ARTÍCULO

Time–Adaptive Support Vector Machines

Guillermo Grinblat, Pablo M. Granitto, Alejandro Ceccatto

Centro Internacional Franco-Argentino de Ciencias de la Información
y de Sistemas (CIFASIS) - Universidad de Marsella/UNR/CONICET
Bvd. 27 de Febrero 210 Bis
Rosario 2000 (República Argentina)
{grinblat, granitto, ceccatto}@cifasis-conicet.gov.ar

Abstract

In this work we propose an adaptive classification method able both to learn and to follow the temporal evolution of a drifting concept. With that purpose we introduce a modified SVM classifier, created using multiple hyperplanes valid only at small temporal intervals (windows). In contrast to other strategies proposed in the literature, our method learns all hyperplanes in a global way, minimizing a cost function that evaluates the error committed by this family of local classifiers plus a measure associated to the VC dimension of the family. We also show how the idea of slowly changing classifiers can be applied to non-linear stationary concepts with results similar to those obtained with normal SVMs using gaussian kernels.

Keywords: Adaptive methods, Support Vector Machine, Drifting concepts.

1 Introduction

Many classification problems associated with real world systems vary over time. For example, a system may change because of physical reasons as the season of the year, or because there is a change in the expectations or interests of its users. In most cases the cause and characteristics of the change are not obviously present in the data under analysis. In these situations, the classifier needs to learn not only the correct input-output function but also of to detect the change in the concept and to adapt to it.

Usually this problem is tackled using a temporal window of a given length and assuming that the change in the concept that must be learned is negligible in that period of time [1]. If the window is too big, such assumption is in general not valid and the responce 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 looses accuracy because it must learn the input-output relationship from only a few examples. As a potential solution, there are algorithms that use an adaptive window size [1], where the data is divided in small batches and the optimal number of batches is used. Even then, the concept must be stationary in all batches. Vicente et al. [2] introduced a different, purely statistical approach, to the learning of evolutive concepts. In a recent work, Kolter and Maloof [4] discussed the use of ensemble methods for drifting concepts, including a lengthly review of the state of the art in the field.

Bartlett et al. [3] proposed to learn a sequence of functions with some restrictions on how diverse those functions can be. In this work we present a new approach to this problem, the use of succesive classifiers that vary following the concept

change, but they are all fitted in a global way. In our method, the interval of validity of each classifier can be as small as needed (even only one point), but the classifiers are trained to minimize a global measure of the error instead of adjusting them locally. To build the sequence of classifiers we selected one of the most powerfull methods nowadays, the Support Vector Machines [5, 6] (SVM), which we adapted accordingly.

An interesting by-product of our method is the possibility of applying it on non–linear stationary classification problems. In that case, the idea described before can be applied simply by replacing the temporal windows by a "neighbourhood window" in the input space. In this way, the SVM classifier can describe complex decision functions (not only simple hyperplanes) without relying on kernels [5]. We show, through a simple example, that with our implementation of adaptive SVM it is possible to achieve results similar to those of a normal SVM with a gaussian kernel.

The rest of this paper is organized as follows. In Section 2 we introduce and justify the new minimization problem we need to solve and in section 3, we derive the corresponding dual problem. Then, in Section 4 we present some preliminary experimental results and, finally, Section 5 closes the work with some conclusions.

2 The new minimization problem

Suppose we have a data set $[(x_1, y_1) \ldots, (x_n, y_n)]$, where the pair (x_i, y_i) was obtained at time i, x_i is a vector in a given vectorial space and $y_i = \pm 1$. In this setting we define:

Family F of classifiers: is composed by a set of hyperplanes that can change with time. That is, using this family, a point readed at a given time t is classified according only to the hyperplane corresponding to that t period of time. The Vapnik-Chervonenkis dimension (VC) [6] of this family is $VC(F) = \infty$. This follows from the fact that the hyperplane can be changed enough from time $i-1$ to time i as to be able to classify correctly the point x_i, no matter where it is.

