Máquinas de Vectores Soporte Adaptativas

Guillermo Grinblat y 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, 2000 Rosario, República Argentina

Resumen Se propone un método de clasificación adaptativo capaz de aprender un concepto y seguir su evolución temporal como consecuencia de cambios lentos en sistemas evolutivos. Para ello se realiza una modificación del clasificador SVM (máquina de vectores soporte), consistente en usar múltiples hiperplanos válidos en pequeñas localidades temporales (ventanas) para realizar la clasificación. A diferencia de otras propuestas de este tipo en la literatura, en este caso se realiza un aprendizaje de todos los hiperplanos en forma global, minimizando una cantidad que contiene al error que comete la familia de clasificadores locales más una medida asociada a la dimensión VC de los mismos. Para conceptos estacionarios, la misma idea aplicada a localidades en el espacio de características permite obtener resultados comparables a los que proporciona SVM con kernel gausiano.

1. INTRODUCCIÓN

Muchos problemas de clasificación asociados a sistemas del mundo real varían con el tiempo. Por ejemplo, un sistema puede variar por razones físicas como las estaciones del año, o puede ser necesaria su adaptación por cambio en las expectativas o intereses de los usuarios del mismo. En la mayoría de los casos, la causa y características de este cambio no se presentan en los datos a analizar de manera obvia, por lo que el clasificador asociado tiene que ser capaz no sólo de aprender la relación entrada-salida correcta en cada instante de tiempo sino además advertir el cambio en el concepto y adaptarse a él.

Habitualmente este problema se trata usando una ventana temporal, suponiendo que el cambio en el concepto a aprender es despreciable dentro de ella [1]. Si la ventana es de ancho muy grande, dicha suposición en general no es válida y el tiempo de adaptación del algoritmo resulta excesivo. Por el contrario, cuando el ancho de la ventana es chico el algoritmo se adapta rápidamente, pero es más sensible al ruido y se torna impreciso ya que debe aprender la relación entrada-salida a partir de unos pocos ejemplos. Dentro de esta aproximación al problema, existen algoritmos que usan un ancho de ventana adaptable [1], donde se dividen los datos en grupos ("batchs") y se usa como ventana la cantidad de grupos óptima. Aún así, es igualmente necesario suponer que el concepto no cambia dentro de cada grupo de datos. Por otro lado, un enfoque puramente estadístico del aprendizaje de conceptos evolutivos puede hallarse en [2].

En este trabajo presentamos una nueva forma de encarar este problema, en la cual los clasificadores sucesivos varían siguiendo la mutación del concepto pero su ajuste se realiza de manera global. Es particular, la ventana temporal de validez de un clasificador puede ser tan chica como se desee (inclusive un sólo punto), pero los clasificadores se entrenan minimizando una medida global de error en lugar de ajustarse localmente. Esta filosofía se aplica onsiderando una adaptación de uno de los métodos de clasificación más potentes y estudiados en la actualidad, las llamadas "maquinas de vectores soporte" o SVM por sus siglas en inglés [3].

En el caso de clasificación estacionaria, la filosofía arriba descripta puede aplicarse reemplazando las ventanas temporales por vecindades de un punto en el espacio de características del problema. De esta forma, el clasificador SVM puede describir fronteras de decisión complejas (no simples hiperplanos) sin necesidad de apelar al truco del kernel ("kernel trick")[3] para lograr un clasificador no lineal. Mostraremos, a través de un ejemplo simple, que con esta implementación de SVM adaptativo es posible alcanzar resultados equivalentes a SVM con kernel gausiano.

2. JUSTIFICACIÓN DEL MÉTODO

Supongamos que tenemos un conjunto de datos $[(x_1, y_1) \ldots, (x_n, y_n)]$, donde (x_i, y_i) fue obtenido en el instante $t = i$ y $y_i = \pm 1$. Definimos:

Clase F: Clase de clasificadores tal que $f \in F$ implica que f implementa una frontera de decisión constituida por hiperplanos que cambian con t. Así, un punto en un instante dado es clasificado de acuerdo al lado del hiperplano correspondiente a ese instante en que se encuentra. Note que la dimensión de Vapnik-Chervonenkis (VC) [4] de F es ∞, ya que si el hiperplano cambia lo suficiente desde el instante $i-1$ al instante i, podrá clasificar bien el punto x_i sin importar dónde se encuentre.

