make the comparison unfair.

In this evaluation, we fixed the number of dimensions at $d = 2$ (for all the datasets) and varied the number m of classifiers in the sequence, in order to evaluate the dependence of the methods with the m/n ratio. In the first place, we used the same rotating hyperplane dataset as in the prediction task. In Fig. 9(a), we show the results for the three methods included this time. We also evaluated a noisy version of this dataset, (b), in which a random 10% of the labels were switched. In (c) of the same figure we show the results corresponding to

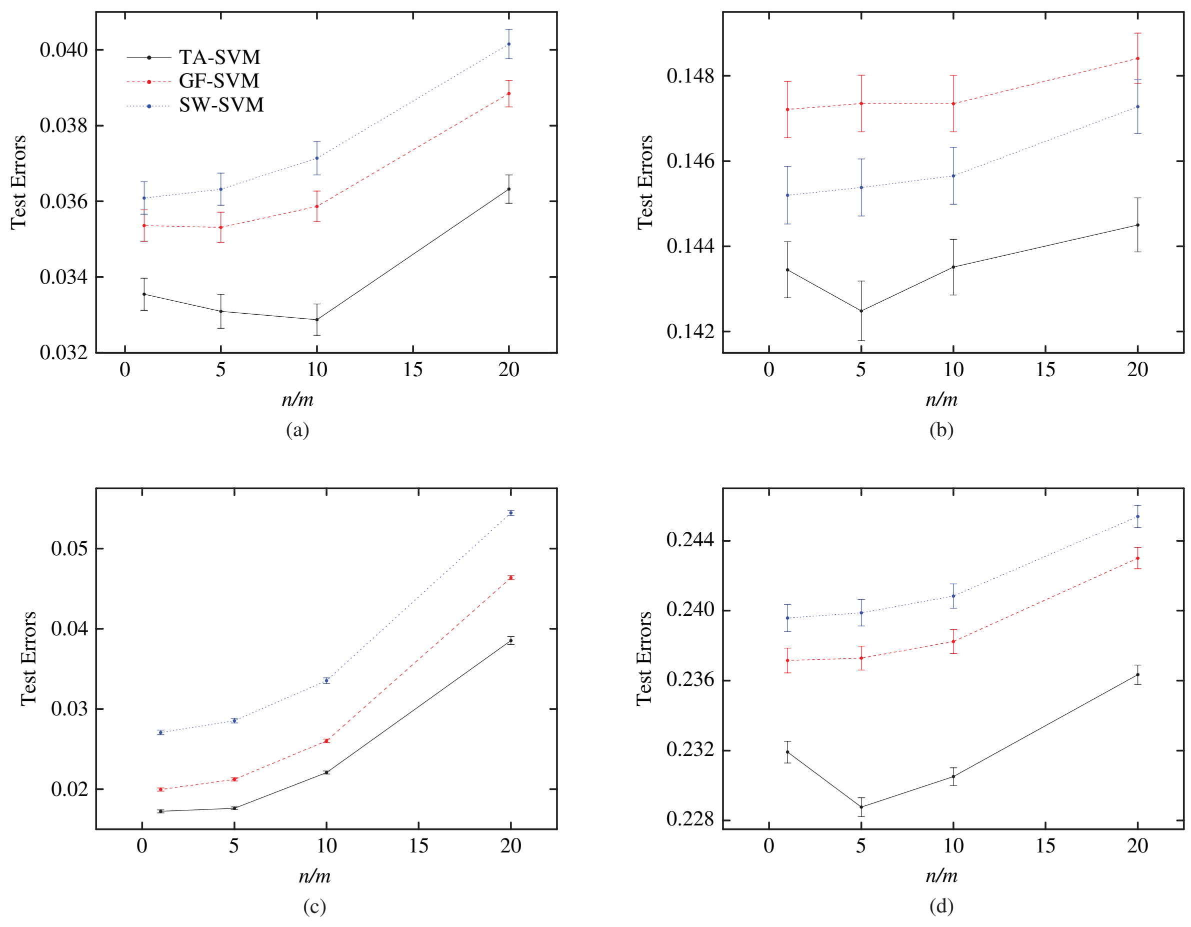


Fig. 9. Estimation test errors as a function of the number m of classifiers in the sequence (a) for the rotating hyperplane problem, (b) for the same problem with 10% noise, (c) for the sliding Gaussians problem, and (d) for the same problem using normal distributions with bigger σ.

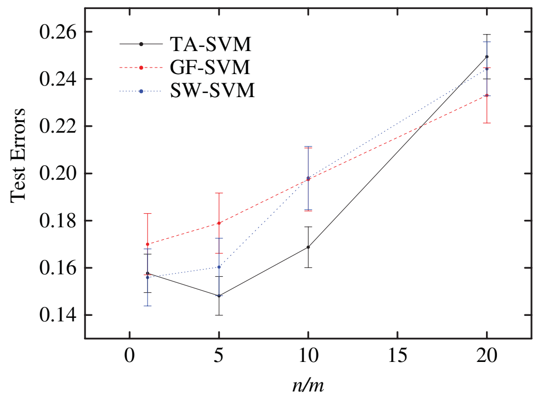
the sliding Gaussians dataset, using the same settings as in Section III-B. Finally, in (d) we use a second version of the sliding Gaussian generated with $\sigma = 0.3$, i.e., with more overlap of the classes, and the other setting equal to (c). In the four situations (two datasets times two noise levels), the qualitative results are similar. The overall performance of SW and GF deteriorates when fewer classifiers are used in the sequence (higher n/m values). Clearly, when using fewer SVMs the concept drift becomes more relevant to the problem. The effect is clearer for the low-noise situations, (a) and (c). For noisy situations, as we discussed before, there is always certain trade-off between noise and drift. TA-SVM clearly outperforms SW and GF in the estimation task. This was expected, as TA-SVM was designed to learn accurately all the sequence at once. We already showed in Section III-D that TA-SVM usually works better when using $n/m > 1$, in particular in noisy situations. This result is evident here in all cases except for (c), which is a problem with low-noise and a relatively fast drift of the decision boundary.

