Adaptive Support Vector Machines para predicción de series de tiempo

Elkin Eduardo García Díaz, Pedro Andrés Rangel, y Fernando Lozano Martínez

***Resumen*— Debido a su capacidad para extraer información a partir de un conjunto de muestras, las máquinas de vectores de soporte (SVM por sus siglas en inglés) se han convertido en una técnica ampliamente usada en problemas clasificación y regresión. En el caso del problema de regresión, se tiene una serie de tiempo y el objetivo es encontrar una función que pueda predecir su comportamiento en el futuro. Una de las suposiciones subyacentes de este algoritmo es que la serie de tiempo debe ser un proceso estacionario. Sin embargo, en la mayoría de casos de interés, la serie de tiempo constituye un proceso no estacionario variante en el tiempo. En este documento se presenta un procedimiento de entrenamiento que adapta SVM a series de tiempo no estacionarias.**

***Palabras Clave*— Support Vector Machines, Predicción de series de tiempo, Procesamiento de señales adaptativo, SVM, SVR.**

I. Introducción

LAS máquinas de vectores de soporte (*SVM*, por sus siglas en inglés) son algoritmos ampliamente usados en problemas de aprendizaje supervisado. El objetivo de un problema de aprendizaje supervisado es encontrar una función $h(x)$ que extraiga la información de un conjunto de datos de entrenamiento $\mathcal{S} = \{\langle x_i, z_i\rangle\}_{i=1}^m$. *SVM* encuentra esta función resolviendo un problema de programación cuadrática, la complejidad temporal del problema es $O(m^3)$. Múltiples investigaciones han propuesto métodos para mejorar esta complejidad [1]–[4] llegando a que sea usualmente $O(m^2)$.

En el procedimiento de entrenamiento de *SVM* se asume que el conjunto de datos de entrenamiento es independiente e idénticamente distribuido. Bajo esta suposición, si se tiene un número suficiente de datos, una máquina de vectores de soporte es capaz de encontrar un clasificador con el mínimo error posible [5]. Para el caso particular de regresión con *SVM* es mejor conocido como *SVR (Support Vector Regression)*. En *SVR*, el que los datos sean independientes e idénticamente distribuidos implica que la serie de tiempo debe ser un proceso estacionario. Sin embargo, en la mayoría de casos de interés, la serie constituye un proceso variante en el tiempo. En este documento se propone un método que permite usar *SVR* en procedimientos no estacionarios. La idea básica de este nuevo propuesto es adaptar el regresor encontrado por *SVR* a medida que el proceso no estacionario va cambiando. El algoritmo resultante es denominado *SVR Adaptativo (ASVR)*.

ASVR utiliza una variación del algoritmo *SMO (Sequential Minimal Optimization)* propuesto por Platt [4]. En cada paso, *SMO* divide el problema de programación cuadrática original en un problema de optimización más pequeño. La ventaja de este procedimiento radica en que el problema de optimización resultante puede resolverse analíticamente. Al usar un algoritmo de este tipo, se hace que el proceso adaptativo sea lo más rápido posible.

El resto del documento está organizado de la siguiente forma: en la sección II se introduce *SVM* para regresión, en la sección III se trata el problema de predicción de series de tiempo usando *SVR*, como resolver eficientemente el problema de optimización, qué métricas se usan para evaluar el desempeño y el nuevo algoritmo *ASVR*. En la sección IV se muestran las pruebas realizadas con *SVR* y *ASVR* y finalmente en la sección V se dan las conclusiones.

Universidad de los Andes; CMUA - Centro de Microelectrónica Universidad de los Andes; Grupo de Machine Learning

II. Preliminares y Notación

A. Support Vector Machines en el problema de regresión

Sea $\mathcal{S} = \{\langle \mathbf{x}_i, z_i\rangle\}_{i=1}^m$ un conjunto de m datos de entrenamiento, donde cada $z_i \in \mathbb{R}$ y donde cada $\mathbf{x}_i$ pertenece a un espacio de muestras $\mathbb{R}^d$. Considere ahora la función lineal $h : \mathbb{R}^d \mapsto \mathbb{R}$ expresada en (1)

$$h(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x}\rangle + b \quad \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R} \tag{1}$$

Donde $\langle \cdot, \cdot \rangle$ denota el producto punto en $\mathbb{R}^d$. El objetivo de *SVR* es encontrar una función lineal que no se equivoque más que ϵ en los datos de entrenamiento, y que adicionalmente sea lo menos compleja posible. La complejidad de la función lineal se calcula usando la norma de $\mathbf{w}$ [6], [7]. A partir de esto el problema que resuelve *SVR* es

$$\begin{aligned} \min_{\mathbf{w}\in\mathbb{R}^d, b\in\mathbb{R}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{s.a.} \quad & z_i - \langle \mathbf{w}, \mathbf{x}_i\rangle - b \leq \epsilon \\ & \langle \mathbf{w}, \mathbf{x}_i\rangle + b - z_i \leq \epsilon \quad \text{para} \quad i = 1, \ldots, m \end{aligned} \tag{2}$$