Family F_v of classifiers: is composed by the classifiers $f \in F$ for which the change between an hyperplane at a given time and the next one is bounded by v. To be more precise, if the hyperplane at time i is defined by $w_i \cdot x_i + b_i$, then

$$f \in F_v \Leftrightarrow (w_{i-1} - w_i)^2 + (b_{i-1} - b_i)^2 \leq v^2 \quad \forall i.$$

It is easy to see that given v, v' such that $v \leq v'$, we have $F_v \subseteq F_{v'}$, which means that $VC(F_v) \leq VC(F_{v'})$. On the other hand, if the input space has d dimensions, $VC(F_{v=0}) = d+1$, since F_0 is a set of hyperplanes that do not change in time. It follows that the VC dimension of F_v grows with v, starting from $d+1$.

According to this, we can control the complexity of the set of classifiers f just limiting them to be in F_v for some small v. Other option, that makes use of the regularization theory, is to search in F the set of classifiers that minimizes

$$Err(f, x, y) + C \text{ comp}(f)$$

where

$$Err(f, x, y) = \sum_i \max[0, 1 - y_i(w_i x_i + b_i)],$$

is the empirical error, $\text{comp}(f)$ is a given measure of the complexity of the set of n classifiers and C is a constant that defines the relative importance of the complexity with respect to the errors.

If we change slightly the definition of F_v to be the family of hyperplanes that change less than v *on average* (instead of bounded by v), we can use a simple measure of complexity given by

$$\text{comp}(f) = \frac{1}{n} \sum_i (w_{i-1} - w_i)^2 + (b_{i-1} - b_i)^2$$

Following the typical reasoning in Structural Risk Minimization [6], the VC dimension should drop when the average margin of all hyperplanes grows. According to this, we can define an improved measure of complexity:

$$\begin{aligned} \text{comp}(f) &= \frac{1}{n} \sum_i w_i^2 + \\ &\quad + \frac{\gamma}{n} \sum_i (w_{i-1} - w_i)^2 + (b_{i-1} - b_i)^2 \end{aligned}$$

where γ can be used to change the relative importance of each term.

Finally we get the problem

$$\begin{aligned}\min \quad & \sum_i \max(0, 1 - y_i(w_i x_i + b_i)) + \\ & + C[\frac{1}{n}\sum_i w_i^2 + \frac{\gamma}{n}\sum_i (w_{i-1} - w_i)^2 + \\ & + (b_{i-1} - b_i)^2],\end{aligned}$$

which can be show to be equivalent to

$$\begin{aligned}\min \quad & \sum_i \xi_i + \\ & + C[\frac{1}{n}\sum_i w_i^2 + \frac{\gamma}{n}\sum_i (w_{i-1} - w_i)^2 + \\ & + (b_{i-1} - b_i)^2],\end{aligned}$$

subject to

$$\begin{aligned}\xi_i &\geq 0 \\ y_i(w_i x_i + b_i) - 1 + \xi_i &\geq 0\ .\end{aligned}$$

The solution of this problem is a set of SVMs that evolve in time, which we will call Time–Adaptive Support Vector Machines (TA–SVM).

3 Solution: deriving the dual

In this section, following the typical strategy in the SVM literature, we will derive the dual of the minimization problem we have recently introduced. Even if our problem is slightly more complicated than the one usually faced, the strategy to solve it remains the same, involving lagrange multipliers and derivatives.

From here on we will call V_i the set of neighbour points of x_i and N_i its cardinality. Also we will use the following notation. Given a matrix A, A_{i*} will be its i^{th} row and A_{*j} its j^{th} column. $P \odot Q$ will be the element-wise product of the matrices P and Q. That is, $(P \odot Q)_{ij} = P_{ij}Q_{ij}$.

We start from the problem

$$\begin{aligned}\min_{w_i, b_i} \quad & \frac{1}{2n}\sum_{i=1}^{n}[||w_i||^2 + \frac{\gamma}{2}\sum_{j\in V_i} ||w_i - w_j||^2 + \\ & + (b_i - b_j)^2] + C\sum_i \xi_i,\end{aligned}$$

with $||w|| = w \cdot w$, subject to

$$\begin{aligned}\xi_i &\geq 0 \\ y_i(w_i x_i + b_i) - 1 + \xi_i &\geq 0.\end{aligned}$$

Note that we have used the common notation in SVM, where the parameter C multiplies the errors instead of the complexity of the solution.