C. Extrapolation

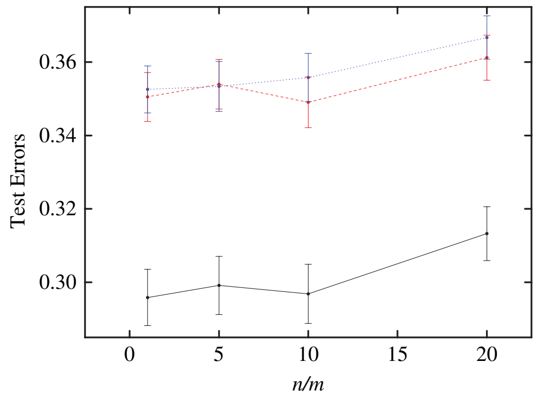
The extrapolation task is an extension of the prediction task, in which we are interested in the prediction of several steps ahead into the future. In this case, we need to extrapolate the position of the decision boundary some steps into the future, starting from the last classifier in the sequence.[3] Our method does not assume any functional form for the time evolution of the sequence of classifiers, the only constraint is that neighboring hyperplanes should be close to each other, so we do not have a principled way to determine the position of each future classifier. In consequence, we must choose an appropriate external method to make an extrapolation based on the position of each classifier in the sequence. In this paper, we use a simple linear extrapolation, but a more complex model could be applied if required (and if there is enough data available).

For the experiments in this task, we generated training sequences with 450 points, and for each one of these training sets we generated 100 test sequences with 500 points each. At each run, we optimized all internal parameters using the first 450 points of the test sequences as validation sets and then evaluated the resulting classifiers on the last 50 points of

[3]A simple way to do it would be to use an extended sequence of classifiers with hyperplanes located in the future period that we want to predict, and let TA-SVM choose the position of each one. Unfortunately, this procedure will not have the effect we are looking for. As we do not have training points for the future period, the solution will be a compromise between only two penalties: the last term in (2), which will make the solution to stay at the position of the last hyperplane before the extrapolation period, and the first term in (2), which will move the solution to the null vector in the extrapolation period.

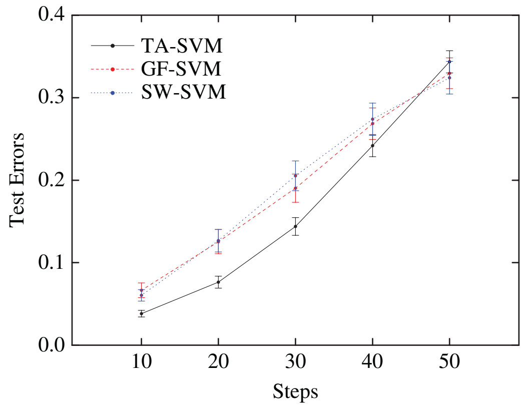


(a)

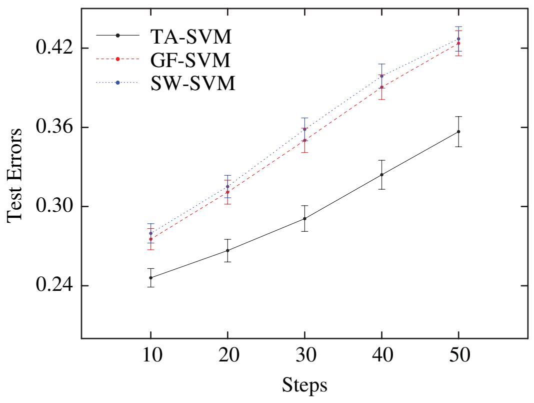


(b)

Fig. 10. Extrapolation test errors as a function of the number m of classifiers in the sequence (a) for the sliding Gaussians problem and (b) for the same problem using distributions with bigger σ.



(a)



(b)

Fig. 11. Extrapolation test errors as a function of the number of predicted steps into the future (a) for the sliding Gaussians problem and (b) for the same problem using distributions with bigger σ.

these test sets. As in the previous case, we fixed the number of dimensions at $d = 2$ and varied the number m of classifiers in the sequence, from $n/m = 1$ to $n/m = 20$. To extrapolate the position of the decision boundary, we used a simple linear extrapolation of the values of each component of $\mathbf{w}$ ($w_i(t) = \alpha_i t + \beta_i$) and of b ($b(t) = \alpha_b t + \beta_b$), fitting the coefficients of the linear models ($\alpha_{i,b}$ and $\beta_{i,b}$) using all the SVMs in the last 50 training points. The number of SVMs included in the extrapolation goes from 50 for $n/m = 1$ to only 2 for $n/m = 20$. We do not use DWM in this case because it produces an ensemble of SVMs and there is no simple way to extrapolate the position of the decision boundary for this method.

For this evaluation, we used the sliding Gaussians dataset in the same two settings that we explained in the estimation task. In Fig. 10, we show the corresponding results. All methods become more unstable in this task, as indicated by the bigger error bars, because we are superposing two error sources, the fitting of the classifiers, and the extrapolation of their position. In Fig. 10(a), we can see again a typical behavior of TA-SVM, with the best performance at $n/m > 1$. For $n/m = 20$, all methods show similar poor results, mainly associated with bad extrapolations of the decision boundaries. For the noisy situation in Fig. 10(b), the difference between TA-SVM and the other methods is clearer. In Fig. 11, we show, for the same two datasets, the evolution of the classification error as a function of the number of time steps into the future predicted by all methods. The results correspond to $n/m = 10$. In the low-noise situation, (a), TA-SVM shows the best performance for all but the maximum number of time steps, when all methods are equivalent. With more noise present, (b), TA-SVM is clearly superior in all situations.

D. Real-World Case: Electricity Pricing

As a last evaluation of TA-SVM we considered a real-world problem, the electricity pricing dataset [55]. The dataset contains 45 312 instances collected at regular 30-min intervals during 30 months, from May 1996 to December 1998. The data was first obtained directly from the electricity supplier in New South Wales, Australia. There are five attributes in total. The first two date the record in day of week (1 to 7) and half-hour period (1 to 48). The last three attributes measure the current demand, the demand in New South Wales, the demand in Victoria, and the amount of electricity scheduled for transfer between the two states. The target is a binary value indicating whether the price of electricity will go up or down.