Sin embargo, la suposición tácita en (2) es que existe una función h que puede aproximar todos los datos de entrenamiento con una precisión ϵ, en otras palabras, que el problema de optimización es factible, aspecto que no necesariamente se cumple. Para solucionar este inconveniente se utiliza la versión *C-SVR* del problema de optimización, la cual incluye variables de holgura ξ_i y ξ_i^* [7]:

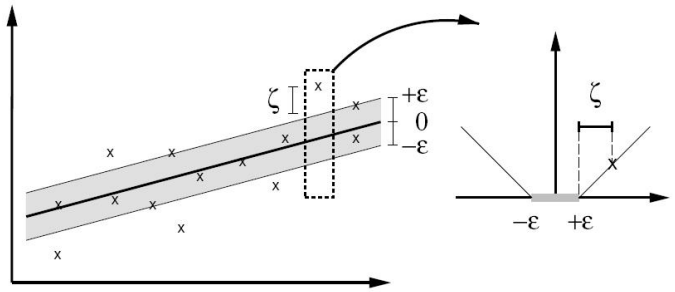


Fig. 1: Problema de optimización *C-SVR* y su función de error [7]

$$\begin{aligned} \min_{\substack{\mathbf{w}\in\mathcal{X}',\boldsymbol{\xi},\boldsymbol{\xi}^*\in\mathbb{R}^m \\ b\in\mathbb{R}}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{m}\sum_{i=1}^{m}(\xi_i+\xi_i^*) \\ \text{s.a.} \quad & \xi_i,\xi_i^* \geq 0 \\ & z_i - \langle \mathbf{w},\mathbf{x}_i\rangle - b \leq \epsilon + \xi_i \\ & \langle \mathbf{w},\mathbf{x}_i\rangle + b - z_i \leq \epsilon + \xi_i^* \\ & \qquad\qquad \text{para} \quad i = 1,\ldots,m \end{aligned} \tag{3}$$

Note que en este problema, la función de costo está constituida por dos partes: la primera intenta mantener el modelo lo menos complejo posible, mientras que la segunda intenta minimizar una función de costo sobre los datos de entrenamiento. La constante C permite equilibrar la complejidad del modelo obtenido con el número de errores que éste comete. El problema de optimización (3) encuentra el modelo con menor error de acuerdo con la función de perdida mostrada a continuación:

$$\epsilon\text{-error}(x) = \begin{cases} 0, & \text{si } |x| \leq \epsilon \\ |x| - \epsilon, & \text{si } |x| > \epsilon \end{cases} \tag{4}$$

Esta interpretación del problema de optimización que resuelve *SVR* es ilustrada en la figura 1. En esta figura, únicamente los puntos que se encuentran fuera del tubo de tamaño ϵ contribuyen en el error. Por otra parte, el error de los puntos por fuera del tubo aumenta linealmente.

Usualmente en lugar de resolver el problema de optimización (3) se resuelve su problema dual, el cual se muestra a continuación:

$$\begin{aligned} \min_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*\in\mathbb{R}^m} \; & f(\boldsymbol{\alpha},\boldsymbol{\alpha}^*) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)\langle \mathbf{x}_i,\mathbf{x}_j\rangle \\ & + \epsilon\sum_{i=1}^{m}(\alpha_i+\alpha_i^*) - \sum_{i=1}^{m} z_i(\alpha_i-\alpha_i^*) \\ \text{s.a.} \quad & \sum_{i=1}^{m}(\alpha_i-\alpha_i^*) = 0 \\ & 0 \leq \alpha_i,\alpha_i^* \leq \frac{C}{m} \qquad \text{para} \quad i = 1,\ldots,m \end{aligned} \tag{5}$$

Al resolver el problema dual (5) se obtienen los multiplicadores de Lagrange α_i y α_i^* de las restricciones del problema primal (3). A partir de estos multiplicadores de Lagrange se puede encontrar el vector $\mathbf{w}$:

$$\mathbf{w} = \sum_{i=1}^{m}(\alpha_i - \alpha_i^*)\mathbf{x}_i \tag{6}$$

B. Kernels

Cuando se usa el producto punto convencional en (1) se obtiene un modelo lineal; sin embargo, es posible obtener modelos no lineales mediante la transformación adecuada de los datos de entrenamiento [7]. Considere una transformación no lineal $\Phi : \mathbb{R}^d \mapsto \mathcal{X}'$ que va del espacio de entrada $\mathbb{R}^d$ al espacio de características $\mathcal{X}'$. Si los datos de entrada son transformados antes de resolver el problema de optimización de *SVR*, entonces el modelo obtenido es de la siguiente forma:

$$h(x) = \langle \mathbf{w}, \Phi(x)\rangle_{\mathcal{X}'} + b \quad \mathbf{w}\in\mathcal{X}', b\in\mathbb{R} \tag{7}$$

