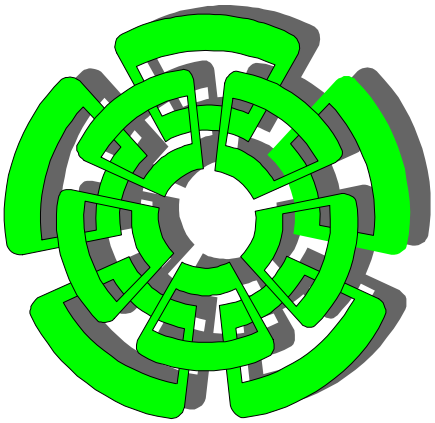

CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS
DEL
INSTITUTO POLITÉCNICO NACIONAL

DEPARTAMENTO DE CONTROL AUTOMÁTICO

Las maquinas de vectores de soporte para identificación en línea

TESIS QUE PRESENTA EL:
Ing. Juan Angel Resendiz Trejo

PARA OBTENER EL GRADO DE
MAESTRO EN CIENCIAS

EN LA ESPECIALIDAD DE
CONTROL AUTOMÁTICO

DIRECTOR DE TESIS:
Dr. Wen Yu Liu

México, D.F., Sep del 2006

Índice general

Índice de figuras

Capítulo 1

Introducción

Es innegable que hasta ahora las redes neuronales han sido el método más popular para el modelado de procesos industriales los cuales son altamente no lineales. Las redes neuronales son aproximadores universales que pueden aproximar cualquier función no lineal ya sea en un ambiente fuera de línea o en línea. La identificación de sistemas no lineales través de redes neuronales permite obtener modelos no lineales de la realidad a partir de un conjunto de entradas las cuales, en última instancia, determinan el funcionamiento del modelo. Este tipo de modelado se basa en el diseño de la estructura de la red neuronal lo cual es muy complicado de describir precisamente y aún más para realizar el ajuste de los pesos las redes neuronales usan métodos como el gradiente descendiente y el de propagación hacia atrás los cuales hacen que la red neuronal sufra de una lenta convergencia y del mínimo local. En trabajos como [1] se han realizado algunas modificaciones donde se propone un nuevo algoritmo de propagación hacia atrás robusto el cual es mas resistente a los efectos de ruido en las entradas y evitar los errores en la aproximación, en [2] se utilizaron funciones de membresía B-spline para minimizar una función robusta, este algoritmo mejoro la velocidad de convergencia del algoritmo. Pero a pesar de lo hecho estos algoritmos siguen sufriendo del mínimo local.

Las maquinas de vectores de soporte (SVM por sus siglas en ingles, "Support Vector Machine"), fueron desarrolladas por Vapnik (1995) [3], para el problema de clasificación pero

la forma actual de SVM para regresión fue desarrollada en los laboratorios de AT&T por Vaptnik [4]. SVM esta ganando gran popularidad como herramienta para la identificación de sistemas no lineales, esto debido principalmente a que SVM esta basado en el principio de minimización del riesgo estructural (SRM por sus siglas en ingles, "Structural Risk Minimization"), principio originado de la teoría de aprendizaje estadístico desarrollada por Vapnik en [3], el cual ha demostrado ser superior al principio de minimización del riesgo empírico (ERM por sus siglas en ingles ."Empirical Risk Minimization"), utilizado por las redes neuronales convencionales. Algunas de las razones por las que este método ha tenido éxito es que no padece de mínimos locales y el modelo solo depende de los datos con mas información llamados vectores de soporte(SV por sus siglas en ingles, "Support Vectors"). Las grandes ventajas que tiene SVM son:

- Una excelente capacidad de generalización, debido a la minimización del riesgo estructurado.
- Existen pocos parámetros a ajustar; el modelo solo depende de los datos con mayor información.
- La estimación de los parámetros se realiza a través de la optimización de una función de costo convexa, lo cual evita la existencia de un mínimo local.
- La solucion de SVM es sparse, esto es que la mayoria de las variables son cero en la solución de SVM, esto quiere decir que el modelo final puede ser escrito como una combinación de un numero muy pequeño de vectores de entrada, llamados vectores de soporte.

En [5] y [6] se puede encontrar una buena introducción a este método. SVM resuelve un problema cuadratico donde el numero de coeficientes es igual al numero de entradas o datos de entrenamiento. Este hecho hace que para grandes cantidades de datos las tecnicas numericas de optimización ,existentes para resolver el problema cuadratico, no sean admisibles en terminos computacionales. Este es un problema que impide el uso de SVM para la

identificacion de sistemas no lineales en linea, esto es, en casos en los que las entradas son obtenidas de manera secuencial y el aprendizaje se realiza en cada paso. En la literatura existen algunas técnicas para la aplicación de SVM en línea. Estos intentos hicieron posible el diseño de algoritmos los cuales mejoran tanto el tiempo como el costo computacional de los algoritmos convencionales existentes para resolver el problema cuadrático. Sin embargo, aun los algoritmos más eficientes de SVM dependen de todos los datos de entrenamiento [7], [8]. En [9] y [8] se propone descomponer el problema en subproblemas mas sencillos de tratar a travez de cuatro pasos. En [7] se utiliza método de optimización mínima secuencial (SMO) para dividir el problema cuadrático en una serie de problemas cuadráticos de menor dimensión los cuales pueden ser resueltos de manera analítica evitando así utilizar técnicas numericas para el problema cuadrático mientras que en [10] se propone un algoritmo recursivo para el entrenamiento de SVM donde la idea basica consiste en encontrar las condiciones apropiadas de Kuhn- Tucker (KT) para la actualización de los coeficientes. Otra forma de lograr que la solución sea sparse es por construcción. Aquí el algoritmo comienza con una representación vacía y se agregan datos de entrenamiento de acuerdo a algún criterio. Más sin embargo la desventaja de estas técnicas es que pueden dar una aproximación de la solución, y pueden requerir muchas operaciones por cada iteración para alcanzar un nivel adecuado de convergencia. Es indiscutible que cada día aumenta el número de aplicaciones para las cuales SVM es una herramienta muy útil. Esto debido al creciente entusiasmo, que se ha dado en las ultimas décadas, por desarrollar nuevos métodos los cuales lleven al mejor desempeño del método SVM. Aun cuando SVM muestra ser un método que supera a las Redes Neuronales en cuanto a su capacidad de generalización y a la ausencia de mínimos locales, SVM sufre de otros problemas como la selección de la mejor función kernel y los problemas computacionales al realizar la identificación sobre un conjunto, muy poblado, de datos de entrenamiento.

1.1. Motivación

Debido a la creciente complejidad que presentan los procesos industriales, se ha hecho extremadamente difícil construir modelos matemáticos para describir el comportamiento de estos. Para solucionar este problema se han utilizado las Redes Neuronales convencionales, incluyendo el método de propagación hacia atrás y las Redes Neuronales de base radial ya que pueden aproximar cualquier función no lineal. Aún cuando las Redes Neuronales poseen una auto adaptación y auto aprendizaje, aún quedan algunos temas que solucionar como la presencia de mínimos locales y la elección de la estructura de la red, mientras que en las Redes Neuronales de base radial el problema reside en la selección de los centros.

Debido a que SVM es una nueva clase de algoritmos de aprendizaje que utiliza el método kernel y el principio de minimización del riesgo estructural para producir algoritmos no lineales de aprendizaje supervisado, hoy en día este método es el estado del arte en aplicaciones de identificación de sistemas. Algunas aplicaciones exitosas del método SVM se pueden ver en [11], [12] y [13].

SVM ha demostrado un gran éxito en muchas aplicaciones en lotes, no a si en las aplicaciones en línea. No obstante, la extensión de los métodos de kernel en línea donde la información fluye de manera secuencial ha sido muy complicada. Este método presenta una dependencia entre el costo computacional y el numero de datos de entrenamiento, haciendo que este método sea admisible solamente para aprendizaje fuera de línea.

Un requerimiento importante para cualquier algoritmo en línea es que su complejidad paso a paso este acotada por una constante independiente del tiempo t, para lo cual se asume que los datos se obtienen de manera constante. Y dado que la complejidad de SVM es dependiente del número de datos de entrenamiento, obtener una solución sparse es esencial para el aprendizaje. Mientras que se puede utilizar el algoritmo por lotes utilizando un buffer deslizante para aplicaciones en línea, es mucho mejor desarrollar un algoritmo que trabaje en línea. En la literatura existen técnicas las cuales buscan aplicar SVM en un ambiente en línea por medio de ventanas deslizantes. En [14] se presenta la detección de fallas por medio de SVM utilizando la técnica de ventanas deslizantes para la detección de defectos en la

industria textil, pero no para la identificación de sistemas no lineales.

El propósito de este trabajo, es proponer un método SVM Recursivo para identificación en línea el cual admita en su representación a un conjunto de datos de entrenamiento los cuales no puedan ser representados por una combinación lineal de los datos de entrenamiento anteriores. El método propuesto es capaz de realizar de manera recursiva la identificación de sistemas no lineales. Asimismo, en este trabajo se estudia las técnicas de ventanas deslizantes en SVM para identificación en línea de sistemas no lineales.

1.2. Justificación

Al investigar la literatura existente de SVM para identificación uno se encuentra con que es poca. Más aun si buscamos SVM para identificación en línea nos encontramos con que la literatura existente es aún más escasa y que el trabajo por realizar en esta área es aún muy extenso. La mayoría de la literatura existente de SVM en línea se emplea en algoritmos para clasificación y no así para identificación. Por eso el objetivo de este trabajo es obtener un algoritmo recursivo el cual permita utilizar SVM para la identificación de sistemas no lineales en un ambiente en línea, así como investigar el uso de la técnicas de ventanas deslizantes, utilizadas ampliamente en el procesamiento de información cuando no es viable su almacenamiento en memoria, para la implementación de los métodos SVM en un ambiente en línea.

1.3. Organización de la tesis

En el primer capitulo se dio una brebe introducción a SVM. El resto de la tesis se organiza de la siguiente forma. En el Capitulo 2. se muestra una pequeña introducción a las herramientas necesarias para comprender de mejor forma el funcionamiento de SVM. También se da una breve introducción al método de SVM y LS SVM. En el Capitulo 3. se presenta el método SVM recursivo para identificación en línea, basado en el método SVM ε insensitive sin termino bias, para obtener una solución sparse, esto es, que el modelo final dependa de

un numero acotado de datos de entrenamiento. También se realizan varias simulaciones para comprobar la efectividad del método en un ambiente en línea. En el Capitulo 4. Se estudia el uso de la técnica de ventanas deslizantes para la implementación de SVM para identificación en línea. Se propondrá la técnica de ventanas deslizantes para obtener algoritmos SVM en línea para la identificación de sistemas no lineales en línea. Se utilizaran los métodos de SVM para identificación: ε insensitive, $\varepsilon-$ insensitive cuadrático y mínimos cuadrados. Por último, el capítulo 5 describe las conclusiones generales del trabajo realizado y las perspectivas de investigación para trabajos futuros.

Capítulo 2

Support Vector Machine

Las máquinas de Vectores de Soporte (Support Vector Machine, por sus siglas en ingles SVM) son un nuevo sistema de aprendizaje el cual ha tenido un desarrollo muy significativo en los últimos años tanto en la generación de nuevos algoritmos como en las estrategias para su implementación. SVM es un sistema de aprendizaje basado en el uso de un espacio de hipótesis de funciones lineales en un espacio de mayor dimensión inducido por un Kernel, en el cual las hipótesis son entrenadas por un algoritmo tomado de la teoría de optimización el cual utiliza elementos de la teoría de generalización. SVM es un sistema para entrenar máquinas de aprendizaje lineal eficientemente tanto que para clasificación como para regresión se han encontrado muchas aplicaciones como clasificación de imágenes, reconocimiento de caracteres, detección de proteínas, clasificación de patrones, identificación de funciones, etc. A continuación describiremos el sistema SVM para la identificación de funciones no lineales. Para lograr esto comenzaremos dando una breve introducción a las funciones kernel, la teoría de generalización y la teoría de optimización las cuales nos ayudaran a comprender mejor al sistema SVM, y por ultimo se describirán los métodos de SVM para regresión.

2.1. Espacios inducidos por la función Kernel

Debido a las limitaciones computacionales de las máquinas de aprendizaje lineal estas no pueden ser utilizadas en la mayoría de las aplicaciones del mundo real. La representación por medio del Kernel ofrece una solución alternativa a este problema, proyectando la información a un espacio de características de mayor dimensión el cual aumenta la capacidad computacional de la máquinas de aprendizaje lineal. La forma más común en que las máquinas de aprendizaje lineales aprenden una función objetivo es cambiando la representación de la función, esto es similar a mapear el espacio de entradas X a un nuevo espacio de características $F = \{\phi(x)|x \in X\}$. Esto es:

$$x = \{x_1, x_2, \cdots, x_n\} \longrightarrow \phi(x) = \{\phi(x)_1, \phi(x)_2, \cdots, \phi(x)_n\}$$

Definición 2.1 *Las cantidades introducidas para describir la información original o atributos son conocidas como características, mientras que a la selección de la mejor representación se le conoce como selección de características.*

En la Figura 2.1 se muestra un mapeo de un espacio de entradas de dos dimensiones a un espacio de características de dos dimensiones, donde la información no puede ser separada por una máquina lineal en el espacio de entradas mientras que en el espacio de características esto resulta muy sencillo.

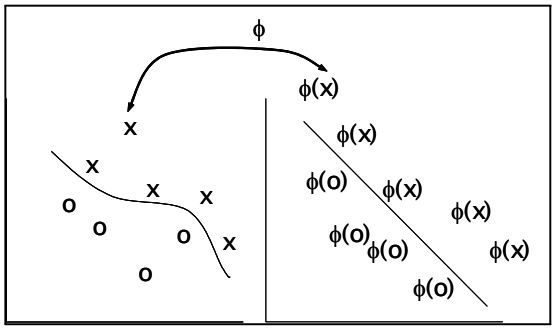

Figura 2.1: Mapeo del espacio de entradas a un espacio de caracteristicas de mayor dimensión.

Las máquinas de aprendizaje lineales son funciones reales $f : X \in \mathbb{R}^n \rightarrow Y \in \mathbb{R}$. La función f se considera como una función lineal de $x \in X$, tal que se pude escribir como

$$\begin{aligned} f(x) &= \langle w \cdot x \rangle + b && (2.1) \\ &= w x^T + b && (2.2) \\ &= \sum_{i=1}^{n} w_i x_i + b \end{aligned}$$

donde w es el vector de pesos y b es el bias, términos tomados de la literatura de redes neuronales. Este tipo de máquinas admiten una representación dual, esto es si definimos a $w = \sum_{i=1}^{n} \alpha_i y_i x_i$ tenemos que la función lineal se puede escribir en su forma dual esto es:

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i \langle x_i \cdot x_i \rangle + b \tag{2.3}$$

donde $\langle \cdot \rangle$ es el producto interno. Una propiedad importante de la representación dual es que la información de entrenamiento entra a la función a través de las entradas de la matriz de Gram $G = \langle x_i \cdot x_i \rangle$. A fin de aprender relaciones no lineales con máquinas lineales es necesario seleccionar un conjunto de características no lineales con las cuales poder rescribir la información original en una nueva representación. De ahí que el conjunto de hipótesis que se consideran son del tipo:

$$f(x) = \sum_{i=1}^{n} w_i \phi_i(x) + b \tag{2.4}$$

donde $\phi : X \longrightarrow F$ es un mapeo no lineal que va del espacio de entradas a algún espacio de características. Debido a que las máquinas de aprendizaje lineal admiten una representación dual podemos escribir la hipótesis como una combinación lineal de la información de entrada (x_i, y_i), de la siguiente manera:

$$f(x) = \sum_{i=1}^{N} \alpha_i y_i \left\langle \phi_i^T(x) \cdot \phi_i(x) \right\rangle + b \tag{2.5}$$

esto es si podemos calcular el producto interno en el espacio de características como una función de las entradas, estaremos realizando un mapeo del espacio de entradas a el espacio de características donde se realizara el aprendizaje con una máquina lineal.

Definición 2.2 *Un Kernel K es una función, tal que para todo $x, z \in X$*

$$K(x,z) = \langle \phi(x) \cdot \phi(z) \rangle = \sum_{i=1}^{l} \phi_i^T(x)\, \phi_i(z) \tag{2.6}$$

donde ϕ es un mapeo del espacio de entradas X al espacio de características F.

El uso de la función kernel hace posible realizar el mapeo de la información de entrada (x, z) al espacio de características $(\phi(x), \phi_i(z))$ de forma implícita y entrenar a la máquina lineal en dicho espacio. La única información necesaria para el entrenamiento es la matriz de Gram, dicha matriz también es conocida como la matriz kernel la cual se denota con la letra K :

$$K(x,z) = \left\langle \phi(x)_i \cdot \phi(z)_j \right\rangle_{i,j=1}^{l} \tag{2.7}$$

Una vez que se define a la matriz K, la hipótesis se puede calcular evaluando al menos l veces la matriz K de la siguiente manera:

$$f(x) = \sum_{i=1}^{l} \alpha_i y_i K(x_i, x) + b \tag{2.8}$$

Comentario 2.1 *Utilizando la función kernel no es necesario calcular explícitamente el mapeo $\phi : X \longrightarrow F$ para aprender en el espacio de características.*

Algunas de las propiedades que debe cumplir la función kernel para definir un espacio de características son las siguientes:

1. Simétrica

$$K(x,z) = \langle \phi(x) \cdot \phi(z) \rangle = \langle \phi(z) \cdot \phi(x) \rangle = \sum_{i=1}^{l} \phi_i(x)\, \phi_i(z) = K(z,x) \tag{2.9}$$

2. Desigualdad de Cauchy- Schwarz

$$\begin{aligned} K(x,z)^2 &= \langle \phi(x) \cdot \phi(z) \rangle^2 \leq \|\phi(x)\|^2 \|\phi(z)\|^2 = \langle \phi(x) \cdot \phi(x) \rangle \langle \phi(z) \cdot \phi(z) \rangle \\ &= K(x,x)K(z,z) \end{aligned} \tag{2.10}$$

Estas condiciones, como sea, no son suficientes para la existencia de un espacio de características. En esta sección se introduce el teorema de Mercer el cual provee las condiciones que debe cumplir una función K para ser un kernel el cual induzca un espacio de características.

Proposición 2.1 *Sea X un espacio de entradas finito con $K(x,z)$ una función simétrica sobre X. Entonces $K(x,z)$ es un kernel si y solo si la matriz*

$$K = \left(K\left(x_i, x_j\right)\right)_{i,j=1}^{n} \tag{2.11}$$

es positiva semidefinida.

En un espacio de Hilbert también se puede definir un kernel, puesto que el espacio de Hilbert es un espacio con producto interno que satisface las propiedades lineales. Introduciendo un elemento de ponderación λ se define el kernel de la siguiente forma

$$\left\langle \phi\left(x\right) \cdot \phi\left(z\right) \right\rangle = \sum_{i=1}^{\infty} \lambda_i \phi_i\left(x\right) \phi_i\left(z\right) = k(x,z) \tag{2.12}$$

el teorema de Mercer da las condiciones necesarias y suficientes para que una función simétrica y continua $k(x,z)$ admita dicha representación, con $\lambda_i \geq 0$. Esto es equivalente a que $k(x,z)$ sea un producto interno en el espacio de características $F \supseteq \phi\left(X\right)$, donde F es un espacio l_2 de secuencias $\boldsymbol{\psi} = \left(\psi_1, \psi_2 \cdots, \psi_i, \cdots\right)$ para el cual $\sum_{i=1}^{\infty} \lambda_i \psi_i^2 < \infty$. Esto induce implícitamente un espacio de características en el cual se puede representar una función lineal

$$f\left(x\right) = \sum_{i=1}^{\infty} \lambda_i \psi_i \phi_i\left(x\right) + b = \sum_{j=1}^{l} \alpha_j y_j K\left(x, x_j\right) + b \tag{2.13}$$

con $\boldsymbol{\psi} = \sum_{j=1}^{l} \alpha_j y_j \phi\left(x_j\right)$. Note que en la segunda representación el número de términos en la sumatoria es igual a la número de ejemplos de entrenamiento.

Teorema 2.1 *(Mercer) Sea X un subconjunto compacto de $\mathbb{R}^n$. Sea K una función simétrica continua tal que el operador $T_K : L_2(X) \longrightarrow L_2(X)$*

$$\left(T_K f\right)(\cdot) = \int_X K(\cdot, x) f(x) dx \tag{2.14}$$

es positivo, esto es

$$\int_{X\times X} K(x,z)\, f(x)\, f(z)\, dxdz \geq 0 \tag{2.15}$$

para todo $f \in L_2(X)$. *Entonces podemos expandir* $K(x,z)$ *en una serie uniformemente convergente en términos de las eigenfunciones de* T_K, $\phi \in L_2(X)$, *con* $\|\phi i\|_{L_2} = 1$ *y los eigenvalores asociados* $\lambda_i \geq 0$.

$$K(x,z) = \sum_{i=1}^{\infty} \lambda_i \phi_i(x)\, \phi_i(z) \tag{2.16}$$

Comentario 2.2 *La condición de positividad es equivalente a requerir que para cualquier subconjunto finito de* X, *su matriz correspondiente sea positiva semidefinida*

$$K(x,z) = \int_{X\times X} K(x,z)\, f(x)\, f(z)\, dxdz \geq 0 \in L_2 \tag{2.17}$$

Esto es para verificar que una función simétrica y continua sea un kernel es verificar la Observación 2.2. Esto es que la matriz definida al restringir la función a un conjunto finito sea definida positiva.

Proposición 2.2 *Sean* K_1 *y* K_2 *kernels sobre* $X \times X$, $X \subseteq \mathbb{R}^n$, $a \in \mathbb{R}^+$, *con* f *una función real sobre* X

$$\phi : X \to \mathbb{R}^m \tag{2.18}$$

con K_3 *un kernel sobre* $\mathbb{R}^m \times \mathbb{R}^m$, $B^{n\times n}$ *una matriz semidefinida positiva, entonces las siguientes funciones son kernels:*

1. $K(x,z) = K_1(x,z) + K_2(x,z)$
2. $K(x,z) = aK_1(x,z)$, $a \in \mathbb{R}$
3. $K(x,z) = K_1(x,z)K_2(x,z)$
4. $K(x,z) = f(x)f(z)$
5. $K(x,z) = K_3(\phi(x), \phi(z))$

6. $K(x,z) = x^T Bz$

A continuacion se dan algunas funciones que satisfacen las condiciones de Mercer.

Lineal Este kernel es una transformación lineal del tipo

$$K(x,z) = \langle Ax \cdot Az \rangle = x^T A^T Az = x^T Bz \tag{2.19}$$

donde $B = A^T A$ es una matriz semidefinida positiva.

Polinomial El mapeo polinomial es un método muy popular para modelar funciones no lineales,

$$\begin{aligned} K(x,x) &= \langle x,x \rangle^d \\ K(x,x) &= (\langle x,x \rangle + c)^d \end{aligned} \tag{2.20}$$

con $c \in \mathbb{R}$. en la practica se prefiere utilizar el segundo kernel debido a que se evitan los problemas de que el Hesiano se vuelva cero.

Funciones de Base Radial

$$K(x,z) = \exp(-\|x-z\|^2 / \sigma^2) \tag{2.21}$$

Las funciones de base radial (RBF)son también conocidas como funciones Gaussianas.

2.2. Teoría de la Generalización

Considere un conjunto de entrenamiento dado $\{x_i, y_i\}_{i=1}^n$, con la entrada $x_i \in \mathbb{R}^n$ y la salida $y_i \in \mathbb{R}$, con función de probabilidad $F(x,y)$. El problema de identificación consiste en encontrar una función $f(x)$ que aproxime de manera óptima a una salida deseada $y(x)$. Se define la función de costo $L(y_i, f(x))$ la cual dictamina como se penalizara el error de estimación. El valor esperado de la función de costo se define por la función de riesgo

$$R(w) = \int L(y_i, f(x)) dF_{x,Y}(x,y) \tag{2.22}$$

Evaluar esta función de riesgo es complicado pues usualmente no se conoce la función de distribución $F_{x,Y}(x, y)$, solo podemos usar el conjunto de entrenamiento dado para realizar la identificación. Para superar este problema podemos utilizar el principio del riesgo empírico el cual nos dice que en lugar de emplear la función de riesgo $R(w)$, podemos construir una función de riesgo empírico $R_{emp}(w)$ de la forma

$$R_{emp}(w) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x)) \tag{2.23}$$

en base a un conjunto de entrenamiento independiente e idènticamente distribuidos, (i.i.d). $\{x_i, y_i\}$, $i = 1, 2, \cdots, n$, también nos dice que si w_{emp} minimiza a $R_{emp}(w)$, entonces $R(w_{emp})$ converge en probabilidad al mínimo de todos los valores de $R(w)$, siempre que converja uniformemente a $R(w)$. Se espera que si el numero de datos de entrenamiento $n \longrightarrow \infty$, entonces el riesgo empírico $R_{emp}(w)$ converge a la función de riesgo $R(w)$. Este método es conocido como Minimización del Error Empírico. Se puede observar que este método funciona cuando se tiene un conjunto de entrenamiento muy grande, mas sin embargo si no se cuenta con dicho conjunto puede presentar el problema de overfiting. Para evitar este problema se agrega un término a la función de riesgo empírico

$$R_{emp}(w) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x)) + \frac{\lambda}{2}\|w\|^2 \tag{2.24}$$

el termino $\|w\|^2$ controla la capacidad, esto es el error de generalización y $\lambda > 0$ es una constante de regularización. Esta función se aproxima muy bien a la función de riesgo si se escoge adecuadamente a la constante de regularización. En la teoría de aprendizaje estadístico la minimización de la ec. 2.24 se le conoce como minimización del riesgo estructurado [3].