Following Harries [55], we considered batches of 1-week length. At each week, we train the classifiers with all previous batches and predict the next batch (i.e., the 336 instances in the current week). We considered this setting as a prediction task, and correspondingly for our method we use the last SVM in the sequence to make predictions. In order to select the free parameter of other methods, we used a simple validation

TABLE II
PREDICTION ACCURACY ON THE ELECTRICITY PRICING DATASET, AVERAGED OVER TIME, FOR ALL METHODS TESTED IN THIS PAPER. THE ROWS LABELED SVM SHOW THE RESULTS OBTAINED WITH STATIONARY SVM

Method	Kernel	Accuracy (%)
SVM	Linear	63.3
GF-SVM		65.1
SW-SVM	Linear	65.3
DWM-SVM		63.3
TA-SVM		65.6
SVM	Gaussian	66.1
GF-SVM		67.2
SW-SVM	Gaussian	67.8
DWM-SVM		66.9
TA-SVM		68.9

scheme. At each step, we set aside the last available week in the training set as a validation set (not the current week, which we want to predict, but the previous one). Once we selected the parameters that are optimal for this validation set, we train again all methods using the complete training set.

As this is a real-world dataset, we do not know in advance what is the real amount of drift in the data. In order to have an estimation of the benefits of using concept drift methods in this case, we also applied a standard SVM using the same procedure as described before (i.e., for each week we determined optimal parameters, trained the SVM, and predicted the current week).

In Table II, we show the corresponding results. In the first rows we show the result obtained using a linear kernel, as in all previous datasets. All adaptive methods outperformed the standard SVM in this case, suggesting the actual presence of some concept drift in the dataset. TA-SVM shows the best performance in this case. For reference, Harries used a decision tree with a sliding window in the same problem, reporting 1-week prediction accuracies for various window sizes between 66% and 67.7%. Looking for a better solution, we repeated the experiment using a Gaussian kernel in this case. All methods improved with the use of nonlinear classifiers. Again, TA-SVM shows the best performance in this dataset.[4]

V. CONCLUSION

In this paper, we presented the TA-SVM, which is a new method for generating adaptive classifiers and capable of learning concepts that change with time. The basic idea of TA-SVM is to use a sequence of classifiers, each one appropriate for a small time window but, in contrast to other proposals, learning all the hyperplanes in a global way. Starting from the solution of independent SVMs, we showed that the addition of a new term in the cost function (which penalizes the diversity between consecutive classifiers) produces in fact a coupling of the sequence. Once coupled, the set of SVMs acts as a single adaptive classifier.

[4]DWM shows a low performance on this setup. Kolter and Maloof [25] applied DWM to this dataset with better results, but using an online learning setting, learning, and predicting one instance at a time (an easier task for this problem), which differs from Harries's methodology.

We evaluated different aspects of the TA-SVM using artificial drifting problems. In particular, we showed that changing the number of classifiers (the n/m ratio) and the coupling constant γ, we can effectively regularize the sequence of classifiers. We compared TA-SVM with other state-of-the-art methods in three different settings: estimation, prediction, and extrapolation, including problems with small datasets, high-dimensional input spaces and noise. TA-SVM showed in all cases to be equivalent to or better than the other methods. Even for the most unfavorable situation for TA-SVM, i.e., the sudden changes of the STAGGER dataset, our new method showed a very good performance. We also applied TA-SVM to a real-world dataset, i.e., Harries's electricity pricing, with very good results.

TA-SVM has two free parameters, m and γ. In our experience, the more efficient way to use them is to fix the n/m ratio in a range of 5 to 10, and then tune γ using an internal cross validation. If the dataset is small or there are indications of high drift levels, one can use $n = m$ to increase the flexibility of the model. The C parameter follows the same rules as in standard SVMs. If there is previous knowledge about noise levels, the C value can be set accordingly. If not, we recommend to begin with a low C value and leave the regularization to the coupling term.

There is nothing in our formulation or the derivation of the dual problem that prevents the use of arbitrary kernel functions to evaluate distances and create nonlinear adaptive classifiers. We already used this possibility in the modeling of the electricity pricing domain. The only potential difficulty can arise in the extrapolation setting. For kernel functions corresponding to finite-dimensional feature spaces, it is always possible, in principle, to use our simple extrapolation. However, this cannot be done if the kernel is associated to an infinite-dimensional feature space.

If needed, there are some simple ways to make TA-SVM scale efficiently to larger problems. For the prediction or extrapolation tasks, the focus is on the performance of the last classifiers in the sequence. In the prediction case, we only use the last classifier to predict the labels of the test samples. For the extrapolation task, we use only a few SVMs from the end of the sequence in order to extrapolate the solution. As the coupling term in (2) involves only interactions to first time neighbors, the influence of old examples to the actual TA-SVM prediction decays exponentially with time. According to this analysis, very old examples quickly become useless to TA-SVM and can be eliminated from the dataset. In practice, for prediction and extrapolation we will pay a reduced cost by limiting TA-SVM to use a fixed number of the more recent examples. In addition, we can easily force the first hyperplane in this reduced sequence to keep the optimal position found in a previous step, which will reduce even more the loss in performance. Going further in the same direction, we can even arrive at a quasi online version of TA-SVM, where only a few hyperplanes are adjusted at each step.

We are currently studying the application of TA-SVM to real problems in slowly drifting systems, in particular to fault prediction in critical mechanical systems. Also, we are evaluating the extension of TA-SVM to one-class classification and regression problems. Finally, we are considering the use of partially overlapping windows for TA-SVM.

Acknowledgment

The authors would like to thank P. F. Verdes and three anonymous reviewers for useful suggestions that considerably improved this manuscript.

Appendix A
Deriving the Dual Problem

First, we introduce the notation we use in this section. We consider the case where we want to adjust a sequence of m hyperplanes to a dataset with n points. We use Greek letters for the indices that vary in the number of hyperplanes and Latin letters for the indices that vary in the number of points.

As we explained in the main text, the hyperplanes are defined by a vector $\mathbf{w}_\mu$ and a scalar b_μ, for $\mu \in \{1, \ldots, m\}$. The hyperplane corresponding to the i-th point is defined by $(\mathbf{w}_{\mu_i}, b_{\mu_i})$. p_μ is the set of points $\{i : \mu_i = \mu\}$. $\mathbf{P}$ is an $m \times n$ matrix defined as $P_{\mu j} = 1$, if $j \in p_\mu$, 0 otherwise.