The corresponding lagrangean is

$$\begin{aligned}L \quad = \quad & \frac{1}{2n}\sum_{i=1}^{n}[||w_i||^2 + \frac{\gamma}{2}\sum_{j\in V_i} ||w_i - w_j||^2 + \\ & + (b_i - b_j)^2] + C\sum_i \xi_i \\ & - \sum_i \alpha_i(y_i(w_i x_i + b_i) - 1 + \xi_i) - \sum_i \beta_i \xi_i,\end{aligned} \tag{1}$$

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

We want to maximize L with respect to α_i and β_i, and minimize it with respect to w_i, b_i and ξ_i. At this point, the derivatives with respect to the primal variables should be zero. The equation

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

implies that

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

On the other hand, taking in account that each x_i is multiplied by $(C - \alpha_i - \beta_i)$, Eq. (1) becomes

$$\begin{aligned}L \quad = \quad & \frac{1}{2n}\sum_{i=1}^{n}[||w_i||^2 + \frac{\gamma}{2}\sum_{j\in V_i} ||w_i - w_j||^2 + \\ & + (b_i - b_j)^2] - \\ & - \sum_i \alpha_i(y_i(w_i x_i + b_i) - 1).\end{aligned} \tag{2}$$

In the case of the w_i we have

$$\frac{\partial L}{\partial w_i} = 0 = \frac{1}{n}\left(w_i + \gamma\sum_{j\in V_i}(w_i - w_j)\right) - \alpha_i y_i x_i,$$

where we have considered that if x_j is a neighbour of x_i then x_i is a neighbour of x_j. This results in

$$\frac{1}{n}\left((1 + \gamma N_i)w_i - \gamma\sum_{j\in V_i} w_j\right) = \alpha_i y_i x_i. \tag{3}$$

If we define the matrix M and vector z as

$$M_{ij} = \begin{cases} (1 + \gamma N_i)/n & \text{if } i = j \\ -\gamma/n & \text{if } j \in V_i \\ 0 & \text{in other case.}\end{cases}$$

$$z_i = \alpha_i y_i x_i$$

we can write (3) in the matrix form $Mw = z$, or $w_i = (M^{-1})_{i*}z$. Replacing in (2) we have

$$\begin{aligned} L &= \frac{1}{2n}\alpha^T((M^{-1})^2 \odot K)\alpha + \\ &+\frac{\gamma}{4n}\alpha^T((M^{-1}QM^{-1}) \odot K)\alpha + \\ &+\frac{\gamma}{4n}\sum_i \sum_{j\in V_i}(b_i - b_j)^2 \\ &-\alpha^T((M^{-1}) \odot K)\alpha + \sum_i \alpha_i - \sum_i \alpha_i y_i b_i. \end{aligned} \quad (4)$$

where we have defined two matrices $N \times N$, $K_{ij} = y_i y_j x_i x_j$ and Q given by

$$Q_{ij} = \begin{cases} N_i & \text{if } i = j \\ -1 & \text{if } i \text{ is a neighbour of } j \\ 0 & \text{in other case} \end{cases}$$

In the case of the b_i,

$$\frac{\partial L}{\partial b_i} = 0 = \frac{\gamma}{n}\sum_{j\in V_i}(b_i - b_j) - \alpha_i y_i,$$

which gives

$$\frac{\gamma N_i}{n} b_i - \frac{\gamma}{n}\sum_{j\in V_i} b_j = \alpha_i y_i. \quad (5)$$

Defining h as:

$$h = \begin{pmatrix} \alpha_1 y_1 \\ \vdots \\ \alpha_n y_n \end{pmatrix}$$

we can write (5) as

$$\frac{\gamma}{n} Qb = h. \quad (6)$$

Since Q is singular,

$$Q\overline{1} = \begin{pmatrix} N_1 - \sum_{j\in V_i} 1 \\ \vdots \\ N_n - \sum_{j\in V_n} 1 \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$

so

$$0 = \frac{\gamma}{n}\overline{0}b = \frac{\gamma}{n}\overline{1}Qb = \overline{1}h = \sum_i \alpha_i y_i.$$