Clase F *reducida*: Sea $f \in F_v$ un clasificador perteneciente a F, tal que el cambio del hiperplano de un instante al siguiente está acotado por v. Para ser más precisos, si en el instante i el hiperplano es $f_i = w_i \cdot x_i + b_i$, entonces

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

Sea $F_{v'}$ el conjunto de clasificadores que no cambian más que v', con $v \leq v'$. Como $F_v \subseteq F_{v'}$, la dimensión VC de $F_v \leq$ la dimensión VC de $F_{v'}$. Por otro lado, la dimensión VC de $F_{v=0}$ en $\Re^n$ es $n+1$, ya que es el conjunto de hiperplanos que no varían. Es decir, la dimensión VC de F_v crece con v.

De acuerdo a esto, si queremos controlar la complejidad de las funciones f podemos limitarnos a elegir funciones de F_v para cierto v. O, por la teoría de regularización, en vez de buscar la función de F_v que minimice cierto error $Err(f, x, y)$, podemos buscar la función de F que minimice

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

donde C es una constante que define la importancia relativa de los errores de clasificación con respecto a la complejidad de la función $\text{comp}(f) = \text{máx}_i[(w_{i-1} - w_i)^2 + (b_{i-1} - b_i)^2]$.

Alternativamente, podemos hacer el mismo razonamiento definiendo F_v como la clase de hiperplanos que en promedio se mueven menos que v, con lo que llegaríamos a minimizar

$$Err(f, x, y) + C\frac{1}{n}\sum_i (w_{i-1} - w_i)^2 + (b_{i-1} - b_i)^2$$

Por un razonamiento similar al realizado anteriormente, podemos llegar a la conclusión de que la dimensión VC baja cuando el margen promedio del hiperplano móvil crece. Definimos entonces la complejidad de f como

$$\text{comp}(f) = \frac{1}{n}\sum_i w_i^2 + \frac{C_2}{n}\sum_i (w_{i-1} - w_i)^2 + (b_{i-1} - b_i)^2$$

donde C_2 indica a qué causa de aumento de la dimensión VC le damos más importancia.

Definiendo ahora

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

llegamos al problema

$$\text{mín}\, C_3 \sum_i \text{máx}(0, 1 - y_i(w_i x_i + b_i)) + \frac{1}{n}\sum_i w_i^2 + \frac{C_2}{n}\sum_i (w_{i-1} - w_i)^2 + (b_{i-1} - b_i)^2,$$

que es equivalente a

$$\text{mín}\, C_3 \sum_i \xi_i + \frac{1}{n}\sum_i w_i^2 + \frac{C_2}{n}\sum_i (w_{i-1} - w_i)^2 + (b_{i-1} - b_i)^2,$$

sujeto a

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

3. SVM ADAPTATIVO

Tal como fuera explicitado más arriba, tenemos n puntos, $x_1 \ldots x_n$, divididos en dos clases, con $y_i = \pm 1$ la clase del punto x_i. Definimos ahora V_i como el conjunto de puntos vecinos a x_i y denotamos N_i al número de puntos en V_i. En lo sucesivo llamaremos M_{i*} a la fila i de la matriz M y M_{*j} a la columna j de dicha matriz. Con $P * Q$ indicaremos el producto miembro a miembro de las matrices P y Q; es decir, $(P * Q)_{ij} = P_{ij} Q_{ij}$.

