PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA INFORMÁTICA

“MÁQUINAS DE SOPORTE VECTORIAL CON ALGORITMOS BASADOS EN POBLACIONES PARA EL PRONÓSTICO DEL PRECIO DE ACCIONES LAN CHILE”

EDUARDO ALBERTO CUEVAS ALFARO

INFORME FINAL DEL PROYECTO
PARA OPTAR AL TÍTULO PROFESIONAL DE
INGENIERO CIVIL EN INFORMÁTICA

JUNIO, 2010.

Dedicatoria

Dedicada a mis padres por todo el amor y sacrificio, a mis hermanos por sus consejos e incondicional apoyo a lo largo de este proceso.

Agradecimientos

A Dios, mis padres,y profesor guía.

RESUMEN

En este proyecto se presenta el estado del arte de las técnicas utilizadas y los modelos desarrollados para el pronóstico del precio de las acciones de LAN Chile. Estos modelos consisten en la optimización de los parámetros de las máquinas de soporte vectorial mediante algoritmos genéticos (AG) y optimización por enjambre de partículas (PSO). El mejor resultado se obtuvo con el modelo basado en PSO, que presenta un 94% de la varianza explicada.

Palabras-claves: Máquinas de soporte vectorial, Máquinas de soporte vectorial de mínimos cuadrados, kernel, algoritmos genéticos , optimización por enjambre de partículas.

ABSTRACT

On this project is presented the state of the art of the used techniques, and the developed models to forecast the LAN Chile's stock prices . These models consist in the optimization of the parameters of support vector machines through genetic algorithms (GA) and particle swarm optimization (PSO).The best performance was obtained with the PSO-based model, which has a 94% of the variance explained

Keywords: Support vector machines, least squares support vector machines, kernel, Hybrid Algorithms, Particle Swarm Optimization.

ÍNDICE DE CONTENIDOS.

LISTA DE ABREVIATURA O SIGLAS.

SVM : Support Vector Machine.
SVMC : Support Vector Machine for Classification.
SVMR : Support Vector Machine for Regression.
LS-SVM : Least Squares Support vector Regression
AG : Algoritmo Genético.
PSO : Particle Swarm Optimization.

ÍNDICE DE FIGURAS.

ÍNDICE DE TABLAS.

1. INTRODUCCIÓN.

1.1. Introducción.

Las acciones corresponden a títulos-valores que representan una parte proporcional del capital social de una empresa y como tales, otorgan a sus titulares la calidad de socios o propietarios de la empresa. Como inversión, supone una inversión en renta variable, dado que no tiene un retorno fijo establecido por contrato, sino que depende de la buena marcha de dicha empresa. Cuando una empresa cotiza en La Bolsa, sus respectivas acciones pueden negociarse en el mercado bursátil; los vendedores y compradores determinan el precio de las acciones.

Al multiplicar el precio de la acción en el mercado por el número de acciones existentes se obtiene como resultado la capitalización o el valor bursátil de una empresa. Este criterio es muy utilizado para establecer el valor real de una empresa [Korn y Korn, 2005] La determinación del precio de las acciones de las empresas supone, en definitiva, la valoración que hace el mercado sobre las expectativas de las empresas que cotizan. En este sentido, la Bolsa puede considerarse como un "barómetro" de la economía.

Las incógnitas más díficiles de responder en relación a las inversiones en acciones son ¿cuándo vender? y ¿cuándo comprar?. Un buen enfoque para la venta de acciones es cuando a la empresa le va bien, pero se considera que el dinero podría estar mejor resguardado en otra inversión o podría generar más dinero en otro lado, o bien, a la empresa le va mal y esto es a causa de una mala administración, de problemas propios de la industria o de cambios en las variables que hicieron comprar la acción inicialmente [korn y korn, 2005]. La decisión de compra es un poco más sencilla. El inversionista compra una acción cuando confía en que esa empresa tiene perspectivas de crecimiento, ventajas sostenibles, buena administración, poca deuda o cuando genera efectivo.

Conocer la tendencia futura del precio de las acciones resultaría muy favorable para un inversionista, pues éste tendría una gran ventaja sobre los demas interesados, ya que al poseer esta información, podría definir previamente si comprar o vender la acción en custión. Desgraciadamente el proceso es complejo debido a númerosas variables que determinan el precio final de una acción, tales como, la tasa de desempleo, el IPC, la situación económica general, decisiones políticas, el comportamiento del consumo, entre otras.

En los últimos años el problema del pronóstico del precio de acciones ha sido foco de investigación en todo el mundo, sin embargo, los estudios han sido escasos a nivel local, por lo que se espera que este proyecto de título sirva como base para futuras investigaciones. A nivel mundial, el problema en cuestión se ha tratado con diversas técnicas entre las que se encuentran las redes neuronales y las máquinas de soporte vectorial, las cuales han resultados ser herramientas muy efectivas para complementar las desiciones financieras.

Las Máquinas de Vector Soporte (SVM) es una área prometedora en máquinas de aprendizaje, desarrollada inicialmente por Vapnik para construir clasificadores [Vapnik, 1995], actualmente son utilizadas también para realizar regresiones mediante la introducción de una nueva variable a la técnica [Smola, 1996]. Está modificación se conoce como SVMR

(por sus siglas en inglés *Support Vector Machine for Regression*) y son una poderosa técnica para el análisis predictivo de datos.

Debido a que la investigación en esta área a nivel nacional es escaso, el principal objetivo de esta investigación es la estimación de los otros parámetros de la SVMR utilizando **Algoritmo genéticos (AG)** y **Optimización por enjambre de partículas (PSO)** para la predicción del precio de las acciones de LAN Chile. Para esto se combinarán las cualidades de exploración y explotación de ambos algoritmos, de manera tal, que sea posible abarcar todo el espacio de búsqueda y una vez encontrada las mejores regiones, encontrar la mejor solución al problema.

1.2.1 Objetivo General.

Desarrollar y evaluar un modelo de pronóstico para el precio de las acciones LAN Chile, utilizando máquinas de soporte vectorial combinada con algoritmos genéticos y optimización por enjambre de partículas.

1.2.2 Objetivos Específicos.

- Comprender el estado de arte de las Máquinas de Soporte Vectorial (SVM), Algoritmos Genéticos (AG) y Optimización por enjambre de partículas (PSO).
- Diseñar la estructura y estimar los parámetros del modelo de pronóstico mediante el uso de AG y PSO.
- Implementar y evaluar el rendimiento de los modelos propuestos.

1.2. Organización del Texto.

Este documento está formado por cinco capítulos y un anexo, distribuido como se muestra a continuación.

En el capítulo 2, se describe el estado del arte de las máquinas de soporte vectorial (SVM), mostrando tanto las SVM para clasificación como las SVM para regresión, además se realiza una descripción de las máquinas de soporte vectorial de mínimos cuadrados para regresión (LS-SVM) que se utilizan en los modelos propuestos.

Luego, en el capítulo 3, se abordan los algoritmos genéticos, comenzando con los lineamientos generales y una breve reseña histórica de la técnica, para continuar con la definición de sus elementos claves, tipo de codificación, métodos de selección, algoritmo narrativo, finalizando con algunas aplicaciones.

El capitulo 4 comprende el estudio de PSO, partiendo con los fundamentos de éste, se muestra el algoritmo tradicional, los tipos de actualizaciones, y se finaliza con algunas aplicaciones.