Also, we use the kernel matrix $\mathbf{K}$, defined by $K_{ij} = y_i y_j \mathbf{x}_i \mathbf{x}_j$. As in conventional SVMs, we can always replace this definition with any other including a useful inner product.

Other required matrix is $\mathbf{Q}$ given by

$$Q_{\mu\nu} = \begin{cases} 1, & \text{if hyperplanes } \nu \text{ and } \mu \text{ are neighbors,} \\ 0, & \text{otherwise} \end{cases}$$

which we symmetrize by $\mathbf{Q}^S = (\mathbf{Q} + \mathbf{Q}^T)/2$.

We also use the notation $\mathbf{P} \odot \mathbf{Q}$ for the entrywise (or Hadamard) matrix product of $\mathbf{P}$ and $\mathbf{Q}$: $(P \odot Q)_{ij} = P_{ij} Q_{ij}$.

We start from the problem

$$\min_{w_\mu, b_\mu} \frac{1}{2m} \sum_{\mu=1}^{m} \left(||\mathbf{w}_\mu||^2 + \frac{\gamma}{2} \sum_{\nu=1}^{m} Q_{\mu\nu} \big(||\mathbf{w}_\mu - \mathbf{w}_\nu||^2 + (b_\mu - b_\nu)^2 \big) \right) + C \sum_{i=1}^{n} \xi_i$$

subject to

$$\begin{aligned} \xi_i &\geq 0 \\ y_i(\mathbf{w}_{\mu_i}\mathbf{x}_i + b_{\mu_i}) - 1 + \xi_i &\geq 0 \end{aligned}$$

where $||\mathbf{w}||^2 = \mathbf{w} \cdot \mathbf{w}$. This is the same problem we introduced in the main text, with small differences that help in the search of the solution.

Given the symmetry of the term including $\mathbf{Q}$, it is easy to rewrite the problem using $\mathbf{Q}^S$

$$\frac{1}{2m} \sum_{\mu=1}^{m} \left(||\mathbf{w}_\mu||^2 + \frac{\gamma}{2} \sum_{\nu=1}^{m} Q^S_{\mu\nu} \big(||\mathbf{w}_\mu - \mathbf{w}_\nu||^2 + (b_\mu - b_\nu)^2 \big) \right) + C \sum_{i=1}^{n} \xi_i.$$

Then, the corresponding Lagrangian is

$$\begin{aligned} L = \frac{1}{2m} \sum_{\mu=1}^{m} &\left(||\mathbf{w}_\mu||^2 + \frac{\gamma}{2} \sum_{\nu=1}^{m} Q^S_{\mu\nu} \big(||\mathbf{w}_\mu - \mathbf{w}_\nu||^2 + (b_\mu - b_\nu)^2 \big) \right) + C \sum_{i=1}^{n} \xi_i \\ &- \sum_{i=1}^{n} \alpha_i \big(y_i(\mathbf{w}_{\mu_i}\mathbf{x}_i + b_{\mu_i}) - 1 + \xi_i \big) \\ &- \sum_{i=1}^{n} \beta_i \xi_i \end{aligned} \tag{3}$$

where $\alpha_i \geq 0$ and $\beta_i \geq 0$.

We have to maximize L with respect to α_i and β_i and minimize it with respect to $\mathbf{w}_i$, b_i and ξ_i. At this point, the derivatives with respect to the primal variables should be zero

$$\frac{\partial L}{\partial \xi_i} = 0, \qquad \frac{\partial L}{\partial \mathbf{w}_\mu} = 0, \qquad \frac{\partial L}{\partial b_\mu} = 0.$$

From these equations, we can eliminate the variables ξ_i, $\mathbf{w}_\mu$, and b_μ from L and obtain the dual problem. We start with the derivative with respect to ξ_i

$$\frac{\partial L}{\partial \xi_i} = 0 = C - \alpha_i - \beta_i$$

which implies that

$$0 \leq \alpha_i \leq C.$$

On the other hand, taking into account that each ξ_i is multiplied by $(C - \alpha_i - \beta_i)$, (3) becomes

$$L = \frac{1}{2m} \sum_{\mu=1}^{m} \left(||\mathbf{w}_\mu||^2 + \frac{\gamma}{2} \sum_{\nu=1}^{m} Q^S_{\mu\nu} \big(||\mathbf{w}_\mu - \mathbf{w}_\nu||^2 + (b_\mu - b_\nu)^2 \big) \right) - \sum_{i=1}^{n} \alpha_i \big(y_i(\mathbf{w}_{\mu_i}\mathbf{x}_i + b_{\mu_i}) - 1 \big). \tag{4}$$

In the case of $\mathbf{w}_\mu$ we have

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{w}_\mu} &= 0 \\ &= \frac{1}{m} \left(\mathbf{w}_\mu + \gamma \sum_\nu Q^S_{\mu\nu} (\mathbf{w}_\mu - \mathbf{w}_\nu) \right) - \sum_{j \in p_\mu} \alpha_j y_j \mathbf{x}_j \end{aligned}$$

which results in

$$\frac{1}{m} \left(\mathbf{w}_\mu + \gamma \sum_\nu Q^S_{\mu\nu} (\mathbf{w}_\mu - \mathbf{w}_\nu) \right) = \sum_{j \in p_\mu} \alpha_j y_j \mathbf{x}_j. \tag{5}$$

Defining the matrix $\mathbf{M}$ as

$$M_{\mu\nu} = \begin{cases} \big(1 + \gamma \sum_\kappa Q^S_{\mu\kappa}\big)/m, & \text{if } \mu = \nu, \\ -\gamma\, Q^S_{\mu\nu}/m, & \text{otherwise} \end{cases}$$

we can write $\mathbf{w}_\mu$ as

$$\mathbf{w}_\mu = \sum_j M^{-1}_{\mu\mu_j} \alpha_j y_j \mathbf{x}_j.$$

Using this, we can rewrite the term $\sum ||\mathbf{w}_\mu||^2$ in (4)

$$\sum_\mu ||\mathbf{w}_\mu||^2 = \sum_\mu \sum_{ij} M^{-1}_{\mu\mu_i} M^{-1}_{\mu\mu_j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j.$$