It is possible to eliminate b from (4). The part that depends on b is

$$\begin{aligned} &\frac{\gamma}{4n}\sum_i \sum_{j\in V_i}(b_i - b_j)^2 - \sum_i \alpha_i y_i b_i = \\ &\quad = \frac{\gamma}{4n}\sum_i \sum_{j\in V_i}(b_i - b_j)^2 - h^T b. \end{aligned}$$

On the other hand,

$$b^T Q b = \sum_i \sum_{j\in V_i}(b_i - b_j)b_i.$$

which gives

$$\frac{\gamma}{4n}\sum_i \sum_{j\in V_i}(b_i - b_j)^2 - h^T b = \frac{\gamma}{2n} b^T Q b - h^T b.$$

From (6) we know that

$$b^T Q = \frac{nh^T}{\gamma},$$

resulting in

$$\frac{\gamma}{2n}\frac{nh^T}{\gamma} b - h^T b = -\frac{h^T b}{2}.$$

Diagonalizing Q we have $Q = PDP^T$, where $P^{-1} = P^T$ since Q is symmetric, so we can obtain from (6)

$$DP^T b = P^T \frac{nh}{\gamma}$$

If we define $b' = P^T b$ and $h' = P^T h$, the last equation can be written as

$$Db' = \frac{nh'}{\gamma}.$$

As D is diagonal, it's easy to find the b'_i for which $D_{ii} = \lambda_i \neq 0$:

$$b'_i = \frac{nh'}{\gamma\lambda_i}, \text{ if } \lambda_i \neq 0.$$

If the eigenvalue $\lambda_i = 0$, from (6) we obtain

$$\begin{aligned} P^T_{i*} Qb &= P^T_{i*}\frac{nh}{\gamma} \\ \overline{0}b &= P^T_{i*}\frac{nh}{\gamma} \\ 0 &= \frac{nh'_i}{\gamma}. \\ 0 &= h'_i \end{aligned}$$

That is, when we cannot determine b'_i with the last equation, $h'_i = 0$.

Let's go back to what we wanted to obtain:

$$\begin{aligned} -\frac{h^T b}{2} &= -\frac{h^t P P^T b}{2} \\ &= -\frac{1}{2} h'^T b' \\ &= -\frac{1}{2}\sum_i h'_i b'_i \\ &= -\frac{1}{2}\sum_i \frac{n}{\gamma}\Lambda_i h'^2_i \end{aligned}$$

where $\Lambda_i = \frac{1}{\lambda_i}$ if $\lambda_i \neq 0$ and $\Lambda_i = 0$ if $\lambda_i = 0$. If we define D' as the diagonal matrix $D'_{ii} = \Lambda_i$ and Y given by $Y_{ij} = y_i y_j$, the last equality is

$$\begin{aligned} -\frac{h^T b}{2} &= -\frac{n}{2\gamma} h^T P D' P^T h \\ &= -\frac{n}{2\gamma} \alpha^T ((P D' P^T) \odot Y)\alpha. \end{aligned}$$

Replacing in (4) we have

$$\begin{aligned} L &= \frac{1}{2n}\alpha^T((M^{-1})^2 \odot K)\alpha + \\ &\quad \frac{\gamma}{4n}\alpha^T((M^{-1}QM^{-1}) \odot K)rr\alpha - \\ &\quad \alpha^T((M^{-1}) \odot K)\alpha + \sum_i \alpha_i - \\ &\quad -\frac{n}{2\gamma}\alpha^T((PD'P^T) \odot Y)\alpha \end{aligned}$$

That is

$$\begin{aligned} L &= \alpha^T[(\frac{1}{2n}M^{-2} + \frac{\gamma}{4n}M^{-1}QM^{-1} - M^{-1}) \odot K - \\ &\quad -\frac{n}{2\gamma}(PD'P^T) \odot Y]\alpha + \sum_i \alpha_i, \end{aligned} \tag{7}$$

which has the form

$$L = \alpha^T R \alpha + \sum_i \alpha_i$$

with the matrix R defined accordingly.

Finally, the dual problem is

$$\max_\alpha \alpha^T R \alpha + \sum_i \alpha_i,$$

subject to

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

which is the same quadratic minimization problem with restrictions solved in SVM (with a different matrix R). In consequence, any technique employed to solve the conventional SVM problem can be used here, as, for example, SMO [5].