Posteriormente, en el capítulo 5, se describen los modelos implementados utilizando LS-SVM, AG y PSO para la búsqueda de los meta-parámetros de la máquina de soporte vectorial. En la primera sección se muestra el proceso y características del entrenamiento del modelo LS-SVM AG, además de lo resultados de el testing realizado.

En la segunda sección del capítulo 5, se presenta la implementación de LS-SVM PSO, se describe el proceso, características de entrenamiento y resultados de las pruebas realizadas al modelo.

Luego, en la tercera sección se muestra el modelo LS-SVM AG+PSO, que corresponde a un híbrido de las dos técnicas utilizadas anteriormente, se presentan el proceso y las características utilizadas en el entrenamiento, finalizando con el resultados del testing realizado.

Finalmente, en el capítulo 6 se presentan las conclusiones finales de la investigación, posterior a este capitulo se encuentran los anexos, donde el anexo A contiene el código fuente de los distintos algoritmos utilizados en el desarrollo de los modelos.

2. MÁQUINAS DE SOPORTE VECTORIAL.

2.1. Introducción.

Las **máquinas de soporte vectorial**, o SVM (por sus siglas en inglés Support Vector Machine), corresponden a un nuevo sistema de aprendizaje, el que en los últimos años, se ha convertido en el centro de atención, tanto en el desarrollo de nuevos algoritmos, como en las estrategias para su implementación. Las SVM fueron desarrolladas por Vapnik en el año 1995, para enfrentar los problemas de clasificación, sin embargo el actual método de **Máquinas se soporte vectorial para regresión** (SVMR por sus siglas en inglés) fue desarrollada en los laboratorios de AT&T por este mismo [Vapnik, 1995].

Las SVM están ganando gran popularidad como herramienta para la identificación de sistemas no lineales, esto es a causa, principalmente, a que las SVM están basadas en el principio de minimización del riesgo estructural (SRM por sus siglas en inglés "Structural Risk Minimization"), principio originado de la teoría de aprendizaje estadístico desarrollada por Vapnik.

Esta teoría permite escoger un clasificador que minimiza una cota superior sobre el riesgo (o error de prueba), y proporciona una buena medida para obtener clasificadores que generalizan bien sobre datos no previamente vistos. Esto le da una ventaja a las SVM sobre otros clasificadores, como redes neuronales artificiales que pueden producir modelos que sobreajustan los datos.

Por otro lado, las máquinas de soporte vectorial tienen la habilidad de construir regiones de decisión no lineales de una manera en que discrimina mediante la introducción de una función Kernel. Este principio ha demostrado ser superior al principio de minimización del riesgo empírico (ERM por sus siglas en inglés), el que se utiliza en las RNA tradicionales. Algunas de las razones por las que este método ha tenido éxito es que no padece de mínimos locales, dependiendo solo de los datos con mas información llamados vectores de soporte.

En algunas aplicaciones las SVM muestran tener un alto desempeño, mas que los métodos de aprendizaje tradicional como es el caso de las redes neuronales [Burges,1998], y han sido introducidas como herramientas poderosas para la resolución de problemas de clasificación y regresión.

En el presente capítulo, en una primera parte se presenta las **máquinas de soporte vectorial para la clasificación**, dividiendo la sección en **casos linealmente separables** y **casos no linealmente separables**. Posteriormente se muestran las **máquinas de soporte vectorial para regresión**, las que se utilizan para el análisis predictivo de datos, mostrando una formulación particular de SVM llamada **Máquina de soporte vectorial de mínimos cuadrados para regresión**. Finalmente se presenta un listado de **aplicaciones** donde se utiliza este método.

2.2. Máquinas de Soporte Vectorial para Clasificación.

Al hablar de clasificación necesariamente se involucra el concepto de conjunto, es decir, que en general este proceso consiste en hacer una separación de los elementos de un conjunto en diferentes subconjuntos, a los cuales se les denominará **clases**. Se determinan las

propiedades de las clases en las que se clasificarán los elementos del conjunto original (modelos), cada uno de los elementos de éste son comparados y clasificados con cada uno de los modelos, para determinar a cual de ellos pertenece.

Matemáticamente, el proceso anteriormente comentado, puede representarse como una función de mapeo entre el conjunto original y el grupo de clases, se parte de la hipótesis de que sin importar la naturaleza del conjunto original, cualquiera de sus elementos pueden ser representados de forma numérica.

Como se mencionó en el párrafo anterior, la SVM mapea los elementos del conjunto original o de entrada, a un espacio de características de dimensión mayor, es decir, si los puntos de entrada se encuentran en R^2, son mapeados por la SVM a R^3, posteriormente busca un **hyperplano** que los separe y que maximice el margen *m* entre las clases del espacio. Maximizar dicho margen es un problema de **programación cuadrática**, pudiendo ser resuelto introduciendo **multiplicadores de Lagrange.**

Sin ningún tipo de conocimiento sobre el mapeo, la Máquina de soporte vectorial busca el hyperplano óptimo utilizando el producto punto con funciones en el espacio de características, las cuales se denominan **kernels**. Es posible escribir la solución del hyperplano óptimo como la combinación de unos pocos puntos de entrada que son llamados vectores de soporte. A continuación se revisará la teoría básica de las SVM en problemas de clasificación.

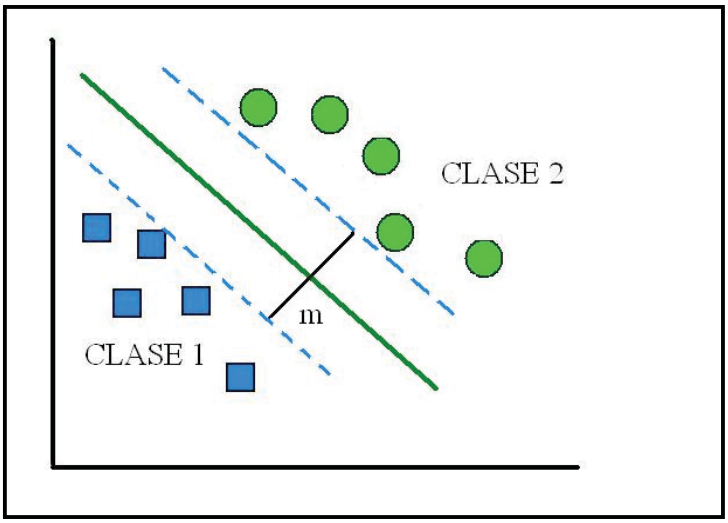

Figura 2- 1. margen *m*.

2.2.1. Caso linealmente separable.

Dado un conjunto C de puntos para entrenamiento, etiquetados en dos clases denominas C_1 y $C_{2,}$, como se puede apreciar en la figura 2-2. cada elemento del conjunto C se representa:

$$(y_1,x_1),(y_2,x_2),.....,(y_i,x_i)$$

Los puntos de entrenamiento $x_i \in R^N$, pertenecen a alguna de las dos clases y se les ha etiquetado $y_i \in \{-1,1\}$, dado i = 1,2,….,l. Normalmente la búsqueda del hyperplano optimo en una entrada es demasiado restrictivo para ser de usa practico.