Partimos del problema

$$\min_{w_i,b_i} \frac{1}{2n}\sum_{i=1}^{n}\left(||w_i||^2 + \frac{C_2}{2}\sum_{j\in V_i}||w_i - w_j||^2 + (b_i - b_j)^2\right) + C_3\sum_i \xi_i,$$

con $||w|| = w \cdot w$, sujeto a

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

El correspondiente lagrangeano es:

$$\begin{aligned} L = \frac{1}{2n}\sum_{i=1}^{n}\left(||w_i||^2 + \frac{C_2}{2}\sum_{j\in V_i}||w_i - w_j||^2 + (b_i - b_j)^2\right) + C_3\sum_i \xi_i \\ - \sum_i \alpha_i(y_i(w_i x_i + b_i) - 1 + \xi_i) - \sum_i \beta_i \xi_i, \end{aligned} \tag{2}$$

donde $\alpha_i \geq 0$ y $\beta_i \geq 0$.

Tenemos que maximizar L con respecto a los α_i y β_i y minimizarlo con respecto a los w_i, b_i y ξ_i. En este punto de mínimo, las derivadas con respecto a las variables primales tienen que ser nulas:

$$\frac{\partial L}{\partial \xi_i} = 0 = C_3 - \alpha_i - \beta_i,$$

lo cual implica que

$$0 \leq \alpha_i \leq C_3.$$

Por otro lado, teniendo en cuenta que cada ξ_i está en L multiplicado por $C_3 - \alpha_i - \beta_i$, (2) queda

$$L = \frac{1}{2n}\sum_{i=1}^{n}\left(||w_i||^2 + \frac{C_2}{2}\sum_{j\in V_i}||w_i - w_j||^2 + (b_i - b_j)^2\right) - \sum_i \alpha_i(y_i(w_i x_i + b_i) - 1). \tag{3}$$

En el caso de los w_i se tiene

$$\frac{\partial L}{\partial w_i} = 0 = \frac{1}{n}\left(w_i + C_2\sum_{j\in V_i}(w_i - w_j)\right) - \alpha_i y_i x_i,$$

donde se ha considerado que si x_j es vecino de x_i, x_i es vecino de x_j. Esta ecuación resulta

$$\frac{1}{n}\left((1 + C_2 N_i)w_i - C_2\sum_{j\in V_i} w_j\right) = \alpha_i y_i x_i. \tag{4}$$

Si definimos la matriz M y el vector z como

$$M_{ij} = \begin{cases} (1 + C_2 N_i)/n \text{ si } i = j \\ -C_2/n \qquad \text{si } j \in V_i \\ 0 \qquad\qquad \text{caso contrario.} \end{cases}$$

$$z_i = \alpha_i y_i x_i$$

podemos poner (4) en la forma $Mw = z$, o bien $w_i = (M^{-1})_{i*} z$. Reemplazando en (3) obtenemos

$$L = \frac{1}{2n}\alpha^T((M^{-1})^2 * K)\alpha + \frac{C_2}{4n}\alpha^T((M^{-1}QM^{-1}) * K)\alpha + \frac{C_2}{4n}\sum_i \sum_{j \in V_i} (b_i - b_j)^2$$
$$-\alpha^T((M^{-1}) * K)\alpha + \sum_i \alpha_i - \sum_i \alpha_i y_i b_i. \tag{5}$$

donde hemos definido dos matrices $N \times N$, $K_{ij} = y_i y_j x_i x_j$ y Q dada por

$$Q_{ij} = \begin{cases} N_i \text{ si } i = j \\ -1 \text{ si } i \text{ es vecino de } j \\ 0 \quad \text{caso contrario} \end{cases}$$

En el caso de los b_i,

$$\frac{\partial L}{\partial b_i} = 0 = \frac{C_2}{n}\sum_{j \in V_i}(b_i - b_j) - \alpha_i y_i,$$

de lo que obtenemos

$$\frac{C_2 N_i}{n} b_i - \frac{C_2}{n}\sum_{j \in V_i} b_j = \alpha_i y_i. \tag{6}$$

Definiendo h como:

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

podemos escribir (6) en la forma

$$\frac{C_2}{n} Qb = h. \tag{7}$$

Dado que Q es singular,

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

se puede afirmar que

$$0 = \frac{C_2}{n}\overline{0}b = \frac{C_2}{n}\overline{1}Qb = \overline{1}h = \sum_i \alpha_i y_i.$$