Donde $\langle\cdot,\cdot\rangle_{\mathcal{X}'}$ representa al producto punto en el espacio de características $\mathcal{X}'$. Para encontrar dicho modelo se plantea el problema primal y dual, tal y como se hizo en la sección A. En el problema de optimización dual (5), los datos de entrada $\mathbf{x}_i$ solamente aparecen en términos de sus productos punto $\langle \mathbf{x}_i,\mathbf{x}_j\rangle$. Para obtener el modelo (7) se utilizará el producto punto de los datos transformados $\langle \Phi(x_i),\Phi(x_j)\rangle_{\mathcal{X}'}$.

Una forma de calcular eficientemente el producto punto en el espacio de características es usar una función $k : \mathbb{R}^d\times\mathbb{R}^d \mapsto \mathbb{R}$ que esté dotada con la siguiente propiedad:

$$k(x_i,x_j) = \langle \Phi(x_i),\Phi(x_j)\rangle_{\mathcal{X}'} \tag{8}$$

A esta función k se le conoce como *kernel*.

Ahora, el problema 5 en el espacio de características $\mathcal{X}'$ es

$$\begin{aligned} \min_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*\in\mathbb{R}^m} \quad & f(\boldsymbol{\alpha},\boldsymbol{\alpha}^*) = \frac{1}{2}\left(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*\right)^T K\left(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*\right) \\ & + \epsilon\sum_{i=1}^{m}(\alpha_i+\alpha_i^*) + \sum_{i=1}^{m} z_i\left(\alpha_i-\alpha_i^*\right) \\ \text{s.a.} \quad & \sum_{i=1}^{m}(\alpha_i-\alpha_i^*) = 0 \\ & 0 \leq \alpha_i,\alpha_i* \leq C \qquad \text{para} \quad i = 1,\ldots,m \end{aligned} \tag{9}$$

Donde $K_{i,j} = k(x_i,x_j)$.

C. Sequential Minimal Optimization: SMO

Considere el problema de programación cuadrática en forma matricial mostrado en la ecuación (10), donde $\boldsymbol{\alpha}\in\mathbb{R}^m$ es el vector que contiene los parámetros a optimizar, $Q\in\mathbb{R}^{m\times m}$ es una matriz definida semi-positiva, $\mathbf{e}\in\mathbb{R}^m$ es un vector de unos, y $\mathbf{y}\in\mathbb{R}^m$ es un vector cualquiera.

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha} - \mathbf{e}^T\boldsymbol{\alpha} \\ \text{s.a.} \quad & \mathbf{y}^T\boldsymbol{\alpha} = 0 \\ & 0 \leq \alpha_i \leq \frac{C}{m} \qquad \text{para} \quad i = 1,\ldots,m \end{aligned} \tag{10}$$

Cuando la dimensión de la matriz Q es muy alta, no es posible utilizar métodos convencionales para resolver (10).

Algorithm: SMO $\left(Q, \frac{C}{m}, \boldsymbol{\alpha}^1 \text{factible}\right)$
Input: SMO $\left(Q, \frac{C}{m}, \boldsymbol{\alpha}^1 \text{factible}\right)$
Output: $\boldsymbol{\alpha}$ óptimo

for $k = 1$ **to** $\boldsymbol{\alpha}^k = \boldsymbol{\alpha}$ *óptimo* **do**

1) Encontrar dos elementos en el conjunto de trabajo $B = \{i, j\} \subset \{1, \ldots, m\}$.
2) Definir $N \equiv \{1, \ldots, m\} - B$
3) Definir $\boldsymbol{\alpha}_N^k$ y $\boldsymbol{\alpha}_B^k$ como los sub-vectores de $\boldsymbol{\alpha}^k$ correspondientes a N y B respectivamente.
4) Resolver el sub-problema de optimización

$$\begin{aligned} \min_{\alpha_i, \alpha_j} \quad & \frac{1}{2} [\alpha_i, \alpha_j] \begin{bmatrix} Q_{ii} & Qij \\ Q_{ji} & Q_{jj} \end{bmatrix} \begin{bmatrix} \alpha_i \\ \alpha_j \end{bmatrix} \\ & + (-\mathbf{e}_B + Q_{BN} \boldsymbol{\alpha}_N^K)^T \begin{bmatrix} \alpha_i \\ \alpha_j \end{bmatrix} \\ \text{s.a.} \quad & 0 \leq \alpha_i, \alpha_j \leq \frac{C}{m} \\ & y_i \alpha_i + y_j \alpha_j = -\mathbf{y}_N^T \boldsymbol{\alpha}_N^T \end{aligned}$$

5) Actualizar α_i^{k+1} y α_j^{k+1} usando los óptimos encontrados en el paso anterior.
6) Actualizar $\boldsymbol{\alpha}_N^{k+1} = \boldsymbol{\alpha}_N^k$

end

Fig. 2: Algoritmo *SMO*.