On the other hand

$$\left(\mathbf{P}^T \mathbf{M}^{-2} \mathbf{P}\right)_{ij} = \sum_\mu M^{-1}_{\mu\mu_i} M^{-1}_{\mu\mu_j}.$$

With this and the definition of $\mathbf{K}$, we have

$$\sum_\mu ||\mathbf{w}_\mu||^2 = \boldsymbol{\alpha}^T \left(\left(\mathbf{P}^T \mathbf{M}^{-2} \mathbf{P}\right) \odot \mathbf{K}\right) \boldsymbol{\alpha}. \tag{6}$$

The $\sum_{\mu\nu} Q^S_{\mu\nu} ||\mathbf{w}_\mu - \mathbf{w}_\nu||^2$ term can be rewritten as

$$\begin{aligned}\sum_{\mu\nu} Q^S_{\mu\nu} ||\mathbf{w}_\mu - \mathbf{w}_\nu||^2 &= \sum_{\mu\nu} Q^S_{\mu\nu} \sum_{ij} \left(M^{-1}_{\mu\mu_i} - M^{-1}_{\nu\nu_i}\right) \\ &\quad \times \left(M^{-1}_{\mu\mu_j} - M^{-1}_{\nu\nu_j}\right) \\ \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j &= \sum_{ij} 2 \Bigg(\sum_\mu M^{-1}_{\mu\mu_i} M^{-1}_{\mu\mu_j} D_{\mu\mu} \\ &\quad - \sum_{\mu\nu} M^{-1}_{\mu\mu_i} M^{-1}_{\nu\mu_j} Q^S_{\mu\nu} \Bigg) \\ &\quad \times \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j \end{aligned}$$

as $\mathbf{Q}^S$ is symmetric and using the $m \times m$ diagonal matrix defined by

$$D_{\mu\nu} = \begin{cases} \sum_\kappa Q^S_{\mu\kappa}, & \text{if } \mu = \nu \\ 0, & \text{otherwise.} \end{cases}$$

Given that

$$\begin{aligned} &\left(\mathbf{P}^T \mathbf{M}^{-1} (\mathbf{D} - \mathbf{Q}^S) \mathbf{M}^{-1} \mathbf{P}\right)_{ij} \\ &\quad = \sum_\mu M^{-1}_{\mu\mu_i} M^{-1}_{\mu\mu_j} D_{\mu\mu} - \sum_{\mu\nu} M^{-1}_{\mu\mu_i} M^{-1}_{\nu\mu_j} Q^S_{\mu\nu} \end{aligned}$$

we can write $\sum_{\mu\nu} Q^S_{\mu\nu} ||\mathbf{w}_\mu - \mathbf{w}_\nu||^2$ as

$$\begin{aligned} &\sum_{\mu\nu} Q^S_{\mu\nu} ||\mathbf{w}_\mu - \mathbf{w}_\nu||^2 \\ &\quad = 2\boldsymbol{\alpha}^T \left(\left(\mathbf{P}^T \mathbf{M}^{-1} (\mathbf{D} - \mathbf{Q}^S) \mathbf{M}^{-1} \mathbf{P}\right) \odot \mathbf{K}\right) \boldsymbol{\alpha}. \end{aligned} \tag{7}$$

Using (6) and (7) we can write (4) as

$$\begin{aligned} L = &\frac{1}{2m} \boldsymbol{\alpha}^T \left(\mathbf{P}^T \mathbf{M}^{-2} \mathbf{P} \odot \mathbf{K}\right) \boldsymbol{\alpha} \\ &+ \frac{\gamma}{2m} \boldsymbol{\alpha}^T \left(\mathbf{P}^T \mathbf{M}^{-1} \left(\mathbf{D} - \mathbf{Q}^S\right) \mathbf{M}^{-1} \mathbf{P} \odot \mathbf{K}\right) \boldsymbol{\alpha} \\ &+ \frac{\gamma}{4m} \sum_{\mu,\nu} Q^S_{\mu\nu} \left(b_\mu - b_\nu\right)^2 - \boldsymbol{\alpha}^T \left(\mathbf{P}^T \mathbf{M}^{-1} \mathbf{P} \odot \mathbf{K}\right) \boldsymbol{\alpha} \\ &+ \sum_i \alpha_i - \sum_i \alpha_i y_i b_{\mu_i}. \end{aligned}$$

The matrices $\mathbf{M}$, $\mathbf{D}$, and $\mathbf{Q}^S$ are related by the equation

$$\mathbf{M} = \frac{\mathbf{I} + \gamma \left(\mathbf{D} - \mathbf{Q}^S\right)}{m}.$$

With this equality, we can simplify the obtained L

$$\begin{aligned} L = &- \frac{1}{2} \boldsymbol{\alpha}^T \left(\mathbf{P}^T \mathbf{M}^{-1} \mathbf{P} \odot \mathbf{K}\right) \boldsymbol{\alpha} \\ &+ \frac{\gamma}{4m} \sum_{\mu,\nu} Q^S_{\mu\nu} \left(b_\mu - b_\nu\right)^2 + \sum_i \alpha_i - \sum_i \alpha_i y_i b_{\mu_i}. \end{aligned} \tag{8}$$

Now we should use the derivatives with respect to b_μ. In the case

$$\frac{\partial L}{\partial b_\mu} = 0 = \frac{\gamma}{m} \sum_{\nu=1}^m Q^S_{\mu\nu} (b_\mu - b_\nu) - \sum_{i \in p_\mu} \alpha_i y_i$$

which gives

$$\frac{\gamma}{m} \left(b_\mu \sum_{\nu=1}^m Q^S_{\mu\nu} - \sum_{\nu=1}^m Q^S_{\mu\nu} b_\nu \right) = \sum_{i \in p_\mu} \alpha_i y_i \tag{9}$$

which, defining $h_i = \alpha_i y_i$, we can write as

$$\frac{\gamma}{m} \left(\mathbf{D} - \mathbf{Q}^S\right) \mathbf{b} = \mathbf{Ph}. \tag{10}$$