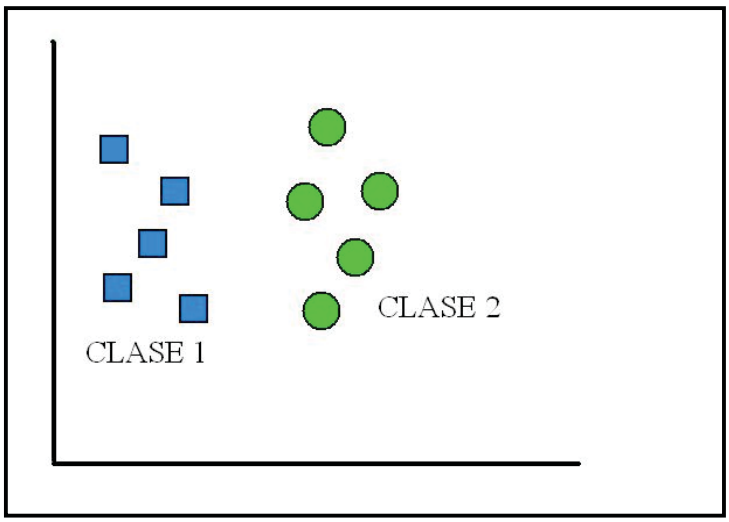

Figura 2- 2. Caso separable linealmente.

Para solucionar esta situación, se mapea el espacio de entrada en un espacio de características de una dimensión mayor, y buscar el hyperplano adecuado en este lugar. Sea Z = φ(x) la notación que corresponde al vector en el espacio de características con un mapeo φ de R^N a un espacio de características Z. Lo que se busca encontrar en el hyperplano denotado por:

$$w \bullet z + b = 0 \qquad (2.2.1.1)$$

este hyperplano está definido por el par *(w,b)*, de manera que sea posible separar el punto *x* de acuerdo a la función:

$$f(x_i) = sign(w \bullet z_i + b) = \begin{cases} 1, & y_i = 1 \\ -1, & y_i = -1 \end{cases} \qquad (2.2.1.2)$$

donde w ∈ Z y b ∈ R. Mas específicamente, el conjunto C es linealmente separable si existe *(w,b)* de tal forma que las inecuaciones:

$$\begin{cases} (w \bullet z_i + b) \geq 1, & y_i = 1 \\ (w \bullet z_i + b) \leq -1, & y_i = -1 \end{cases} \qquad (2.2.1.3)$$

Sean validas para todos los elementos del conjunto. Para el caso linealmente separable de C es posible determinar que la existencia de un hyperplano tal, determinado por un vector w, no es única, de echo existen una infinidad de tales vectores, pero existe solo un hyperplano óptimo, para el cual, el margen entre las proyecciones de los puntos de entrenamiento de dos diferentes clases es maximizado.

2.2.2. Caso no linealmente separable.

Dado un conjunto C con datos que no son linealmente separables, se deben permitir violaciones a la clasificaron en la formulación de la SVM.

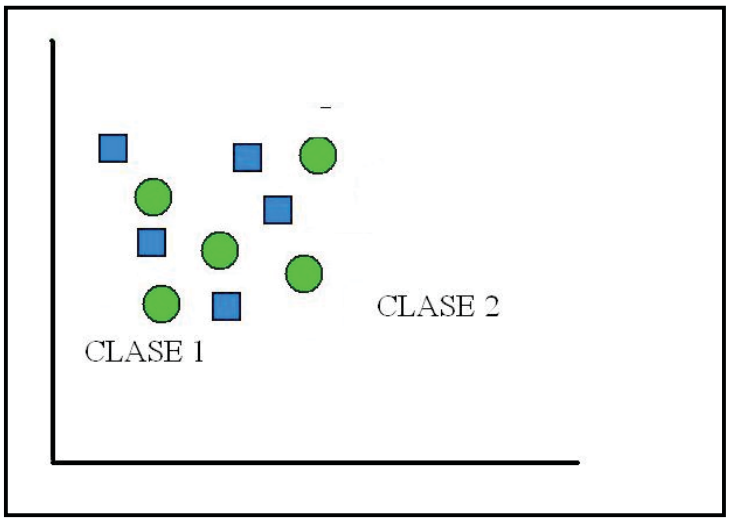

Figura 2- 3. Caso no linealmente separable.

Es posible generalizar el análisis previo introduciendo variables no negativas $\xi_i \geq 0$, de modo que la ecuación (2.2.1.3) sea modificada a:

$$y_i\left(w \bullet z_i + b\right) \geq 1 - \xi_{i,} \qquad \textbf{i=1,2,…,l} \qquad (2.2.2.1)$$

los $\xi_i \neq 0$ en la ecuación anterior, son aquellos para los cuales xi no satisface la ecuación (2.2.2.1). Por lo tanto el termino $\sum_{i=1}^{l} \xi_i$ puede ser considerado como algún tipo de error en la clasificación. Luego el problema del hyperplano optimo se vuelve a definir como la solución del siguiente problema:

$$\min\left\{\frac{1}{2} w \bullet w + C\sum_{i=1}^{l} \xi_i\right\} \qquad (2.2.2.2)$$

$$\textit{Sujeto} \quad a \quad \begin{array}{l} y_i\left(w \bullet z_i + b\right) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{array}$$

donde C es una constante que puede ser definida como un parámetro de regularización, éste es el único parámetro libre de ser ajustado en la formulación de la máquina de soporte vectorial. Dicho ajuste puede ser un balance entre la maximización del margen y la violación a la clasificación. Buscando el hyperplano optimo en (2.2.2.2) es un problema de programación cuadrática, el cual puede ser resuelto construyendo un lagrangiano y trasformándolo en el dual:

$$\text{Max } W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j z_i \bullet z_j \qquad (2.2.2.3)$$

$$\text{Sujeto a} \qquad \sum_{i=1}^{l} y_i \alpha_i = 0, 0 \le \alpha_i \le C, \; i = 1,.....,l \qquad (2.2.2.4)$$

Donde $\alpha = (\alpha_1,\ldots, \alpha_l)$ corresponde a un vector de multiplicadores de Lagrange positivos y que se encuentran asociados con las constantes en (2.2.2.1). El teorema de Khun-Tucker cumple un rol importante en la teoría de las SVM, pues de acuerdo a este teorema la solución α_i del problema en (2.2.2.3) satisface las siguientes condiciones:

$$\begin{aligned} &\overline{\alpha}\left(y_i\left(\overline{w} \bullet z_i + \overline{b}\right) - 1 + \overline{\xi_i}\right) = 0, && i = 1,....,l \\ &\left(c - \overline{\alpha_i}\right) \cdot \xi_i = 0, && i = 1,....,l \end{aligned} \qquad (2.2.2.5)$$

de esta igualdad es posible deducir que los únicos valores $\overline{\alpha_i} \neq 0$ en (2.2.2.5) son aquellos que para las constantes en (2.2.2.1) son satisfechos con el signo de igualdad. Se denomina **vector de soporte,** al punto x_i correspondiente con $\overline{\alpha_i} > 0$, existen dos tipos de vectores de soporte en un caso no separable, por un lado en el caso en que $0 < \overline{\alpha_i} < C$, el vector de soporte correspondiente x_i, debe satisfacer las igualdades $y_i\left(\overline{w} \bullet z_i + \overline{b}\right) = 1$ y $\xi_i = 0$, mientras que en el caso $\overline{\alpha_i} = C$, el vector de soporte x_i correspondiente no satisface (2.2.1.3), y $\xi_i \neq 0$. Se refiere a estos vectores de soporte como errores, el punto xi correspondiente con $\overline{\alpha_i} = 0$ es clasificado correctamente y esta alejado del margen de decisión, tal como se muestra en la figura 2-4.