En este caso, se pueden usar algoritmos de descomposición para resolver el problema de optimización. Los algoritmos de descomposición son procedimientos iterativos que optimizan solamente un pequeño subconjunto de las variables en cada ronda. En el caso extremo se optimizan solo dos variables por ronda, a este algoritmo se le conoce como algoritmo secuencial de optimización mínima, *SMO* por sus siglas en inglés, y fue usado por primera vez en el entrenamiento de *SVM* por Platt [4]. *SMO*, tal y como es explicado en [8], es ilustrado en la figura 2.

Ahora bien, reescribiendo en forma matricial el problema (9) se obtiene

$$\begin{aligned} \min_{\boldsymbol{\alpha}, \boldsymbol{\alpha}^* \in \mathbb{R}^m} \quad & f(\boldsymbol{\alpha}, \boldsymbol{\alpha}^*) = \frac{1}{2} \left[\boldsymbol{\alpha}^T, \boldsymbol{\alpha}^{*T}\right] \begin{bmatrix} K & -K \\ -K & K \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\alpha}^* \end{bmatrix} \\ & + \left[\epsilon\, \mathbf{e}^T + \mathbf{z}^T, \epsilon\, \mathbf{e}^T - \mathbf{z}^T\right] \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\alpha}^* \end{bmatrix} \\ s.a. \quad & \left[\mathbf{e}^T, -\mathbf{e}^T\right] \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\alpha}^* \end{bmatrix} \\ & 0 \leq \alpha_i, \alpha_i^* \leq C \qquad \text{para} \qquad i = 1, \ldots, m \end{aligned} \tag{11}$$

Es claro que (11) es un problema de programación cuadrática de las mismas características que (10) y es fácilmente resuelto utilizando *SMO*.

La combinación de *SMO* con técnicas de *shrinking* y *caching* [3] ha permitido que la complejidad computacional del algoritmo de entrenamiento de *SVM* esté entre cuadrática y cúbica. Dicha complejidad computacional depende del tipo de problema y está básicamente dominada por el número de evaluaciones requeridas de la función kernel [9].

III. Regresión de Series de Tiempo

En el caso de regresión de series de tiempo se cuenta con un conjunto de datos $\mathcal{Z} = \{z_1, z_2, \ldots, z_n\}$ y se buscan predecir los siguientes valores de la serie $z_{n+1}, z_{n+2}, \ldots$. En estos casos a cada z_i se le debe asignar un vector x_i que son los d valores anteriores de la serie, es decir que $x_i = [z_{i-d}, z_{i-d+1}, \ldots z_{i-2}, z_{i-1}]$ con lo cual escogido un valor de d para una serie $\mathcal{Z}$ de n elementos, el número de datos de entrenamiento es $n - (d + 1)$, formando el conjunto de datos de entrenamiento $\mathcal{S} = \{\langle x_i, z_i \rangle\}_{i=d+1}^n$.

Adicionalmente, cada vector x_i puede contener información adicional a los z_i proveniente de información correlacionada, por ejemplo para predecir el valor de una acción en la bolsa, puede ser de utilidad tener información sobre los precios del dolar, razón por la cual estos valores se incorporan a x_i

A. Medida de desempeño

Un aspecto de gran importancia a la hora de medir el desempeño de un predictor es tener una métrica adecuada que permita comparar el desempeño de diferentes métodos, uno de los más comunes es el *error cuadrático medio normalizado* [10], *NMSE* por sus siglas en inglés.

El *NMSE* se define como

$$NMSE = \frac{1}{\sigma^2 N} \sum_{i=1}^{N} (z_i - \hat{z}_i)^2 \tag{12}$$

En donde z_i es el valor real de la serie, $\hat{z}_i$ es la predicción y σ^2 es la varianza de la serie real sobre el periodo de duración N. Un valor de $NMSE = 1$ corresponde a predecir la media, e idealmente se busca que $NMSE = 0$, es decir que la predicción sea igual al valor real.