4 Experimantal Results

4.1 Artificial datesets

We first applied the new TA–SVM to three artificial datasets, designed to evaluate different aspects of the method.

The first two examples point to check the time evolution of the classifier for different values of the γ parameter and to compare it with the solution found by a standard SVM with a linear kernel (which does not take into account the time at which each sample was collected). The γ parameter regulates how diverse the differenct classifiers in the TA–SVM can be. High γ values force the classifiers to remain stationary (see Eq. 1, for example).

Example 1: We generated a dataset with 500 points sampled from a normal distribution in $\Re^2$, divided in two classes. Class 1 contains 250 points centered in $(0, 1)$ with a standard deviation of 0.1 for each dimension. The other 250 points, corresponding to class -1, were equally generated but centered in $(0, -1)$. To simulate a drift in time, we added an angle of $\frac{i\pi}{500}$ radians to each point x_i, represented in polar coordinates. The resulting points (again in rectangular coordinates) were presented to both (TA–SVM and SVM) algorithms. Figure 1 shows the resulting (final) dataset. When time evolves, both classes drift along the unitary circle in counterclockwise sense.

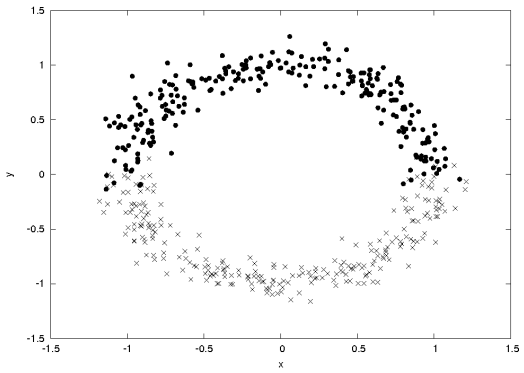

Figure 1: The dataset used in the Example 1

In order to force the TA–SVM to search for a temporal evolution, only the points x_{i-1} and x_{i+1} were considered as neighbours of x_i (with the exception of both extremes). That is, we used a matrix Q defined as

$$\begin{aligned} Q_{11} = Q_{nn} &= 1 \\ Q_{ii} &= 2 \quad \text{for } i \neq 1 \text{ and } i \neq n \\ Q_{i,i+1} = Q_{i,i-1} &= -1 \\ Q_{ij} &= 0 \quad \text{in other case.} \end{aligned}$$

The parameter C in TA–SVM was fixed to 1 while γ was changing. According to this, we also used $C = 1$ for the SVM parameter.

Figure 2 shows the time evolution of the classifier for $\gamma \in \{10, 10^3, 10^6, 10^8\}$. The lines in the Figure join the extreme of each vector w_i with the corresponding extremes of w_{i+1} and w_{i-1}; i.e. they are the curve $w(t)$. For TA–SVM, the training errors for these four cases were 0, 0, 0, and 2.6%, respectively. Class 1 points are plotted for comparison. The Figure shows that the classifier evolves, following the data, for low γ, and tends to the solution found by standard SVM as γ grows. The training error is 0, except for high values of γ where the TA–SVM is forced to remain almost constant (it is worth mentioning that the Bayes error for this example is zero).

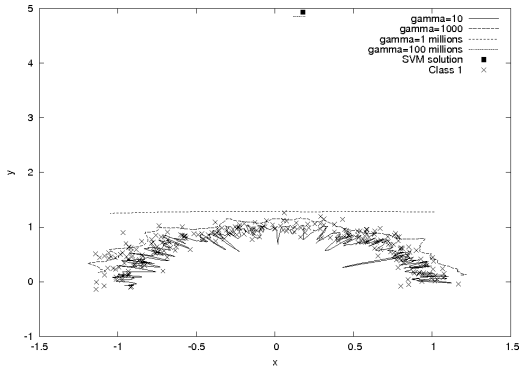

Figure 2: Results of TA–SVM and SVM on Example 1. The different lines show the time evolution of the TA–SVM classifier for growing values of γ.