2.3. Teoría de Optimización

La teoría de optimización nos da las propiedades que debe cumplir la solución del problema de optimización las cuales nos llevan a la implementación de algoritmos de aprendizaje.

Definición 2.3 *(Problema primario de Optimización). Dadas las funciones f y g_i, $i = 1, 2, \cdots, k.$ y h_i, $i = 1, 2, \cdots, m.$ definidas sobre $\Omega \subseteq \mathbb{R}^n$.*

$$\begin{array}{lll} \textit{minimizar} & f(w) & w \in \Omega \\ \textit{sujeto a} & : \quad h_i(w) = 0, & i = 1, 2, \cdots, m \\ & \quad g_i(w) \leq 0, & i = 1, 2, \cdots, k \end{array} \tag{2.25}$$

donde $f(w)$ es la función objetivo, $h_i(w)$ y $g_i(w)$ son las restricciones de igualdad y desigualdad respectivamente, $w \in \Omega$.

Al conjunto de valores que satisfacen las condiciones 2.25 se le conoce como conjunto admisible $R = \{w \in \Omega, g(w) \leq 0, h(w) = 0\}$. Para simplificar la notación escribiremos $h(w) = 0$ y $g(w) \leq 0$ las cuales son las restricciones de igualdad y desigualdad respectivamente.

Un punto que es solución del problema de optimización es un punto $w^* \in \mathbb{R}$, tal que no existe otro punto w para el cual $f(w) < f(w^*)$, a dicho punto también se le conoce como mínimo global. Un punto $w^* \in \Omega$ se le conoce como mínimo local de $f(x)$ si existe un $\varepsilon > 0$ tal que para $f(w) \geq f(w^*)$, $\forall w \in \Omega$ tal que $\|w - w^*\| < \varepsilon$.

Definición 2.4 *(Programa Lineal y Programa Cuadrático). A un problema de optimización para el cual la función objetivo y las funciones de igualdad y desigualdad son lineales se le conoce como programa lineal, mientras que al problema de optimización para el cual la función objetivo es cuadrática y las funciones de igualdad y desigualdad son lineales se le conoce como programa cuadrático.*

La desigualdad $g_i(w) \leq 0$ se dice activa si la solución w^* satisface $g_i(w^*) = 0$ de otra forma se dice inactiva. Las restricciones de igualdad son siempre activas. Algunas veces, se introducen las variables de perdida, denotadas por ξ, para transformar a las restricciones de desigualdad $g(w)$ en restricciones de igualdad, esto es

$$g_i(w) \leq 0 \Longleftrightarrow g_i(w) + \xi_i = 0, \ \xi_i \geq 0$$

en general las variables de perdida asociadas con restricciones activas son cero, mientras que para restricciones inactivas estas indican el nivel de relajación o robustez de la restricción.

Definición 2.5 *(Función convexa). Una función real $f(x)$ es convexa $\forall w \in \mathbb{R}^n$ y para $\theta \in (0,1)$ si*

$$f\left(\theta w + (1-\theta)\, u\right) \leq \theta f\left(w\right) + (1-\theta)\, f\left(u\right) \tag{2.26}$$

si la desigualdad es estricta, la función es estrictamente convexa.

Una función $f(w)$ es convexa si su matriz Hessiana es positiva semidefinida, esto es

$$H(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \geq 0$$

Uno de los métodos para resolver el problema de optimización con restricciones de igualdad es a través de la teoría de Lagrange el cual fue extendido al problema de optimización con restricciones. Los conceptos principales de esta teoría son el concepto de multiplicadores de Lagrange y la función Lagrangiano. Dicha teoría es una generalización del resultado de Fermat en 1629, el cual plantea el problema de optimización sin restricciones. In 1951 Kuhn y Tucker generalizaron la teoría de Lagrange permitiendo la introducción de restricciones de desigualdad en el problema de optimización.

Definición 2.6 *(Función Lagrangiano). Se define la función Lagrangiano como*

$$L\left(w, \beta\right) = f\left(w\right) + \sum_{i=1}^{m} \beta_i h_i\left(w\right) \tag{2.27}$$

donde $h_i\left(w\right)$ son restricciones de igualdad y β_i son los multiplicadores de Lagrange.

Teorema 2.2 *(Lagrange). Una condición necesaria para que un punto w^* sea un mínimo de $f(w)$ sujeto a $h_i\left(w\right) = 0$, $i = 1, 2, \cdots, m$, con $f \in C^1$, es*

$$\frac{\partial L\left(w^*, \beta^*\right)}{\partial w} = 0$$

$$\frac{\partial L\left(w^*, \beta^*\right)}{\partial \beta} = 0$$

para algunos valores β^*. La condición también es suficiente siempre que $L\left(w^*, \beta^*\right)$ sea una función convexa de w.

Para el caso mas general del problema de optimización de la definición 2.25, se define el Lagrangiano generalizado.

Definición 2.7 *(Lagrangiano generalizado). Sea el problema de optimización con dominio* $\Omega \in \mathbb{R}^n$,

$$\begin{array}{lll} \textit{minimizar} & f(w), & w \in \Omega \\ \textit{sujeto a} & : \quad g_i(w) \leq 0, & i = 1, 2, \cdots, k \\ & \quad h(w) = 0 & i = 1, 2, \cdots, m \end{array} \tag{2.28}$$

se define el Lagrangiano generalizado como

$$\begin{aligned} L(w, \alpha, \beta) &= f(w) + \sum_{i=1}^{m} \alpha_i g_i(w) + \sum_{i=1}^{k} \beta_i h_i(w) h_i(w) \\ &= f(w) + \alpha g(w) + \beta h(w) \end{aligned}$$

con $\alpha = (\alpha_1, \cdots, \alpha_m)^T$ *y* $\beta = (\beta_1, \cdots, \beta_k)^T$.

Definición 2.8 *El problema dual del problema de optimización primario de la Definición 2.25, es:*

$$\begin{array}{lll} \textit{maximizar} & \theta(\alpha, \beta) \\ \textit{sujeto a} & : \quad \alpha \geq 0 \end{array} \tag{2.29}$$

donde $\theta(\alpha, \beta) = \text{ínf}_{w \subset \Omega} L(w, \alpha, \beta)$.

Otro teorema que es de gran importancia es el teorema de Dualidad débil, el cual es una de las relaciones fundamentales entre el problema primario y el problema dual.

Teorema 2.3 *(Dualidad Débil). Sea* $w \in \Omega$ *una solución admisible del problema primario de optimización y* (α, β) *una solución admisible del problema de optimización Dual. Entonces* $f(w) \geq \theta(\alpha, \beta)$.

Demostración. De la definición de $\theta(\alpha, \beta)$ para $w \in \Omega$ tenemos

$$\theta(\alpha, \beta) = \inf_{w \subset \Omega} L(w, \alpha, \beta) \leq L(w, \alpha, \beta) = f(w) + \alpha g(w) + \beta h(w)$$

luego se sigue que $\theta(\alpha, \beta) \leq f(w)$, esto debido a que w es admisible implica que $g(w) \leq 0$, $h(w) = 0$, mientras que como (α, β) es admisible implica que $\alpha \geq 0$. ■

Corolario 2.1 *El valor del problema Dual esta acotado superiormente por el valor del problema primario, esto es*

$$\sup\left\{\theta\left(\alpha,\beta\right)\right\} \leq \text{ínf}\left\{f\left(w\right) : g_i\left(w\right) \leq 0,\ h(w) = 0\right\}$$

Corolario 2.2 *Si* $\theta\left(\alpha^*,\beta^*\right) = f\left(w^*\right)$, *donde* $\alpha^* \geq 0$, $g\left(w^*\right) \leq 0$, $h\left(w^*\right) = 0$, *entonces* (α^*,β^*) *y* w^* *resuelven el problema dual y primario respectivamente. En este caso* $\alpha g(w) = 0$.

Una forma de verificar si se ha alcanzado la solución es comparar la diferencia entre los valores del problema dual y primario, si esta es cero entonces se ha alcanzado la solución. A la diferencia entre ambas soluciones se conoce como brecha de dualidad.

Teorema 2.4 *(Teorema de Dualidad Fuerte). Dado un problema de optimización con dominio* $\Omega \in \mathbb{R}^n$, *convexo,*

$$\begin{array}{lll} \textit{minimizar} & f\left(w\right)\ w \in \Omega \\ \textit{sujeto a} & : & g_i\left(w\right) \leq 0,\ i = 1,2,\cdots,k \\ & & h_i\left(w\right) = 0,\ i = 1,2,\cdots,m \end{array}$$

donde $g_i\left(w\right)$ *y* $h_i\left(w\right)$ *son funciones a fin tales que*

$$h\left(w\right) = Aw - b$$

para alguna matriz A *y un vector* b, *la brecha de dualidad es cero esto es* $\theta\left(\alpha^*,\beta^*\right) = f\left(w^*\right)$.

Este teorema garantiza que la solución del problema dual y primario tienen el mismo valor para los problemas de optimización. Ahora podemos citar el teorema de Kuhn- Tucker el cual da condiciones para la solución óptima del problema de optimización de la definición 2.25.

Teorema 2.5 *(Kuhn- Tucker). Dado un problema de optimización con dominio* $\Omega \in \mathbb{R}^n$, *convexo*

$$\begin{array}{lll} \textit{minimizar} & f\left(w\right),\ w \in \Omega \\ \textit{sujeto a} & : & g_i\left(w\right) \leq 0,\ i = 1,2,\cdots,k \\ & & h_i\left(w\right) = 0,\ i = 1,2,\cdots,m \end{array}$$

con $f \in C^1$ convexa y g_i y h_i afines, las condiciones necesarias y suficientes para que un punto w^* sea óptimo, es la existencia de α^* y β^* tales que

$$\begin{aligned} \frac{\partial L(w^*, \alpha^*, \beta^*)}{\partial w} &= 0 \\ \frac{\partial L(w^*, \alpha^*, \beta^*)}{\partial \beta} &= 0 \\ \alpha_i^* g_i(w^*) &= 0,\ i = 1, 2, \cdots, k \\ g_i(w^*) &\leq 0,\ i = 1, 2, \cdots, k \\ \alpha_i^* &\geq 0,\ i = 1, 2, \cdots, k \end{aligned}$$

La condición $\alpha_i^* g_i(w^*) = 0$ se conoce como la condición complementaria de Karush-Kuhn- Tucker. Esta condición implica que solo las restricciones activas van a tener variables duales diferentes de cero, esto significa que para ciertas optimizaciones el numero de variables envueltas va a ser mucho menor que el del conjunto de entrenamiento.

Comentario 2.3 *Este resultado nos dice que la solución puede estar en dos posiciones con respecto a las restricciones de desigualdad, puede estar en el interior de la región admisible, con la restricción inactiva, o en la frontera de la restricción con la restricción activa. Esto es si la restricción es activa significa que $g_i(w^*) = 0$, o $\alpha_i^* = 0$.*

Mas adelante se vera que los multiplicadores de Lagrange $\alpha \neq 0$, se les llama Vectores de Soporte.

Debido a que tratar el problema 2.25 directamente con restricciones de desigualdad resulta complicado, podemos definir un problema dual el cual es mas sencillo de resolver. El método de Lagrange para problemas de optimización convexos tiene una representación dual la cual se obtiene introduciendo al Lagrangiano los multiplicadores de Lagrange o variables duales. Los métodos duales se basan en la idea de que las variables fundamentales del problema son las duales. La obtención del problema dual se obtiene igualando a cero la primer derivada parcial con respecto a las variables primarias del Lagrangiano y sustituyendo las relaciones obtenidas dentro del Lagrangiano. Esto es

$$\theta(\alpha, \beta) = \inf_{w \subset \Omega} L(w, \alpha, \beta)$$

la función resultante contiene solo variables duales la cual debe ser maximizada con respecto a restricciones simples. Recordando que maximizar $f(w)$ es lo mismo que minimizar $(-f(w))$.

2.4. SVM para Regresión

En SVM la meta es encontrar una función $f(x)$ que tenga a lo más una desviación ε de la salida y_i para todos los datos de entrenamiento, y al mismo tiempo, que sea lo más mínima posible. En otras palabras, no nos preocupamos por errores menores a ε, pero si de aquellos que sean mayores. Considere el problema de aproximar un conjunto de entrenamiento $\{x_i, y_i\}$, $x \in \mathbb{R}^n$, $y \in \mathbb{R}$, por medio de una función lineal $f : X \subseteq \mathbb{R}^n \to \mathbb{R}$,

$$\begin{aligned} f(x) &= \langle w \cdot x \rangle + b \\ &= x^T w + b \\ &= \sum_{i=1}^{n} w_i x_i + b \end{aligned} \tag{2.30}$$

donde $\langle \cdot \rangle$ es el símbolo del producto interno, y $x = \{x_1, x_2, \cdots, x_n\}$, $y = \{y_1, y_2, \cdots, y_n\}$ y $w = \{w_1, w_2, \cdots, w_n\}$.donde w es el vector de pesos y b es el bias. En el caso de 2.30 se busca encontrar w lo más mínima posible. Para asegurar esto, una forma es mínimizar la norma Euclidiana, i.e. $\|w\|^2$. Formalmente podemos escribir este problema como un problema de optimización convexo: fue qe

$$\begin{aligned} &\text{minimizar} && \tfrac{1}{2}\|w\|^2 \\ &\text{sujeto a :} && y_i - (\langle w \cdot x_i \rangle + b) \leq \varepsilon \\ & && y_i - (\langle w \cdot x_i \rangle + b) \geq -\varepsilon \end{aligned} \tag{2.31}$$

La sunción clave en 2.31 fue que la función $f(x)$ existe y aproxima a todos los pares (x_i, y_i) con una precisión ε, o en otras palabras que el problema convexo es factible. Algunas veces este no es el caso por lo cual se introducen las variables de perdida. En SVM para regresión la idea básica consiste en realizar un mapeo de los datos de entrenamiento $x \in X$, a un espacio de mayor dimensión F a través de un mapeo no lineal $\phi : X \longrightarrow F$, donde podemos

realizar una regresión lineal. En la Figura 2.2 se puede observar la arquitectura generada por SVM para regresión, tambien se puede observar que el regresor final solo depende de todos aquellos datos que contienen la mayor información posible para generar el regresor, llamados vectores de soporte.

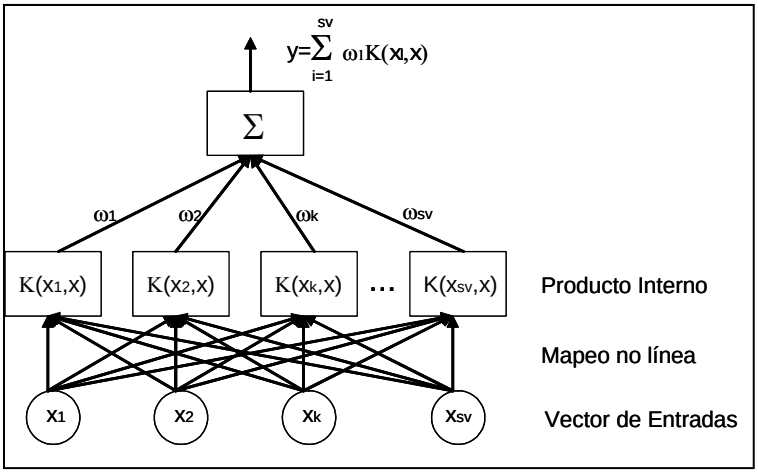

Figura 2.2: Arquitectura de SVM para regresión

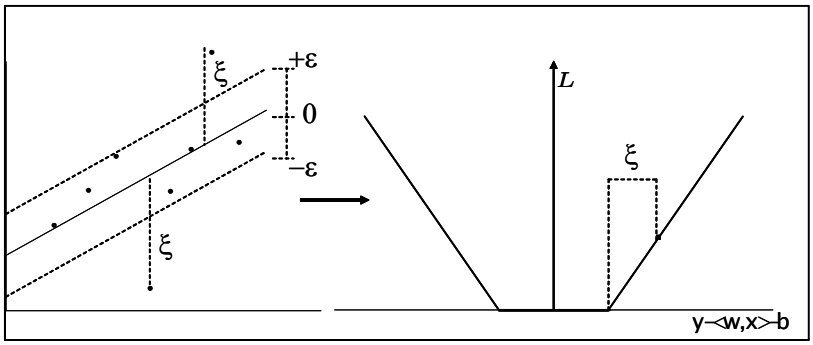

Figura 2.3: Variables de Perdida ξ.

En la Figura 2.3 se muestra las variables de perdida para un ejemplo de identificación con banda ε insensitive.

SVM utiliza una función de costo originalmente propuesta por Vaptnik(1995) se define a continuación la función de costo $\varepsilon-$ insensitive.

Definición 2.9 *La función de costo, o discrepacia, $\varepsilon-$ insensitive, $L^{\varepsilon}(x, y, f)$ se define como*

$$L^{\varepsilon}(x, y, f) = |y - f(x)| = \text{máx}\,(0, |y - f(x)| - \epsilon)$$

donde f es una función real con dominio X, x∈ X, y ∈ ℝ.

2.5. LS-SVM

El método de Mínimos cuadrados SVM para identificación trabaja fuera de línea pero la ventaja que tiene en comparación con los métodos ε insensitive y cuadrático es que la solución del problema se realiza a través de un conjunto de ecuaciones. Este método puede tratar una cantidad mas considerable de datos de entrenamiento.

Dado un conjunto de entrenamiento $\{x_i, y_i\}_{i=1}^{n}$, $x \in \mathbb{R}^n$, $y \in \mathbb{R}$. En el espacio de características el modelo dado por las Maquinas de Vectores de Soporte toma la siguiente forma:

$$f(x) = \sum_{i=1}^{n} w_i \phi_i(x) + b \tag{2.32}$$

donde el mapeo $\phi : X \longrightarrow F$, mapea el espacio de entradas a algún espacio de características de mayor dimensión. En el metodo LS-SVM se plantea el siguiente problema de minimización

$$\begin{array}{ll} \text{mín}\, J(w, \xi) & = \frac{1}{2}\|w\|^2 + \frac{C}{2}\sum_{i=1}^{l}\xi_i^2 \\ \text{sujeto a} \;: & y_i - \langle w \cdot \phi(x_i)\rangle = \xi_i \end{array} \tag{2.33}$$

este problema corresponde al problema de SVM Ridge regresión. El Lagrangiano esta dado por

$$L(w, b, \alpha, \xi) \;\; = \frac{1}{2}\|w\|^2 \frac{C}{2}\sum_{i=1}^{l}\xi_i^2 - \sum_{i=1}^{l}\alpha_i\left(\langle w \cdot \phi(x_i)\rangle + b + \xi_i - y_i\right) \tag{2.34}$$

donde α_i son lo multiplicadores de Lagrange. Calculando las derivadas parciales del Lagrangiano 2.34, con respecto de todas las variables e igualando a cero podemos obtener las

condiciones de optimización, esto debido a que la función es convexa

$$\begin{array}{rcl} \frac{\partial L}{\partial w} = 0 & \longrightarrow & w = \sum_{i=1}^{l} \alpha_i \phi\left(x_i\right) \\ \frac{\partial L}{\partial b} = 0 & \longrightarrow & \sum_{i=1}^{l} \alpha_i = 0 \\ \frac{\partial L}{\partial \xi} = 0 & \longrightarrow & \alpha_i = C\xi_i \\ \frac{\partial L}{\partial \alpha} = 0 & \longrightarrow & y_i = \langle w \cdot \phi\left(x_i\right)\rangle + b + \xi_i \end{array} \tag{2.35}$$

eliminando w y ξ y utilizando la función kernel, obtenemos el siguiente sistema de ecuaciones lineales:

$$\begin{array}{rl} y_i = & \sum_{i=1}^{l} \alpha_i \phi\left(x_i\right) \phi\left(x_i\right) + b + \frac{\alpha_i}{C} \\ \sum_{i=1}^{l} \alpha_i = & 0 \end{array} \tag{2.36}$$

$$\left[\begin{array}{cc} 0 & \bar{1}^T \\ \bar{1} & K + \frac{1}{C} I \end{array}\right] \left[\begin{array}{c} b \\ \alpha \end{array}\right] = \left[\begin{array}{c} 0 \\ y \end{array}\right] \tag{2.37}$$

donde $y = \left[y_1, \cdots, y_i\right]^T$. Entonces el vector $1 = \left[1; \cdots 1\right]^T$. Dado este sistema de ecuaciones 2.37, el modelo utilizado por el método LS-SVM se expresa de la siguiente forma

$$\hat{f}\left(x\right) = K\left(x, x_i\right)\alpha + b$$

Notese que este método no requiere de la determinación del paramétro de precsión ϵ el cual esta relacionado con la función de costo $\epsilon-$ insensitive del método propuesto por Vaptnik [4]. La gran desventaja de este método es que la solución del modelo no es sparse esto es que ninguna de las soluciónes α_i se desvanece en la solución optima, por ende el modelo depende de todos los datos de entrenamiento.

Capítulo 3

SVM recursivo para identificación en línea

En esta capitulo se plantea el método SVM para identificación en línea, basado en el método propuesto en [15], para obtener una solución sparse, en el sentido de que el modelo no dependa de todos lo datos de entrenamiento. El modelo propuesto es el utilizado en [16] donde el regresor no utiliza el termino bias. La solución dada por el método SVM para identificación en línea al problema de regresión es de la forma

$$\tilde{f}(x) = \sum_{i=1}^{t} \alpha_i k(x_i, x) \tag{3.1}$$

donde $\{x_i\}_{i=1}^{t}$ son los puntos de entrenamiento y $k(\cdot,\cdot)$ es la función Kernel la cual depende del numero de datos de entrenamiento, es decir del tiempo. Para evitar esto se utiliza un método de Aproximación Linealmente Dependiente en línea que admite secuencialmente dentro de la representación del Kernel solo aquellos ejemplos que pueden ser representados como una combinación lineal de los ejemplos anteriores en el espacio de características.

3.1. Aproximación linealmente dependiente en línea

El termino sparse significa que al final de la solución de SVM la mayoría de las variables se desvanecen, en nuestro caso por sparse nos referiremos a que podemos aproximar a los datos futuros por una combinación lineal de los datos anteriores obteniendo así un modelo que no dependa de todos los datos de entrenamiento. En un escenario en línea, para el estimado de f en el tiempo t, la solución esta dada por

$$\begin{aligned} \tilde{f}(x) &= \sum_{i=1}^{t} \alpha_i k(x_i, x) \\ &= \sum_{i=1}^{t} \alpha_i \langle \phi(x_i), \phi(x) \rangle \end{aligned} \tag{3.2}$$

la cual es de hecho un regresor en el espacio de Hilbert. La solución mas sencilla es considerar todos los datos de entrenamiento $x_1, \cdots, x_t$ en la solución. Esto nos lleva a un algoritmo el cual crece en complejidad cada ves que se agrega un nuevo dato a la representación del Kernel. Si el punto x_l satisface que $\phi(x_t) = \sum_{i=1}^{t-1} a_i \phi(x_i)$, entonces no hay necesidad de tener coeficientes α_i distintos de cero para $\phi(x_t)$ en el tiempo t, debido a que puede ser representado por los datos de entrenamiento anteriores. Para esto se asume que se da un conjunto de datos de entrenamiento secuencialmente de la forma $\{(x_1, y_1), (x_2, y_2), \cdots\}$, $x_i \in \mathcal{X}$. Asumiendo que para el tiempo t, después de haber observado $t-1$ datos de entrenamiento $\{x_i\}_{i=1}^{t-1}$, hemos recopilado un *"diccionario"* el cual consistente de un subconjunto de datos de entrenamiento $D = \{\tilde{x}_j\}_{j=1}^{m}$, con $m = t-1$, donde, por construcción, $\{\phi(\tilde{x}_j)\}_{j=1}^{m}$ son vectores linealmente independientes en el espacio de características.

Al recibir el nuevo dato de entrenamiento x_t tenemos que verificar si $\phi(x_t)$ puede ser representado como una combinación lineal de los datos del diccionario, esto es, verificar si $\phi(x_t)$ es aproximadamente linealmente dependiente (ALD) de los vectores del diccionario. Si este no es el caso, agregamos el nuevo dato al diccionario.

Para evitar agregar el nuevo dato x_t al diccionario necesitamos encontrar coeficientes

$a = (a_1, \cdots, a_m)^T$ que satisfagan la condición 3.3 (ALD).

$$\delta_t = \min \left\| \sum_{i=1}^{m} a_i \phi(x_i) - \phi(x_l) \right\|^2 \leq \nu \tag{3.3}$$

donde ν es el nivel de precisión que determina el nivel de aproximación o que tan sparse sea la solución. Si se cumple la condición ALD, $\phi(x_t)$ puede ser aproximado dentro del error cuadrático ν por alguna combinación lineal de los vectores del diccionario, $D = \{\tilde{x}_j\}_{j=1}^{m}$.