Para construir el hyperplano óptimo $\overline{w} \bullet z + \overline{b}$, se utiliza la siguiente ecuación:

$$\overline{w} = \sum_{i=1}^{l} \overline{\alpha_i} y_i z_i \qquad (2.2.2.6)$$

y el escalar b puede determinarse de las condiciones del teorema de Kuhn-Tucker, la función generalizada de decisión de (2.2.1.2) y (2.2.2.6) es tal que:

$$f(x) = sign(w \bullet z \bullet b) = sign\left(\sum_{i=1}^{l} \alpha_i y_i z_i \bullet z + b\right) \qquad (2.2.2.7)$$

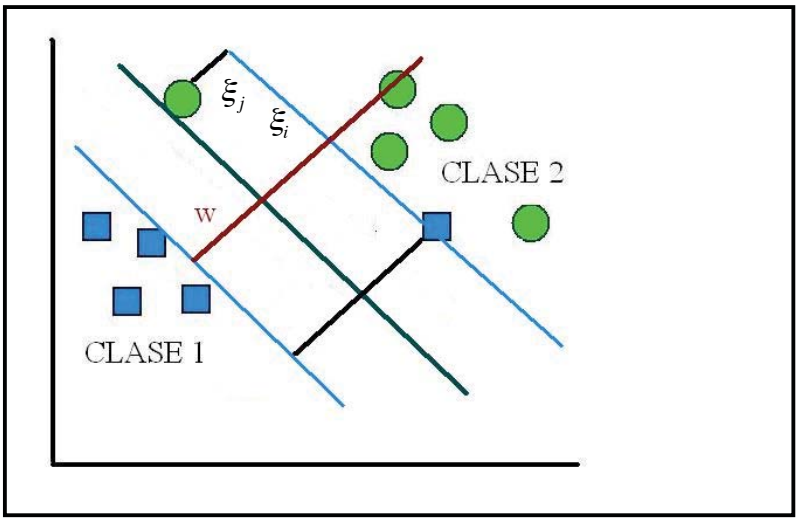

Figura 2- 4. Muestra parámetros $\xi_{i,j}$

2.3. Máquinas de soporte vectorial para regresión (SVMR).

Las máquinas de soporte vectorial también pueden ser utilizadas en problemas de regresión mediante la introducción de una función de pérdida alternativa que se modifica para incluir una medida de distancia, esta función de pérdida se denomina ε-insensitive [Vapnik, 1995] [Smola, 1996]. Las SVMR (por sus siglas en inglés *Support Vector Machine for Regression*) son una poderosa técnica para el análisis predictivo de datos, donde se tiene el problema de aproximar un conjunto de entrenamiento.

Sea el problema de aproximación de un conjunto de datos $\{(x_1,y_1),.....,(x_m\ y_m)\}, x \in R^n, y \in R$, con una función lineal dada por $f(x) = \langle w,x \rangle + b$, donde $\langle \bullet,\bullet \rangle$ denota el producto interno.

Con la función de pérdida mas general, posee ξ-zonas insensibles descritas como:

$$|y_i - f(x_i,\alpha)|_\xi = \begin{cases} \xi & si \quad y_i - f(x_i,\alpha) \le \xi \\ |y_i - f(x_i,\alpha)| & de \quad otra \quad forma \end{cases} \qquad (2.3.1)$$

El objetivo ahora es encontrar una función $f(x,\alpha)$, que maximice la desviación de ξ con respecto a los objetivos y_i para todos los datos de entrenamiento, al mismo tiempo sea lo mas plana posible, que la función sea lo mas plana posible, se refiere a que se quiere buscar el menor parámetro posible para *w*. Una forma de asegurar esto es a través de la minimización de la norma $\|w\|^2 = \langle w,w \rangle$, dicho problema puede ser reescrito como un problema de optimización convexa:

$$\begin{aligned} &\min imizar \quad \frac{1}{2}\|w\|^2 \\ &sujeto \quad a \quad \begin{cases} y_i - \langle w, x_i \rangle - b \le \varepsilon \\ \langle w, x_i \rangle + b - y_i \le \varepsilon \end{cases} \end{aligned} \tag{2.3.2}$$

para esto se ha supuesto que la función f existe, pero en ocasiones, esto no ocurre. Para esto se introducen variables de flexibilidad denominadas $\xi_i \quad y \quad \xi_i^*$, llegando a la formula:

$$\begin{aligned} &Minimizar \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\left(\xi_i + \xi_i^*\right) \\ &Sujeto \quad a \quad \begin{cases} y_i - \langle w, x_i \rangle - b \le \varepsilon + \xi_i \\ \langle w, x_i \rangle + b - y_i \le \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \ge 0 \end{cases} \end{aligned} \tag{2.3.4}$$

donde el valor C determina la compensación entre que tan plana es f y el porcentaje de error sobre el cual las desviaciones mayores a ξ son permitidas. La función y las restricciones de (2.3.4), pueden ser escritas como la siguiente función de Lagrange:

$$\begin{aligned} L = &-\frac{1}{2}|w|^2 + C\sum_{i=1}^{N}\left(\xi_i + \xi_i^*\right) - \sum_{i=1}^{N}\alpha_i\left(\varepsilon + \xi_i - y_i + \langle w \bullet x_i \rangle + b\right) \\ &-\sum_{i=1}^{N}\alpha_i^*\left(\varepsilon + \xi_i^* - y_i + \langle w \bullet x_i \rangle - b\right) - \sum_{i=1}^{N}\left(\eta_i\xi_i + \eta_i^*\xi_i^*\right) \end{aligned} \tag{2.3.5}$$

con la teoría de Lagrange se obtienen las condiciones para α:

$$\begin{aligned} \partial_b L &= \sum_{i=1}^{N}\left(\alpha_i^* - \alpha_i\right) = 0 \\ \partial_w L &= w - \sum_{i=1}^{N}\left(\alpha_i^* - \alpha_i\right)x_i = 0 \\ \partial_{\xi_i} L &= C - \alpha_i - \eta_i = 0 \\ \partial_{\xi_i^*} L &= C - \alpha_i^* - \eta_i^* = 0 \end{aligned} \tag{2.3.6}$$

Sustituyendo las ecuación (2.3.6) en (2.3.5) se obtiene el problema en forma dual

$$\textit{Maximizar} \quad -\frac{1}{2}\sum_{i,j=1}^{N}\left(\alpha_i^* - \alpha_i\right)\left(\alpha_j^* - \alpha_j\right)\left\langle x_i, x_j\right\rangle + \sum_{i=1}^{N} y_i\left(\alpha_i^* - \alpha_i\right) - \varepsilon\sum_{i=1}^{N}\left(\alpha_i^* - \alpha_i\right)$$

$$\textit{Sujeto} \quad a \quad \sum_{i=1}^{N}\alpha_i^* = \sum_{i=1}^{N}\alpha_i, \quad 0 \le \alpha_i^* \le C \quad y \quad 0 \le \alpha_i \le C \tag{2.3.7}$$

debido a que todas las sumatorias van desde 1 hasta N, la solución del problema puede ser reescrita como:

$$f(x) = \sum_{i=1}^{N}\left(\alpha_i^* - \alpha_i\right)\left\langle x_i - x\right\rangle + b \tag{2.3.8}$$

normalmente $\alpha_i^*, \alpha_i \neq 0$, donde sus vectores correspondientes se denominan de soporte, es un valor pequeño, y para representarlos se requiere de escasos miembros, es más, la formulación se generaliza al caso no lineal, que normalmente para realizar un modelado adecuado de los datos, reemplazando el producto interior por una función Kernel, de esta forma el problema a solucionar queda definida como:

$$\begin{aligned} &\textit{Maximizar} \quad -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\alpha_i^* - \alpha_i\right)\left(\alpha_j^* - \alpha_j\right)k\left(x_i, x_j\right) + \sum_{i=1}^{N} y_i\left(\alpha_i^* - \alpha_i\right) - \varepsilon\sum_{i=1}^{N}\left(\alpha_i^* - \alpha_i\right) \\ &\textit{Sujeto} \quad a \quad \sum_{i=1}^{N}\alpha_i^* = \sum_{i=1}^{N}\alpha_i, \quad 0 \le \alpha_i^* \le C \quad y \quad 0 \le \alpha_i \le C \end{aligned} \tag{2.3.9}$$