Example 2: In this case we used the same points sampled for the previous example, but we changed the added angle to $\frac{3i\pi}{1000}$ radians (instead of $\frac{i\pi}{500}$). In this case we have some overlap between the classes, as can be seen in Figure 3.

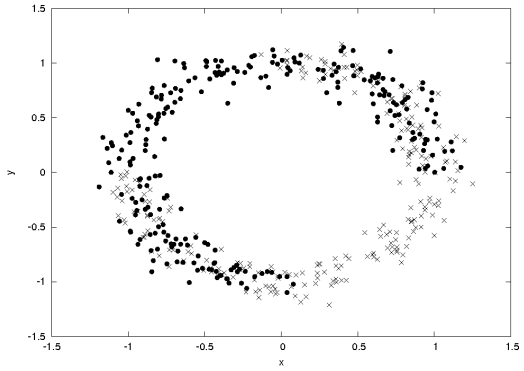

Figure 3: The dataset used in the Example 2

The results in this case were qualitatively similar to those of the first example. Figure 4 shows the results, including the curves corresponding to $\gamma \in \{10^3, 10^6, 10^7, 10^8\}$. The training errors of the TA–SVM for these values were 0, 0, 0,2% y 28,8%, respectively. It is worth mentioning that the high error rate for $\gamma = 10^8$ is approximately equal to the result obtained with the standard SVM, given the high overlap between the classes.

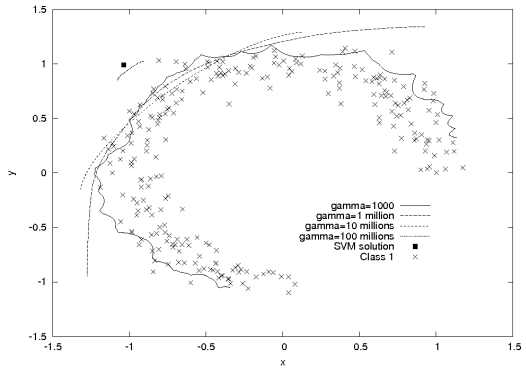

Figure 4: Results of TA–SVM and SVM on Example 2. As in Fig. 2, the different lines show the time evolution of the TA–SVM classifier for growing values of γ.

Example 3: With this example we want to study the behaviour of the method in problems where the decision frontiers are not a simple hyperplane (i.e., we address the situation of a smooth change in the input space of the problem instead of a smooth change in time). Even though the decision frontier is a complex function of the input, the classes can be separated by a set of hyperplanes that varies slowly as one of the coordinates changes. Note that in this case the method is being applied over a stationary system, so this example actually explores the capacity of the algorithm of becoming an alternative to SVM with non linear kernels.

We created a dataset sampling 500 points from a uniform distribution in $x \in \Re^2$, where $x_1 \in [-0.2; 1.2]$ and $x_2 \in [-5; 5]$. A point x_i belongs to class 1 if

$$x_{2,i} \geq \frac{1}{1 + e^{-x_{1,i}}}$$

and to class -1 in the other case. Figure 5 shows the resulting dataset.

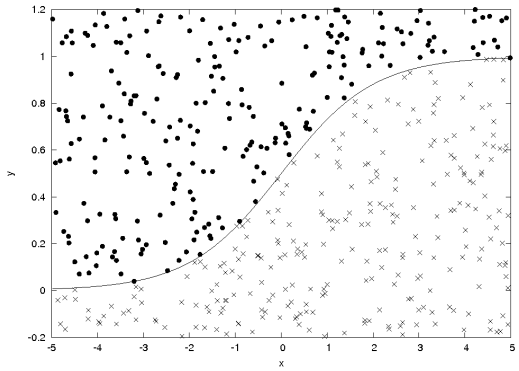

Figure 5: The dataset used in the Example 3

As in the previous cases, in order to force the TA–SVM to search for solutions that vary locally, the points x_i and x_j were considered neighbours if x_i is one of the 5 nearest–neighbours of x_j or x_j is one of the 5 nearest–neighbours of x_i. As before, both parameters C (in TA–SVM and in SVM) were fixed to 1. We use as test set a fixed grid of points inside the same rectangle as the training points. Each point in the test set was classified using the hyperplane assosiated to its nearest neighbour in the training data.