Since $(\mathbf{D} - \mathbf{Q}^S)$ is singular, given that $(\mathbf{D} - \mathbf{Q}^S)\overline{1} = \overline{0}$, we can write

$$0 = \frac{\gamma}{n} \overline{0} \mathbf{b} = \frac{\gamma}{n} \overline{1} \left(\mathbf{D} - \mathbf{Q}^S\right) \mathbf{b} = \overline{1} \mathbf{Ph} = \sum_{i=1}^n \alpha_i y_i.$$

In this case, the solution to the system (10) is

$$\mathbf{b} = \frac{m}{\gamma} \left(\mathbf{D} - \mathbf{Q}^S\right)^+ \mathbf{Ph} \tag{11}$$

[where $(\mathbf{D} - \mathbf{Q}^S)^+$ is the pseudoinverse of $(\mathbf{D} - \mathbf{Q}^S)$]. We still need to eliminate the b_μ from L. The part that depends on $\mathbf{b}$ is

$$\begin{aligned} &\frac{\gamma}{4m} \sum_{\mu,\nu} Q^S_{\mu\nu} \left(b_\mu - b_\nu\right)^2 - \sum_i \alpha_i y_i b_{\mu_i} \\ &\qquad = \frac{\gamma}{4m} \sum_{\mu,\nu} Q^S_{\mu\nu} \left(b_\mu - b_\nu\right)^2 - \mathbf{b}^T \mathbf{Ph}. \end{aligned} \tag{12}$$

We can write this as

$$\begin{aligned} &\frac{\gamma}{4m} \sum_{\mu,\nu} Q^S_{\mu\nu} (b_\mu - b_\nu)^2 - \mathbf{b}^T \mathbf{Ph} \\ &\quad = \frac{\gamma}{4m} \sum_{\mu,\nu} Q^S_{\mu\nu} \left(b_\mu^2 - 2 b_\mu b_\nu + b_\nu^2\right) - \mathbf{b}^T \mathbf{Ph} \\ &\quad = \frac{\gamma}{2m} \left(\sum_{\mu,\nu} b_\mu^2 Q^S_{\mu\nu} - \mathbf{b}^T \mathbf{Q}^S \mathbf{b} \right) - \mathbf{b}^T \mathbf{Ph} \\ &\quad = \frac{\gamma}{2m} \left(\mathbf{b}^T \left(\mathbf{D} - \mathbf{Q}^S\right) \mathbf{b} \right) - \mathbf{b}^T \mathbf{Ph} \\ &\quad = \frac{\mathbf{b}^T \mathbf{Ph}}{2} - \mathbf{b}^T \mathbf{Ph} \\ &\quad = -\frac{\mathbf{b}^T \mathbf{Ph}}{2} \\ &\quad = -\frac{m}{2\gamma} \mathbf{h}^T \mathbf{P}^T \left(\mathbf{D} - \mathbf{Q}^S\right)^+ \mathbf{Ph} \\ &\quad = -\frac{m}{2\gamma} \boldsymbol{\alpha}^T \left(\left(\mathbf{P}^T \left(\mathbf{D} - \mathbf{Q}^S\right)^+ \mathbf{P} \right) \odot \mathbf{Y} \right) \boldsymbol{\alpha} \end{aligned}$$

where $\mathbf{Y} = \mathbf{y}\mathbf{y}^T$. Taking into account the last equality, L becomes

$$L = -\frac{1}{2}\boldsymbol{\alpha}^T\Big(\big(\mathbf{P}^T\mathbf{M}^{-1}\mathbf{P}\big)\odot\mathbf{K} + \big(\mathbf{P}^T(\mathbf{M}-\mathbf{I})^{+}\mathbf{P}\big)\odot\mathbf{Y}\Big)\boldsymbol{\alpha} + \sum_i \alpha_i$$

which has the form

$$L = -\frac{1}{2}\boldsymbol{\alpha}^T\mathbf{R}\boldsymbol{\alpha} + \sum_i \alpha_i$$

with the matrix $\mathbf{R}$ defined accordingly. Finally, the dual problem is

$$\max_{\boldsymbol{\alpha}} -\frac{1}{2}\boldsymbol{\alpha}^T\mathbf{R}\boldsymbol{\alpha} + \sum_i \alpha_i \tag{13}$$

subject to

$$0 \le \alpha_i \le C$$
$$\sum \alpha_i y_i = 0$$

which is the same quadratic minimization problem with restrictions solved in SVM (with a different matrix $\mathbf{R}$). In consequence, any technique employed to solve the conventional SVM problem can be used here, as, for example, SMO [43].

Appendix B
Complexity Evaluation

As follows from (13), the complexity of the whole problem is given by the computation of the matrix $\mathbf{R}$ and the solution of the optimization problem. As we mentioned before, this last step is equivalent to a conventional SVM optimization problem, which is $O(n^2)$.

The computation of $\mathbf{R}$ involves the inversion of $\mathbf{M}$ and the computation of the pseudoinverse of $(\mathbf{M}-\mathbf{I})$. The general solutions of these problems are costly but in our case, given that we consider only interactions to first time neighbors, both problems can be solved analytically.

After this, the computation of $\mathbf{P}^T\mathbf{M}^{-1}\mathbf{P}$ and $\mathbf{P}^T(\mathbf{M}-\mathbf{I})^{+}\mathbf{P}$ is trivial, given that

$$(\mathbf{P}^T\mathbf{M}^{-1}\mathbf{P})_{ij} = M^{-1}_{\mu_i\mu_j}$$
$$(\mathbf{P}^T(\mathbf{M}-\mathbf{I})^{+}\mathbf{P})_{ij} = (\mathbf{M}-\mathbf{I})^{+}_{\mu_i\mu_j}.$$

Hence, the computation of each element of the Hadamard product is $O(1)$, which means that the computation of $\mathbf{R}$ is $O(n^2)$, i.e., no greater than the optimization step.

References

[1] J. C. Schlimmer and R. H. Granger, "Beyond incremental processing: Tracking concept drift," in *Proc. 5th Nat. Conf. Artif. Intell.*, Irvine, CA, 1986, pp. 502–507.

[2] R. Klinkenberg and T. Joachims, "Detecting concept drift with support vector machines," in *Proc. 17th Int. Conf. Mach. Learn.*, San Mateo, CA, 2000, pp. 487–494.

[3] R. Vicente, O. Kinouchi, and N. Caticha, "Statistical mechanics of online learning of drifting concepts: A variational approach," *Mach. Learn.*, vol. 32, no. 2, pp. 179–201, Aug. 1998.