Realizando la minimización de la condición ALD podemos verificar si se satisface la condición y encontrar los coeficientes a para el algoritmo.

$$\begin{aligned}
\delta_t &= \min_a \left\{ \sum_{i,j=1}^{m_{t-1}} a_i a_j \left\langle \phi(\tilde{x}_i), \phi(\tilde{x}_j) \right\rangle - 2 \sum_{j=1}^{m_{l-1}} a_j \left\langle \phi(\tilde{x}_j), \phi(x_t) \right\rangle + \left\langle \phi(x_t), \phi(x_t) \right\rangle \right\} \qquad (3.4) \\
\delta_t &= \min_a \left\{ \sum_{i,j=1}^{m_{t-1}} a_i a_j k\left(\tilde{x}_i, \tilde{x}_j\right) - 2 \sum_{j=1}^{m_{t-1}} a_j k\left(\tilde{x}_j, \tilde{x}_t\right) + k\left(\tilde{x}_t, \tilde{x}_t\right) \right\} \\
\delta_t &= \min_a \left\{ a^T \tilde{K}_{t-1} a - 2 a^T \tilde{k}_{t-1} + k_{tt} \right\}
\end{aligned}$$

donde

$$\begin{aligned}
\left[\tilde{K}_{t-1}\right]_{i,j} &= k\left(\tilde{x}_i, \tilde{x}_j\right) \\
\left(\tilde{k}_{t-1}\left(x_t\right)\right)_i &= k\left(\tilde{x}_j, \tilde{x}_t\right) \\
k_{tt} &= k\left(\tilde{x}_t, \tilde{x}_t\right)
\end{aligned}$$

con $i, j = 1, 2, \cdots, m$. Resolviendo el problema de minimización de la ecuación (3.4), tenemos que:

$$\begin{aligned}
\frac{\partial \left(a^T \tilde{K}_{t-1} a - 2 a^T \tilde{k}_{t-1}\left(x_l\right) + k_{tt} \right)}{\partial a} &= 0 \qquad (3.5) \\
2\tilde{K}_{t-1} a - 2\tilde{k}_{t-1}\left(x_t\right) &= 0 \\
\tilde{K}_{t-1}^{-1} \tilde{k}_{t-1}\left(x_t\right) &= a
\end{aligned}$$

sustituyendo la solución (3.5) en el problema 3.4 obtenemos

$$\begin{aligned}
\delta_t &= \left(\tilde{K}_t^{-1}\tilde{k}_t\right)^T \tilde{K}_t\tilde{K}_t^{-1}\tilde{k}_t - 2\left(\tilde{K}_t^{-1}\tilde{k}_t\right)^T \tilde{k}_t + k_{tt} \qquad (3.6)\\
\delta_t &= \tilde{k}_t^T\tilde{K}_t^{-1T}\tilde{K}_t\tilde{K}_t^{-1}\tilde{k}_t - 2\tilde{k}_t^T\tilde{K}_t^{-1T}\tilde{k}_t + k_{tt}\\
\delta_t &= \tilde{k}_t^T\tilde{K}_t^{-1T}\tilde{k}_t - 2\tilde{k}_t^T\tilde{K}_t^{-1T}\tilde{k}_t + k_{tt}\\
\delta_t &= k_{tt} - \tilde{k}_t^T\tilde{K}_t^{-1T}\tilde{k}_t\\
\delta_t &= k_{tt} - \tilde{k}_t^T a
\end{aligned}$$

esto debido a que K es simétrica. Si no se cumple la condición 3.3, esto es, si $\delta_t > \nu$, entonces debemos expandir el diccionario aumentando el dato x_t esto es: $D_t = D_{t-1} \cup x_t$ y $m_t = m + 1$. Usando el diccionario actualizado $\phi(x_t)$ se representara por si solo debido a que se encuentra en el diccionario. Consecuentemente para cada paso i hasta t podemos representar $\phi(x_t)$ como

$$\phi(x_t) = \sum_{i=1}^{m} a_{i,j}\phi(\tilde{x}_j) + \phi_i^{res}, \qquad \|\phi_i^{res}\|^2 \leq \nu \qquad (3.7)$$

donde ϕ_i^{res} denota al componente de residuo del vector. Escogiendo ν lo suficientemente pequeña, podemos hacer el error, en la aproximación de $\Phi = \phi(x_t) \approx \sum_{j=1}^{m} a_{i,j}\phi(\tilde{x}_j)$, lo bastante pequeño como para obtener una buena aproximación de K. La aproximación en términos de matrices esta dada por

$$K_t \approx A_t\tilde{K}_t A_t^T \qquad (3.8)$$

donde $[K_t]_{i,j} = k(x_i, x_j)$ con $i, j = 1, \cdots, t$ es la matriz Kernel completa y $[A_t]_{i,j} = a_{i,j}$ es una matriz creciente $t \times m$ a la cual se le agrega el vector a^T en cada paso t.

Teorema 3.1 *(Yaakov Engel, Shie Mannor 2004) Asumiendo que i) K es un Kernel de Mercer y ii) y que $\mathcal{X}$ es un subconjunto de un espacio de Banach. Entonces para cualquier secuencia de datos de entrenamiento $\{x_i\}_{i=1}^{\infty}$ y para cualquier $\nu > 0$ el numero de vectores en el diccionario es finito.*

Este teorema nos asegura que después de un periodo inicial de tiempo, el algoritmo se vuelve independiente del tiempo y solo depende de la dimensión del diccionario. Esto hace que el algoritmo sea adecuado para un ambiente en línea.

3.2. SVM Recursivo

En esta seccion se mostrara un algoritmo con cotas en la memoria y el tiempo, por paso, que sea dependiente de solo de los datos del diccionario de dimensión m en el espacio de caracteristicas. Acontinuación se muestra que la idea de aproximación linealmente dependiente en linea puede ser utilizada para generar una versión de SVM para identificación en línea.

Para realizar la identificación se utiliza una función la cual es lineal en los parámetros w

$$\tilde{f}(x) = \sum_{i=1}^{t} K(x, x_i)\,\alpha_i = \langle w \cdot \phi(x) \rangle \tag{3.9}$$

donde $w = \sum_{i=1}^{t} \alpha_i \phi_i(x)$. Empleando la función de perdida $L_1 = \max\{0, |y - f(x)|\}$, Laplace [5], (i.e., $\epsilon = 0$), en la función de riesgo 2.24, obtenemos la siguiente función de riesgo

$$R = \frac{1}{2}\|w\|^2 + \sum_{i=1}^{t} |y_i - f(x)| \tag{3.10}$$

introduciendo las variables de perdida, (3.10) puede ser minimizada esto es

$$\begin{aligned} \text{minimizar} \quad & \tfrac{1}{2}\|w\|^2 + C\sum_{i=1}^{t}(\xi_i + \xi_i^*) \\ \text{sujeto a:} \quad & y_i - \langle w \cdot \phi(x) \rangle \leq \xi_i, \\ & \langle w \cdot \phi(x) \rangle - y_i \leq \xi_i^*, \\ & \xi_i \geq 0, \xi_i^* \geq 0,\ i = 1, 2, \cdots, t \end{aligned} \tag{3.11}$$

Para resolver el problema de minimización 3.11, se introducen los multiplicadores de Lagrange para reescribir el problema de la siguiente manera

$$\begin{aligned} \mathcal{L}(w, \xi, \xi^*, \hat{\alpha}, \alpha^*, \beta, \beta^*) \;=\; & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{t}(\xi_i + \xi_i^*) - \sum_{i=1}^{t}(\beta_i \xi_i + \beta_i^* \xi_i^*) \\ & - \sum_{i=1}^{t} \hat{\alpha}_i (\xi_i - y_i + \langle w \cdot \phi_i(x) \rangle) - \sum_{i=1}^{t} \alpha_i^* (\xi_i^* + y_i - \langle w \cdot \phi_i(x) \rangle) \end{aligned} \tag{3.12}$$

y obteniendo las derivadas parciales con respecto a las variables primarias y estableciendo condiciones estacionarias tenemos

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial w} &= 0 \quad \Rightarrow w = \sum_{i=1}^{t} \left(\hat{\alpha}_i - \alpha_i^*\right) \phi_i\left(x\right) \\ \frac{\partial \mathcal{L}}{\partial \xi^{(*)}} &= 0 \quad \Rightarrow C - \alpha_i^{(*)} = \beta_i^{(*)} \end{aligned} \tag{3.13}$$

donde $\xi^{(*)}$=ξ, ξ^*, $\alpha_i^{(*)} = \alpha^*, \hat{\alpha}$ y $\beta_i^{(*)} = \beta, \beta^*$. sustituyendo 3.13 en 3.11 obtenemos el problema dual de 3.11.

$$\begin{aligned} \text{minimizar} \quad & \begin{cases} \frac{1}{2} \sum_{i,j=1}^{t} \left(\hat{\alpha}_i - \alpha_i^*\right) \left(\hat{\alpha}_j - \alpha_j^*\right) \left(\left\langle \phi_i\left(x\right) \cdot \phi_j\left(x\right) \right\rangle\right) \\ -\sum_{i=1}^{t} \left(\hat{\alpha}_i - \alpha_i^*\right) y_i \end{cases} \\ \text{sujeto a} \quad & \left\{ -C \leq \hat{\alpha}_i, \alpha_i^* \leq C \right. \end{aligned} \tag{3.14}$$

.

Sustituyendo $\alpha = \hat{\alpha}_i - \alpha_i^*$, usando la relación $\hat{\alpha}_i \alpha_i^* = 0$ y el truco del kernel para calcular el producto interno en el espacio de características, es posible reescribir el problema de optimización dual 3.14 como

$$\text{minimizar} \quad \begin{cases} \frac{1}{2} \sum_{i,j=1}^{t} \alpha_i \alpha_j K\left(x, x_i\right) - \sum_{i=1}^{t} \alpha_i y_i \end{cases} \tag{3.15}$$

Por conveniencia escribimos a 3.15 en su forma vectorial

$$W(\alpha) = \alpha^T K \alpha - y^T \alpha \tag{3.16}$$

Diferenciando a 3.16 con respecto a la variable dual y estableciendo condiciones estacionarias obtenemos la solución

$$K\alpha - y = 0 \tag{3.17}$$

donde la solución $\alpha = \left[\alpha_1, \cdots, \alpha_t\right]^T$ esta dada por $\alpha = K^{\dagger} y$, la cual puede ser resuelta por el método convencional de mínimos cuadrados recursivo (RLS), pero en un ambiente en línea,

donde los datos son adquiridos de manera recursiva, resulta imposible utilizar este método. Primero, resulta difícil mantener en memoria a la matriz kernel $K\left(x, x_i\right)$, debido a que esta aumenta su dimensión conforme se van adquiriendo los datos de entrenamiento y segundo la dimensión del modelo es igual al numero de datos de entrenamiento causando overfitting.

Puesto que el algoritmo debe ser independiente del tiempo, el método de mínimos cuadrados recursivo no es aceptable. Haciendo uso del método de aproximación linealmente independiente en línea se puede resolver el problema de la dependencia del tiempo, utilizando una matriz kernel $\tilde{K}_t \in m_l \times m_l$, con $l < t$, en lugar de la matriz $K_t \in m_t \times m_t$ la cual depende de todos los datos de entrenamiento. Para esto redefinimos a w_t como

$$w_t = \Phi_t \alpha_t = \tilde{\Phi}_t A_t^T \alpha_t = \tilde{\Phi}_t \tilde{\alpha}_t \tag{3.18}$$

donde $\Phi_t = \left[\phi\left(x_1\right), \cdots, \phi\left(x_t\right)\right]$ y $\tilde{\Phi}_t = \left[\phi\left(\tilde{x}_1\right), \cdots, \phi\left(\tilde{x}_m\right)\right]$ y $\tilde{\alpha}_t = A_t^T \alpha_t$, $\tilde{\alpha}_t \in \mathbb{R}^m$. Ahora podemos minímizar el error cuadrático

$$\begin{aligned} L(\alpha) &= \left\|\Phi_t^T w - y_t\right\|^2 \\ &= \left\|\Phi_t^T \tilde{\Phi}_t \tilde{\alpha}_t - y_t\right\|^2 \\ &= \left\|A_t \tilde{\Phi}_t^T \tilde{\Phi}_t \tilde{\alpha}_t - y_t\right\|^2 \\ &= \left\|A_t \tilde{K}_t \tilde{\alpha}_t - y_t\right\|^2 \end{aligned} \tag{3.19}$$

a la expreción 3.19 la podemos escribir de la siguiente forma

$$\begin{aligned} L(\tilde{\alpha}) &= \left\|A_t \tilde{K}_t \tilde{\alpha}_t - y_t\right\|^2 \\ &= \left(A_t \tilde{K}_t \tilde{\alpha}_t - y_t\right)^T \left(A_t \tilde{K}_t \tilde{\alpha}_t - y_t\right) \\ &= \left(A_t \tilde{K}_t \tilde{\alpha}_t\right)^T A_t \tilde{K}_t \tilde{\alpha}_t - \left(A_t \tilde{K}_t \tilde{\alpha}_t\right)^T y_t - y_t^T A_t \tilde{K}_t \tilde{\alpha}_t - y_t^T y_t \\ &= \tilde{\alpha}_t^T \tilde{K}_t^T A_t^T A_t \tilde{K}_t \tilde{\alpha}_t - \tilde{\alpha}_t^T \tilde{K}_t^T A_t^T y_t - \tilde{\alpha}_t^T \tilde{K}_t^T A_t^T y_t - y_t^T y_t \end{aligned} \tag{3.20}$$

tomado la derivada parcial de $L(\tilde{\alpha})$ con respecto a $\tilde{\alpha}$ e imponiendo condiciones estacionarias tenemos

$$\frac{\partial L(\tilde{\alpha})}{\partial \tilde{\alpha}} = 2\tilde{K}_t^T A_t^T A_t \tilde{K}_t \tilde{\alpha}_t - 2\tilde{K}_t^T A_t^T y_t = 0 \tag{3.21}$$

apartir de la relación 3.21 tenemos que $\tilde{K}_t^T A_t^T A_t \tilde{K}_t \tilde{\alpha}_t = \tilde{K}_t^T A_t^T y_t$ de lo cual se obtiene la solución

$$\begin{aligned} \tilde{\alpha}_l &= \left(A_t \tilde{K}_t\right)^{-1} y_t \\ &= \tilde{K}_t \left(A_t^T A_t\right)^{-1} A_t^T y_t \end{aligned} \tag{3.22}$$

la cual puede ser resuelta utilizando el método propuesto en [17], por lo cual nos podemos encontrar con dos casos.

1. El dato de entrenamiento $\phi(x_t)$ es ALD del diccionario D_{t-1}, esto es $\delta_t \leq \nu$, y a_t esta dada por la ecuación 3.5. En este caso $D_t = D_{t-1}$ y actualizamos a $m_t = m_{t-1}$ y $\tilde{K}_t = \tilde{K}_{t-1}$.

2. $\delta_t > \nu$, por lo tanto, $\phi(x_t)$ no es ALD de D_{t-1}. x_t es agregado al diccionario esto es $D_t = D_{t-1} \cup \{x_t\}$, $m_t = m_{t-1} + 1$, y $\tilde{K}_t$ crece.

Caso 3.1 *En este caso, solamente A, cambia en cada paso, esto es: $A_t = \left[A_{t-1}^T, a_t\right]^T$. Por lo tanto $A_t^T A_t = A_{t-1}^T A_{t-1} + a_t a_t^T$, y $A_t^T y_t = A_{t-1}^T y_{t-1} + a_t y_t$. Note que $\tilde{K}_t$ no cambia.*

$$\tilde{\alpha}_t = \tilde{K}_t \left(A_t^T A_t\right)^{-1} A_t^T y_t$$

definiendo a

$$\begin{aligned} G_t &= A_t^T A_t \\ &= A_{t-1}^T A_{t-1} + a_t a_t^T \\ &= G_{t-1} + a_t a_t^T \end{aligned}$$

utilizando el lema de la inverción de matrices,

$\left(A + BC^{-1}D\right)^{-1} = A^{-1} - A^{-1}B\left(C + DA^{-1}B\right)^{-1} DA^{-1}$, *con* $A = A_{t-1}^T A_{t-1}$, $B = a_t$, $C^{-1} = I$, *y* $D = a_t^T$, *podemos rescribir a* G_t^{-1} *como:*

$$\begin{aligned} G_t^{-1} &= \left(A_{t-1}^T A_{t-1}\right)^{-1} - \left(A_{t-1}^T A_{t-1}\right)^{-1} a_t \left(I + a_t^T \left(A_{t-1}^T A_{t-1}\right)^{-1} a_t\right)^{-1} a_t^T \left(A_{t-1}^T A_{t-1}\right)^{-1} \\ &= G_{t-1}^{-1} - G_{t-1}^{-1} a_t \left(I + a_t^T G_{t-1}^{-1} a_t\right)^{-1} a_t^T G_{t-1}^{-1} \end{aligned}$$

definiendo $P_t = G_t^{-1} = \left(A_t^T A_t\right)^{-1}$ *y* $P_{t-1} = G_{t-1}^{-1} = \left(A_{t-1}^T A_{t-1}\right)^{-1}$, *tenemos:*

$$P_t = P_{t-1} - \frac{P_{t-1} a_t a_t^T P_{t-1}}{(I + a_t^T P_{t-1} a_t)} \tag{3.23}$$

definiendo $q_t = P_{t-1} a_t / \left(I + a_t^T P_{t-1} a_t\right)$*podemos rescribir esta ecuación de la siguiente manera:*

$$P_t = P_{t-1} - q_t a_t^T P_{t-1}$$

Sustituyendo esta expreción en $\tilde{\alpha}_t$ *obtenemos la siguiente expresión*

$$\begin{aligned} \tilde{\alpha}_t &= \tilde{K}_t P_t A_t^T y_t \\ &= \tilde{K}_t \left(P_{t-1} - q_t a_t^T P_{t-1}\right) A_t^T y_t \end{aligned}$$

observemos que $A_t^T y_t = A_{t-1}^T y_{t-1} + a_t y_t$

$$\begin{aligned} \tilde{\alpha}_t &= \tilde{K}_t \left(P_{t-1} - q_t a_t^T P_{t-1}\right) \left(A_{t-1}^T y_{t-1} + a_t y_t\right) \\ &= \left(\tilde{K}_t P_{t-1} - \tilde{K}_t q_t a_t^T P_{t-1}\right) \left(A_{t-1}^T y_{t-1} + a_t y_t\right) \\ &= \tilde{K}_t P_{t-1} A_{t-1}^T y_{t-1} + \tilde{K}_t P_{t-1} a_t y_t - \tilde{K}_t q_t a_t^T P_{t-1} A_{t-1}^T y_{t-1} - \tilde{K}_t q_t a_t^T P_{t-1} a_t y_t \\ &= \tilde{\alpha}_{t-1} + \tilde{K}_t P_{t-1} a_t y_t - \tilde{K}_t q_t a_t^T P_{t-1} A_{t-1}^T y_{t-1} - \tilde{K}_t q_t a_t^T P_{t-1} a_t y_t \\ &= \tilde{\alpha}_{t-1} + \tilde{K}_t \left[P_{t-1} a_t y_t - q_t a_t^T P_{t-1} A_{t-1}^T y_{t-1} - q_t a_t^T P_{t-1} a_t y_t\right] \\ &= \tilde{\alpha}_{t-1} + \tilde{K}_t \left[P_{t-1} a_t y_t - q_t a_t^T P_{t-1} a_t y_t - q_t a_t^T \left(\tilde{K}_t \tilde{K}_t^{-1}\right) P_{t-1} A_{t-1}^T y_{t-1}\right] \\ &= \tilde{\alpha}_{t-1} + \tilde{K}_t \left[\left(P_{t-1} - \frac{P_{t-1} a_t a_t^T P_{t-1}}{(I + a_t^T P_{t-1} a_t)}\right) a_t y_t - q_t a_t^T \tilde{K}_t \left(\tilde{K}_t^{-1} P_{t-1} A_{t-1}^T y_{t-1}\right)\right] \\ &= \tilde{\alpha}_{t-1} + \tilde{K}_t \left[P_t a_t y_t - q_t a_t^T \tilde{K}_t \tilde{\alpha}_{t-1}\right] \\ &= \tilde{\alpha}_{t-1} + \tilde{K}_t q_t \left[y_t - k_{t-1}\left(x_t\right) \tilde{\alpha}_{t-1}\right] \end{aligned} \tag{3.24}$$

con $q_t = P_t a_t$ *y* $k_{t-1}\left(x_t\right) = \tilde{K}_t a_t$.

Caso 3.2 *Aqui* $\tilde{K}_t \neq \tilde{K}_{t-1}$, *pero se puede obtener de una manera recursiva de la siguiente manera:*

$$\tilde{K}_t = \begin{bmatrix} \tilde{K}_{t-1} & \tilde{k}_{t-1}\left(x_t\right) \\ \tilde{k}_{t-1}\left(x_t\right)^T & k_{tt} \end{bmatrix}$$

y su inversa es

$$
\begin{aligned}
\tilde{K}_t^{-1} &= \begin{bmatrix} \tilde{K}_{t-1}^{-1} + \frac{\tilde{K}_{t-1}^{-1}\tilde{k}_{t-1}(x_t)\tilde{k}_{t-1}(x_t)^T\tilde{K}_{t-1}}{\left(k_{tt}-\tilde{k}_{t-1}(x_t)^T\tilde{K}_{t-1}^{-1}\tilde{k}_{t-1}(x_t)\right)} & \frac{-\tilde{K}_{t-1}^{-1}\tilde{k}_{t-1}(x_t)}{\left(k_t-\tilde{k}_{t-1}(x_t)^T\tilde{K}_{t-1}^{-1}\tilde{k}_{t-1}(x_t)\right)} \\ \frac{-\tilde{k}_{t-1}(x_t)^T\tilde{K}_{t-1}^{-1}}{\left(k_t-\tilde{k}_{t-1}(x_t)^T\tilde{K}_{t-1}^{-1}\tilde{k}_{t-1}(x_t)\right)} & \frac{1}{\left(k_{tt}-\tilde{k}_{t-1}(x_t)^T\tilde{K}_{t-1}^{-1}\tilde{k}_{t-1}(x_t)\right)} \end{bmatrix} \\
&= \begin{bmatrix} \tilde{K}_{t-1}^{-1} + \frac{a_t a_t^T}{\left(k_{tt}-\tilde{k}_{t-1}(x_t)^T a_t\right)} & \frac{-\tilde{K}_{t-1}^{-1}\tilde{k}_{t-1}(x_t)}{\delta_t} \\ \frac{-\tilde{k}_{t-1}(x_t)^T\tilde{K}_{t-1}^{-1}}{\delta_t} & \frac{1}{\delta_t} \end{bmatrix} \\
&= \begin{bmatrix} \tilde{K}_{t-1}^{-1} + \frac{a_t a_t^T}{\delta_t} & \frac{-\tilde{K}_{l-1}^{-1}\tilde{k}_{l-1}(x_l)}{\delta_t} \\ \frac{-\tilde{k}_{t-1}(x_t)^T\tilde{K}_{t-1}^{-1}}{\delta_t} & \frac{1}{\delta_t} \end{bmatrix} \\
&= \frac{1}{\delta_t}\begin{bmatrix} \delta\tilde{K}_{t-1}^{-1} + a_t a_t^T & -\tilde{K}_{t-1}^{-1}\tilde{k}_{t-1}(x_t) \\ -\tilde{k}_{t-1}(x_t)^T\tilde{K}_{t-1}^{-1} & 1 \end{bmatrix} \\
&= \frac{1}{\delta_t}\begin{bmatrix} \delta_t\tilde{K}_{t-1}^{-1} + a_t a_t^T & -a_t \\ -a_t^T & 1 \end{bmatrix}
\end{aligned} \tag{3.25}
$$

con a y δ dadas por las ecuaciones 3.5 y 3.6 respectivamente. Debido a que x_t se agrega al diccionario, $\phi(x_t)$ se representa de manera propia. Consecuentemente

$$
A_t = \begin{bmatrix} A_{t-1} & 0 \\ 0^T & 1 \end{bmatrix}; \qquad A_t^T A_t = \begin{bmatrix} A_{t-1}^T A_{t-1} & 0 \\ 0^T & 1 \end{bmatrix} \tag{3.26}
$$

$$
P_t = \left(A_t^T A_t\right)^{-1} = \begin{bmatrix} P_{t-1}^{-1} & 0 \\ 0^T & 1 \end{bmatrix} \tag{3.27}
$$

donde $0 = [0_1, \cdots, 0_m]^T$. *La actualización de las variables* $\tilde{\alpha}$ *esta dada por:*

$$\begin{aligned}
\tilde{\alpha}_t &= \tilde{K}_t^{-1}\left(A_t^T A_t\right)^{-1} A_t^T y_t \qquad (3.28)\\
&= \frac{1}{\delta_t}\begin{bmatrix} \delta_t \tilde{K}_{t-1}^{-1} + a_t a_t^T & -a_t \\ -a_t^T & 1 \end{bmatrix}\begin{bmatrix} \left(A_{t-1}^T A_{t-1}\right)^{-1} A_{t-1}^T y_{t-1} \\ y_t \end{bmatrix}\\
&= \frac{1}{\delta_t}\begin{bmatrix} \delta_t \tilde{K}_{t-1}^{-1}\left(\left(A_{t-1}^T A_{t-1}\right)^{-1} A_{t-1}^T y_{t-1}\right) + a_t a_t^T\left(\left(A_{t-1}^T A_{t-1}\right)^{-1} A_{t-1}^T y_{t-1}\right) - a_t y_t \\ -a_l^T\left(\left(A_{l-1}^T A_{l-1}\right)^{-1} A_{l-1}^T y_{l-1}\right) + y_l \end{bmatrix}\\
&= \frac{1}{\delta_t}\begin{bmatrix} \delta_t \tilde{\alpha}_{t-1} - a_t\left(y_t - a_t^T\left(\tilde{K}_{t-1}\tilde{K}_{t-1}^{-1}\right)\left(A_{t-1}^T A_{t-1}\right)^{-1} A_{t-1}^T y_{t-1}\right) \\ y_t - a_t^T\left(\tilde{K}_{t-1}\tilde{K}_{t-1}^{-1}\right)\left(A_{t-1}^T A_{t-1}\right)^{-1} A_{t-1}^T y_{t-1} \end{bmatrix}\\
&= \frac{1}{\delta_t}\begin{bmatrix} \delta_t \tilde{\alpha}_{t-1} - a_t\left(y_t - \tilde{k}_{t-1}\left(x_t\right)^T \tilde{\alpha}_{t-1}\right) \\ y_t - \tilde{k}_{t-1}\left(x_t\right)^T \tilde{\alpha}_{t-1} \end{bmatrix}
\end{aligned}$$

donde $\tilde{k}_{l-1}\left(x_l\right)^T = a_l^T \tilde{K}_{l-1}$. *Con* $A \in \mathbb{R}^{l\times m}, P \in \mathbb{R}^{m\times m}$, $\tilde{K}_t \in \mathbb{R}^{m\times m}$, $a_l \in \mathbb{R}^{m\times 1}$ *y* $\alpha_l \in \mathbb{R}^{m\times 1}$.