3.1. Eliminación de los b_i

Es posible eliminar b de (5). La parte que depende de b es

$$\frac{C_2}{4n}\sum_i\sum_{j\in V_i}(b_i-b_j)^2-\sum_i\alpha_i y_i b_i=\frac{C_2}{4n}\sum_i\sum_{j\in V_i}(b_i-b_j)^2-h^Tb.$$

Por otro lado,

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

con lo que

$$\frac{C_2}{4n}\sum_i\sum_{j\in V_i}(b_i-b_j)^2-h^Tb=\frac{C_2}{2n}b^TQb-h^Tb.$$

De (7) sabemos que

$$b^TQ=\frac{nh^T}{C_2},$$

con lo cual queda

$$\frac{C_2}{2n}\frac{nh^T}{C_2}b-h^Tb=-\frac{h^Tb}{2}.$$

Diagonalizando Q resulta $Q=PDP^T$, con $P^{-1}=P^T$ ya que Q es simétrica, con lo cual de (7) podemos obtener

$$DP^Tb=P^T\frac{nh}{C_2}$$

Si definimos $b'=P^Tb$ y $h'=P^Th$, esto es

$$Db'=\frac{nh'}{C_2}.$$

Como D es diagonal, es fácil encontrar los b'_i para los cuales $D_{ii}=\lambda_i\neq 0$:

$$b'_i=\frac{nh'}{C_2\lambda_i},\text{ si }\lambda_i\neq 0.$$

Si el autovalor $\lambda_i=0$, de (7) tenemos que

$$\begin{aligned}P_{i*}^TQb&=P_{i*}^T\frac{nh}{C_2}\\ \overline{0}b&=P_{i*}^T\frac{nh}{C_2}\\ 0&=\frac{nh'_i}{C_2}.\\ 0&=h'_i\end{aligned}$$

Es decir, cuando no podemos determinar b'_i por la ecuación anterior, $h'_i = 0$.

Volvamos a lo que queremos obtener:

$$\begin{aligned} -\frac{h^T b}{2} &= -\frac{h^t P P^T b}{2} \\ &= -\frac{1}{2} h'^T b' \\ &= -\frac{1}{2} \sum_i h'_i b'_i \\ &= -\frac{1}{2} \sum_i \frac{n}{C_2} \Lambda_i h'^2_i \end{aligned}$$

donde $\Lambda_i = \frac{1}{\lambda_i}$ si $\lambda_i \neq 0$ y $\Lambda_i = 0$ si $\lambda_i = 0$. Si se define D' como la matriz diagonal $D'_{ii} = \Lambda_i$ e Y dada por $Y_{ij} = y_i y_j$, esto último queda

$$\begin{aligned} -\frac{h^T b}{2} &= -\frac{n}{2C_2} h^T P D' P^T h \\ &= -\frac{n}{2C_2} \alpha^T ((P D' P^T) * Y) \alpha. \end{aligned}$$

Reemplazando en (5) obtenemos

$$\begin{aligned} L = \frac{1}{2n} \alpha^T ((M^{-1})^2 * K) \alpha + \\ \frac{C_2}{4n} \alpha^T ((M^{-1} Q M^{-1}) * K) r r \alpha - \\ \alpha^T ((M^{-1}) * K) \alpha + \sum_i \alpha_i - \\ -\frac{n}{2C_2} \alpha^T ((P D' P^T) * Y) \alpha \end{aligned}$$

Es decir,

$$L = \alpha^T \left[\left(\frac{1}{2n} M^{-2} + \frac{C_2}{4n} M^{-1} Q M^{-1} - M^{-1} \right) * K - \frac{n}{2C_2} (P D' P^T) * Y \right] \alpha + \sum_i \alpha_i,$$

lo que implica que es de la forma

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

con la matriz R definida de manera adecuada.