Por lo que la solución está dada por:

$$f(x) = \sum_{i=1}^{N}\left(\alpha_i^* - \alpha_i\right)k\left(x_i, x\right) + b \tag{2.3.10}$$

Los vectores de soporte son aquellos puntos x_i en los cuales el error de interpolación es $\geq \xi$. Los puntos en los que el error de interpolación es $< \xi$ nunca son vectores de soporte, y no forman parte de la solución. Una vez que se han encontrado, pueden ser eliminados del conjunto de datos, y si se resuelve el problema de programación nuevamente sobre el conjunto reducido se encuentra la misma solución.

Maximizar (2.3.9), corresponde a un problema que tradicionalmente es resuelto a través de **programación cuadrática**, introduciendo **multiplicadores de Lagrange.**

2.4. Máquinas de soporte vectorial de mínimos cuadrados para regresión (LS-SVM).

Como se mencionó anteriormente las máquinas de soporte vectorial se resuelven un problema de programación cuadrática , obteniendo una solución tras dicha optimización. Sin embargo existe la posibilidad de simplificar aspectos de la formulación de las SVMR, sin perder sus ventajas, para esto se utilizan las Máquinas de soporte vectorial de mínimos cuadrados (LS-SVM).

Las LS-SVM son una reformulación de las SVMs de Vapnik, en las que la optimización lleva a resolver un sistema de ecuaciones lineales, esto es mas simple de utilizar que las soluciones de programación cuadrática anteriormente descrita. En este documento sólo se abordará el caso de LS-SVM utilizado para regresión.

La formulación Vapnik descrita en el punto anterior se modifica en dos puntos; en primer lugar, en vez de inecuaciones, la formulación de los LS-SVM utilizan ecuaciones de igualdad, donde el valor de la derecha de la ecuación se toma como un valor objetivo, mas que un valor umbral. Sobre este valor, se permite un valor de estimación variable, dicha variable juega un papel similar a las variables de holgura en SVMR, es por esto que ésta se denomina ξ.

En segundo lugar, la función de pérdida de Vapnik, se sustituye por una función de pérdida cuadrática. Estas dos modificaciones simplifican enormemente el problema. Considere la siguiente ecuación:

$$f(x) = w \bullet k(x) + b \tag{2.4.1}$$

donde $x \in R^n$, $y \in R$, $f(x) : R^n \rightarrow R^m$ es un mapeo hacia un espacio de características mayor. Además, dado un conjunto de entrenamiento $\{(x_1, y_1), ..., (x_m, y_m)\}$, podemos expresar el problema de Optimization como:

$$\begin{aligned} &Minimizar \quad \frac{1}{2}\|w\|^2 + \frac{C}{2}\sum_{i=1}^{m}(\xi_i^2) \\ &tal \quad que \quad y_i = \langle w \bullet k(x) \rangle + b + \xi_i \end{aligned} \tag{2.4.2}$$

esta formulación es equivalente a la función de coste de una regresión ridge, formulada en el espacio de características. Sin embargo este problema puede no ser resoluble en el espacio primario, es por esto que se pasa a su forma dual mediante el teorema de Lagrange,

$$L(w, b, \alpha, \xi) = \frac{1}{2}\|w\|^2 + \frac{C}{2}\sum_{i=1}^{N}\xi_i^2 - \sum_{i=1}^{N}\alpha_i \langle w \bullet k(x_i) \rangle + b + \xi_i - y_i \tag{2.4.3}$$

donde α son los multiplicadores de Lagrange. Calculando las derivadas parciales del Lagrangiano, con respecto a todas las variables e igualando a cero podemos obtener las condiciones de optimización,

$$\begin{aligned} \partial_b L &= \sum_{i=1}^{N}(\alpha_i) = 0 \\ \partial_w L &= w - \sum_{i=1}^{N}(\alpha_i)x_i = 0 \\ \partial_{\xi_i} L &= C\xi_i - \alpha_i = 0 \\ \partial_\alpha L &= \langle w \bullet k(x_i)\rangle + b + \xi_i - y_i \end{aligned} \tag{2.4.4}$$

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

$$y_i = \sum_{i=1}^{N}\alpha_i \bullet k(x_i) \bullet k(x_i) + b + \frac{\alpha_i}{C}$$

$$\sum_{i=1}^{N}\alpha_i = 0 \tag{2.4.5}$$

$$\begin{bmatrix} 0 & \bar{1}^T \\ \bar{1} & K + \frac{1}{C}I \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ y \end{bmatrix}$$

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

$$\overline{f}(x) = k(x,x_i)\alpha + b \tag{2.4.6}$$

Un punto importante a señalar es que este método no requiere de la determinación del parámetro de precisión ε, el cual está relacionado con la función de costo ε-insensitive del método propuesto por Vapnik. La gran desventaja de este método es que la solución del modelo no es "sparse", esto quiere decir que ninguna del las soluciones α_i se desvanece en la solución optima, por ende el modelo depende de todos los datos de entrenamiento.

Las máquinas de soporte vectorial para regresión han sido utilizadas para múltiples problemas de distintas áreas tales como, la estimación de parámetros petrofísicos, diseño de antenas de agrupación, predicción de hipertensión arterial, entre muchos otros.

2.5. Funciones Kernel.

Como se mencionó en la sección anterior cuando la regresión lineal no es apropiada, como es el caso de la mayoría de las aplicaciones en ingeniería, un kernel de mapeo no lineal K, se utiliza para mapear los datos a un espacio de características de dimensión mayor en donde se pueda realizar la regresión lineal.

Algunos de los Kernel mas utilizados son:

- Polinomio: $k(x,y) = (\langle x,y \rangle + 1)^d, \quad d = 1,2,....$
- Funciones de base radial: $k(x,y) = \exp\{-|x-y|^2/\sigma^2\}$
- Redes neuronales de dos capas: $k(x,y) = \tanh(b\langle x,y \rangle - c)$, *Para cierto valores de b y c*

2.6. Aplicaciones.

Las Máquinas de soporte vectorial han sido estudiadas intensamente en los ultimo años y aplicadas exitosamente en gran variedad de temas como: estimación de densidades probabilísticas , y la predicción de series de tiempo. A continuación se listan algunas otras aplicaciones de las SVM:

- **Estimación de parámetros petrofísicos:** Se utilizaron máquinas de soporte vectorial en regresión para la estimación del volumen de arcilla mediante registro de rayos Gamma, dicho parámetro, junto con porosidad y saturación de agua son muy importantes en la caracterización de yacimientos de hidrocarburos [del Mar y Urdaneta, 2009].

- **Predicción de fuga de clientes para una institución financiera:** Se utilizan SVM para desarrollar un modelo predictivo para clasificar los clientes con tendencias a la fuga en un banco. De esta forma es posible hacer más efectivas las políticas comerciales de retención, ser más eficientes en la asignación de los recursos al focalizar a los ejecutivos y mejorar las relaciones con los clientes al detectar los principales focos de deficiencias del servicio [Miranda, Rey, Weber, 2005].