Notese que este algoritmo es una forma del método de mínimos cuadrados SVM sin el termino bias, para el cual no tenemos que definir el termino ε requerido en el método convencional de SVM ε insensitive.

En la Figura 3.1 se presenta el algoritmo SVM Recursivo. El algoritmo comienza calculando la matriz $\tilde{K}_t$, $\tilde{K}_{t-1}$, $\tilde{\alpha}_t$ y P_1 para el tiempo $t = 1$. De manera recursiva se adquiere el nuevo dato de entrenamiento para el tiempo $t = 2, 3, \cdots$, al cual se le realiza la prueba ALD, si $\delta_t > \nu$ entonces el nuevo dato es agregado al diccionario $D_t = D_{t-1} \cup \{x_t\}$, se obtienen 3.24, 3.26 para actualizar el coeficiente $\tilde{\alpha}_t$, en caso contrario el diccionario permanece sin modificaciones $D_t = D_{t-1}$, se obtiene 3.22, y se actualiza el parámetro $\tilde{\alpha}_t$. Por ultimo se obtiene el modelo a partir del diccionario y los coeficientes $\tilde{\alpha}_t$ actualizados para el tiempo t. Obsérvese que la dimensión del diccionario depende del valor seleccionado para ν, este valor determina el numero de datos de entrenamiento de los cuales dependerá el modelo final.

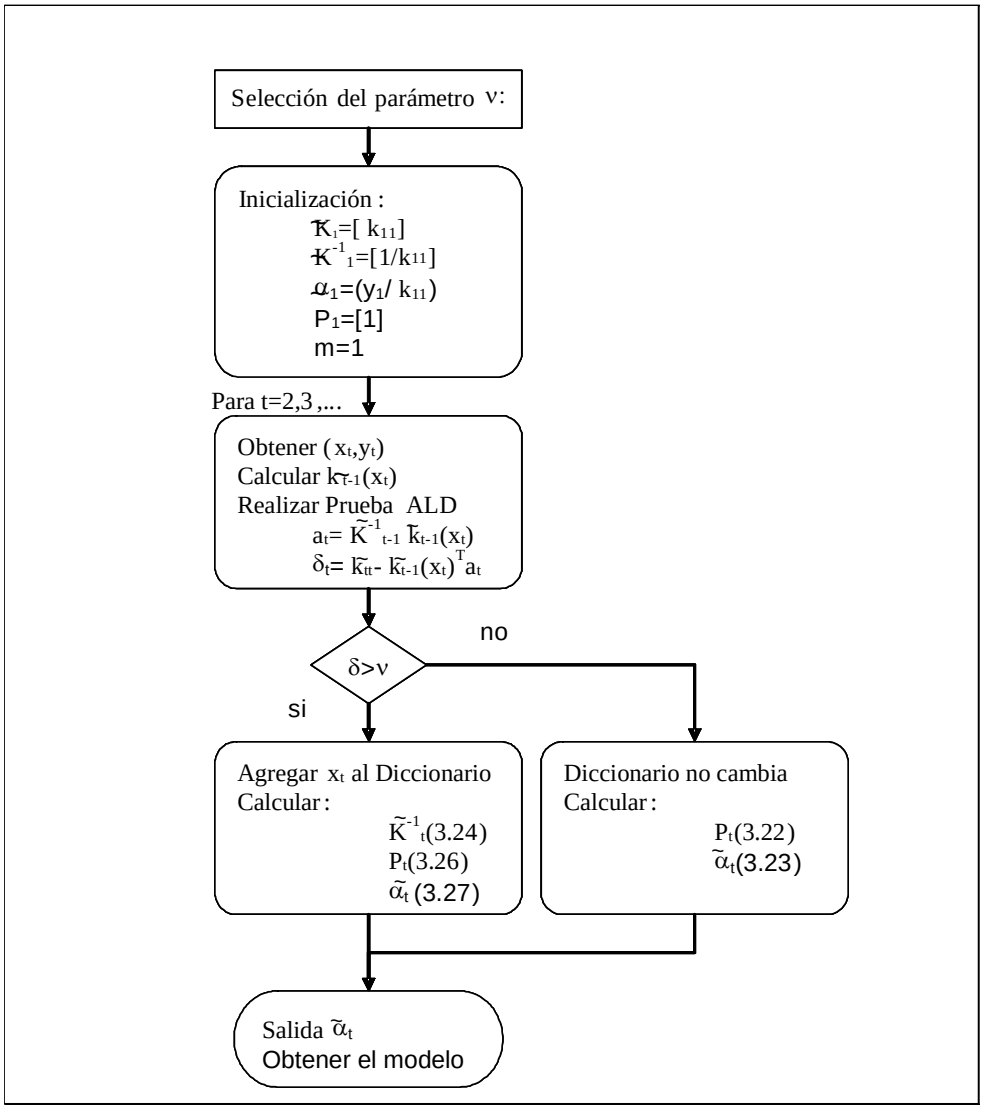

Figura 3.1: Algoritm SVM Recursivo

3.3. Simulaciones

A continuación se realizarán algunas simulaciones con los algoritmos para la identificación de una función no lineal y de un sistema no lineal MISO.

3.3.1. Identificación de la función sinc

Queremos utilizar los algoritmos SVM recursivo y mínimos cuadrados SVM para la identificación de la función

$$y(x) = sin(pi * x)/(pi * x) \tag{3.29}$$

donde $x \in [0, 10]$, con pasos de 0.1. La entrada se define como

$$X(k) = [x(k)]$$

En las Figura 3.2, 3.3 y 3.4 se muestran los resultados obtenidos en la identificación de la función Sinc utilizando el algoritmo SVM Recursivo empleando el kernel RBF, $\sigma = 3$, con ν =0.1, ν =0.2 y ν =0.001 respectivamente. Para el método LS-SVM se utilizo un kernel RBF y se eligieron los valores $C = 20$, σ = ,02, los cuales mostraron tener buenos resultados. Los resultados de la simulación con el algoritmo de mínimos cuadrados SVM se muestran en la Figura 3.5. También se realizo la identificación de la función Sinc utilizando SVM clásico utilizando el kernel RBF $\sigma = 1$ y los siguientes valores $C = 20$ y ε =0.007. Los resultados de la identificación se muestran el la Figura 3.6. Para evaluar el comportamiento de ambos algoritmos utilizamos el error medio cuadrático, $e = \sqrt{\frac{\sum_{k=1}^{N}\left(\hat{f}(i)-y(i)\right)}{k}}$. En la identificación con el algoritmo SVM $e = 5 \times 10^{-4}$, LS SVM e =1.07×10^{-3}, mientras que para el método SVM recursivo los mejores resultados se obtuvieron con ν =0.001, el error medio cuadrático fue e =9.90×10^{-3}, el error medio cuadratico vario dependiendo del valor elegido para ν. A partir de las gráficas se observa que el método SVM Recursivo depende de la selección que se haga del valor ν. En este ejemplo se observo que los métodos SVM clásico y LS SVM obtuvieron un error de identificación mucho más pequeño que el obtenido por el método SVM recursivo esto se debe a que factor de precisión ν, empleado en el método SVM recursivo, produce un error de aproximación de la matriz kernel, el cual se ve reflejado en el modelo final. Mas sin

embargo, aun cuando los métodos SVM y LS SVM obtuvieron un error medio cuadrático menor que el obtenido por el método SVM Recursivo, su costo computacional fue mayor que el del método SVM Recursivo para este ejemplo. En la Tabla 3.1 se puede ver el tiempo de computo requerido por los algoritmos para un numero diferente de datos de entrenamiento esté tiempo también varia de acuerdo a la selección de ν.

# Datos	SVM Clasico	LSSVM	SVMR
100	11.6410	3.9380	0.4840
1000	1 hr(aprox)	45.3600	10.4530

Tabla 3.1: Tiempo de Computo (seg.)

En este ejemplo también se observo que el método SVM recursivo solo depende de 3 vectores de entrenamiento, lo cual demuestra que la solución es sparse, mientras que el método LS SVM utiliza todos los datos disponibles, esto hace que el método SVM recursivo sea muy viable para su uso en línea.

3.3.2. Identificación de un sistema MISO

A continuación utilizaremos los algoritmos para realizar la identificación de un sistema MISO el cual se toma de [21]. El sistema a identificar es

$$y(k) = \frac{y(k-1)y(k-2)y(k-3)u(k-2)\left[y(k-3)-1\right]+u(k-1)}{1+y(k-1)^2+y(k-2)^2} \tag{3.30}$$

donde $u(k)$ es la entrada del sistema y $y(k)$ es la salida del sistema. La entrada del algoritmo se define como

$$X(k) = [y(k-1), y(k-2), y(k-3), u(k-1), u(k-2)]$$

Los estados iniciales son cero y el numero de vectores de entrenamiento es de 50. La señal de entrada se define como

$$u(k) = 0.9\sin(2\pi k/10)$$

Para el método SVM Recursivo se utilizo el kernel RBF para el cual se escogío el valor $\sigma = 3$ y se emplearon los siguientes valores de precisión ν =0.1, ν =0.003 y ν =0.001, este

ultimo mostró ser un buen nivel de aproximación. Los resultados de las simulaciones se muestran en las Figuras 3.7, 3.8 y 3.9, respectivamente. Para el método LS SVM se utilizo el kernel RBF y se eligieron los valores $C = 20$, σ =0.00000002 los cuales durante varias simulaciones mostraron tener los mejores resultados para este método. Los resultados de la simulación se muestra en las Figura 3.10, mientras que los resultados de la identificación con el algoritmo SVM clásico se muestran en la Figura 3.11. En las simulaciones se encontró que el error medio cuadrático para el metodo LS SVM fue e =3.51×10^{-4} mientras que para el algoritmo SVM clásico e =1.37×10^{-4}. El error obtenido por el algoritmo SVM Recursivo fue e =2.84×10^{-4}. En este ejemplo se encontró que el método SVM Recursivo funciona de una manera muy adecuada obteniendo un error de identificación muy bajo y solo depende de un conjunto de 3 vectores de entrenamiento en el diccionario, mientras que los métodos LS SVM y SVM utilizan todos los datos disponibles para generar el modelo. De los resultado también se puede ver que el método SVM Recursivo tiene un desempeño computacional mucho mejor que el método LS SVM, esto debido a que el espacio de memoria utilizado por el método SVM Recursivo se reduce significativamente al utilizar un numero muy reducido de datos de entrenamiento. El método SVM Recursivo redujo significativamente el costo computacional utilizado por los métodos convencionales SVM y LS SVM. En la Tabla 3.2 se muestran los tiempos de computo de los diferentes algoritmos en la identificación del sistema MISO utilizando un numero diferente de datos de entrenamiento. Se puede observar que los algoritmos SVM y LS SVM pueden manejar cantidades pequeñas de datos de entrenamiento pero no así con grandes cantidades. SVM Recursivo mostró reducir significativamente el tiempo de computo esto debido que el modelo no depende de todos los datos de entrenamiento disponibles. Se aprecio que la selección del valor de precisión ν resulta de gran importancia, puesto que este determina tanto el nivel de precisión que se requiere, así como el número de datos de entrenamiento de los cuales dependerá el modelo.

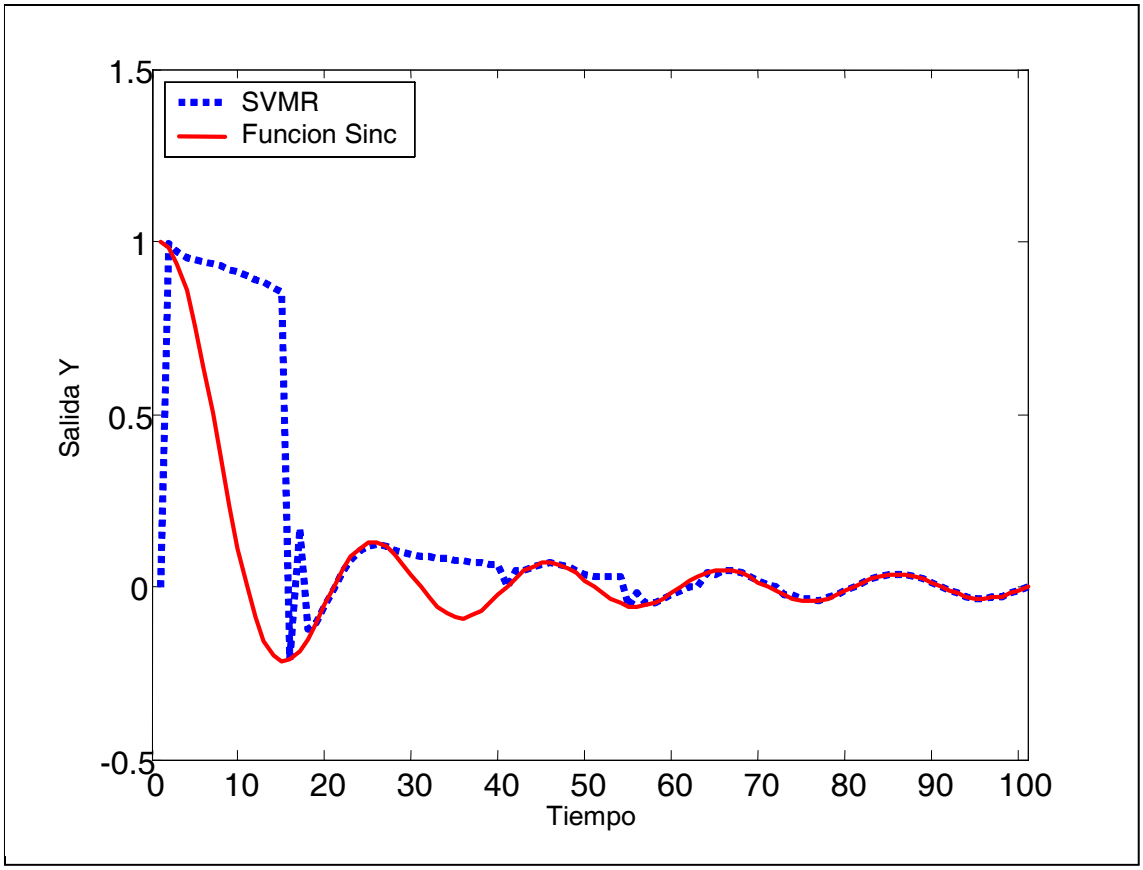

Figura 3.2: Identificación de la función Sinc utilizando el algoritmo SVM Recursivo, con ν =0.2.

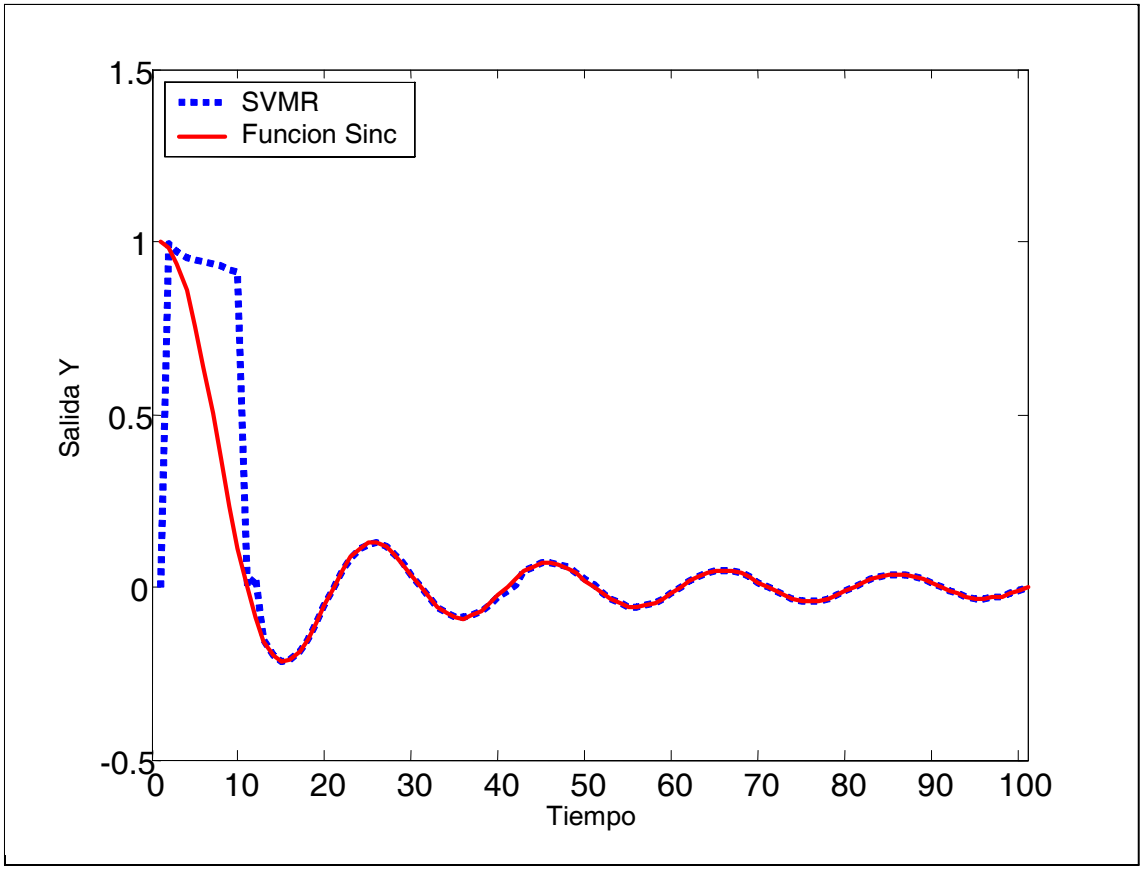

Figura 3.3: Identificación de la función Sinc utilizando el algoritmo SVM Recursivo, con ν =0.1.

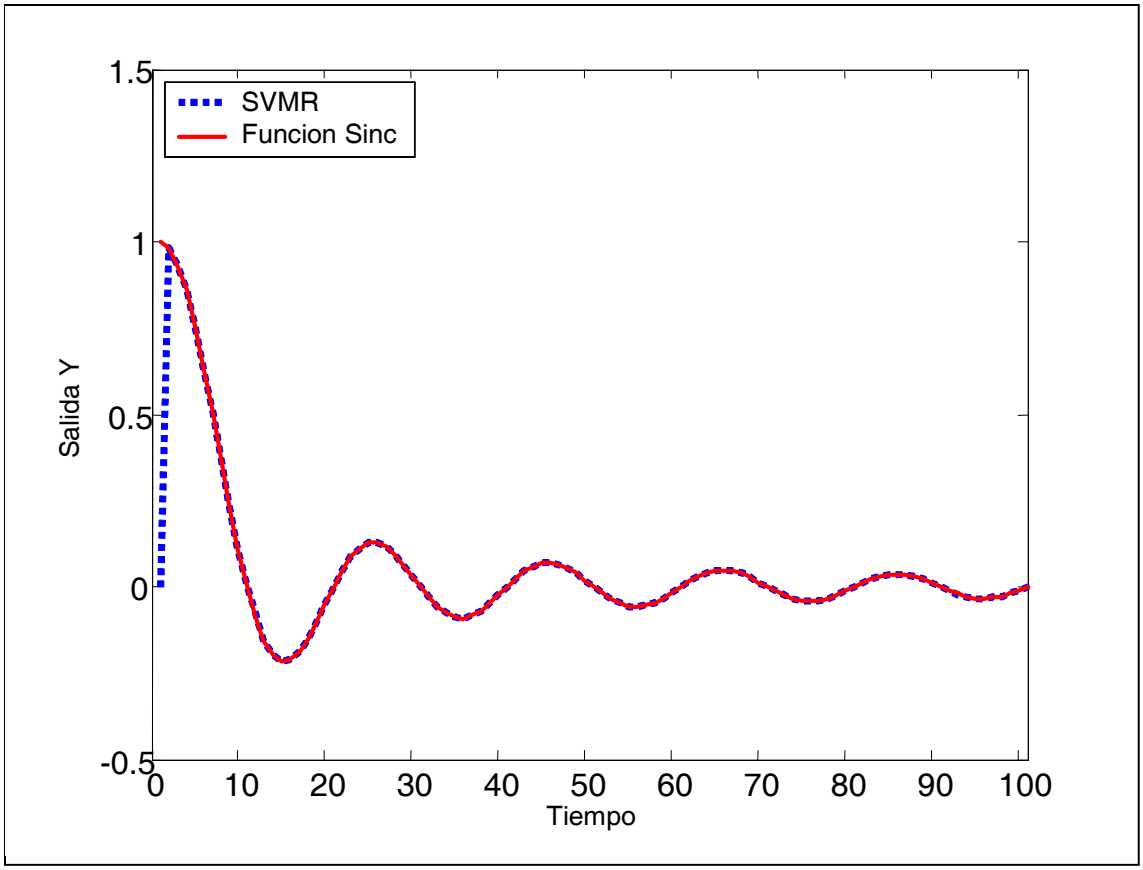

Figura 3.4: Identificación de la función Sinc utilizando el algoritmo SVM Recursivo, con ν =0.001.

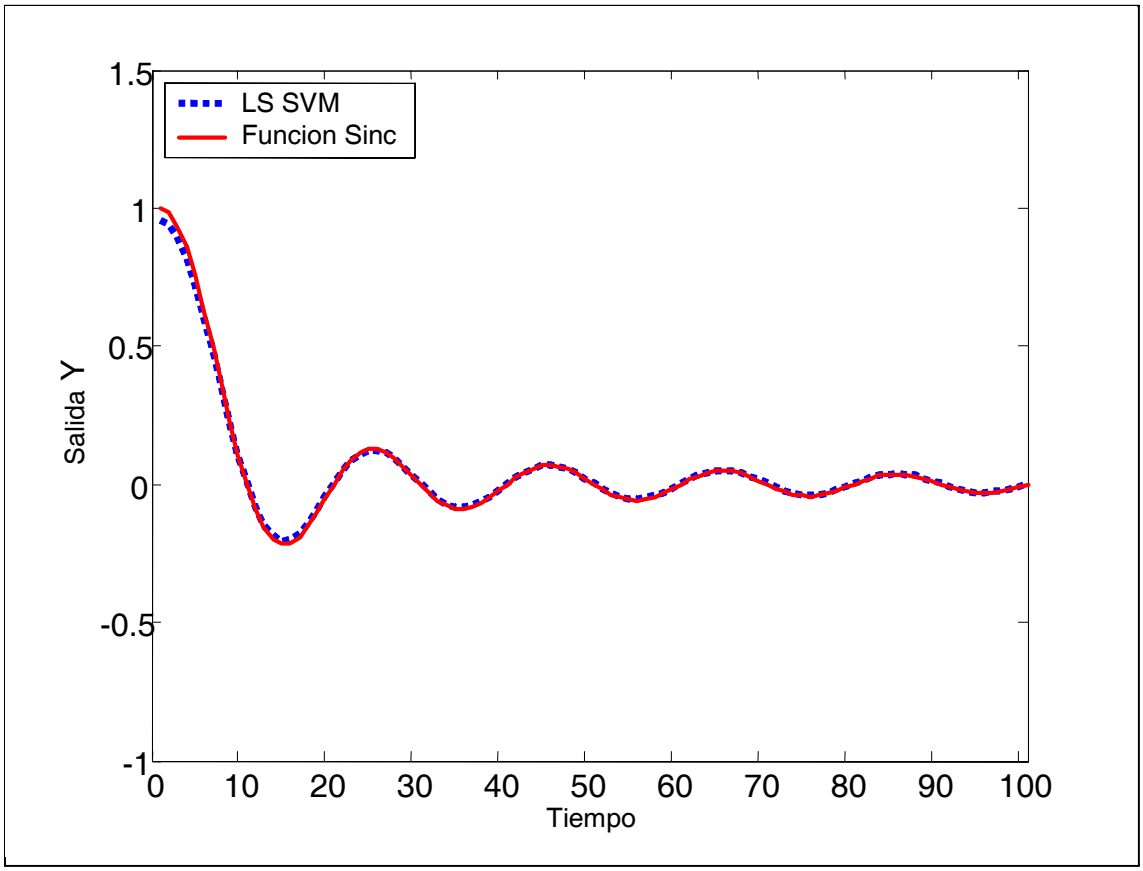

Figura 3.5: Identificación de la función Sinc utilizando el algoritmo LS SVM, con $C = 20$.