El problema dual queda entonces

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

sujeto a

$$0 \leq \alpha_i \leq C_3$$
$$\sum \alpha_i y_i = 0,$$

que es el problema resuelto en SVM (aunque con otra R). En consecuencia, para resolverlo se puede usar cualquiera de las técnicas empleadas para SVM convencional, como por ejemplo SMO[3].

4. EXPERIMENTOS

4.1. Bases de Datos Artificiales

Se probó el método en las siguientes tres bases de datos artificiales:

Dataset 1: Se generó un dataset de puntos $x_i \in \Re^2$, divididos en dos clases, la clase 1 formada por 250 puntos centrados en $(0, 1)$, con un desvío estandar de 0,1 en cada dimensión y la clase -1 formada por 250 puntos centrados en $(0, -1)$, con el mismo desvío.

Para simular un cambio a través del tiempo, a cada punto x_i, representado en coordenadas polares, se le sumó un ángulo de $\frac{i\pi}{500}$ radianes. Al algoritmo se le presentaron los puntos en coordenadas rectangulares. El resultado se grafica en la figura 1.

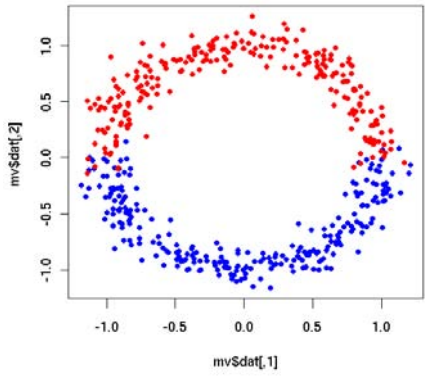

Figura 1. Dataset 1

Dataset 2: A los mismos puntos que en el dataset anterior se les sumó un ángulo $\frac{3i\pi}{1000}$ en vez de $\frac{i\pi}{500}$. En este caso el solapamiento de las clases es mayor (ver figura 2).

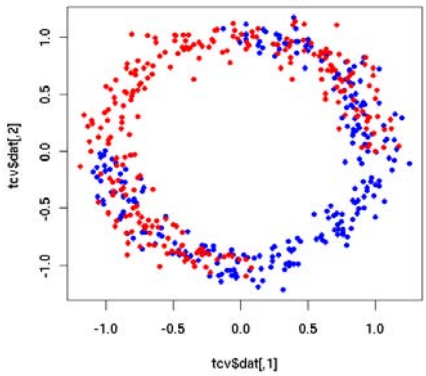

Figura 2. Dataset 2

Dataset 3: Se tomaron 500 puntos $x \in \Re^2$ al azar con $x_1 \in [-0{,}2; 1{,}2]$ y $x_2 \in [-5; 5]$, con distribución uniforme. Al punto x_i se le asignó la clase 1 si

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

y la clase -1 en caso contrario (ver Figura 3).

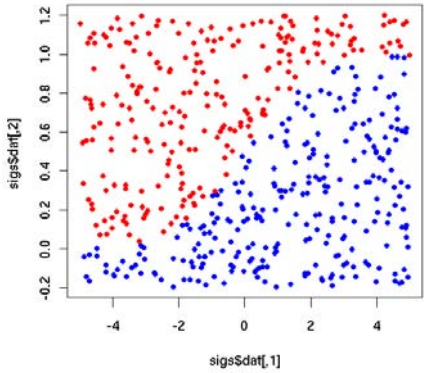

Figura 3. Dataset 3

La intención de las pruebas con los dos primeros datasets fue ver cómo evoluciona el clasificador a través del tiempo al variar el parámetro C_2 y compararlo con la solución encontrada por SVM estándar con kernel lineal (que no tiene en cuenta el tiempo en que cada una de las muestras fue colectada).