Figures 6 to 9 show the results obtained on the test set for a γ of 10^3, 10^4, 5×10^4 y 5×10^5 respectively. As expected, the decision frontier for a big γ is very similar to the one produced by SVM with a linear kernel.

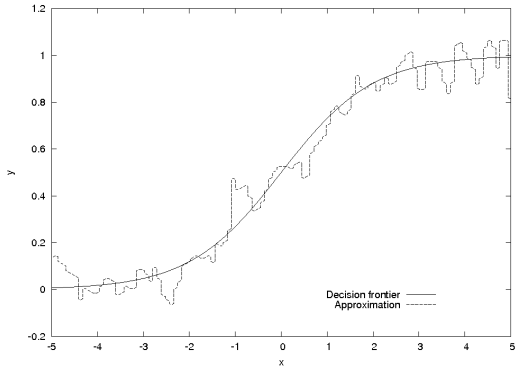

Figure 6: The decision frontier for Example 3 obtained with TA–SVM using $\boldsymbol{\gamma = 10^3}$.

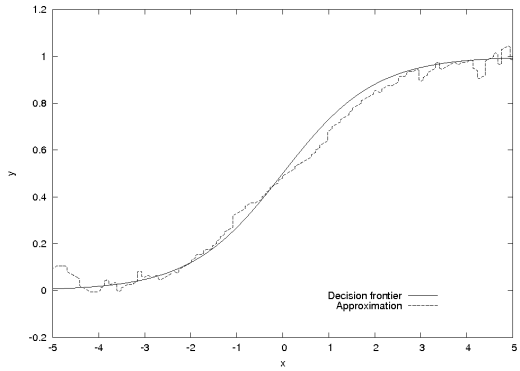

Figure 7: The decision frontier for Example 3 obtained with TA–SVM using $\boldsymbol{\gamma = 10^4}$.

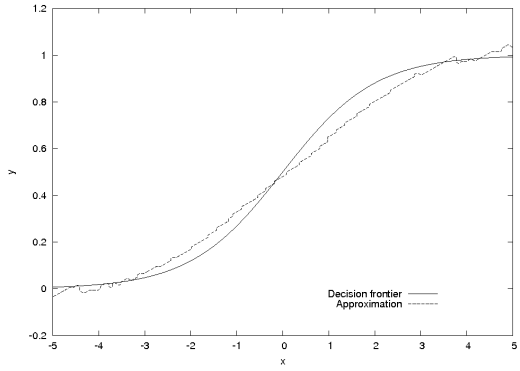

Figure 8: The decision frontier for Example 3 obtained with TA–SVM using $\boldsymbol{\gamma == 5 \times 10^4}$.

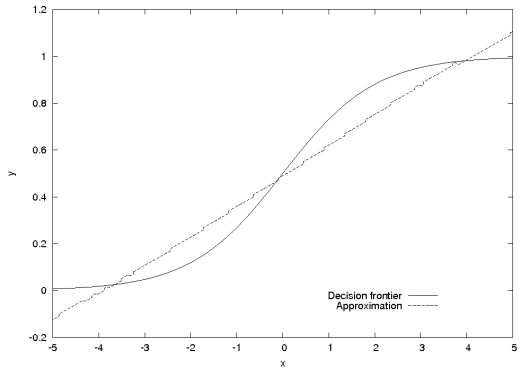

Figure 9: The decision frontier for Example 3 obtained with TA–SVM using $\boldsymbol{\gamma == 5 \times 10^5}$.

We also compared the error rates on the test set produced by TA–SVM with those corresponding to a SVM with a gaussian kernel. Table 1 shows these results. The error rates reported in the Table are the best that could be obtained by optimizing the γ parameter of the gaussian kernel and using a fixex value of $C = 1$. As can be seen in the table, TA–SVM outperforms the best result of SVM with a gaussian kernel, even though the results must be considered as preliminary since neither of the methods were exhaustively optimized.

Method	Error rate (%)
TA–SVM $\gamma = 10^3$	3.61
TA–SVM $\gamma = 10^4$	2.12
TA–SVM $\gamma = 5 \times 10^4$	2.81
TA–SVM $\gamma = 5 \times 10^5$	5.17
SVM (linear kernel)	5.73
SVM (gaussian kernel, $\gamma = 2.94$)	2.37

Table 1: Error rates on the test set for Example 3 and different methods.