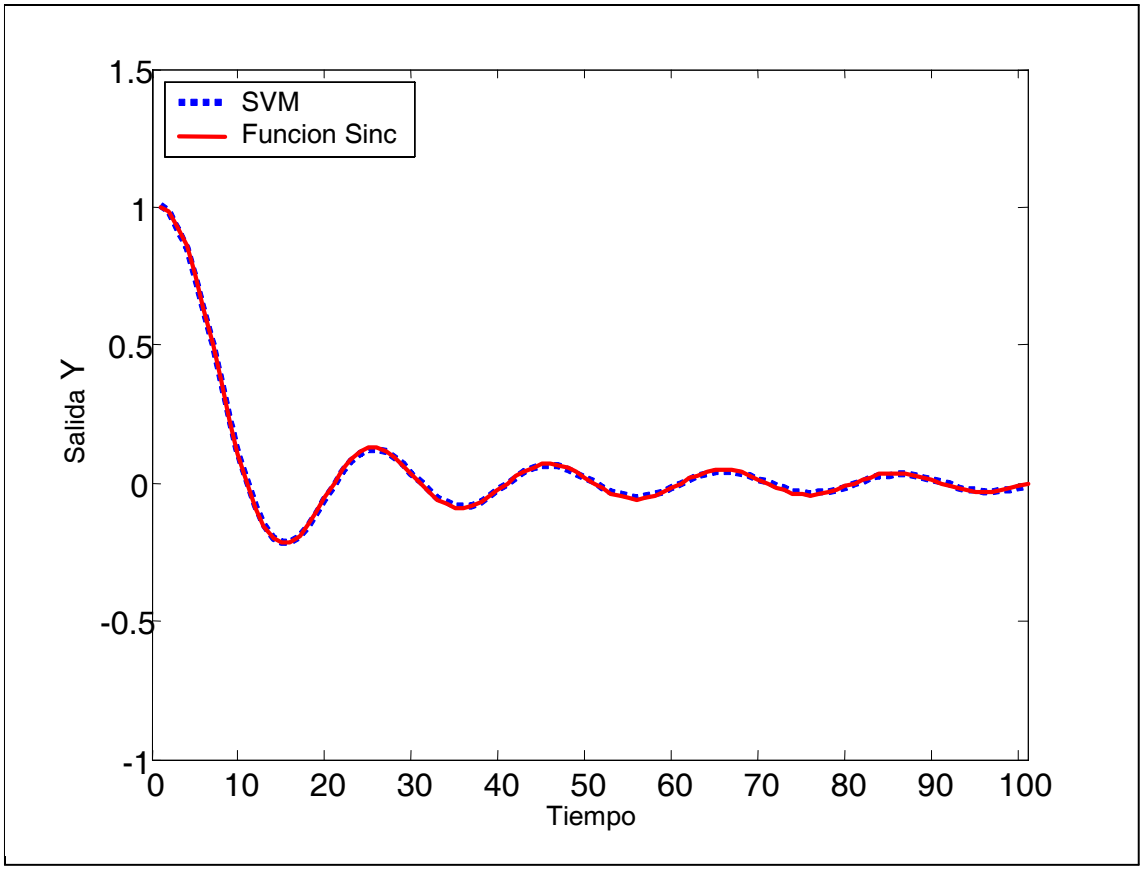

Figura 3.6: Identificación de la función Sinc utilizando el algoritmo SVM, con $C = 20$ y ε =0.007.

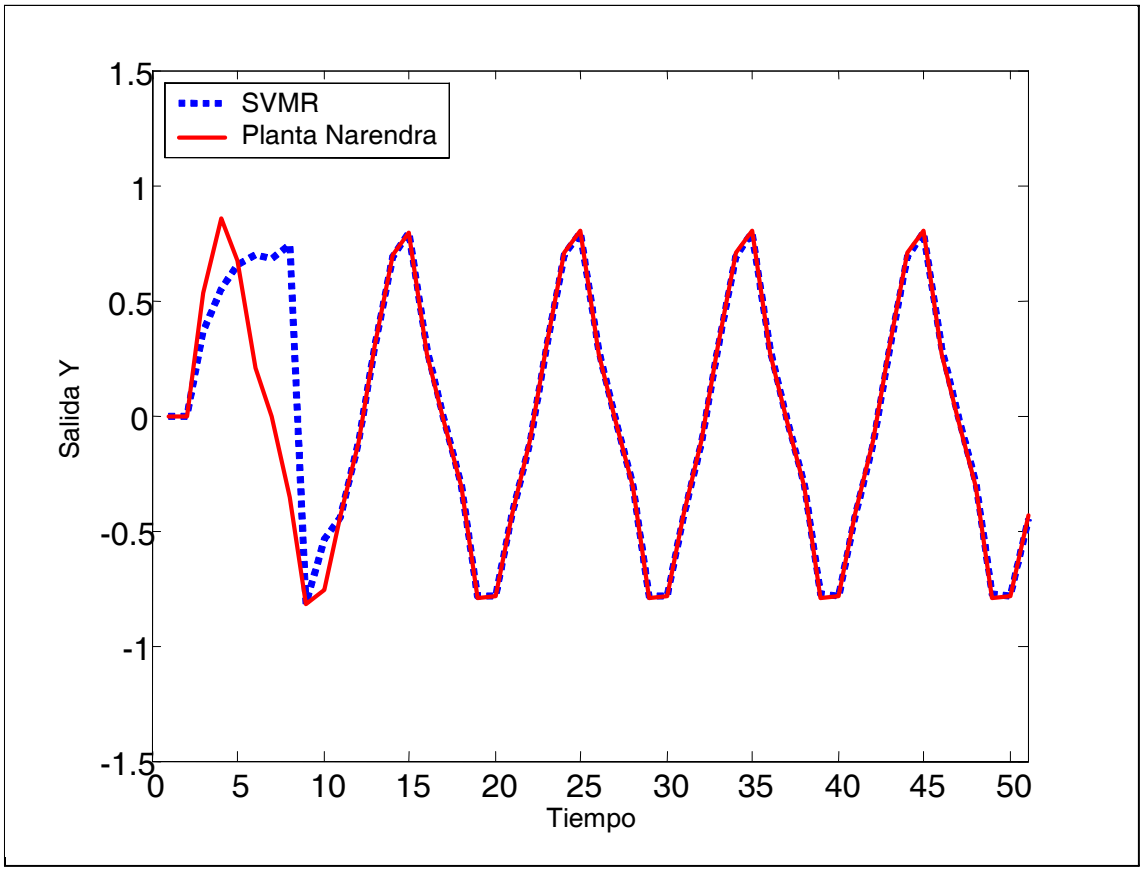

Figura 3.7: Identificación del sistema MISO utilizando el algoritmo SVMR con $\nu = 0{,}1$.

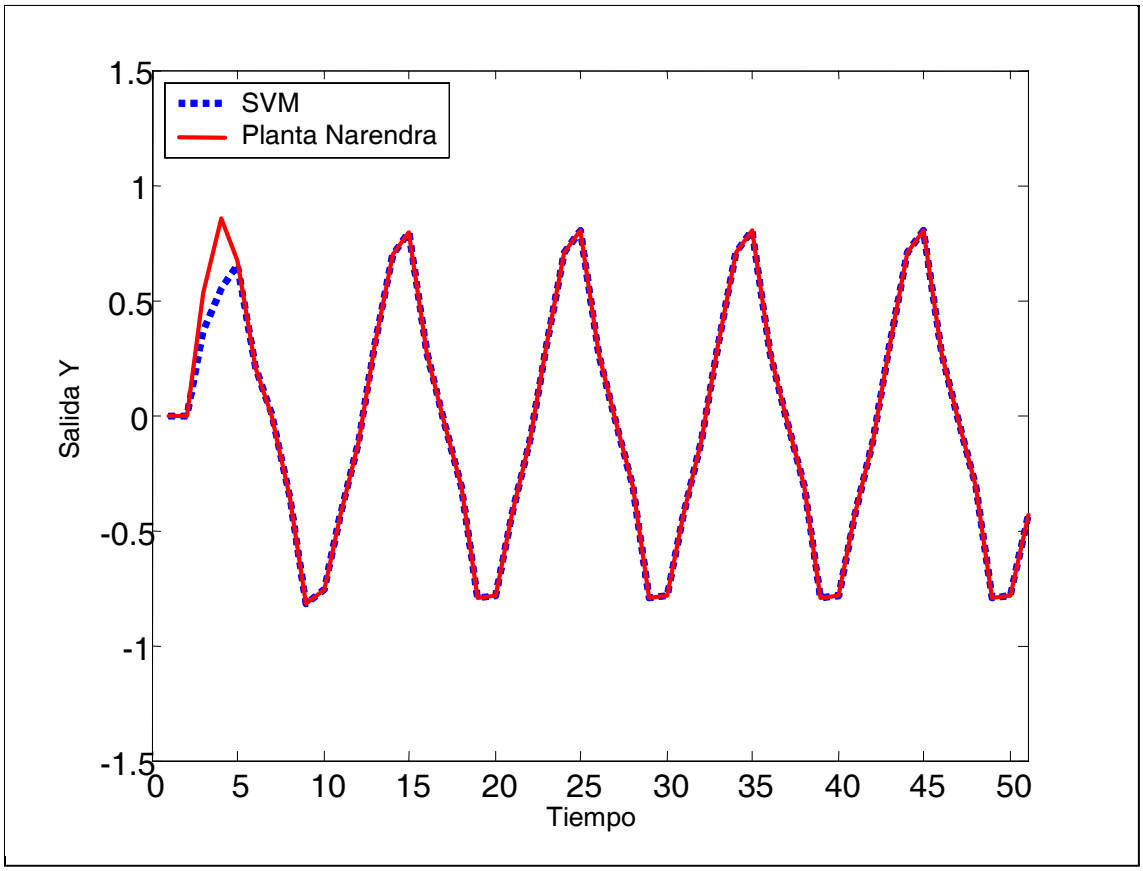

Figura 3.8: Identificación del sistema MISO utilizando el algoritmo SVMR con $\nu = 0{,}003$.

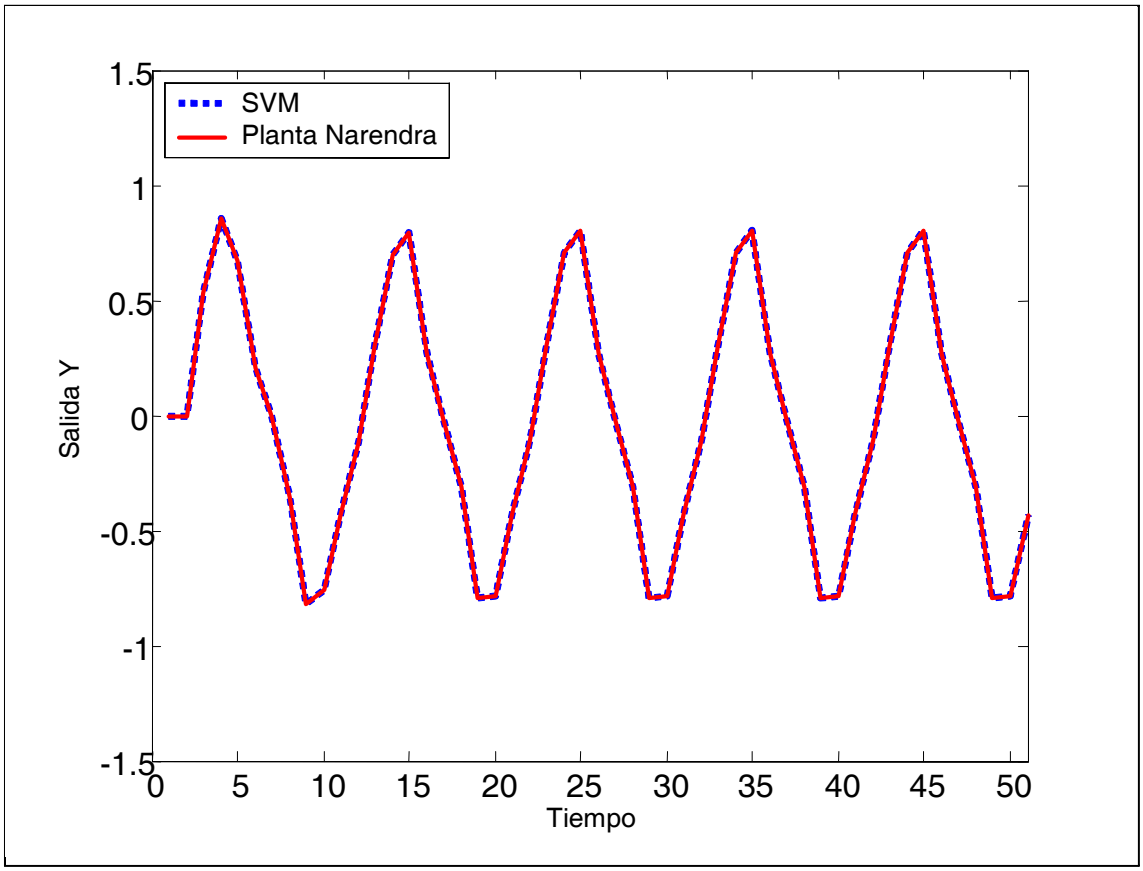

Figura 3.9: Identificación del sistema MISO utilizando el algoritmo SVMR con $\nu = 0{,}001$.

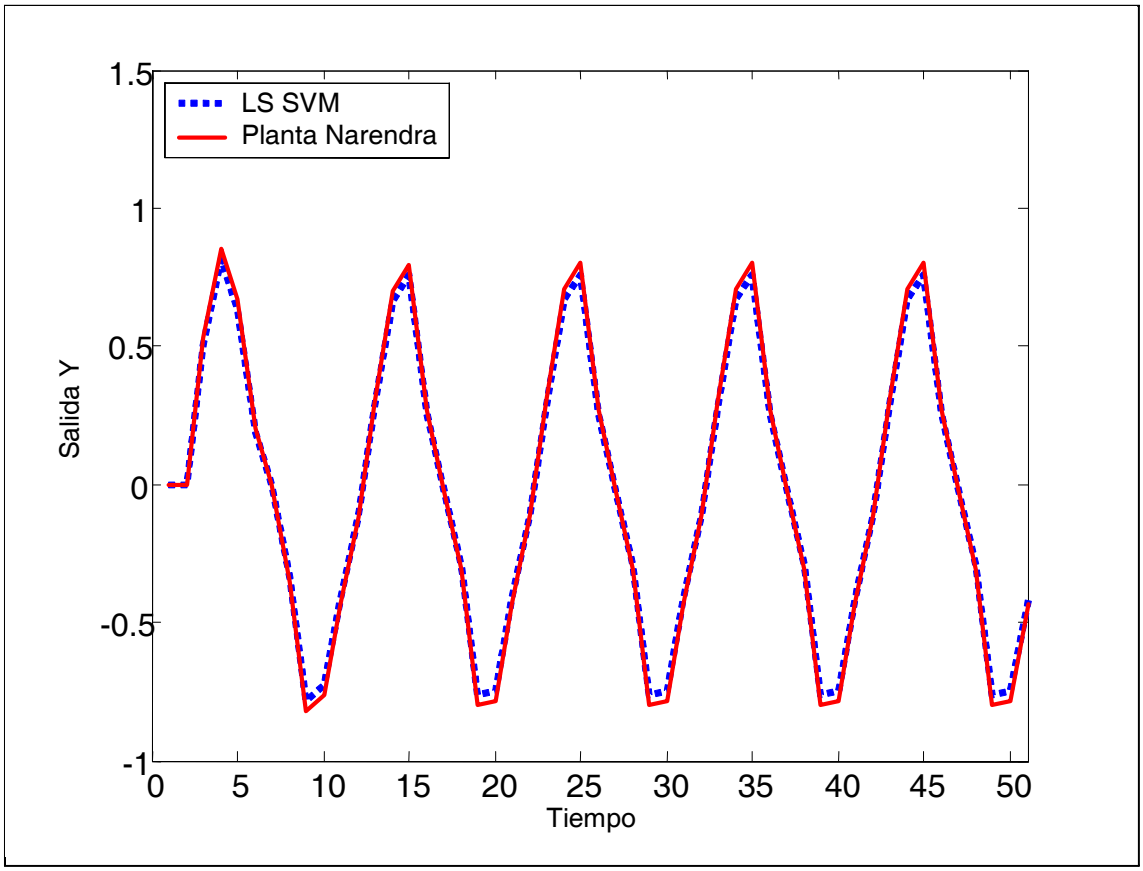

Figura 3.10: Identificación del sistema MISO utilizando el algoritmo LS SVM, con $C = 20$.

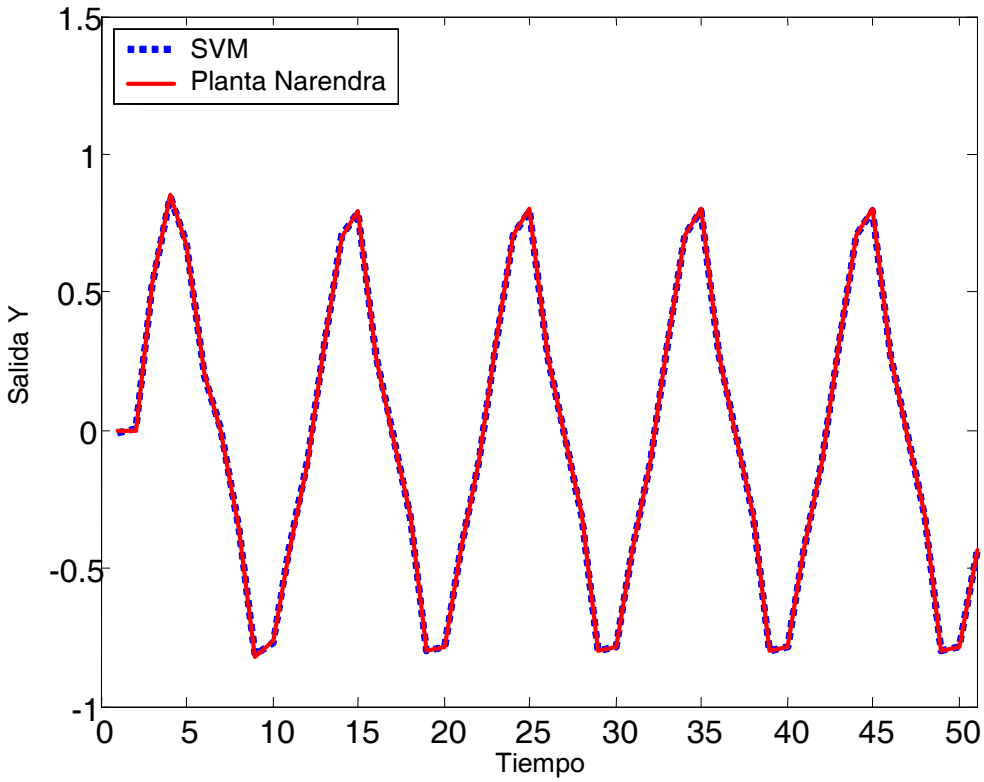

Figura 3.11: Identificación del sistema MISO utilizando el algoritmo SVM, con $C = 20$ y $\varepsilon = 0{,}007$.

# Datos	SVM Clasico	LSSVM	SVMR
50	1.216	0.312	0.156
1000	1 hr(aprox)	48.438	0.453
10000	———-	———-	8.359000

Tabla 3.2: Tiempo de Computo (seg.)

Capítulo 4

Ventanas Deslizantes SVM para identificación en linea

4.1. Introduction

Es sabido que las Máquinas de Vectores de Soporte consumen mucho tiempo en el cálculo de la solución óptima ya que SVM resuelve un problema cuadrático el cual considera a todos los datos de entrenamiento como posibles vectores de soporte. El método, introducido por J. A. K. Suykens, LS- SVM incrementa significativamente la velocidad del calculo de de la solución pero sigue manteniendo el problema del calculo del Kernel para un numero muy grande de datos de entrenamiento. Uno de los problemas más importantes por los cuales es casi imposible utilizar SVM en un ambiente en línea es el procesamiento de los datos de entrenamiento requeridos. Las técnicas actuales para resolver el problema cuadrático no son aceptables para un ambiente en línea ya que su costo computacional es muy elevado. En un ambiente en línea donde los datos de entrenamiento se recopilan de manera secuencial, seria imposible el almacenamiento de la matriz kernel en memoria.

Debido a que en un ambiente en línea la adquisición de datos permanece sin acotar, haciendo indeseable su almacenamiento, para SVM es importante almacenar información para resolver el problema cuadrático. Para resolver este problema usando una cantidad acotada de

datos, es necesario revisar técnicas para almacenamiento de cadenas de información. Estas técnicas se basan en la relación de la dimensión de la cadena o ventana de información y la habilidad de proveer la información necesaria para la tarea a realiza. Este tipo de técnicas se conocen como Ventanas de Tiempo.

En esta sección se propondrá la técnica de Ventanas Deslizantes, (SW, por sus siglas en ingles, "Sling Window"), para aplicar SVM en línea para la identificación de sistemas no lineales. Se utilizaran los dos métodos de SVM para identificación: ε insensitive, $\varepsilon-$ insensitive cuadrático y mínimos cuadrados. Las ventanas deslizantes son ampliamente utilizadas en aplicaciones donde se requiere el procesamiento de grandes cantidades de información. El procesamiento de grandes cantidades de información a través de las ventanas deslizantes ha sido ampliamente estudiado en [18], [19] y [20].

4.2. Ventanas de Tiempo Deslizantes

Muchas aplicaciones de identificación en la vida real toman como entrada una secuencia de datos. El estudio de dichos procesos se pueden resolver utilizando una ventana de tiempo sobre un conjunto de datos de entrada. Técnicamente la ventana de tiempo es una secuencia de información $((x_1, y_1), \cdots, (x_L, y_L))$ que puede ser procesada por un algoritmo, en una secuencia prescrita. Esta técnica es un algoritmo que usa una cantidad de información muy pequeña en comparación con todo el conjunto de información disponible. Estos modelos no requieren cotas sobre el tiempo computacional, pensando que el tiempo requerido por dicho modelo es, típicamente, mas pequeño que el requerido para procesar toda la información disponible.

En la técnica de ventanas de tiempo deslizantes se quiere mantener una cadena de información estática o dinámica, pero por las limitaciones de almacenamiento no es posible mantener en memoria toda la información disponible. En el caso general las ventanas de tiempo deslizantes se busca mantener una cadena o ventana de información estática, conteniendo a L nuevos elementos en la cadena. En algunos casos el valor de L es fijado, pero en algunas aplicaciones este valor puede variar. En nuestro caso el valor de L se mantendrá

estático.

En la Figura 4.1 se ilustra el funcionamiento de la ventana de tiempo. Se asume que los nuevos datos arriban por la parte derecha y los elementos a la izquierda son aquellos datos que ya han sido procesados u observados. Cada elemento tiene un tiempo de arribo, el cual incrementa en uno en cada arribo. El dato activo corresponde a la posición activa de un elemento en la ventana actual, estos pasos se toman de izquierda a derecha, con el elemento primero en la posición 1. Claramente el elemento activo cambia cada ves que un nuevo dato arriba, una ves que se ha almacenado L datos, el algoritmo puede realizar la tarea requerida y después la ventana se mueve para comenzar la recopilación de la información nuevamente. Los datos procesados pueden ser eliminados o mantenidos en la ventana para su procesamiento.

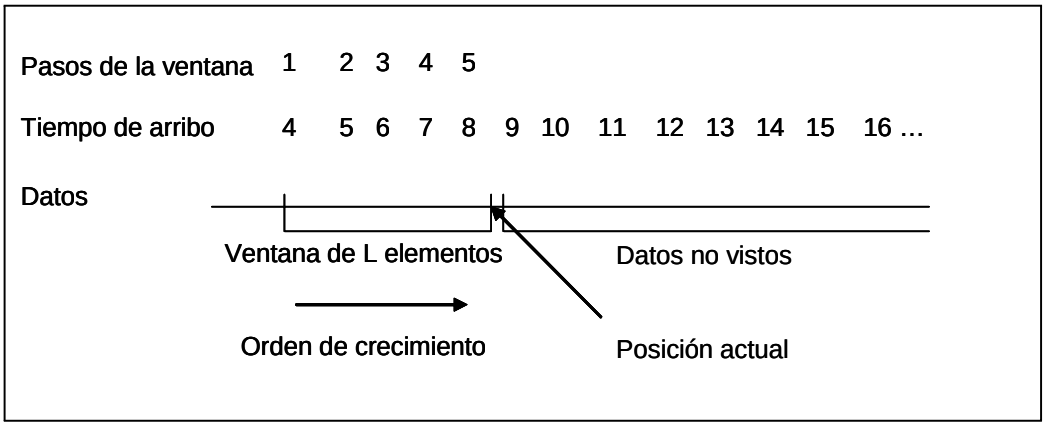

Figura 4.1: Convención de la ventana de tiempo deslizante.

El desempeño del algoritmo depende en mucho del tamaño de la ventana puesto que esta define el número de datos que serán procesados por el algoritmo en cada paso.

4.3. Ventanas deslizantes SVM ε insensitive para identificación en línea

Es esta sección se presenta el algoritmo ventanas deslizantes SVM ε insensitive. La aplicación de técnicas como las ventanas deslizantes son ampliamente utilizadas en problemas en los cuales no es viable el almacenamiento de grandes cantidades de información. El método de SVM ε insensitive ha demostrado ser muy efectivo en la identificación de sistemas no lineales fuera de línea o por lotes, esto se puede ver en el trabajo realizado por Liu y San [21], donde se realiza la identificación de un sistema no lineal utilizando una función de costo lineal ε insensitive y como kernel una función RBF, demostrando, a través de resultados experimentales, que el método es efectivo. En [22] se utiliza SVM para el modelado de sistemas no lineales donde a partir de experimentos numéricos, se muestra que el método puede reducir los efectos del ruido en la identificación, y su desempeño es mejor que el de las redes neuronales. En [23] proponen la selección de la mejor función kernel y de los parámetros a través de un gran numero de experimentos utilizando diferentes funciones kernel para la identificación de un sistema SISO no lineal. Las funciones kernel utilizadas fueron; Polinomial, RBF, Sigmoide, Fourier y por ultimo Spline, formando así diferentes modelos SVM, mostrando que el método con la función RBF como kernel es el mas efectivo para realizar la identificación de sistemas no lineales además de que la elección de los parámetros es pequeña. Más sin embargo la investigación de SVM para identificación en línea es muy escasa.

4.3.1. Método ε insensitive

Acontinuación se presenta una breve introducción a los métodos que se implementarán con las ventanas deslizantes.