Para que el método busque una evolución temporal, se consideró como conjunto de puntos vecino a x_i a $\{x_{i-1}, x_{i+1}\}$ (salvo en el caso de los extremos). Es decir, se usó una matriz Q dada por

$$\begin{aligned} Q_{11} = Q_{nn} &= 1 \\ Q_{ii} &= 2 \quad \text{para } i \neq 1 \text{ e } i \neq n \\ Q_{i,i+1} = Q_{i,i-1} &= -1 \\ Q_{ij} &= 0 \quad \text{en otros casos.} \end{aligned}$$

El parámetro C_3 se dejó fijo en 1 mientras se variaba C_2. Acorde a esto, se usó $C = 1$ para el parámetro de SVM.

Los resultados obtenidos muestran que el clasificador cambia de la misma forma en que se generaron los puntos para los valores más bajos de C_2 y tiende a la solución encontrada por SVM estándar a medida que crece C_2. El error de entrenamiento es 0, salvo para valores altos de C_2, que obligan a que el clasificador no varíe mucho y en consecuencia no puede seguir la evolución de los datos (cabe aclarar que los datasets no tienen puntos mal clasificados). En la figura 4 se ve la evolución del clasificador para $C_2 \in \{10, 10^3, 10^6, 10^8\}$. Las líneas unen el extremo de cada vector w_i con el de los vectores w_{i+1} y w_{i-1}; es decir, constituyen la gráfica de la curva $w(t)$. Los errores de entrenamiento para estos cuatro casos fueron 0, 0, 0 y 2,6 %, respectivamente.

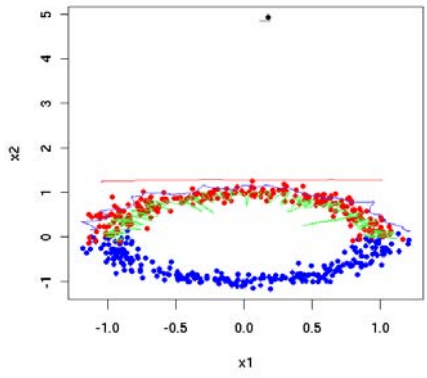

Figura 4. Dataset 1. Las líneas muestran la evolución $w(t)$ para $C_2 = 10$ (verde), 10^3 (azul), 10^6 (roja) y 10^8 (negra); el punto negro corresponde al w obtenido con SVM no adaptivo.

Para el dataset 2 se obtuvieron resultados similares. En la figura 5 se ven las curvas correspondientes a $C_2 \in \{10^3, 10^6, 10^7, 10^8\}$. Los errores de entrenamiento fueron de 0, 0, 0,2 % y 28,8 % para cada uno de estos valores. Cabe destacar que el alto valor de error correspondiente a $C_2 = 10^8$ constituye aproximadamente el resultado esperado para SVM estándar (no adaptativo), debido a la alta superposición de las clases.

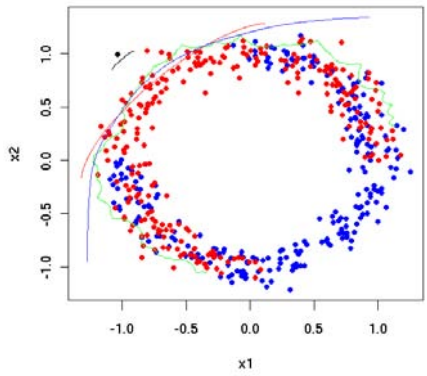

Figura 5. Dataset 2. Las líneas muestran la evolución $w(t)$ para $C_2 = 10$ (verde), 10^3 (azul), 10^6 (roja) y 10^8 (negra); el punto negro corresponde al w obtenido con SVM no adaptativo.

Con el tercer dataset se ejemplifica cómo se comporta el método en problemas cuya frontera de decisión no sea un simple hiperplano (tratando esta situación como un cambio adaptativo en el espacio de características del problema en lugar del tiempo). Si bien la frontera de decisión es una función compleja de la entrada, se pueden separar las clases con un hiperplano que varíe lentamente a medida que cambia una de las coordenadas. Note que en este caso la aplicación del método se realiza sobre un sistema estacionario, por lo que este ejemplo en realidad explora la capacidad del algoritmo aquí propuesto de constituirse en una alternativa al uso de SVM con kernels no-lineales.