- **Sistema de verificación de hablantes:** Su objetivo principal es la implementación y evaluación del desempeño del sistema, basado en máquinas de soporte vectorial como sistema de clasificación [Pedroza, 2007].

3. ALGORITMOS GENÉTICOS.

3.1. Introducción.

Los algoritmos genéticos (AG) corresponden a métodos adaptativos que suelen utilizarse para resolver problemas de búsqueda y optimización. Estos pretenden imitar el comportamiento planteado por Darwin en 1859, basándose en el proceso genético de los seres vivos, en donde las poblaciones a lo largo de las generaciones evolucionan dentro de la naturaleza acorde con los principios de selección natural y la supervivencia de el más fuerte. De esta forma los AG son capaces de ir evolucionando hacia soluciones para problemas del mundo real.

Los AG intentan emular la evolución natural de los seres vivos, utiliza una población de individuos, cada uno de éstos representa una solución factible a un problema determinado, a cada individuo de dicha población se le asigna un valor que se encuentra relacionado con la bondad de dicha solución, de la misma forma en que un ser vivo compite por sobrevivir en la naturaleza.

Mientras mayor grado de adaptación tenga un individuo al problema, mayor será la probabilidad de que éste sea seleccionado para reproducirse, cruzado material genético con otro individuo de la población, y que fue seleccionado de igual forma, dicho cruce generará nuevos individuos (descendientes), los cuales compartirán algunas características de sus padres.

De esta manera se produce una nueva generación de posibles soluciones, que reemplazará a la anterior y tendrá un mayor porcentaje de buenas cualidades que su predecesora. Así, y a medida que aumentan las generaciones, continuará la búsqueda de la población que converja a una solución óptima del problema.

En el presente capítulo se parte presentando la **historia** de los Algoritmos genéticos, mostrando como a partir de la teoría de Darwin se llegó a lo que se conoce hoy como AG. Posteriormente se muestran los **elementos claves** del algoritmo en cuestión, para luego mostrar los tipos de **codificación** de cromosomas más utilizados, continuando con alguno de los **métodos de selección** mas nombrados en la literatura especializada. Luego se presenta el **algoritmo narrativo de un AG básico**, para así, finalizar nombrado las **ventajas y desventajas** de éste Algoritmo y mencionar un listado de **aplicaciones.**

3.2. Historia.

A fines de la década del 50 y principios de los años 60 surgieron los primeros indicios de lo que hoy se conoce como Algoritmos Genéticos, éstos fueron programados por biólogos evolutivos que buscaban encontrar modelos de aspectos de a evolución natural.

Box, Bledsoe, Friedman y Brememann, en el año 1962, habían desarrollado independientemente algoritmos inspirados en la evolución para optimizar funciones. Posteriormente, en el año 1965, se introdujo en la teoría una técnica llamada **estrategia evolutiva**, ésta fue postulada por Rechenberg, en la cual no existía ni población ni cruzamiento, solo se mutaba un padre para generar sólo un descendiente y se quedaba con la mejor de los dos [Haupt y Haupt, 1998].

John Holland establece las bases para desarrollos posteriores, con su trabajo sobre **sistemas adaptativos** en 1962. Fue el primero en proponer el cruzamiento y otros operadores de recombinación de manera explícita.

Años mas tarde, en 1966, se produjo el siguiente aporte importante, Fogel, Owens y Walsh introducen la **programación evolutiva**, ésta es muy similar a la estrategia evolutiva, salvo que las soluciones candidatas para los problemas se representan como máquinas de estado finitas sencillas.

Sin lugar a dudas, el acontecimiento fundamental para los algoritmos genéticos ocurre en 1975 con la publicación del libro basado las investigaciones y papers de Holland y otros investigadores de la Universidad de Michigan llamado "Adaptación de sistemas naturales y artificiales", éste fue el pionero en mencionar de manera rigurosa y sistémica el concepto de sistemas digitales adaptativos utilizando mutación, selección y cruzamiento, emulando el proceso natural de la evolución como estrategia para resolver problemas.

Durante la década posterior los AG se aplicaban en distintas áreas, pasando desde problemas matemáticos abstractos hasta problemas de ingeniería como el control de flujo de una línea de ensamble, optimización estructural y reconocimiento y clasificación de patrones [Goldberg, 1989].

En la actualidad los AG resuelven problemas de interés cotidiano en diversas áreas, como es el caso de la predicción en la bolsa, planificación de la cartera de valores, biología molecular, líneas de montaje. Todo esto en base a la idea de Charles Darwin, la cual se ha convertido en una poderosa técnica de resolución de problemas.

3.3. Elementos claves de un Algoritmo Genético.

Al no existir una rigurosa definición de Algoritmo Genético, generalmente se encasilla dentro del concepto aquellos algoritmos que al menos comparten las características que se mencionan a continuación.

Todo AG se basa en la existencia de una **población de cromosomas**, la cual corresponden a individuos que son posibles soluciones de un problema dado, normalmente se representan como un conjunto de parámetros llamados **genes,** éstos, agrupados ,conforman los denominados cromosomas. Generalmente para representar un cromosoma se utiliza codificación binaria (1,0), cada uno de los individuos obtenidos tendrá una determinada adaptación al problema, que dependerá exclusivamente de la evaluación del **fenotipo,** que corresponde a la información necesaria para construir el cromosoma.

Los Algoritmos Genéticos necesitan de una función **fitness**, la cual es distinta para cada problema y es diseñada de forma específica**,** dicha función asigna un valor a cada cromosoma de la población actual, este valor depende de la exactitud con que un cromosoma determinado resuelva el problema en cuestión. Cuando cada individuo es evaluado se selecciona el que tenga un mejor desempeño.

Continuando con el algoritmo, el siguiente paso corresponde a la etapa de reproducción, donde los individuos mejor evaluados serán seleccionados para cruzarse y producir una nueva

generación. La reproducción consistirá en combinar sus cromosomas utilizando los operadores que se describen a continuación:

Cruzamiento: Consiste en la mezcla de dos cromosomas, que asumen el rol de padres, a partir de una posición en base a un patrón o bien de manera aleatoria, de esta forma se crea una nueva generación de descendientes. Los nuevos individuos presentan soluciones potencialmente mejores que las de sus predecesores.

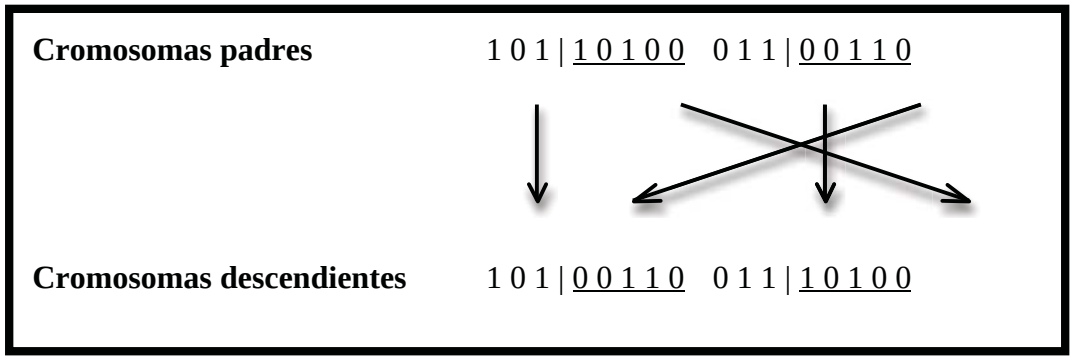

Figura 3- 1. Operador de cruzamiento.