La función de costo ε insensitive, definida en 2.9, se puede ver en la Figura 4.2

Esta función de costo no penaliza errores por debajo de un $\varepsilon > 0$. De la teoría de generalización se tiene que para optimizar la generalización del regresor tenemos que minimizar

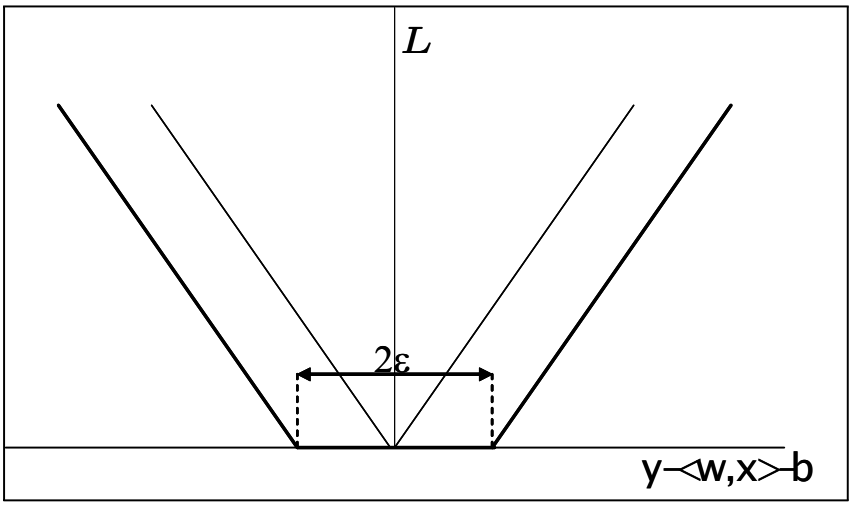

Figura 4.2: Función de costo ε insensitive.

la función de riesgo

$$\|w\|^2 + \sum_{i=1}^{n} L^{\varepsilon}(x_i, y_i, f(x, w)) = \|w\|^2 + \sum_{i=1}^{n} |y - \langle w \cdot x \rangle + b| \tag{4.1}$$

A continuación reformulamos este problema de optimización introduciendo dos variables de perdida no negativas ξ_i y $\hat{\xi}_i$, $i = 1, 2, \cdots, n$, una para aquellos valores que exceden por mas de ε el valor de la función objetivo y la otra para valores que están a una distancia ε por debajo de la función objetivo.

Definición 4.1 *Se definen las variables de perdida de un ejemplo* $(x_i, y_i) \in X \times \mathbb{R}$ *con respecto a una función* $f \in \mathcal{F}$*, como:*

$$\begin{aligned} y_i - (\langle w \cdot x_i \rangle + b) &\leq \varepsilon + \widehat{\xi}_i, \quad i = 1, 2, \ \cdots, n \\ y_i - (\langle w \cdot x_i \rangle + b) &\geq -\varepsilon - \xi_i, \ i = 1, 2, \ \cdots, n \\ \xi_i &\geq 0, \widehat{\xi}_i \geq 0, \ i = 1, 2, \cdots, n \end{aligned} \tag{4.2}$$

Nótese que las variables de perdida $\{\xi_i\}_{i=1}^{n}$ y $\left\{\hat{\xi}_i\right\}_{i=1}^{n}$ describen a la función de costo ε

insensitive de la definición 2.9. Por lo cual podemos definir el problema primario de optimización como:

$$\begin{array}{ll} \text{minimizar} & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\left(\xi_i + \widehat{\xi}_i\right) \\ \text{sujeto a :} & y_i - (\langle w \cdot x_i\rangle + b) \leq \varepsilon + \widehat{\xi}_i, \quad i = 1, 2, \cdots, l \\ & y_i - (\langle w \cdot x_i\rangle + b) \geq -\varepsilon - \xi_i, \ i = 1, 2, \cdots, l \\ & \xi_i \geq 0, \widehat{\xi}_i \geq 0, \ i = 1, 2, \cdots, l \end{array} \tag{4.3}$$

la constante C controla la relación entre minimizar el error de entrenamiento y minimizar el termino de complejidad del modelo $\|w\|^2$. La obtención del problema dual se obtiene igualando a cero la primer derivada parcial con respecto a las variables primarias del Lagrangiano y sustituyendo las relaciones obtenidas dentro del Lagrangiano, tomando en cuenta que $\xi_i\widehat{\xi}_i = 0$ y $\alpha_i\widehat{\alpha}_i = 0$. El Lagrangiano del problema es:

$$\begin{aligned} L_P \ = \ & \frac{1}{2}\sum_{i=1}^{l}\langle w_i \cdot w_i\rangle + C\sum_{i=1}^{l}\left(\xi_i + \widehat{\xi}_i\right) + \sum_{i=1}^{l}\alpha_i\left(y_i - (\langle w \cdot x_i\rangle + b) - \varepsilon - \xi_i\right) \\ & - \sum_{i=1}^{l}\hat{\alpha}\left(y_i - (\langle w \cdot x_i\rangle + b) + \varepsilon + \widehat{\xi}_i\right) \end{aligned} \tag{4.4}$$

donde $L_P = L(w, x, \alpha, \hat{\alpha}, \xi, \widehat{\xi})$. Ahora obtenemos las derivadas parciales del Lagrangiano 4.4

e imponemos condiciones estacionarias.

$$\begin{aligned}
\frac{\partial L}{\partial w} &= w - \sum_{i=1}^{l} \alpha_i x_i + \sum_{i=1}^{l} \hat{\alpha}_i x_i = 0 \\
&\Longrightarrow \quad w = \sum_{i=1}^{l} (\alpha_i - \hat{\alpha}_i) x_i \\
\frac{\partial L}{\partial b} &= -\sum_{i=1}^{l} \alpha_i + \sum_{i=1}^{l} \hat{\alpha} = 0 \\
&\Longrightarrow \quad \sum_{i=1}^{l} (\alpha_i - \hat{\alpha}_i) = 0 \\
\frac{\partial L}{\partial \xi} &= C - \alpha = 0 \\
\frac{\partial L}{\partial \hat{\xi}} &= C - \hat{\alpha} = 0
\end{aligned} \tag{4.5}$$

a continuación sustituimos las relaciones obtenidas 4.5 en el Lagrangiano 4.4.

$$\begin{aligned}
L_D &= \frac{1}{2}\left(\sum_{i=1}^{l} (\alpha_i - \hat{\alpha}_i) x_i \sum_{j=1}^{l} (\alpha_j - \hat{\alpha}_j) x_j\right) + C\sum_{i=1}^{l}\left(\xi_i + \widehat{\xi}_i\right) \\
&\quad + \sum_{i=1}^{l} \alpha_i y_i - \sum_{i,j=1}^{l} \alpha_i (\alpha_j - \hat{\alpha}_j) \langle x_i \cdot x_j \rangle - \varepsilon \sum_{i=1}^{l} \alpha_i - \sum_{i=1}^{l} \alpha_i \xi_i \\
&\quad - \sum_{i=1}^{l} \hat{\alpha}_i y_i + \sum_{i,j=1}^{l} \alpha_j (\alpha_i - \hat{\alpha}_i) \langle x_i \cdot x_j \rangle - \varepsilon \sum_{i=1}^{l} \hat{\alpha}_i - \sum_{i=1}^{l} \hat{\alpha}_i \hat{\xi}_i \\
L_D &= \frac{1}{2} \sum_{i,j=1}^{l} (\alpha_i - \hat{\alpha}_i)(\alpha_j - \hat{\alpha}_j) \langle x_i \cdot x_j \rangle + \sum_{i=1}^{l} (\alpha_i - \hat{\alpha}_i) y_i \\
&\quad - \varepsilon \sum_{i=1}^{l} (\alpha_i + \hat{\alpha}_i) - \sum_{i,j=1}^{l} (\alpha_i - \hat{\alpha}_i)(\alpha_j - \hat{\alpha}_j) \langle x_i \cdot x_j \rangle \\
L_D &= -\frac{1}{2} \sum_{i,j=1}^{l} (\alpha_i - \hat{\alpha}_i)(\alpha_j - \hat{\alpha}_j) \langle x_i \cdot x_j \rangle + \sum_{i=1}^{l} (\alpha_i - \hat{\alpha}_i) y_i - \varepsilon \sum_{i=1}^{l} (\alpha_i + \hat{\alpha}_i)
\end{aligned} \tag{4.6}$$

donde $L_D = L(\alpha, \hat{\alpha})$ es el Lagrangiano Dual. Obsérvese que el Lagrangiano dual esta expresa-

do solamente en función de variables duales y es una función convexa. Entonces el problema dual correspondiente se define como

$$\begin{aligned} \text{maximizar} \quad & -\tfrac{1}{2}\sum_{i,j=1}^{l}(\alpha_i-\hat{\alpha}_i)(\alpha_j-\hat{\alpha}_j)\langle x_i\cdot x_j\rangle+\sum_{i=1}^{l}(\alpha_i-\hat{\alpha}_i)y_i-\varepsilon\sum_{i=1}^{l}(\alpha_i+\hat{\alpha}_i) \\ \text{sujeto a} \ : \quad & 0\le\hat{\alpha}_i,\alpha_i\le C,\ i=1,2,\cdots,l \\ & \sum_{i=1}^{l}(\alpha_i-\hat{\alpha}_i)=0,\ i=1,2,\cdots,l \end{aligned} \tag{4.7}$$

cuyas condiciones complementarias de Karush- Kuhn- Tuker son

$$\begin{aligned} &\alpha_i\left(y_i-(\langle w\cdot x_i\rangle+b)-\varepsilon-\xi_i,\right)=0, && i=1,2,\cdots,l \\ &-\hat{\alpha}\left(y_i-(\langle w\cdot x_i\rangle+b)+\varepsilon+\widehat{\xi}_i,\right)=0, && i=1,2,\cdots,l \\ &\xi_i\widehat{\xi}_i=0\ \alpha_i\widehat{\alpha}_i=0, && i=1,2,\cdots,l \\ &(\alpha_i-C)\xi_i=0,\ (\hat{\alpha}_i-C)\hat{\xi}_i=0 && i=1,2,\cdots,l \end{aligned} \tag{4.8}$$

Comentario 4.1 *Los puntos que se encuentren fuera del tubo $\pm\varepsilon$ alrededor de la función de salida se llaman vectores de soporte, sv, esto es los puntos x_i para los cuales α_i, $\widehat{\alpha}_i\neq 0$ seran los vectores de soporte, notese que si $\alpha_i\neq 0$ entonces $\widehat{\alpha}_i=0$.*

Obsérvese que si tomamos a $\alpha=(\alpha_i-\hat{\alpha}_i)$ y tomando en cuenta que $\alpha_i\widehat{\alpha}_i=0$, podemos rescribir el problema dual 4.7 de la siguiente forma

$$\begin{aligned} \text{maximizar} \quad & L_D=-\tfrac{1}{2}\sum_{i,j=1}^{l}\alpha_i\alpha_j(\langle x_i\cdot x_j\rangle)+\sum_{i=1}^{l}\alpha_i y_i-\varepsilon\sum_{i=1}^{l}|\alpha_i| \\ \text{sujeto a} \ : \quad & \sum_{i=1}^{l}\beta=0,\ i=1,2,\cdots,l \end{aligned} \tag{4.9}$$

Proposición 4.1 *Suponga que se quiere realizar la identificación sobre un conjunto de entrenamiento*

$$S=((x_1,y_1)\cdots(x_l,y_l))$$

utilizando el kernel $K(x_i, x_j)$ *para definir un espacio de características implícito, y suponiendo que* α^* *es solución del siguiente programa cuadratico*

$$\begin{aligned} &\textit{maximizar} \quad L_D = -\tfrac{1}{2}\sum_{i,j=1}^{l}\alpha_i\alpha_j K(x_i, x_j) + \sum_{i=1}^{l}\alpha_i y_i - \varepsilon\sum_{i=1}^{l}|\alpha_i| \\ &\textit{sujeto a} \quad : \quad \sum_{i=1}^{l}\alpha_i = 0, \;\; -C \leq \alpha_i \leq C, \; i = 1, 2, \cdots, l \end{aligned} \tag{4.10}$$

sea $f(x) = \sum_{i=1}^{l}\alpha_i^* K(x_i, x) + b^*$, *donde* b^* *se escoge tal que* $f(x_i) - y_i = -\varepsilon$, $\forall i$ *con* $0 \leq \alpha_i^* < C$. *Entonces la función* $f(x)$ *es equivalente al hiperplano en el espacio de características definido por el kernel* $K(x_i, x_j)$ *que resuelve el problema de optimización 2.7.*

Mientras que en [13] se utilizo SVM ε insensitive para la identificación de un sistema no lineal. A partir de los resultados obtenidos por el método SVM ε insensitive se demostro que tiene las siguientes ventajas:

- Se necesita poco conocimiento del Sistema No lineal.
- No es necesario especificar la estructura del sistema. Comparado con las Redes Neuronales, la estructura se determina automáticamente. No tiene problemas de mínimos locales.
- Tiene una buena habilidad de generalización.

Pero todos estos trabajos no contemplan un ambiente en línea. Se puede ver, a partir de estos trabajos, que en general SVM ε insensitive resulta ser una herramienta muy útil cuando se quiere realizar la identificación de sistemas no lineales en un ambiente fuera de linea pero no son viables para trabajar en linea. Acontinuación se presenta la versión de SVM ε insensitive Sliding Windows para identificación en línea.

Para tratar con este problema se propone un tipo de método en línea basado en las ventanas de tiempo, el cual utiliza sucesivamente L datos en el cálculo de la solución óptima. Usando $((x_1, y_1), \cdots, (x_L, y_L))$ datos de entrenamiento para desarrollar el modelo de

identificación en cada ventana de tiempo, podemos minimizar el costo computacional que representa el calculo del Kernel para la solución del método SVM ε insensitive.

El conjunto de entradas- salida se representaran como (X, y) donde X es el vector de datos de entrenamiento de dimensión t. Los datos de entrenamiento en alguna región pueden ser expresados como $(x_1, y_1), \cdots, (x_k, y_k), (x_L, y_L) \in \mathbb{R}^n \times \mathbb{R}$. Suponga que el problema cuadrático se resuelve con L datos de entrenamiento. Esto es que se puede identificar el modelo a partir de L datos de entrenamiento del pasado al presente. Con el proceso en marcha, se reciben nuevos datos de entrenamiento de manera recursiva y constante. Para poder resolver el problema de SVM de manera adecuada es necesario agregar el nuevo conjunto de datos para actualizar los parámetros del modelo.

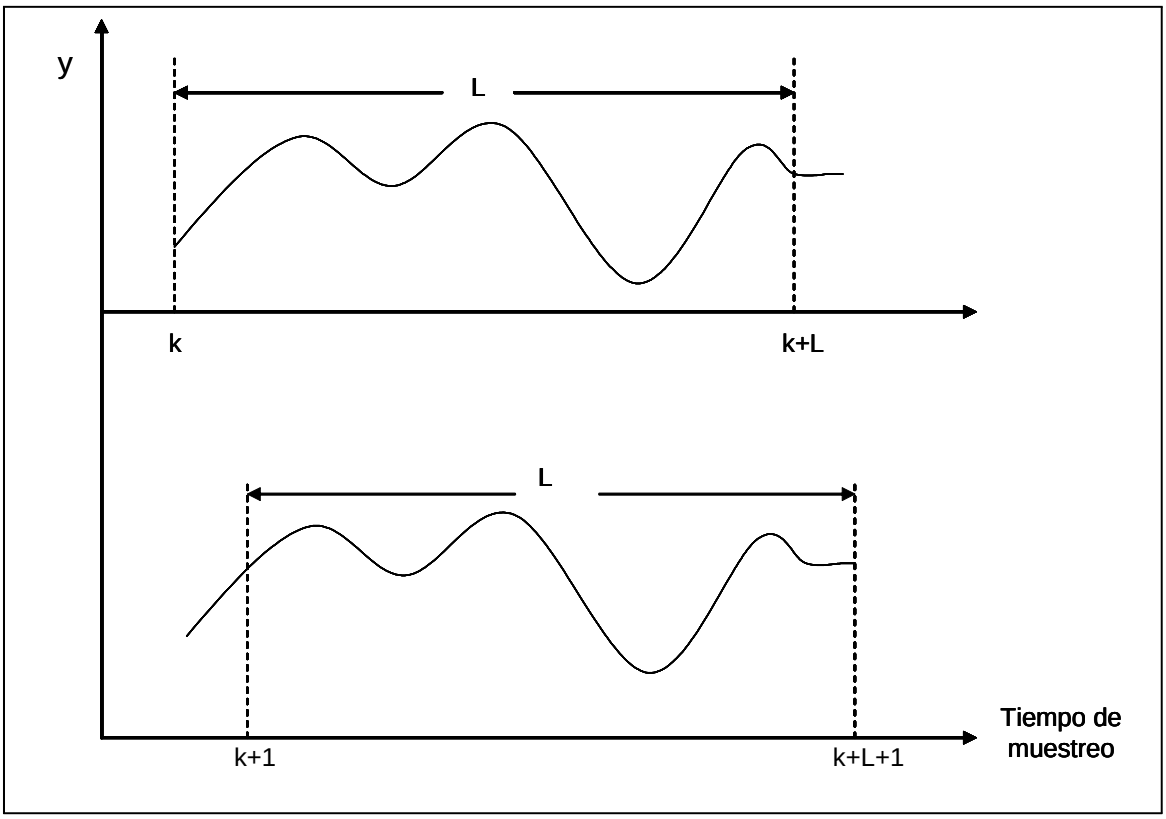

Figura 4.3: Funcionamiento de laVentana de Tiempo Deslizante para el metodo SVM.

Sin perdida de la generalidad se considerara una ventana deslizante con incrementos unitarios aplicada al método SVM. En cada paso se resuelve el problema cuadrático. La Figura 4.3 muestra como trabaja la ventana de tiempo propuesta para el método de SVM ε insensitive Sliding Windows para identificación en línea. Después de transcurrido un tiempo $k = L$ y haber recopilado un conjunto de L datos de entrenamiento se realiza la identificación del modelo. En el tiempo $k + L$, a saber el conjunto de datos de entrenamiento recopilados son L desde el tiempo k hasta el tiempo actual $k + L$. Primero se utilizan los L datos de entrenamiento recopilados para identificar el sistema. Cuando el nuevo dato de entrenamiento $k+L+1$ arriba, se agrega a la ventana y se elimina el primer dato recopilado por el algoritmo en el tiempo k. Los datos de la ventana deslizante cambian del tiempo $k+1$ al tiempo $k+L+1$. El numero de elementos de la ventana no cambia solo se modifican los elementos. Entonces la ventana se mueve conforme el tiempo avanza pero su dimensión no se modifica. El conjunto de entrenamiento se define por la ventana de tiempo deslizante, y el modelo se obtiene en cada paso k a parir del paso $k + L + 1$.

A continuación se presenta el algoritmo de ventanas deslizantes SVM ε insensitive para identificación en línea con paso unitario.

Algoritmo 4.1 *1. Recopilación del conjunto de datos de entrenamiento*

$x_L = \{(x_1, y_1), \cdots, (x_{k+L}, y_{k+L})\}$ *con* L *fijo.*

2. *Obtener el modelo a través del método SVM* ε *insensitive a partir del conjunto de datos de entrenamiento* x_L.

3. *Actualizar la ventana de tiempo. Se recopila el dato nuevo* (x_{k+L+1}, y_{k+L+1}) *y se elimina el dato de entrenamiento* (x_1, y_1) *para obtener el nuevo conjunto de entrenamiento* $x_{L+1} = \{(x_2, y_2), \cdots, (x_{k+L+1}, y_{k+L+1})\}$, *y colocarlo en la posición activa.*

4. *Se obtiene el modelo para el tiempo* $k + L + 1$ *a través del método SVM* ε *insensitive utilizando los datos de la ventana deslizante* x_{L+1}.

5. *Se realiza la actualización del paso 3 y se continúa con el algoritmo de manera recursiva.*

4.4. Ventanas deslizantes SVM ε insensitive cuadratico con factor de olvido λ

En esta sección se presenta el algoritmo de ventanas veslizantes SVM insensitive cuadrático con factor de olvido. Sin perdida de la generalidad utilizaremos las ventanas deslizantes, con incrementos unitarios y aplicaremos un factor de olvido a los datos de entrenamiento anteriores al nuevo dato.

Las funciones lineales han sido ampliamente utilizadas en el área de SVM para penalizar el error. Recientemente la función de costo ε insensitive cuadrática para la penalización del error de identificación ha ido captado más y más la atención de los investigadores. El método SVM ε insensitive cuadrático es una variación del método SVM ε insensitive la cual utiliza una función de costo distinta, en este caso la función de costo cuadrática. Este método también se resuelve como un programa cuadrático para el cual las técnicas de optimización no permiten su uso en un ambiente en línea. El método puede ser utilizado ampliamente, esto debido a que los problemas en los cuales se utiliza SVM ε insensitive pueden ser reformulados para la utilización de SVM ε insensitive cuadrático, utilizando una función de costo cuadrática.

4.4.1. Método ε insensitive cuadratico

En este método se utiliza la función de perdida ε insensitive cuadratica, la cual se define de la siguiente manera.

Definición 4.2 *La función de perdida ε insensitive cuadratica $L_2^{\varepsilon}(x, y, f)$ esta definida como*

$$L_2^{\varepsilon}(x_i, y_i, f) = |y - f(x)|_{\varepsilon}^2 \tag{4.11}$$

La función de costo ε insensitive cuadratica se puede ver en la Figura 4.4.

Tenemos que minimizar la función de perdida ε insensitive cuadrática

$$\|w\|^2 + L_2^{\varepsilon}(x_i, y_i, f) = \|w\|^2 + \sum_{i=1}^{n} |y - \langle w \cdot x \rangle|^2 \tag{4.12}$$

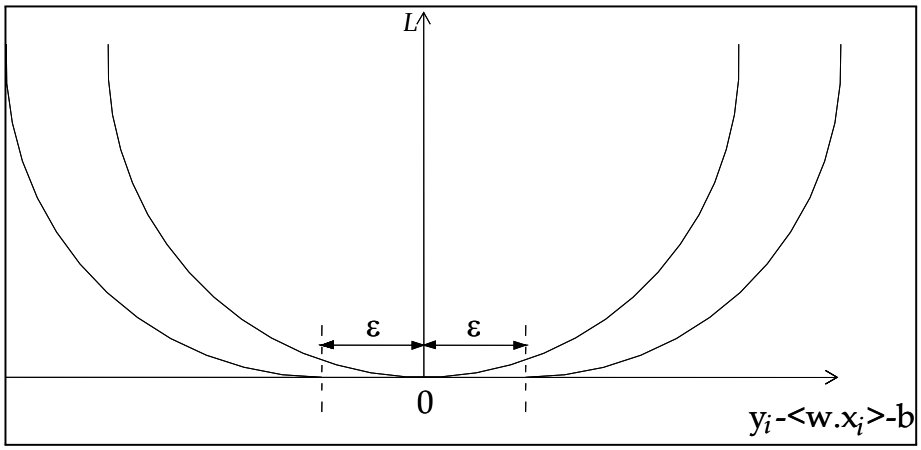

Figura 4.4: Función de costo ε insensitive cuadratica.