Otra medida común es el *error absoluto medio* [10], *MAE* por sus siglas en inglés, definido como

$$MAE = \frac{\sum_{i=1}^{N} |\hat{z}_i - z_i|}{N} \tag{13}$$

Otras medidas de mayor complejidad toman en cuenta el error de la predicción en la dirección, tal es el caso de *DS (Directional Symmetry)* y *WDS (Weighted Directional Symmetry)*

B. Regresión de series de tiempo Adaptativas

Cuando las series de tiempo no son estacionarias, el modelo utilizado para su predicción usando *SVR* pierde su validez a medida que el proceso cambia sus características, sin embargo hay dos aspectos a tener en cuenta:

- La mayoría de estos procesos cambian sus características de forma gradual y suave.
- Para ventanas de tiempo de determinado tamaño, los cambios en el proceso son tan pequeños que el proceso se puede considerar estacionario en esa ventana.

Una solución a este problema es tener una hipótesis $h_j(x)$ para cada instante de tiempo futuro $t_{n+1}, \ldots, t_j, \ldots, t_k$, así se podrían predecir de forma paralela $\hat{z}_{n+1}, \ldots, \hat{z}_k$ dado x, para tener en cuenta que $\hat{z}_j$ no necesariamente está relacionado

con $\hat{z}_i$ si $i \neq j$. Sin embargo el primer inconveniente que surge es que el tener k hipótesis implica no solo un costo computacional k veces más grande sino que el problema se vuelve intratable si se piensa que no hay restricciones de tiempo para la predicción a futuro.

Por otra parte, existiría un esfuerzo computacional innecesario si se piensa que en intervalos de tiempo cortos, la señal es estacionaria y no serían necesarios tantos predictores. Según esto sería conveniente encontrar a priori estos intervalos y así disminuir el esfuerzo computacional de hallar tantos predictores, aunque por las características de estas señales esto puede ser utópico puesto que de poderse hallar en un tiempo lejano, sería un indicio de comportamiento estacionario.

Por estas razones y dado que es posible conocer paulatinamente el desempeño del modelo propuesto al comparar los valores predichos con los valores reales, se propone un nuevo algoritmo que se adapte a los cambios de las series no estacionarias, modificando el modelo en la medida de lo necesario, de acuerdo al desempeño de éste en el transcurso del tiempo, minimizando el esfuerzo computacional.

El algoritmo *ASVR* tal como se muestra en la figura 3 parte de una hipótesis producida por *SVR*, obtenida a partir de la solución del problema de programación cuadrática (11), sin embargo a medida que se realizan predicciones más alejadas en el tiempo, debido a que están cambiando las características del proceso, éstas se alejan de la realidad, lo cual se puede corroborar a medida que se comparan las predicciones de los tiempos anteriores respecto a los valores reales por medio del error e, a partir de ello, se puede determinar el momento indicado de reajustar el modelo. Esto es posible gracias a que los cambios en el proceso son lentos y el nuevo modelo guarda similitud con el modelo inmediatamente anterior.

Para el caso particular de la actualización del modelo de *SVR* se deben tener en cuenta dos aspectos para resolver eficientemente el nuevo problema de programación cuadrática, dado el cambio de los datos de entrada:

- Se toma como punto inicial factible del nuevo modelo el punto factible del modelo anterior.
- Cada vez que se halle un nuevo dato de la predicción es necesario actualizar el gradiente utilizado en la resolución del problema de programación cuadrático inicial.

Estos aspectos combinados con técnicas poderosas de optimización como *SMO* hacen este algoritmo bastante simple y eficiente a nivel de recursos computacionales.

Algorithm: ASVR $(\mathcal{Z}, p, SVR, \mathrm{C}, \mathrm{kernel}, \epsilon, e_{max})$
Input: ASVR $(\mathcal{Z} = \{z_i\}_{i=1}^{n}, p, SVR, \mathrm{C}, \mathrm{kernel}, \epsilon, e_{max})$
Output: $\hat{\mathcal{Z}}$

A partir de d y $\mathcal{Z}$ hallar los x_i para formar la secuencia $\mathcal{S} = \{x_i, z_i\}_{i=d+1}^{n}$
Obtener la hipótesis $h(x)$ con *SVR*

$$h(x) \leftarrow SVR(\mathcal{S}, \mathrm{kernel}, \mathrm{C}, \epsilon)$$

for $t = n$ **to** *Condición de terminación* **do**
Hacer $x_{IN} = [z_{t-d+1}, z_{t-d+2}, \ldots z_t]$

$$\hat{z}_{t+1} = h(x_{IN})$$

Recibir z_{t+1} y hallar el error de la predicción

$$e = |\hat{z}_{t+1} - z_{t+1}|$$

if $e \leq e_{max}$ **then**

$$z_{t+1} = \hat{z}_{t+1}$$

else
Formar la secuencia $\mathcal{S} = \{x_i, z_i\}_{i=t-n+d+2}^{t+1}$
Reentrenar el modelo

$$h(x) \leftarrow SVR(\mathcal{S}, \mathrm{kernel}, \mathrm{C}, \epsilon)$$

end
end

$$\hat{\mathcal{Z}} = \{\hat{z}_{n+1}, \hat{z}_{n+2}, \ldots\}$$

Fig. 3: Algoritmo *ASVR*.