4.2 A real–world example

The synthetic examples 1 and 2 clearly demonstrated the capacity of TA–SVM of resolving drifting classification problems. It is remarkable, however, that TA-SVM can also solve stationary, non–linear classification problems with a precision comparable to SVM with a gaussian kernel, as the analysis of the third example suggests. To explore deeper this capacity, we also tested the new method over a real–world problem. In this case we used the **breast cancer** dataset from the *IDA repository* [7]. As in Müller et al. [8], 100 splits of the data in a training set and a test set were used (200 samples for training and 77 for testing). To optimize the free parameters of each method we used an internal 10-fold cross validation for each of the 100 runs. The parameter γ was chosen from $\{10, 10^2, 10^3, 5\times10^3, 10^4, 5\times10^4\}$ and the number of neighbours from $\{3, 4, 5, 7, 10, 15, 20, 25\}$. Table 2 shows our best result and the ones reported by Müller et al. [8] for standard SVM.

Method	Error rate (%)
TA–SVM	27.0 ± 5.7
SVM	26.0 ± 4.7

Table 2: Average test set errors for the "breast cancer" dataset.

The small difference in average errors, in this case on the side of satndard SVM, does not have statistical significance. Again, this time over a real–world problem, we obtained a result comparable to SVM with a gaussian kernel.

5 Conclusions

In this work we presented TA–SVM, a new method for generating adaptive classifiers, capable of learning concepts that change with time. The basic idea of TA–SVM is to use multiple hyperplanes valid only in small temporal intervals (windows) for making the classification but, in contrast to other proposals, learning all the hyperplanes in a global way. We derived an appropriate minimization problem including the error commited by the family of local classifiers plus a measure associated to the VC dimension of the set of classifiers.

We evaluated the method using two artificial examples, showing that TA–SVM can easily follow the temporal drift of a simple clasification problem.

We also argued the idea of temporal locality can be extended to localities in the input space. We used the TA–SVM method to solve artificial and real–world non–linear classification problems, obtaining results similar to SVM with a gaussian kernel. This capacity of TA–SVM is remarkable, since the method can generate a non–linear classifier in the original input space without having to choose a map to a high dimentional space (through a kernel function).

The presented results are preliminary and the method still requires a more exhaustive experimentation to establish its advantages and shortcomings. Nevertheless, at least for non stationary systems, the results look promising enough as for being optimistic in regard to the application of TA–SVM to real problems in slowly drifting systems.

Acknowledgements

We acknowledge partial support for this project from ANPCyT (grant PICT–2003 11–15132 and PICT–2006 2226).

References

[1] R. Klinkenberg, T. Joachims, Detecting Concept Drift with Support Vector Machines, *Proceedings of ICML-00, 17th International Conference on Machine Learning*, 2000.

[2] Renato Vicente, Osame Kinouchi, and Nestor Caticha, Statistical Mechanics of Online Learning of Drifting Concepts: A Varia-

tional Approach, *Machine Learning*, 32:179-201, 1998.

[3] P. L. Bartlett, S. Ben-Dabid and S. R. Kulkarni, Learning Changing Concepts by Exploiting the Structure of Change, *Machine Learning*, 41 Issue 2:153-174, 2000.

[4] J. Zico Kolter and Marcus A. Maloof, Dynamic–Weighted Majority: An Ensemble Method for Drifting Concepts, *Journal of Machine Learning Research*, 8:2755-2790, 2007.

[5] N. Cristianini and J. Shawe-Taylor, An Introduction to Support Vector Machines, Cambridge University Press, Cambridge, 2000.

[6] V. Vapnik, Statistical Learning Theory, Wiley, 1998.

[7] IDA Benchmark repository used in several boosting, KFD and SVM papers http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm

[8] K. R. Müller, S. Mika, G. Rätsch, K. Tsuda, B. Schölkopf, An Introduction to Kernel-Based Learning Algorithms, *IEEE Transactions on Neural Networks* 12 No. 2, 2001.