Al igual que en el caso de ε insensitive se introducen las variables de perdida ξ_i y $\hat{\xi}_i$, $i = 1, 2, \cdots, n$. Así definimos el problema primario optimización de la siguiente forma:

$$\begin{aligned} \text{minimizar} \quad & \frac{1}{2}\left(\|w\|^2 + C\sum_{i=1}^{l}\left(\xi_i^2 + \widehat{\xi}_i^2\right)\right) \\ \text{sujeto a} \ : \quad & y_i - (\langle w \cdot x_i\rangle + b) \leq \varepsilon + \widehat{\xi}_i, \quad i = 1, 2, \cdots, l \\ & y_i - (\langle w \cdot x_i\rangle + b) \geq -\varepsilon - \xi_i, \ i = 1, 2, \cdots, l \\ & \xi_i \geq 0, \widehat{\xi}_i \geq 0, \ i = 1, 2, \cdots, l \end{aligned} \tag{4.13}$$

El problema Dual puede ser derivado a través del método común y tomando en cuenta que $\xi_i\widehat{\xi}_i = 0$ por lo tanto $\alpha_i\widehat{\alpha}_i = 0$. El lagrangiano de 4.13 esta dado por

$$\begin{aligned} L_P \ = \ & \frac{1}{2}\sum_{i=1}^{l}\langle w_i \cdot w_i\rangle + C\frac{1}{2}\sum_{i=1}^{l}\left(\xi_i^2 + \widehat{\xi}_i^2\right) + \sum_{i=1}^{l}\alpha_i\left(y_i - (\langle w \cdot x_i\rangle + b) - \varepsilon - \xi_i,\right) \\ & - \sum_{i=1}^{l}\hat{\alpha}\left(y_i - (\langle w \cdot x_i\rangle + b) + \varepsilon + \widehat{\xi}_i,\right) \end{aligned} \tag{4.14}$$

donde $L_P = L(w, x, \alpha, \hat{\alpha}, \xi, \widehat{\xi})$ es el Lagrangiano del problema primario. A continuación obteniendo las derivadas parciales de 4.14 con respecto a las variables primarias e imponiendo

condiciones estacionarias.

$$\begin{aligned}
\frac{\partial L}{\partial w} &= w - \sum_{i=1}^{l} \alpha_i x_i + \sum_{i=1}^{l} \hat{\alpha}_i x_i = 0 \\
&\Longrightarrow \quad w = \sum_{i=1}^{l} (\alpha_i - \hat{\alpha}_i) x_i \\
\frac{\partial L}{\partial b} &= -\sum_{i=1}^{l} \alpha_i + \sum_{i=1}^{l} \hat{\alpha} = 0 \\
&\Longrightarrow \quad \sum_{i=1}^{l} (\alpha_i - \hat{\alpha}_i) = 0 \\
\frac{\partial L}{\partial \xi} &= C\xi - \hat{\alpha} = 0 \Longrightarrow \xi = \frac{\hat{\alpha}}{C} \\
\frac{\partial L}{\partial \hat{\xi}} &= C\hat{\xi} - \hat{\alpha} = 0 \Longrightarrow \hat{\xi} = \frac{\hat{\alpha}}{C}
\end{aligned} \tag{4.15}$$

sustituyendo 4.15 en el Lagrangiano 4.14 se tiene

$$\begin{aligned}
L_D &= \frac{1}{2} \sum_{i,j=1}^{l} (\alpha_i - \hat{\alpha}_i)(\alpha_i - \hat{\alpha}_j) \langle x_i \cdot x_j \rangle - \frac{1}{2C} \sum_{i,j=1}^{l} (\alpha_i - \hat{\alpha}_i)(\alpha_i - \hat{\alpha}_j) \\
&\quad + \sum_{i=1}^{l} (\alpha_i - \hat{\alpha}_i) y_i - \sum_{i,j=1}^{l} (\alpha_i - \hat{\alpha}_i)(\alpha_i - \hat{\alpha}_j) \langle x_i \cdot x_j \rangle - \varepsilon \sum_{i=1}^{l} (\alpha_i + \hat{\alpha}_i) \\
L_D &= -\frac{1}{2} \sum_{i,j=1}^{l} (\alpha_i - \hat{\alpha}_i)(\alpha_j - \hat{\alpha}_j) \left(\langle x_i \cdot x_j \rangle + \frac{1}{C} \delta_{ij} \right) + \sum_{i=1}^{l} (\alpha_i - \hat{\alpha}_i) y_i - \varepsilon \sum_{i=1}^{l} (\alpha_i + \hat{\alpha}_i)
\end{aligned} \tag{4.16}$$

donde $L_D = L(\alpha, \hat{\alpha})$ es el Lagrangiano Dual mientras que δ_{ij} es la matriz cuya diagonal su valor es $1/C$ en otro lado vale cero. Obsérvese que al igual que en el método de ε insensitive, el Lagrangiano Dual esta expresado solamente en función de variables duales y es una función

convexa. Entonces el problema dual correspondiente se define como

$$\begin{array}{ll} \text{maximizar} & -\frac{1}{2}\sum_{i,j=1}^{l}\left(\alpha_i-\hat{\alpha}_i\right)\left(\alpha_i-\hat{\alpha}_j\right)\left(\langle x_i\cdot x_j\rangle+\frac{1}{C}\delta_{ij}\right)+\sum_{i=1}^{l}\left(\alpha_i-\hat{\alpha}_i\right)y_i-\varepsilon\sum_{i=1}^{l}\left(\alpha_i+\hat{\alpha}_i\right) \\ \text{sujeto a} & : \quad \sum_{i=1}^{l}\left(\alpha_i-\hat{\alpha}_i\right)=0 \\ & \hat{\alpha}_i,\alpha_i\geq 0 \end{array} \tag{4.17}$$

cuyas condiciones complementarias de Karush- Kuhn- Tuker son

$$\begin{array}{ll} \alpha_i\left(y_i-\left(\langle w\cdot x_i\rangle+b\right)-\varepsilon-\xi_i,\right)=0, & i=1,2,\cdots,l \\ -\hat{\alpha}\left(y_i-\left(\langle w\cdot x_i\rangle+b\right)+\varepsilon+\widehat{\xi}_i,\right)=0, & i=1,2,\cdots,l \\ \xi_i\widehat{\xi}_i=0\ \alpha_i\widehat{\alpha}_i=0, & i=1,2,\cdots,l \end{array} \tag{4.18}$$

nótese que tomando a $\beta=\alpha_i-\hat{\alpha}_i$ y usando $\alpha_i\widehat{\alpha}_i=0$ podemos escribir el problema dual de la siguiente forma

$$\begin{array}{ll} \text{maximizar} & -\frac{1}{2}\sum_{i,j=1}^{l}\beta_i\beta_j\left(\langle x_i\cdot x_j\rangle+\frac{1}{C}\delta_{ij}\right)+\sum_{i=1}^{l}\beta_i y_i-\varepsilon\sum_{i=1}^{l}|\beta_i| \\ \text{sujeto a} & : \quad \sum_{i=1}^{l}\beta=0,\ i=1,2,\cdots,l \end{array} \tag{4.19}$$

Comentario 4.2 *Para el caso en que $\varepsilon=0$ el problema de función de costo ε insensitive cuadratica corresponde a el problema de minimos cuadrados con un factor de decaimiento C el cual controla la importancia entre minimizar la complejidad del modelo y minimizar el error de entrenamiento. Así mismo si $C\rightarrow\infty$ el problema se vuelve el problema de minimos cuadrados sin restricciones.*

Proposición 4.2 *Suponga que se quiere realizar la identificación sobre un conjunto de entrenamiento*

$$S=((x_1,y_1)\cdots(x_l,y_l))$$

utilizando el kernel $K(x_i, x_j)$ *para definir un espacio de características implícito, y suponiendo que* α^* *es solución del siguiente programa cuadratico*

$$\begin{aligned} &\textit{maximizar} \quad w(\alpha) = -\tfrac{1}{2}\sum_{i,j=1}^{l}\alpha_i\alpha_j\left(K(x_i,x_j)+\tfrac{1}{C}\delta_{ij}\right)+\sum_{i=1}^{l}\alpha_i y_i-\varepsilon\sum_{i=1}^{l}|\alpha_i| \\ &\textit{sujeto a} \quad : \quad \sum_{i=1}^{l}\alpha_i = 0,\ i=1,2,\cdots,l \end{aligned} \tag{4.20}$$

sea $f(x)=\sum_{i=1}^{l}\alpha_i^* K(x_i,x)+b^*$, *donde* b^* *se escoge tal que* $f(x_i)-y_i=-\varepsilon-\alpha_i^*/C$, $\forall i$ *con* $\alpha_i^*>0$. *Entonces la función* $f(x)$ *es equivalente al hiperplano en el espacio de características definido por el kernel* $K(x_i,x_j)$ *que resuelve el problema de optimización 2.8.*

El método SVM ha sido muy exitoso en problemas de estimación. En [24] se realiza una aplicación del método SVM ε insensitive cuadrático. A partir de experimentos numéricos se muestra que el método obtiene buenos resultados, sin previo conocimiento del modelo. Acontinuación se presenta el algoritmo de ventanas deslizantes SVM ε insensitive para identificación en línea con paso unitario y factor de olvido λ.

Algoritmo 4.2 *1. Recopilación del conjunto de datos de entrenamiento*

$x_L=\{(x_1,y_1),\cdots,(x_{k+L},y_{k+L})\}$, *con* L *fija.*

2. *Obtener el modelo a través del método SVM* ε *insensitive cuadratico a partir del conjunto de datos de entrenamiento* x_L.

3. *Actualizar la ventana de tiempo. Se retienen en la ventana los datos anteriores*

4. $D_{old}=\{(x_2,y_2),\cdots,(x_{k+L},y_{k+L})\}$ *y se agrega el dato nuevo* $D_{new}=\{(x_{k+L+1},y_{k+L+1})\}$, *para formar el nuevo conjunto de entrenamiento* $x_{L+1}=D_{old}\lambda\cup D_{new}$ *donde* λ *es el factor de olvido,* $0<\lambda<1$.

5. *Se obtiene el modelo para el tiempo* $k+L+1$ *a través del método SVM* ε *insensitive cuadratico utilizando los datos de la ventana deslizante* x_{L+1}.

6. *Se realiza la actualización del paso 3 y se continúa con el algoritmo de manera recursiva.*

4.5. Ventanas deslizantes SVM Minimos Cuadrados con paso L

En esta sección se plantea el uso de las ventanas de tiempo deslizantes con el afán de obtener un algoritmo en línea basado en el método mínimos cuadrados SVM por segmentos de L pasos. El método Mínimos Cuadrados Support Vector Machine (LS- SVM, por sus siglas en ingles) es una versión del método SVM en el cual se elimina el términos de sensibilidad ε, esto es $\varepsilon = 0$. Mientras que el método de SVM resuelve un programa cuadrático, mínimos cuadrados SVM resuelve un programa lineal con restricciones de igualdad en lugar de restricciones de desigualdad, dicho método fue propuesto por Suykens y Vandellawe. En este método se utiliza la función de costo cuadrática. El problema de optimización para el método LS- SVM esta dado por la ecuación 2.33 cuyo problema dual se reduce a un sistema de ecuaciones 2.37.

Mínimos cuadrados SVM ha demostrado tener buenos resultados para la identificación fuera de línea, en [25],[26], [27] y [28] se describe el método mínimos cuadrados SVM para identificación de funciones en un ambiente fuera de línea. El método de mínimos cuadrados SVM resulta ser una herramienta muy poderosa para la identificación de sistemas no lineales, esto se puede ver en [29], [30] y [31]. Sin embargo no se ha realizado, aún, la extensión al caso de identificación en línea. En [32] se proponen un método mejorado para la obtención de la solución del método SL- SVM y muestran que el problema puede ser resuelto utilizando un sistema de ecuaciones lineales reducido. Pese a los esfuerzos realizados por reducir el número de cálculos realizados por el método de mínimos cuadrados SVM, estos mantienen el problema de la dependencia del tiempo por lo cual no son admisibles para la identificación en línea. A continuación se presenta el algoritmo de ventanas deslizantes SVM ε insensitive para identificación en línea con paso fijo L.

Algoritmo 4.3 *1. Recopilación del conjunto de datos de entrenamiento*

$x_L = \{(x_1, y_1), \cdots, (x_{k+L}, y_{k+L})\}$, *con* L *fija.*

2. Obtener el modelo a través del método L SVM a partir del conjunto de datos de entre-

namiento x_L.

3. *Actualizar la ventana de tiempo. Se eliminan los datos anteriores*

   $D_{old} = \{(x_1, y_1), \cdots, (x_{k+L}, y_{k+L})\}$ *y se mueve la ventana* L *pasos para recopilar los datos nuevos*

   $D_{new} = \{(x_{k+L+1}, y_{k+L+1}), \cdots, (x_{k+2L+1}, y_{k+2L+1})\}$.

4. *Se obtiene el modelo a partir del nuevo conjunto de entrenamiento* D_{new}.

5. *Se realiza la actualización del paso 3 y se continúa con el algoritmo de manera recursiva.*

4.6. Simulaciones

En esta sección se investigara el costo computacional de los tres algoritmos expuestos anteriormente, distintos valores para el tamaño de las ventanas serán propuestos así como valores del parámetros ε y σ. Se utilizara un sistema no lineal para ilustrar el desempeño de los algoritmos de ventanas deslizantes SVM. La planta es seleccionada es:

$$y(k) = \frac{y(k-1)\, y(k-2)\, y(k-3)\, u(k-1)\, [y(k-3)-1] + u(k-1)}{1 + y(k-2)^2 + y(k-3)^2} \tag{4.21}$$

la señal de entrada utilizada es:

$$u(k) = \begin{cases} \sin\left(\frac{\pi}{15}\right) & k < 200 \\ 1 & 200 \leq k < 400 \\ -1 & 400 \leq k < 600 \\ 0.3\sin\left(\frac{\pi}{15}k\right) + 0.1\sin\left(\frac{\pi}{22}k\right) + 0.603\sin\left(\frac{\pi}{8.5}k\right) & 600 \leq k < 1000 \end{cases} \tag{4.22}$$

Los datos de entrada para el algoritmo se definen como:

$$X(k) = [y(k-1), y(k-2), y(k-3), u(k-1), u(k-2)] \tag{4.23}$$

El procedimiento a seguir en las simulaciones será el siguiente. Primero se utilizará el algoritmo de ventanas deslizantes SVM ε insensitive para identificación en línea para identificar la planta 4.21 con la entrada 4.22 empleando ventanas deslizantes con $L = 5$, y

ε =0.01, ε =0.03 y ε =0.06. En segundo lugar se realizara la identificación de la misma planta utilizando el algoritmo de ventanas deslizantes SVM ε insensitive cuadrático con factor de olvido probando distintos valores para el factor de olvido λ y distintos valores de ε, ε =0.01, ε =0.03 y ε =0.06. Y por ultimo se utilizara el algoritmo ventanas deslizantes SVM Mínimos Cuadrados por segmentos donde se utilizaran distintos valores de L y σ.Las Figuras 4.5, 4.6 y 4.7 muestran los resultados de la identificación con el algoritmo ventanas deslizantes SVM ε insensitive. De estas simulaciones se observo que las ventanas deslizantes son una técnica la cual permite implementar SVM ε insensitive en un ambiente en línea. Tambien se puede apreciar que, como era de esperar, el factor ε determina el nivel de aproximación que se requiere. La selección de ε determina el nivel de precisión, conforme elegimos más pequeño a este parámetro la identificación es menos deficiente. Las Figuras 4.8, 4.9 y 4.10 muestran los resultados de las identificación con el algoritmo ventanas deslizantes SVM ε insensitive cuadrático con factor de olvido $\lambda = 1$, utilizando diferentes valores ε. Se puede observar que el nivel de aproximación esta definido de acuerdo a la selección del parámetro ε, conforme este parámetro es elegido mas pequeño obtenemos una mejor aproximación. En la Tabla 4.24 se muestran los resultados del costo computacional requerido por el algoritmo ventanas deslizantes SVM ε insensitive cuadrático para distintos valores de λ.

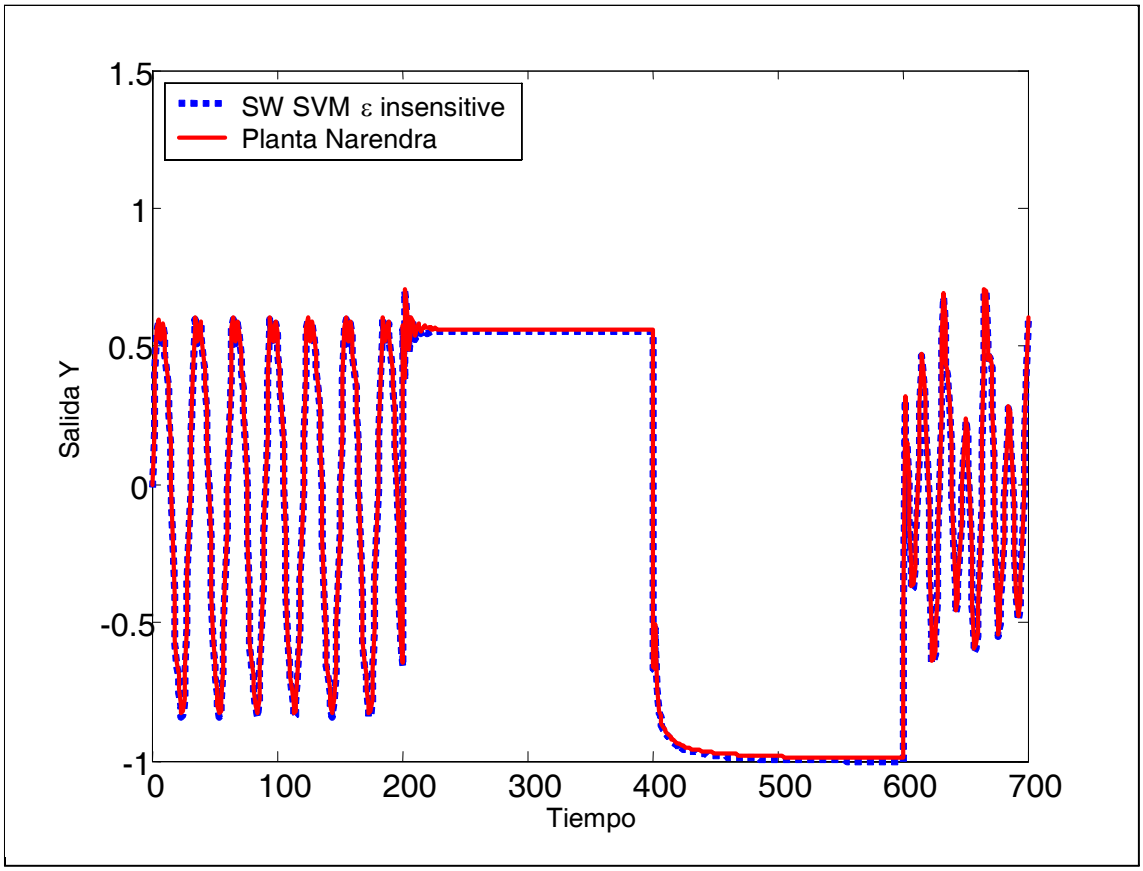

Figura 4.5: Identificación de la planta no líneal utilizando el algoritmo ventanas deslizantes SVM ε insensitive con $\varepsilon = 0{,}01$.

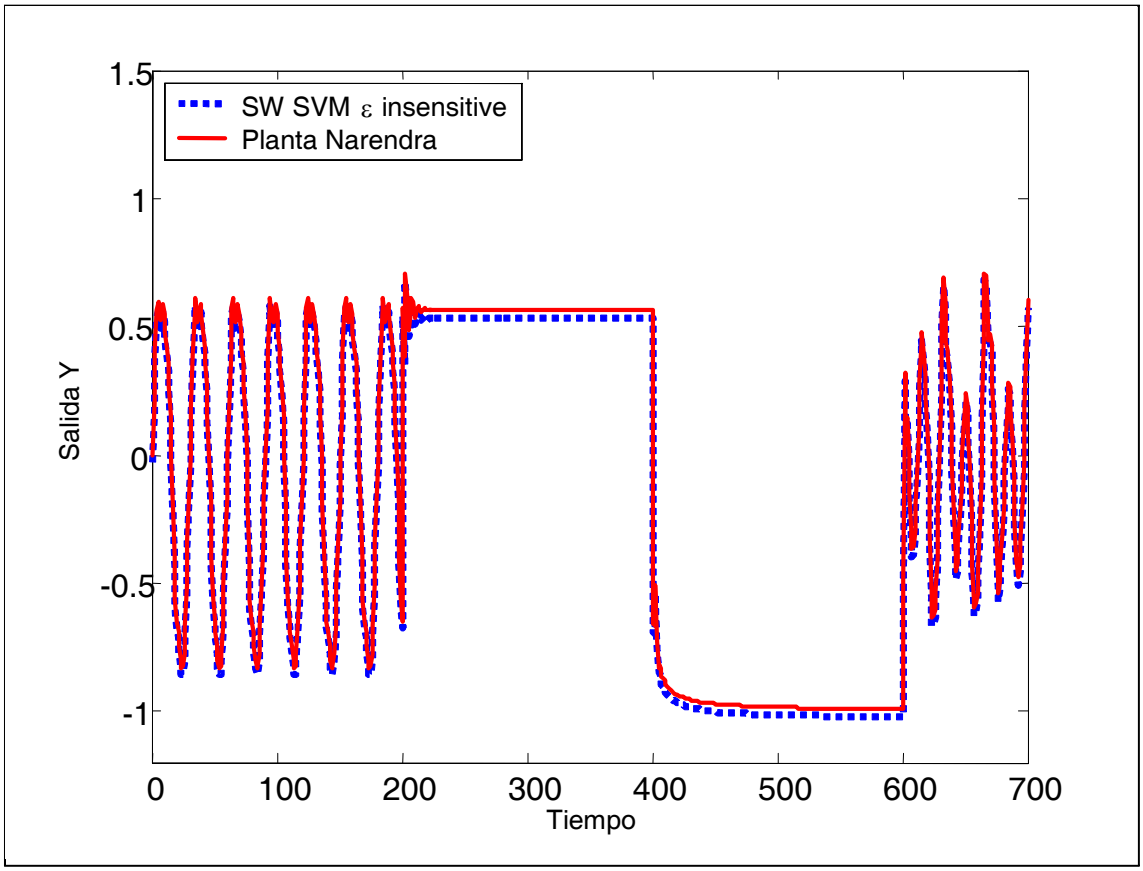

Figura 4.6: Identificación de la planta no líneal utilizando el algoritmo ventanas deslizantes SVM ε insensitive con $\varepsilon = 0{,}03$.

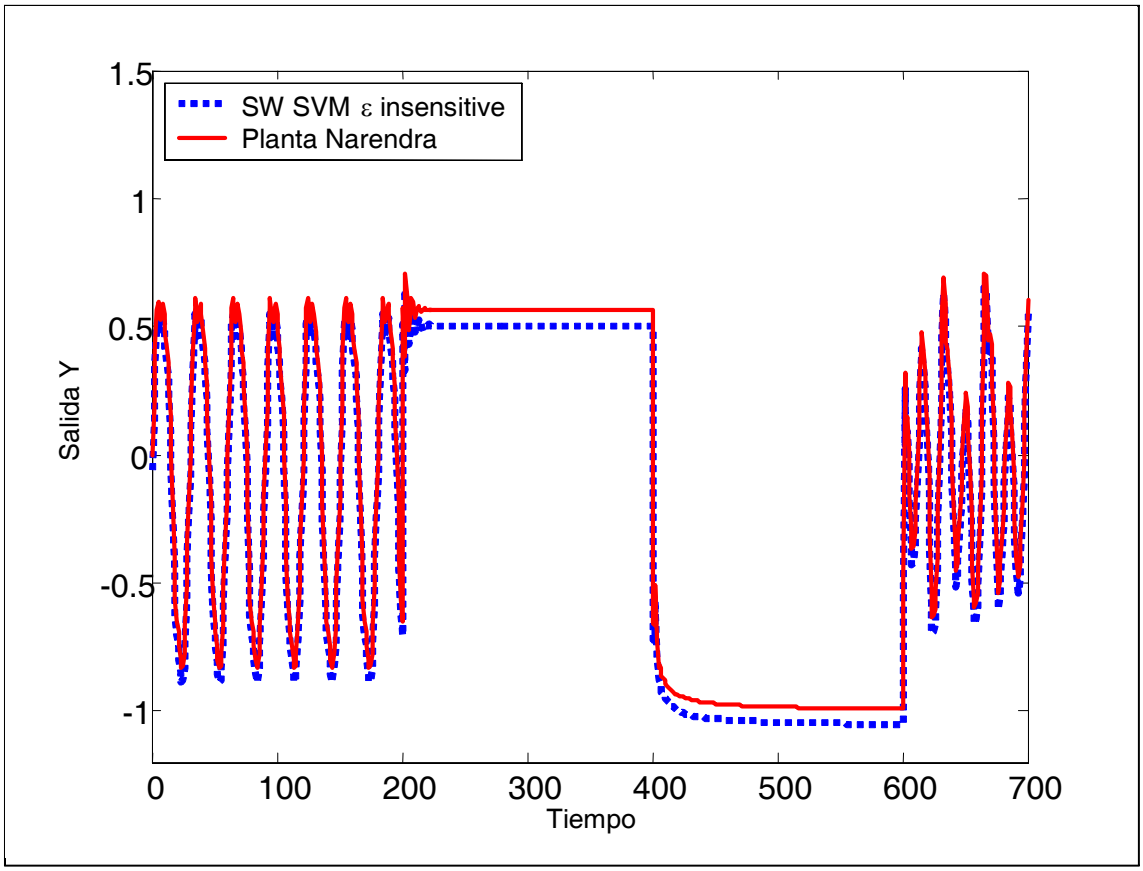

Figura 4.7: Identificación de la planta no líneal utilizando el algoritmo ventanas deslizantes SVM ε insensitive con $\varepsilon = 0{,}06$.

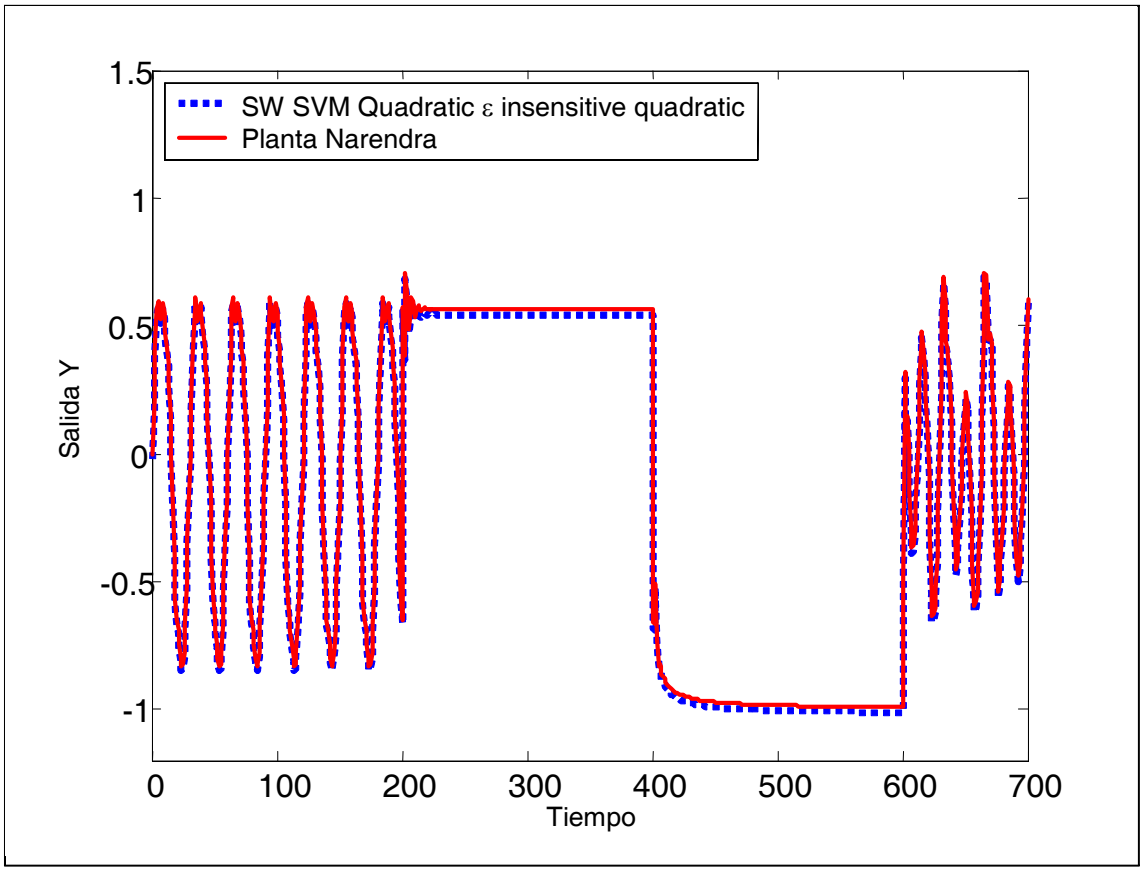

Figura 4.8: Identificación de la planta no líneal utilizando el algoritmo ventanas deslizantes SVM ε insensitive cuadratico con $\varepsilon = 0{,}02$.