Para que el método busque soluciones que varíen localmente se consideró como vecinos a los puntos x_i y x_j sii x_i es uno de los 5 vecinos más próximos a x_j o x_j es uno de los 5 vecinos más próximos a x_i. Lo mismo que antes, el parámetro C_3 se dejó fijo en 1 y se usó también 1 para el parámetro C de SVM. Como conjunto de test consideramos una grilla de puntos dentro del mismo rectángulo que los datos de entrenamiento. A cada punto de test se lo clasificó con el hiperplano asociado a su vecino más próximo dentro del dataset original usado para aprendizaje del clasificador.

En las figuras 6-9 se muestran los resultados obtenidos sobre el conjunto de test para valores de C_2 de 10^3, 10^4, 5×10^4 y 5×10^5 respectivamente. Como era esperable, para C_2 grande la frontera de decisión obtenida es muy similar a la que produciría el SVM estacionario con kernel lineal.

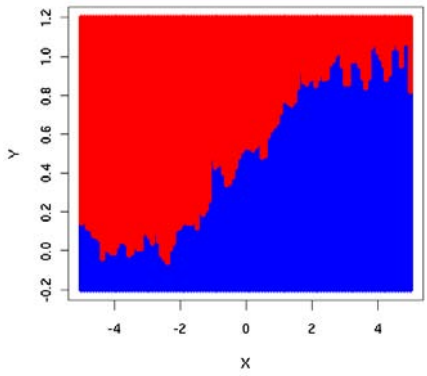

Figura 6. Dataset 3. Resultados para $C_2 = 10^3$.

También se compararon los resultados con los obtenidos con SVM con kernel gausiano. El reportado en el cuadro 1 es el mejor que se pudo conseguir variando el parámetro γ de la gausiana y dejando el parámetro C fijo en 1.

Cuadro 1. Errores de test para el dataset 3.

Prueba	Error (%)
$C_2 = 10^3$	3,61
$C_2 = 10^4$	2,12
$C_2 = 5 \times 10^4$	2,81
$C_2 = 5 \times 10^5$	5,17
SVM lineal	5,73
SVM gausiano ($\gamma = 2{,}94$)	2,37

Como puede verse en la tabla, nuestro método supera al resultado óptimo de SVM con kernel gausiano, aunque los resultados tienen que considerarse preliminares dado que para ninguno de los dos métodos se realizó un ajuste exhaustivo de los parámetros.

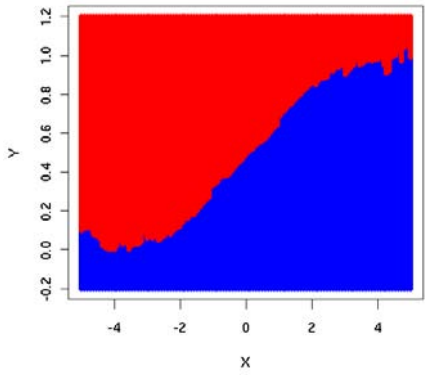

Figura 7. Dataset 3. Resultados para $C_2 = 10^4$.

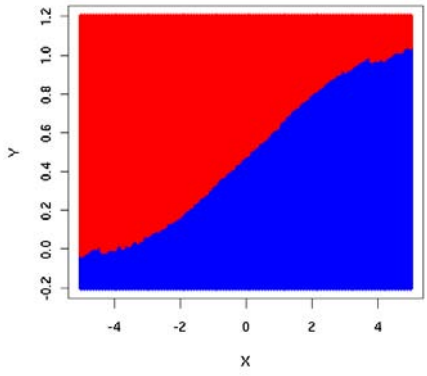

Figura 8. Dataset 3. Resultados para $C_2 = 5 \times 10^4$.