[4] P. L. Bartlett, S. Ben-Dabid, and S. R. Kulkarni, "Learning changing concepts by exploiting the structure of change," *Mach. Learn.*, vol. 41, no. 2, pp. 153–174, Nov. 2000.

[5] K. Stanley, "Learning concept drift with a committee of decision trees," Dept. Comput. Sci., Univ. Texas, Austin, Tech. Rep. UTAI-TR-03-302, 2003.

[6] A. Tsymbal, "The problem of concept drift: Definitions and related work," Dept. Comput. Sci., Trinity College Dublin, Dublin, Ireland, Tech. Rep. TCD-CS-2004-15, Apr. 2004.

[7] T. Mitchell, R. Caruana, D. Freitag, J. McDermott, and D. Zabowski, "Experience with a learning personal assistant," *Commun. ACM*, vol. 37, no. 7, pp. 81–91, 1994.

[8] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Mach. Learn.*, vol. 23, no. 1, pp. 69–101, Apr. 1996.

[9] M. Salganicoff, "Tolerating concept and sampling shift in lazy learning using prediction error context switching," *Artif. Intell. Rev.*, vol. 11, nos. 1–5, pp. 133–155, Feb. 1997.

[10] W. N. Street and Y. Kim, "A streaming ensemble algorithm (SEA) for large-scale classification," in *Proc. 7th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*. San Francisco, CA, 2001, pp. 377–382.

[11] S. H. Bach and M. A. Maloof, "Paired learners for concept drift," in *Proc. IEEE Int. Conf. Data Mining*, Los Alamitos, CA, 2008, pp. 23–32.

[12] C. Alippi and M. Roveri, "Just-in-time adaptive classifiers—part I: Detecting nonstationary changes," *IEEE Trans. Neural Netw.*, vol. 19, no. 7, pp. 1145–1153, Jul. 2008.

[13] C. Alippi and M. Roveri, "Just-in-time adaptive classifiers—part II: Designing the classifier," *IEEE Trans. Neural Netw.*, vol. 19, no. 12, pp. 2053–2064, Dec. 2008.

[14] D. P. Helmbold and P. M. Long, "Tracking drifting concepts by minimizing disagreements," *Mach. Learn.*, vol. 14, no. 1, pp. 27–45, Jan. 1994.

[15] P. L. Bartlett, "Learning with a slowly changing distribution," in *Proc. 5th Annu. Workshop Comput. Learn. Theory*, Pittsburgh, PA, 1992, pp. 243–252.

[16] R. D. Barve and P. M. Long, "On the complexity of learning from drifting distributions," in *Proc. 9th Annu. Workshop Comput. Learn. Theory*, San Mateo, CA, 1996, pp. 170–193.

[17] Y. Freund and Y. Mansour, "Learning under persistent drift," in *Proc. 3rd Eur. Conf. Comput. Learn. Theory*, London, U.K., 1997, pp. 109–118.

[18] C. Alippi, G. Boracchi, and M. Roveri, "Just in time classifiers: Managing the slow drift case," in *Proc. Int. Joint Conf. Neural Netw.*, Atlanta, GA, 2009, pp. 114–120.

[19] R. Klinkenberg and I. Renz, "Adaptive information filtering: Learning in the presence of concept drifts," in *Workshop Notes of the ICML/AAAI Workshop Learning for Text Categorization*. Menlo Park, CA: AAAI Press, 1998, pp. 33–40.

[20] C. Lanquillon, "Enhancing test classification to improve information filtering," Ph.D. thesis, Faculty Comp. Sci., Univ. Magdeburg, Magdeburg, Germany, 2001.

[21] K. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf, "An introduction to kernel-based learning algorithms," *IEEE Trans. Neural Netw.*, vol. 12, no. 2, pp. 181–201, Mar. 2001.

[22] V. Vapnik, "An overview of statistical learning theory," *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 988–999, Sep. 1999.

[23] G. L. Grinblat, P. M. Granitto, and H. A. Ceccatto, "Time-adaptive support vector machines," *Inteligencia Artif.*, vol. 12, no. 40, pp. 39–50, 2008.

[24] M. Karnick, M. Ahiskali, M. D. Muhlbaier, and R. Polikar, "Learning concept drift in nonstationary environments using an ensemble of classifiers based approach," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Hong Kong, China, Jun. 2008, pp. 3455–3462.

[25] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: An ensemble method for drifting concepts," *J. Mach. Learn. Res.*, vol. 8, pp. 2755–2790, Dec. 2007.

[26] J. C. Schlimmer and R. H. Granger, Jr., "Incremental learning from noisy data," *Mach. Learn.*, vol. 1, no. 3, pp. 317–354, 1986.

[27] G. Castillo, J. Gama, and P. Medas, "Adaptation to drifting concepts," in *Proc. Progress Artif. Intell., 11th Portuguese Conf. Artif1. Intell. (EPIA)*, LNCS 2902. Beja, Portugal, 2003, pp. 279–293.

[28] R. Klinkenberg, "Learning drifting concepts: Example selection versus example weighting," *Intell. Data Anal.*, vol. 8, no. 3, pp. 281–300, Aug. 2004.

[29] T. Joachims, "Estimating the generalization performance of a SVM efficiently," in *Proc. 17th Int. Conf. Mach. Learn.*, San Francisco, CA, 2000, pp. 431–438.

[30] I. Koychev and R. Lothian, "Tracking drifting concepts by time window optimization," in *Proc. 25th SGAI Int. Conf. Innov. Tech. Appl. Artif. Intell.*, New York, 2005, pp. 46–59.

[31] C. Alippi and M. Roveri, "Just-in-time adaptive classifiers in non-stationary conditions," in *Proc. Int. Joint Conf. Neural Netw.*, Orlando, FL, 2007, pp. 1014–1019.

[32] I. Koychev, "Tracking changing user interests through prior-learning of context," in *Adaptive Hypermedia*, LNCS 2347. New York: Springer-Verlag, 2002, pp. 223–232.

[33] M. Maloof and R. Michalski, "Selecting examples for partial memory learning," *Mach. Learn.*, vol. 41, no. 1, pp. 27–52, Oct. 2000.