TABLA I: Características Bases de Datos

Bases de Datos	σ	C	ϵ	d
sen-cte	20	10	0.01	5
Serie-A	5400	500	1	20
sen-var	20	10	0.01	5

IV. EXPERIMENTOS

En esta sección se muestran algunos experimentos de regresión de series de tiempo usando el algoritmo propuesto *ASVR* con diferentes conjuntos de datos tanto reales como artificiales comparando su desempeño con el algoritmo tradicional *SVR*.

El kernel utilizado fue un kernel gaussiano puesto que es un kernel universal sencillo y presenta un buen desempeño en predicción de series [7].

Se utilizaron 3 conjuntos de datos, el primero es una señal seno de frecuencia constante *sen-cte* dada por $z(n) = \mathrm{sen}(2\pi f n)$ tomando como conjunto de entrenamiento un número de puntos suficiente para representar un periodo. El segundo es la serie de tiempo A del Santa Fe Institute *Serie-A* [10]. Esta serie corresponde a las fluctuaciones de un láser infrarrojo lejano, descrito "aproximadamente" por 3 ecuaciones diferenciales ordinarias no lineales acopladas. Es claro que esta serie es bastante compleja pues corresponde a un proceso altamente no lineal y caótico, lo que dificulta su predicción, aunque aún es un proceso estacionario y los datos de entrenamiento y evaluación no tienen ruido apreciable. La tercera serie corresponde a un proceso no estacionario, una señal seno de frecuencia variable *sen-var*, la cual se incrementa linealmente y es de la forma $z[n] = \mathrm{sen}(2\pi f[n]n)$, que aunque es un cambio sencillo, dificulta su predicción. Los datos adicionales del modelo para cada una de las series se aprecian en la tabla I

Con respecto a *sen-cte* se tomaron los primeros 100 puntos para predecir los siguientes 400 usando *SVR*, esto fue suficiente para tener un error imperceptible, $NMSE < 10^{-5}$, es de destacar que al utilizar *ASVR* la predicción es la misma pues no se hace necesario ninguna clase de actualización por el error tan bajo. Los resultados se pueden apreciar en la figura 4

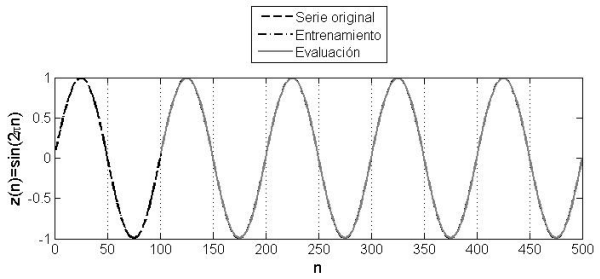


Fig. 4: Seno de frecuencia constante Entrenamiento y Evaluación

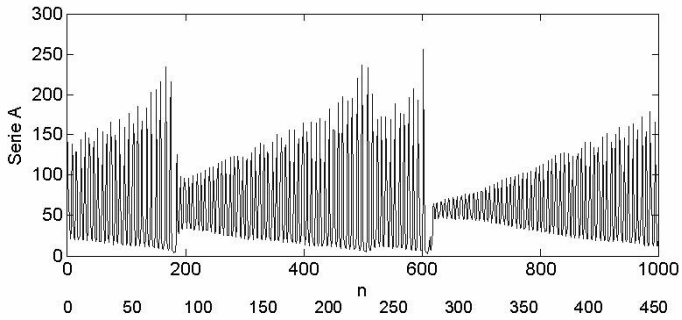


Fig. 5: Serie A Santa Fe Institute - Entrenamiento

Para la *Serie-A* se tomaron los primeros 1000 datos para predecir los siguientes 100, de la misma forma que la competencia del Santa Fe Institute [10]. Los datos de entrenamiento se aprecian en la figura 5. En este caso al ser un proceso estacionario [10], *SVR* y *ASVR* producen la misma respuesta, la cual se destaca por ser bastante precisa, como se aprecia en la figura 6.

Como medida de desempeño, se destaca que el *NMSE* es de $0{,}0213$ un valor bastante bueno si se tiene en cuenta que en la competencia en donde se usó esta serie el mejor $NMSE$ fue de $0{,}0273$ [10], el cual se mejoró en este caso, con lo que se resalta la versatilidad de *SVR* para predecir series de tiempo. Es importante destacar que en [10] ninguno de los métodos utilizados incluye *SVR* o algún método basado en *SVM*.