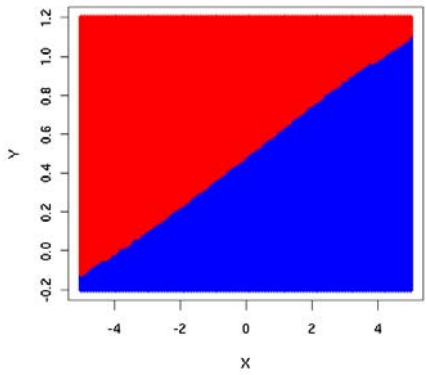

Figura 9. Dataset 3. Resultados para $C_2 = 5 \times 10^4$.

4.2. Bases de Datos Reales

De los dos primeros ejemplos sintéticos considerados en la sección anterior surge claramente la capacidad del algoritmo propuesto para resolver problemas de clasificación adaptativa. Ello no es sorprendente en tanto el método fue específicamente diseñado teniendo en mente este tipo de aplicaciones. Es destacable, en cambio, que el mismo permita resolver problemas de clasificación estándar con precisión comparable a la de SVM no lineal (Kernel SVM), tal como surge del análisis del tercer dataset considerado previamente. Para explorar más en detalle esta capacidad, se probó el método propuesto en un problema real. Para ello se usó el dataset **breast cancer** del *IDA repository* [5]. Al igual que en [6], se consideraron 100 divisiones de los datos para entrenamiento y test (200 para el entrenamiento y 77 para test). Para seleccionar los parámetros del método se hizo en cada una de las 100 pruebas una validación cruzada con 10 "folds". Se probó variar el parámetro C_2 en $\{10, 10^2, 10^3, 5 \times 10^3, 10^4, 5 \times 10^4\}$ y el número de vecinos usados en $\{3, 4, 5, 7, 10, 15, 20, 25\}$.

En el cuadro 2 se muestra el mejor resultado obtenido y el reportado en [6]:[1]

Cuadro 2. Errores de test para el dataset "breast cancer".

Prueba	Error (%)
Nuestro Método	$27{,}0 \pm 5{,}7$
SVM	$26{,}0 \pm 4{,}7$

[1] En [6] el σ reportado para SVM es de 0,47, mientras que en el resumen en [5] figura 4,7.

La pequeña diferencia en los promedios de error, que favorece al mejor resultado obtenido con SVM convencional, no es significativa dada la gran dispersión observada en dichos promedios. Es decir, nuevamente aquí, sobre un problema real, se obtiene que SVM adaptativo es comparable a SVM con kernel gausiano. Esta capacidad del método propuesto constituye un importante valor agregado del mismo, ya que permite generar un clasificador no lineal en el espacio de características original, independizándonos de la elección de un mapa a un espacio de mayor dimensión (a través del kernel utilizado).

5. CONCLUSIONES

En el presente trabajo hemos propuesto un nuevo método de generación de clasificadores adaptativos, capaces de aprender conceptos que mutan con el tiempo. La idea se ejemplificó desarrollando en detalle una versión local del clasificador SVM, que permite recuperar la evolución temporal de un problema de clasificación simple. La idea básica consiste en usar múltiples hiperplanos válidos en pequeñas localidades temporales (ventanas) para realizar la clasificación pero, a diferencia de otras propuestas de este tipo, realizando un aprendizaje de todos los hiperplanos en forma global. Para ello se minimiza una cantidad que contiene al error que comete la familia de clasificadores locales más una medida asociada a la dimensión de VC de los mismos. Por otro lado, la misma idea aplicada a localidades en el espacio de características del problema de clasificación permite obtener resultados comparables a los que proporciona SVM con kernel gausiano.

Los resultados presentados son preliminares y requieren aún una experimentación más exhaustiva para establecer ventajas y desventajas del método propuesto. No obstante, al menos para sistemas no estacionarios, los mismos son lo suficientemente prometedores como para ser optimista en cuanto a la aplicación de esta técnica a problemas reales en sistemas que mutan lentamente.

Referencias

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 Variational Approach*, Machine Learning **32**, 179-201 (1998)
3. N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines*, Cambridge University Press, Cambridge (2000)
4. V. Vapnik, *Statistical Learning Theory*, Wiley (1998)
5. *IDA Benchmark repository used in several boosting, KFD and SVM papers* http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm
6. 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)