[34] I. Koychev, "Gradual forgetting for adaptation to concept drift," in *Proc. ECAI Workshop Current Issues Spatio-Temporal Reason.*, Berlin, Germany, 2000, pp. 101–106.

[35] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proc. 9th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Washington D.C., 2003, pp. 226–235.

[36] J. Gao, B. Ding, J. Han, W. Fan, and P. Yu, "Classifying data streams with skewed class distributions and concept drifts," *IEEE Internet Comput.*, vol. 12, no. 6, pp. 37–49, Nov.–Dec. 2008.

[37] S. Hashemi, Y. Yang, Z. Mirzamomen, and M. Kangavari, "Adapted one-versus-all decision trees for data stream classification," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 5, pp. 624–637, May 2009.

[38] R. Elwell and R. Polikar, "Incremental learning in nonstationary environments with controlled forgetting," in *Proc. Int. Joint Conf. Neural Netw.*, Atlanta, GA, 2009, pp. 771–778.

[39] M. D. Muhlbaier, A. Topalis, and R. Polikar, "Learn ++.NC: Combining ensemble of classifiers with dynamically weighted consult-and-vote for efficient incremental learning of new classes," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 152–168, Jan. 2009.

[40] Z. Kolter and M. Maloof, "Dynamic weighted majority: A new ensemble method for tracking concept drift," in *Proc. 3rd IEEE Int. Conf. Data Mining*, Nov. 2003, pp. 123–130.

[41] N. Littlestone and M. K. Warmuth, "The weighted majority algorithm," *Inform. Comput.*, vol. 108, no. 2, pp. 212–261, Feb. 1994.

[42] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge, U.K.: Cambridge Univ. Press, 2000.

[43] J. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods-Support Vector Learning*. Cambridge, MA: MIT Press, 2000, pp. 185–208.

[44] N. Littlestone, "Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm," *Mach. Learn.*, vol. 2, no. 4, pp. 285–318, Apr. 1987.

[45] S. Ferrari and M. Jensenius, "A constrained optimization approach to preserving prior knowledge during incremental training," *IEEE Trans. Neural Netw.*, vol. 19, no. 6, pp. 996–1009, Jun. 2008.

[46] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-agressive algorithms," *J. Mach. Learn. Res.*, vol. 7, pp. 551–585, Dec. 2006.

[47] J. Kivinen and M. Warmuth, "Exponentiated gradient versus gradient descent for linear predictors," *Inform. Comput.*, vol. 132, no. 1, pp. 1–63, Jan. 1997.

[48] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2165–2176, Aug. 2004.

[49] M. Herbster and M. K. Warmuth, "Tracking the best linear predictor," *J. Mach. Learn. Res.*, vol. 1, pp. 281–309, Sep. 2001.

[50] R. Caruana, "Multi-task learning," *Mach. Learn.*, vol. 28, no. 1, pp. 41–75, Jul. 1997.

[51] S. Thrun and L. Pratt, *Learning to Learn*. Norwell, MA: Kluwer, 1997.

[52] T. Evgeniou, C. M. Micchelli, and M. Pontil, "Learning multiple tasks with kernel methods," *J. Mach. Learn. Res.*, vol. 6, pp. 615–637, Dec. 2005.

[53] W. Fan, "Systematic data selection to mine concept-drifting data streams," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Seattle, WA, 2004, pp. 128–137.

[54] A. Tsymbal, M. Pechenizkiy, P. Cunningham, and S. Puuronen, "Dynamic integration of classifiers for handling concept drift," *Inform. Fusion*, vol. 9, no. 1, pp. 56–68, Jan. 2008.

[55] M. Harries, "Splice-2 comparative evaluation: Electricity pricing," School Comput. Sci. & Eng., Univ. New South Wales, Sydney, Australia, Tech. Rep. NSW-CSE-TR-9905, 1999.

Guillermo L. Grinblat was born in Miramar, Buenos Aires, Argentina, in 1976. He received the Licenciate degree in computer sciences from the National University of Rosario, Rosario, Argentina, in 2006.

He currently holds a Fellowship at the French Argentine International Center for Information and Systems Sciences, Rosario. He is also a Teaching Assistant at the National University of Rosario. His current research interests include drifting problems, kernel methods, and deep architectures.

Lucas C. Uzal was born in Pergamino, Argentina, in 1982. He received the Licentiate and M.Sc. degrees in physics from Balseiro Institute, San Carlos de Bariloche, Argentina, in 2005 and 2006, respectively.

He has been with the French Argentine International Center for Information and Systems Sciences, Rosario, Argentina, since 2007, on a research grant from Consejo Nacional de Investigaciones Cientficas y Tecnolgicas, Rosario. His current research interests include complex systems and time series analysis.

H. Alejandro Ceccatto was born in Argentina in 1953. He received the M.Sc. degree in physics from the Universidad Nacional de Rosario, Rosario, Argentina, in 1979, and the Ph.D. degree in physics from the Universidad Nacional de La Plata, La Plata, Argentina, in 1985.

He was a Post-Doctoral Fellow at the Department of Applied Physics, Stanford University, Stanford, CA, in 1988, and at the Institut fuer Theoretische Physik, Universitaet zu Koeln, Cologne, Germany, in 1989. Since 1995, he has been the Director of the Intelligent Systems Group, Instituto de Fisica Rosario, Rosario. He is currently a full Professor at the Universidad Nacional de Rosario, and Director of the French Argentine International Center for Information and Systems Sciences, Rosario. He has supervised 20 M.Sc. and 11 Ph.D. theses.

Pablo M. Granitto was born in Rosario, Argentina, in 1970. He received the Degree in physics, and the Ph.D. degree, also in physics, in 1997 and 2003, respectively, both from Universidad Nacional de Rosario (UNR), Rosario, Argentina.

He was a Post-Doctoral Researcher at Istituto Agrario San Michele all'Addige, Trento, Italy. Since 2006, he has been a full-time Researcher at Consejo Nacional de Investigaciones Cientficas y Tecnolgicas, Rosario, and UNR. He leads the Machine Learning Group at the French Argentine International Center for Information and Systems Sciences, Rosario. His current research interests include application of modern machine learning techniques to agro-industrial and biological problems, involving feature selection, clustering, and ensemble methods.