Estas pruebas muestran que el método tradicional de *SVR* es suficiente para series de tiempo sin ruido y estacionarias, características que por lo general no cumplen la mayoría de las series de tiempo de la vida real, en donde el ruido es elemento cotidiano tanto en los conjuntos de entrenamiento como de evaluación y los procesos cambian características importantes que hacen que dejen de ser estacionarios. Sin embargo respecto

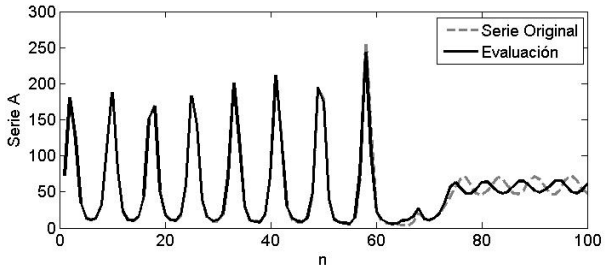


Fig. 6: Serie A Santa Fe Institute - Evaluación

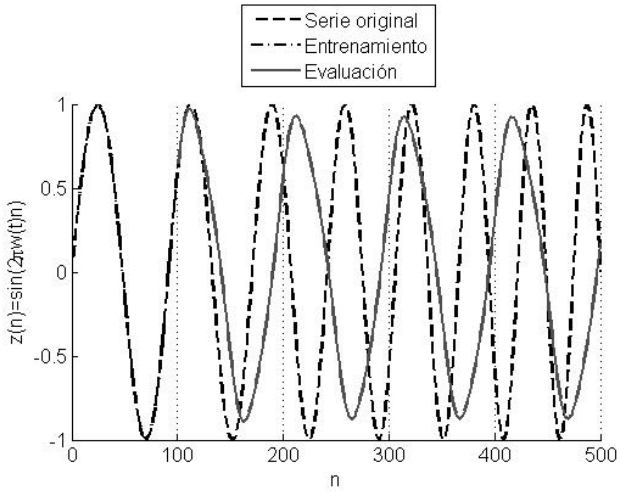


Fig. 7: Seno de frecuencia variable: Entrenamiento y Evaluación con *SVR*

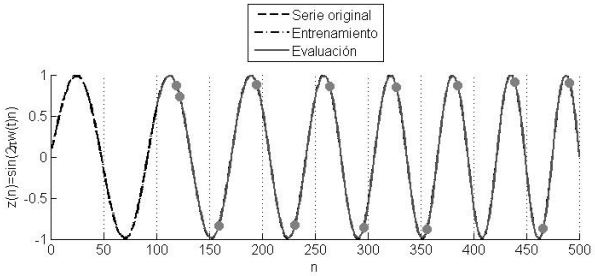


Fig. 8: Seno de frecuencia variable: Entrenamiento y Evaluación con *ASVR*

a esta última característica, como se mencionaba en la sección B, en una ventana de tiempo determinada la mayoría se pueden aproximar a procesos estacionarios.

Como se describe anteriormente, *sen-var* es una señal no estacionaria, al utilizar *SVR* los resultados de la serie predicha se alejan bastante de la señal original como se muestra en la figura 7. Es bastante claro que la señal predicha trata de conservar una frecuencia cercana a la de la señal original y a medida que avanza el tiempo, no toma en cuenta los cambios que tiene la señal (su característica de ser no estacionaria), razón por la cual el modelo utilizado para predecirla no es válido después de cierto tiempo.

Por esta razón se utiliza *ASVR* con un error permitido $e_{max} = 0{,}025$, los resultados se aprecian en la figura 8 en donde es claro el desempeño superior respecto a *SVR*, se resaltan los momentos en los que se realiza una actualización del modelo a medida que va cambiando la frecuencia paulatinamente, corrigiéndolo antes que el modelo pierda validez.

Adicionalmente, para probar la eficacia del algoritmo propuesto frente al ruido, a la señal anterior se le suma ruido blanco gaussiano con media cero y varianza creciente de forma cuadrática respecto al tiempo. Los resultados de la predicción se aprecian en la figura 9. A pesar de tener una señal con ruido, la predicción es similar a la señal original pero además, la predicción es una señal más suave, esto puede tomarse como si la presencia del ruido se disminuyera, de donde este

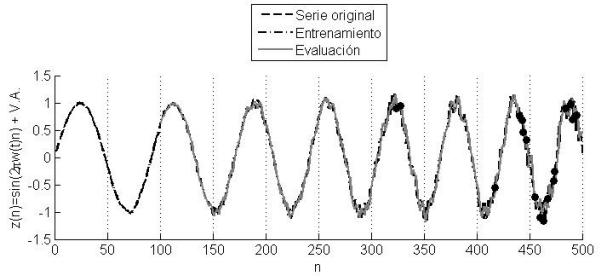


Fig. 9: Seno de frecuencia variable con ruido: Entrenamiento y Evaluación con *ASVR*

método favorece la predicción de series ruidosas, claro está que en este caso el error permitido fue $e_{max} = 0{,}3$ respecto a $e_{max} = 0{,}025$ para la señal sin ruido, puesto que de lo contrario el modelo se reentrenaría innecesariamente debido a los cambios bruscos de la señal producidos por el ruido. Aunque es claro que a medida que aumenta el nivel de ruido, se hace más frecuente el reentrenamiento.