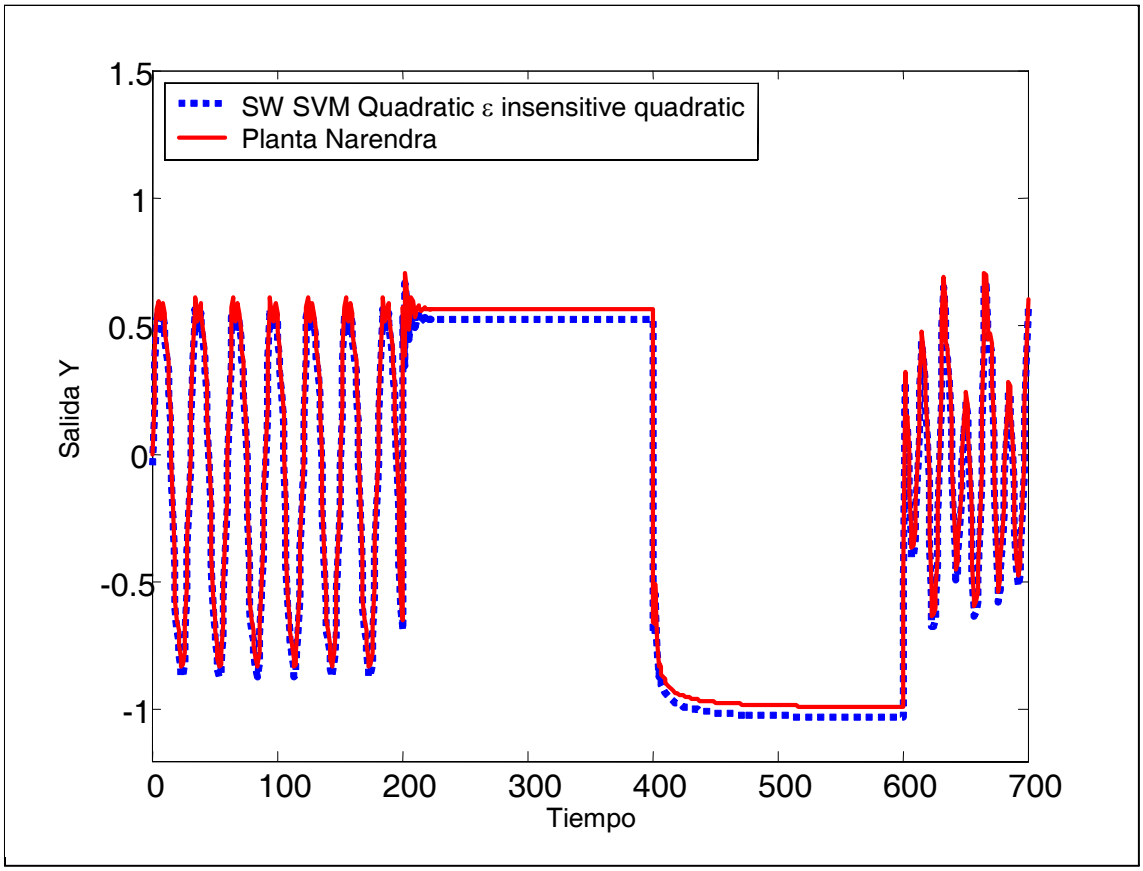

Figura 4.9: Identificación de la planta no líneal utilizando el algoritmo ventanas deslizantes SVM ε insensitive cuadratico con $\varepsilon = 0{,}04$.

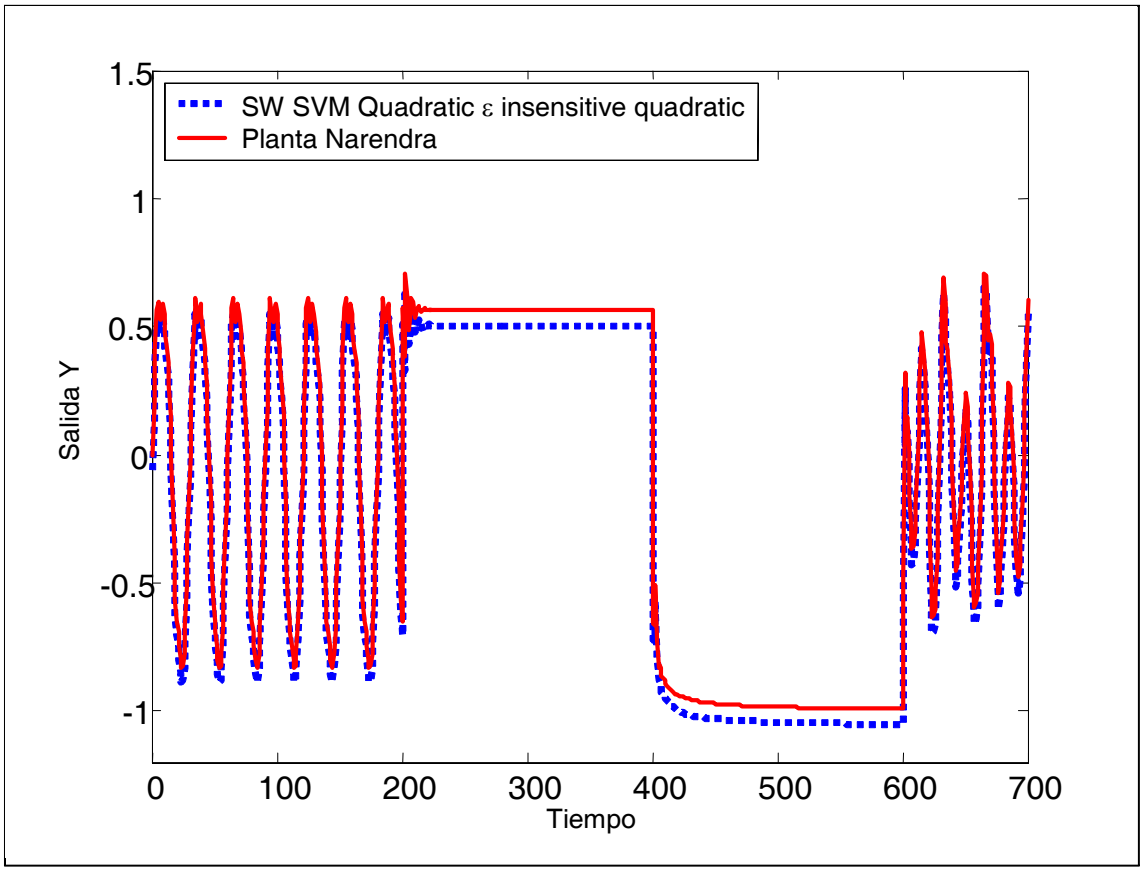

Figura 4.10: Identificación de la planta no líneal utilizando el algoritmo ventanas deslizantes SVM ε insensitive cuadratico con $\varepsilon = 0{,}06$.

λ	Tiempo de Computo (seg.)
$\lambda_1 = 0{,}1$	1.5
$\lambda_1 = 0{,}2$	1.546
$\lambda_1 = 0{,}3$	1.516
$\lambda_1 = 0{,}4$	1.547
$\lambda_1 = 0{,}5$	1.516
$\lambda_1 = 0{,}6$	1.547
$\lambda_1 = 0{,}7$	1.516
$\lambda_1 = 0{,}8$	1.516

Tabla 4.1: Resultados de algoritmo ventanas deslizantes SVM ε insensitive cuadratico con L=5, $\varepsilon = 0{,}001$, C=10000, Kernel RBF y $\sigma = 3$.

(4.24)

Por ultimo, en las Figuras 4.11 y 4.12 se muestran los resultados obtenidos con el algoritmo ventanas deslizantes SVM Mínimos Cuadrados con paso L. Para esta simulación se escogieron los valores C=1 y C=50, Kernel RBF y $\sigma = 3$. En general el algoritmo trabaja bien, variando el tiempo de computo de acuerdo a la dimensión de la ventana. También se observo que el parametro C determina el nivel en que el algoritmo sancionea a los errores de identificación. Al elegir un valor más grande para C lo que hacemos es sancionar con mayor rigidez a los errores de identificación por lo cual este es un parámetro de precisión de algoritmo.

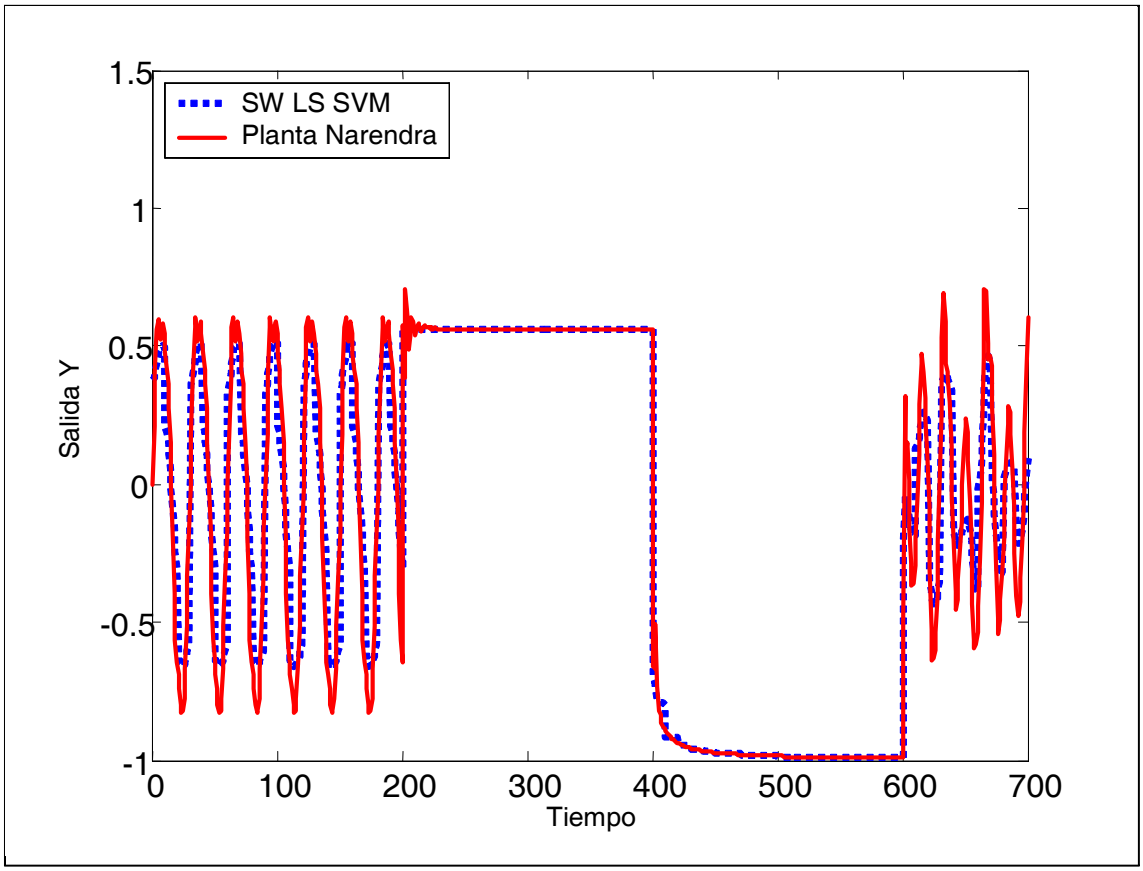

Figura 4.11: Identificación de la planta no líneal utilizando el algoritmo SW LSSVM con L=10 y C=1;

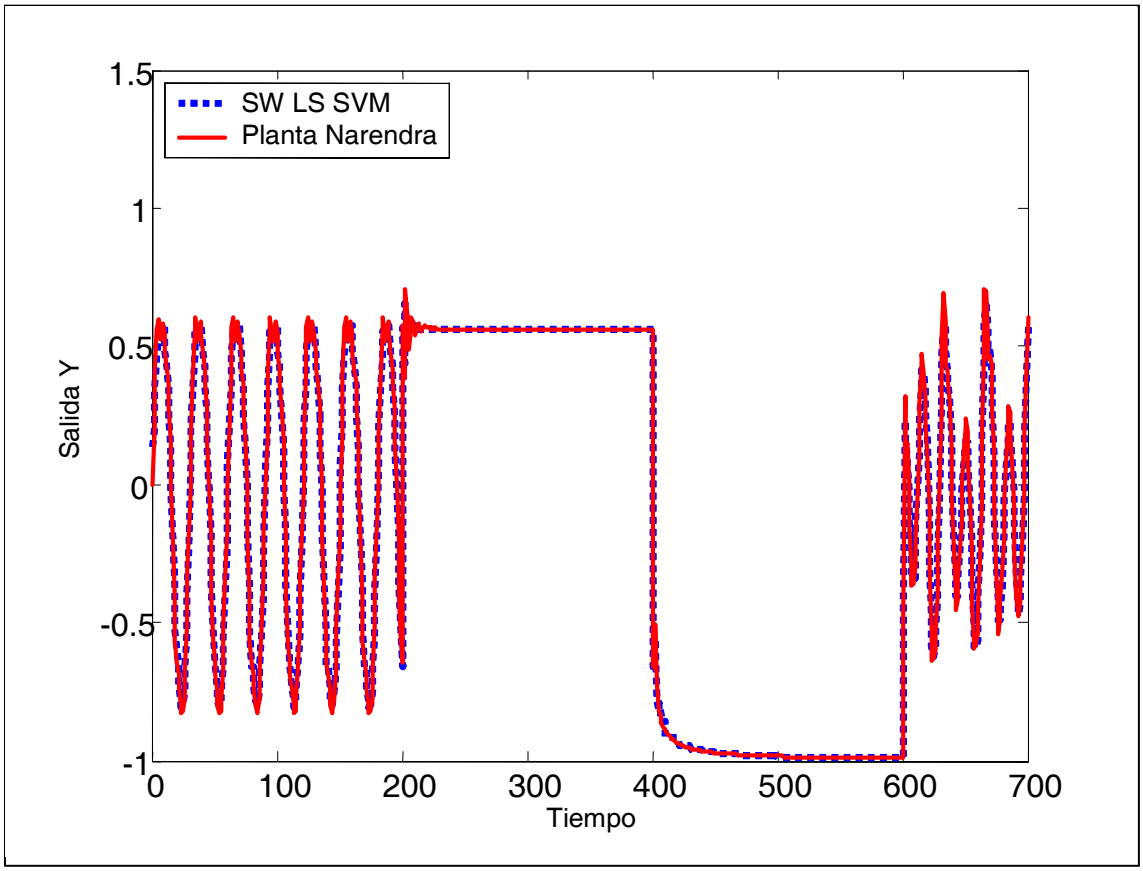

Figura 4.12: Identificación de la planta no líneal utilizando el algoritmo SW LSSVM con L=10 y C=50;

Capítulo 5

Concluciones

Se presento el método de identificación en línea supervisado SVM recursivo para identificación en línea. El método demostró eliminar el problema de la dependencia del tiempo que sufre el método convencional SVM. Esto se logro utilizando el método de aproximación linealmente independiente en línea. Aun cuando los experimentos realizados apenas palpen la vasta cantidad de aplicaciones en los cuales puede se usado el método, este mostró ser adecuado para su aplicación en un ambiente en línea, observando un dato en cada paso.

En esta tesis el método SVM Recursivo se comparo con los métodos convencionales LS SVM y SVM obteniendo un error medio cuadrático similar al de estos dos métodos, más sin embargo, computacionalmente hablando, SVM Recursivo obtuvo mejores resultados que SVM y LS SVM. A partir de las simulaciones el método SVM Recursivo para identificación en línea demostró mantener la dependencia del número de vectores de entrenamiento acotada. Como resultados de las simulaciones se observo que la selección del parámetro ν es de gran importancia ya que este define el nivel de precisión en la identificación. Los experimentos realizados se hicieron con un kernel RBF el cual mostró ser el más adecuado para identificación de sistemas no lineales. El método de dependencia lineal aproximada que se utilizo en SVM recursivo, depende en gran medida de la selección del parámetro σ, una buena selección de este parámetro permite el uso de SVM recursivo para aplicaciones en línea. Si la selección del término no es buena, la dimensión del diccionario crecerá de tal forma

que no sea posible la aplicación del algoritmo en un ambiente en línea. Podemos concluir que el método SVM Recursivo se puede aplicar en un ambiente en línea obteniendo buenos resultados de identificación.

Debido a su gran simplicidad y en particular al hecho de que pueden ser aplicadas de manera recursiva, las ventanas deslizantes mostraron ser una técnica admisible para la implementación de SVM en un ambiente en línea. Se propusieron los algoritmos: ventanas deslizantes SVM insensitive, ventanas deslizantes SVM insensitive cuadrático con factor de olvido y ventanas deslizantes SVM mínimos cuadrados con paso L para la identificación de sistemas no lineales. En esta tesis se observo que las técnicas de ventanas SVM presentadas proporcionan una actualización adecuada cuando la ventana deslizante es usada para definir un conjunto de datos de entrenamiento. Las simulaciones realizadas muestran que el tiempo de cómputo requerido por los algoritmos depende de la dimensión de la ventana de tiempo. Esta puede ser definida basándose en un conocimiento previo del proceso, pero en general no se cuenta con un método para determinar la dimensión.

5.1. Trabajos futuros

- Estudiar la mejor selección del parametro parámetro ν y σ para el algoritmo SVM Recursivo.
- Obtener metodos, asi como realizar teoria para seleccionar la mejor dimensión para las ventanas deslizantes.
- Utilizar SVM Recursivo en aplicaciones en tiempo real.

Bibliografía

[1] D.S.Chen and R.C.Jain, A robust back propagation learning algorithm for function approximation, *IEEE Trans. Neural Networks*, Vol.5, No.3, 1994.

[2] S.Wu and M.J.Er, Dynamic fuzzy neural networks- a novel approach to function approximation, *IEEE Trans. Syst., Man, Cybern. B*, Vol.30, 358-364, 2000.

[3] Vaptnik, V.N., 1995 The nature of Statical Learning Theory, New York: Wiley.

[4] V. Vaptnik, S. Golowich, and A. Smola. "Support vector method for function approximation. regression, estimation, and signal processing", In M. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems* 9, pg. 281- 287, Cambridge, MA, 1997. Mit Press.

[5] A.J. Smola and B.Schölkopf, "*Atutorial on Support Vector Regression*," Neuro COLT2 Technical Report Series, October 1998.

[6] N. Cristianini and J. Shawe- Taylor, .An Introduction to Support Vector Machines and other kernel based learning methods", Cambridge University Press, 2004.

[7] J.C. Platt, "Fast Training of Support Vector Machines Using Sequential Minimum Optimization," in Schölkopf, Burges and Smola, Eds., Advances in Kernel Methods– Support Vector Learning, Cambridge MA: MIT Press, 1998, pp 185-208.

[8] R. Collobert and S. Bengio, "SVMTorch: support vector machines for large-scale regression problems," J. Machine Learning Res., vol. 1, pp. 143–160, 2001.

[9] E. Osuna, R. Freund and F. Girosi, "An Improved Training Algorithm for Support Vector Machines,"Proc. 1997 IEEE Workshop on Neural Networks for Signal Processing, pp 276-285, 1997.

[10] Martin, M. (2002). On-line support vector machines for function approximation. (Tech. Rep. LSI-02-11-R).Catalunya, Spain: Software Department, Universitat Politecnica de Catalunya.

[11] Li- Na Li, Chao- Zhen Hou, The Identification of Industrial Processes Based on SVM, *Proceedings of the First International Conference on Machine Learning and Cybernetics,* Beijin, 4- 5 November 2002.

[12] Xunkai Wei, Yinghong Li, Chen Wang and Jianming Lu, *A Novel Identification Model of Aeroengine Based on Support Vector Machines, Proceedings of the 5^{th} World Congress on Intelligent Control and Automation,* 200- 203, June 2004.

[13] Ming Guang, Wei Wu Yan and Zhan Ting Yuan, Study of Nonlinear Identification Based on Support Vector Machine, *Proceedings of 3^{th} International Conference on Machine Learning and Cybernetics,* 3287- 3290, August, 2004.

[14] D.A. Karras, "Improved Defect Detection in Textile Visual Inspection Using Wavelet Analysis and Support Vector Machines", *ICGST International Journal on Graphics, Vision and Image Processing*, Vol. 05, 2005.

[15] Yaakov Engel, Shie Mannor, and Ron Meir, "Sparse Online Greedy Support Vector Regression", in Proc. 13th Eur. Conf. Machine Learning, 2002.

[16] M. Vogt, "SMO algorithms for support vector machines without bias term," Technische Univ. Darmstadt, Inst. Automat. Contr., Lab. Contr. Syst. Process Automat., Darmstadt, Germany, 2002.

[17] Yaakov Engel, Shie Mannor, and Ron Meir, "The Kernel Recursive Least-Squares Algorithm"*, IEEE Transactions on signal Processing, Vol. 52. No. 8, August 2004.*

[18] Phillip B. Gibbons, Srikanta Tirthapura, "Distributed Streams Algorithms for Sliding Windows", *In Proceedings of the Fourteenth Annual ACM Symposium on Parallel Algorithms and Architectures*, Winnipeg, Manitoba, Canada, August 2002.

[19] Lukasz Golab, David DeHaan, Erik D. Demaine, Alejandro L´opezOrtiz and J. Ian Munro, "Identifying Frequent Items in Sliding Windows over OnLine Packet Streams", *IMC'03*, October 27–29, 2003,Miami Beach, Florida, USA.

[20] M. Datar, A. Gionis, P. Indyk, and R. Motwani, "Maintaining stream statistics over sliding windows", *Proc. 13th ACM-SIAM Symp. on Discrete Algorithms*, pages 635-644, Jan. 2002.

[21] Liu San and Ming Ge, An Effective Approach for Nonlinear System Modeling, *International Symposium on Intelligent Control,* 73-77, September 2004.

[22] Hao Ran Zhang, Xiao Dong Wang, Chang Jiang Zhang, Xiu Ling Xu, Modeling Nonlinear Dynamical Systems Using Support Vector Machine, *Proceedings of the Hourth International Conference on Machine Learning and Cybernetics,* 3204- 3209, August 2005.

[23] Haina Rong, Gexiang Zhang, Cuifang Zhang, Application of Support Vector Machines to Nonlinear Systems Identification, *Autonomus Decentralized Systems,* April 2005, page(s): 501- 507.

[24] Dug Hung Hong and Changha Hwang, "Interval Regression Analysis Using Quadratic Support Vector Machine", *IEEE Transactions on Fuzzy Systems,* Vol. 13, No. 2, April 2005.

[25] J. A. Suykens, L. Lukas, J. Vandewalle, "Sparse Aproximation Using Least Squares Support Vector Machines", IEEE International Symposium on Circuits and Systems, 757- 759, May 2000.

[26] Ivan Goethals, Kristiaan Pelckmans, Johan A. K. Suykens, and Bart De Moor, Subspace Identification of Hammerstein Systems Using Least Squares Support vector Machines, *IEEE Transactions on Automatic Control,* Vol. 50, No. 10, October 2005.

[27] Tony Van Gestel, Johan A. K. Suykens, Dirk Emma Baestaens, Annemie Lambrechts, Gert Lanckriet, Bruno Vandaele, Bart De Moor and Joos Vandewalle, Financial Time Series Prediction Using Least Squares Support Vector Machines Within the Evidence Frame Work, *IEEE Transactions on Neuronal Networks,* Vol. 12, No. 4, July 2001.

[28] Johan A. Suykens, "Nonlinear Modelling and Support vector Machines", *IEEE Instrumentation and Measurement Technology Conference,* 287- 294, May 2001.

[29] Ming- Guang, Zhan- Ming Li and Wen-Hui Li, Study on Lest Squares Vector Machines Algorithm and its Aplication, *Proceeedings of the 17th International Conference on Tools with Artificial Intelligence,* 2005.

[30] Haisheng Li, Xuefeng Zhu, Buhai Shi, Non Linear Identification Based On Least Squares Support Vector Machine, *8th International Conference on Control, Automation, Robotics and Vision,* 2331- 2335, December 2004.

[31] G. T. Jemwa and C. Aldrich, Identification of Chaotic Process Systems with Least Squares Support Vector Machines, 2003.

[32] Wei Chu, Chong Jin Ong, and S. Sathiya Keerthi, An Improved Conjugate Gradient Schem to the Solution of Least Squares SVM, IEEE Transactions on Neural Networks, Vol, 16, No. 2, 498- 501, March 2005.

Apéndice A

Publicaciones

Juan Angel Resendiz-Trejo,Wen Yu, Support Vector Machine for Nonlinear System Online Identification . In *Int. Conf. on Electrical and Electronic Enginnering.*