Mutación: Consiste en el intercambio al azar de algún bit del cromosoma, con esto se entrega un pequeño grado de aleatoriedad a los individuos de la población, lo cual contribuye en disminuir la probabilidad de que queden espacios de búsqueda sin examinar.

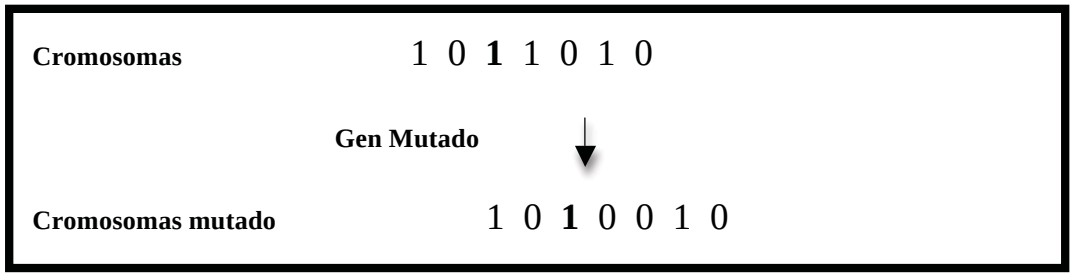

Figura 3- 2. Operador de mutación.

3.4. Codificación del cromosoma.

El principal factor en el éxito de un Algoritmo Genético corresponde a la decisión sobre el tipo de codificación del cromosoma que se utilizará para abordar el problema, a continuación se presentan algunas de las alternativas que se tienen al momento de codificar.

3.4.1. Codificación Binaria.

Es el tipo de codificación mas utilizada, a cada gen se le asignan valores entre 0 y 1, como se aprecia en la figura. De esta forma cada gen, en cada posición dentro del cromosoma, representa una característica del problema.

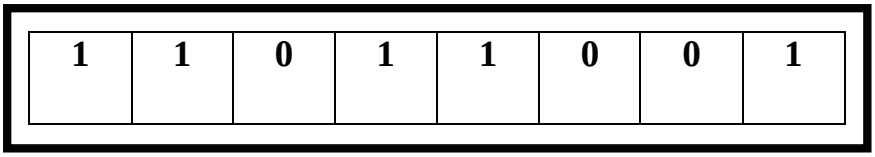

Figura 3- 3. Codificación binaria.

3.4.2. Codificación por valor.

En algunas aplicaciones, es mas natural utilizar caracteres del alfabeto, números reales y enteros para codificar el cromosoma. Si bien, Holland, argumento que éste tipo de codificación presenta peores resultados que la binaria, algunos estudios comparativos han demostrado que la codificación por valor ha presentado mejor rendimiento [Wright, 1991].

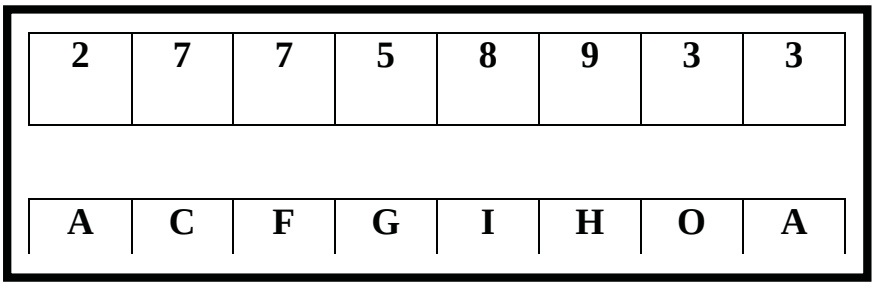

Figura 3- 4. Codificación por valor.

3.5. Métodos de selección.

Una vez que se ha definido el tipo de codificación del cromosoma, se debe decidir la manera en que se seleccionarán los individuos para que se reproduzca y formen una nueva generación, esperando que los descendientes presenten un mejor fitness o desempeño. Aunque la lógica es elegir a aquellos individuos con mejor desempeño, la practica a mostrado que no siempre es la mejor opción, pues al elegir los mejores la capacidad de exploración del algoritmo se ve disminuida, aumentando la probabilidad d quedar atrapado en un desempeño sub-optimo. A continuación se presentan algunos de los métodos mas utilizados.

3.5.1. Selección por torneo.

Es un método donde se eligen individuos al azar y en base a comparaciones directas entre los desempeños obtenidos de cado uno de los seleccionados se selecciona el mejor cromosoma para una nueva reproducción. Existen dos versiones de este tipo de selección:

Determinística: Se escogen al azar un número x de individuos, normalmente x=2, de los escogidos se selecciona al que presente un mejor desempeño o fitness.

Probabilística: En esta versión no se elige el con mejor desempeño, sin que se genera un número aleatorio dentro del intervalo [0,1], se fija un parámetro *p,* generalmente $0{,}5 < p \leq 1$. Si el número aleatorio es mayor que *p*, se escoge el individuo que presente mejor fitness, de lo contrario el que presente un menor desempeño.

Si el grupo seleccionado es pequeño la probabilidad de seleccionar un valor bajo respecto al total de la población aumenta, lo que permite una mayor variabilidad de la población y una mayor capacidad de exploración del algoritmo. Por otro lado, si la selección es grande, cercano al número total de individuos de la población, aumenta la probabilidad de seleccionar al mejor individuo de la población. Mejorando así, la capacidad de explotación del algoritmo .

3.5.2. Selección por ruleta.

Este método consiste en que el número de veces que un individuo será seleccionado para reproducirse, estará dado por:

$$p\left(x_j^i\right) = \frac{f\left(x_j^i\right)}{\sum_{m=1}^{n} f\left(x_m^i\right)} \qquad (3.5.2.1)$$

De esta forma se persigue asignarle un mayor probabilidad a los individuos con mejor desempeño, sin embargo, permite que en cada iteración pueda ser elegido un individuo con baja probabilidad, aumentando así, la diversidad de la población. El riesgo de este método radica en que al existir un grupo de individuos con alta probabilidad en comparación al resto, la población perdería su diversidad y convergería prematuramente.

3.5.3. Selección de Boltzman.

Como se comentó en el método anterior, durante la selección no sólo se debe seleccionar al individuo con mejor desempeño, el método de selección de Boltzman utiliza distintos grados de exigencia del fitness, el que aumenta a medida que se incrementa el número de iteraciones, es decir, en las primeras iteraciones, no se es tan riguroso con el desempeño de los individuos de la población que se seleccionan, con lo que se gana una mayor variabilidad y maximizando la exploración, pero a medida que aumentan las iteraciones, se puede incrementar la exigencia del fitness, con lo que se maximiza la capacidad de explotación del algoritmo.

3.5.4. Selección por elitismo.

Este método fuerza a los algoritmos genéticos a mantener algunos de los individuos de cada generación que presenten un mejor desempeño, esto se hace para no perder los lugares ya explorados del espacio de búsqueda que son buenos candidatos, pues estos individuos pueden perderse sino se reproducen o si son destruidos en cruce o mutación.