V. Conclusiones

Las pruebas realizadas demuestran la versatilidad y eficacia de *SVR* para predecir series de tiempo, sin embargo al suponer que estas señales son procesos estacionarios, se restringen los campos de aplicación de esta clase de métodos pues es una característica poco frecuente en el mundo real.

El método propuesto *ASVR* es una extensión de *SVR* para series de tiempo no estacionarias que permite de forma sencilla y eficiente modificar el modelo de predicción en la medida que se requiera, dado que los cambios de los procesos no estacionarios son lentos. Las pruebas realizadas demuestran como, un solo modelo adaptable es suficiente para predecir series no estacionarias, obteniendo desempeños bastante buenos, aún en presencia de ruido, el cual afectaba muy poco el desempeño del predictor.

Quedan abiertos como trabajos futuros, la implementación de este tipo de algoritmos en series de tiempo reales, de mayor duración y multidimensionales como sonido estéreo o video, así como la evaluación de esta clase de métodos para compresión de este tipo de señales apoyados en la predicción en tiempo real.

Referencias

[1] V. Vapnik, *Estimation of Dependences Based on Empirical Data [in Russian]*. Moscow: Nauka, 1979, (English translation: Springer Verlag, New York, 1982).

[2] E. Osuna, R. Freund, and F. Girosi, "An improved training algorithm for support vector machines," in *Neural Networks for Signal Processing VII — Proceedings of the 1997 IEEE Workshop*, J. Principe, L. Gile, N. Morgan, and E. Wilson, Eds. New York: IEEE, 1997, pp. 276– 285.

[3] T. Joachims, "Making large–scale SVM learning practical," in *Advances in Kernel Methods — Support Vector Learning*, B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds. Cambridge, MA: MIT Press, 1999, pp. 169–184.

[4] J. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods — Support Vector Learning*, B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds. Cambridge, MA: MIT Press, 1999, pp. 185–208.

[5] I. Steinwart, "Cosistency of support vector machines and other regularized kernel classifiers," *IEEE Transactions on Information Theory*, vol. 51, no. 1, pp. 128–142, January 2005.

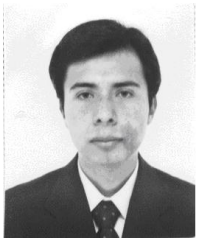
[6] A. Smola and B. Schölkopf, "On a kernel-based method for pattern recognition, regression, approximation and operator inversion," *Algorithmica*, vol. 22, pp. 211– 231, 1998.

[7] B. Schölkopf and A. Smola, *Learning With Kernels*. Cambridge, MA: MIT Press, 2002.

[8] P. H. Chen, R. E. Fan, and C. J. Lin, "A study on smo-type decomposition methods for support vector machines," National Taiwan University, Tech. Rep., 2005.

[9] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001, software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[10] A. S. Weigend and N. A. Gershenfeld (Eds.), *Time Series Prediction: Forecasting the Future and Understanding the Past*. Addison-Wesley, 1994, santa Fe Institute Studies in the Sciences of Complexity.



Elkin Eduardo García Obtuvo su Magíster en Ingeniería Electrónica y de Computadores de la Universidad de los Andes en 2006. Se graduó como Ingeniero Electrónico de la Pontificia Universidad Javeriana en 2003. Sus áreas de interés son las técnicas de aprendizaje supervisado y no supervisado y el diseño electrónico análogo y digital. Actualmente es profesor del Departamento de Ingeniería Eléctrica y Electrónica de la Universidad de los Andes.

Pedro Andrés Rangel Se graduó como ingeniero electrónico de la Universidad Distrital en 2003. Durante el pregrado perteneció al grupo de investigación en lógica programable y técnicas digitales (GILP) de dicha universidad. Recibió su titulo de maestría por parte de la Universidad de los Andes en 2005, especializándose en el área de señales. Actualmente se desempeña como profesor de análisis de señales en las Universidades de los Andes, Javeriana y Distrital.

Fernando Lozano Obtuvo su Ph. D en Electrical Engineering en la Universidad de New Mexico en 2000, Magíster en Ingeniería Eléctrica en la Universidad de los Andes en 1994, y pregrado en Ingeniería Electrónica en la Pontificia Universidad Javeriana en 1990. Su área de interés es Machine Learning Computacional y Estadístico. Actualmente es profesor del Departamento de Ingeniería Eléctrica y Electrónica de la Universidad de los Andes.