3.6. Algoritmo Narrativo de Algoritmo Genético básico.

Dado un problema, y considerando los elementos mencionados en la sección 3.3, un Algoritmo genético básico esta dado por las siguientes líneas:

Generar *una población de n individuos de manera aleatoria.*

Calcular *el desempeño (fitness), dado por f(x), para cada individuo (cromosoma) de la población.*

Hacer

> ***Seleccionar*** *dos individuos de la generación anterior para ser padres (la probabilidad de selección de estos individuos esta dada en función a f(x)).*
>
> ***Cruzar****, en base a la probabilidad de cruzamiento (p_c), el par seleccionado en un punto escogido de manera aleatoria para formar dos descendientes.*
>
> > *Si, no existe cruzamiento, generar dos copias exactas de los padres.*
>
> ***Mutar****, con probabilidad (p_m) los dos descendientes.*
>
> > ***Calcular*** *el fitness de los dos individuos mutados.*
>
> ***Insertar*** *los descendientes mutados en la nueva generación.*
>
> > *Si, n es impar, descartar un nuevo miembro de la población aleatoriamente.*

Mientras *población no converja*

Cada iteración se conoce como generación, comúnmente, un Algoritmo Genético, se itera entre 50 y 500 veces, incluso mas (lo que implica un alto costo computacional), normalmente existen uno o más cromosomas que se acoplan correctamente con el problema en cuestión.

El criterio de termino es dependiente del problema, generalmente los criterios de termino son: número de iteraciones dado, algún criterio de convergencia (diferencia de resultado converja a 0).

3.7. Ventajas y Desventajas de los Algoritmos genéticos.

3.7.1. Ventajas.

- Una de las ventajas mas sobresaliente de los AG corresponde a su característica de ser intrínsecamente paralelos, es decir, que a diferencia de los algoritmos en series, los que exploran el espacio de búsqueda de una única dirección, los Algoritmos Genéticos, al tener descendencia múltiple, exploran el espacio de búsqueda en varias direcciones, con esto, no debe comenzar de nuevo en caso de encontrarse con una solución sub-óptima.
- Son muy efectivos eludiendo los óptimos locales y descubriendo el óptimo global, esto se debe a que durante el cruzamiento ocurre una transferencia de información entre los candidatos con mejor desempeño, beneficiándose unos de otros, mezclándose y potenciándose, pues eventualmente, la descendencia puede tener todas las virtudes de los padres y ninguna de sus debilidades.
- Los AG no dependen de un conocimiento previo, parten con una población aleatoria, realizando cambios de la misma forma en los individuos, y usando el fitness para determinar si los cambios producen mejora. En contraste las técnicas que dependen del conocimientos previo fracasan cuando éste se encuentra ausente, además, parten descartando camino y perdiendo así cualquier solución novedosa que exista.

3.7.2. Desventajas.

- Una de las principales desventajas radica en la dificultad para definir la representación del problema. La forma para especificar las soluciones debe ser tolerante a cambios aleatorios sin producir errores fatales o resultados sin sentidos, como se mencionó en la sección 3.4. la manera generalmente utilizada es definir a los individuos como listas de números, ya sean binarios, reales o enteros.
- La dificultad para definir una función fitness, pues el proceso no es trivial, ya que debe permitir encontrar el mejor desempeño y que éste signifique una mejor solución para un problema determinado, una definición deficiente o inadecuada puede ser incapaz de resolver el problema o bien dar solución a un problema distinto.
- El cuidado que requiere en la selección de elementos claves como; tamaño de la población, ritmo de mutación, cruzamiento y el tipo de selección, pues si el tamaño de la población es muy pequeño, es posible que el AG no sea capaz de abarcar todo el espacio de posibles soluciones o si el ritmo de mutación, o el método de selección no es el adecuado, la población puede entrar en una catástrofe de errores.

3.8. Aplicaciones

Según la literatura cuando un problema posee un espacio de búsqueda amplio, y no se tiene mucha información de éste, probablemente los Algoritmos Genéticos tengan un buen desempeño.

- **Problemas de Predicción:** Se utilizaron Algoritmos Genéticos para predecir el futuro rendimiento de un total de 1600 acciones ofertadas públicamente [Mahfoud y Mani, 1996].
- **Problemas de Optimización**: Se utilizaron AG para definir la ubicación de las orbitas de los satélites de manera de minimizar los apagones a causa de la cobertura [Williams, Crossley y Lang, 2001].
- **Evolución de redes neuronales**: Los AG fueron utilizados para evolucionar Redes Neuronales Artificiales que jugaran damas, los resultados fueron exitoso, llegando a jugar con el mejor jugador de damas del mundo, y aunque perdió 2-4, la causa fue simplemente la carencia de una base de datos de finales de partida.

4. OPTIMIZACIÓN POR ENJAMBRE DE PARTÍCULAS.

4.1. Introducción.

En otro intento por emular y mimetizar el comportamiento de los procesos naturales, surge al igual que otras técnicas estocásticas del cálculo evolutivo , la **Optimización por Enjambre de Partículas**, conocida en la literatura como Particle Swarm Optimization (PSO).

Se ha convertido en uno de los métodos mas usados en la inteligencia computacional, éste busca imitar el comportamiento social de un colectivo a partir de la interacción entre sus individuos y a la vez, la de estos con su entorno.

Los primeros aportes en la materia fueron realizados en el año 1995 por Kennedy y Eberhart, ellos pretendían emular de manera gráfica el movimiento impredecible y sincronizado de bandadas de aves o cardúmenes de peces, fascinados con la capacidad de comunicación entre los individuos del grupo, que eran capaces de separarse, reagruparse y encontrar su alimento. [Kennedy y Eberhart, 1995]

Fueron estos mismos autores lo que utilizan por primera vez el término **partícula** para representar a cualquier individuo que interactúe con otro y muestre algún tipo de comportamiento social. En base a lo planteado teóricamente en el método, el movimiento de cada partícula depende de dos factores: la **memoria autobiográfica** de cada partícula y la **influencia** de todo el enjambre. En la práctica cada una de las partículas de la población representa una de las posibles soluciones y, dependiendo del problema y de la cantidad de variables que intervengan será el número de dimensiones del espacio, es decir, un espacio d-dimensional, consta con d incógnitas para el problema a solucionar.

El proceso evolutivo consiste básicamente en mover cada partícula dentro del espacio de soluciones, con una velocidad que dependerá de la **velocidad actual**, de la **memoria** y de la **información global** que comparte con resto de las partículas del enjambre, además, se utiliza una función **fitness** para evaluar la calidad de cada partícula en base a la posición que ésta tiene.

El presente capítulo parte mostrando los **fundamentos** de la optimización por enjambre de partículas, en donde se definen los cinco principios básicos de la inteligencia de grupo, posteriormente se explica detalladamente el **algoritmo tradicional de PSO**, mostrando también, el **algoritmo narrativo** de este. Luego se muestra la diferencia entre **actualización síncrona y asíncrona**, continuando con **el modelo de óptimo local y el modelo de óptimo global**. Mas adelante se listan algunas de las **variaciones del algoritmo**, se presenta a inserción de nuevas variables al método. Finalmente se muestra una lista con algunas **aplicaciones** donde se utiliza el algoritmo.

4.2. Fundamentos de PSO.

En la inteligencia de grupo se deben respetar cinco principios básicos. El primero es **proximidad**, este término se refiere a que la población debe tener la capacidad para realizar cálculos de espacio y tiempo, lo que en PSO se ve reflejado en los movimientos en d-dimensiones que se llevan a cabo durante un intervalo de tiempo y que coinciden con movimientos de la población a una determinada velocidad.

Para determinar los factores de **calidad** en PSO, se debe considerar la memoria de la partícula en conjunto con la historia o conocimiento social que comparte con las demás. Un tercer principio corresponde a la **diversidad de respuesta**, que es representada por las tendencias marcadas por la memoria personal de cada partícula y por la historia de la mejor posición visitada por el conjunto.

Finalmente están dos aspectos contrarios, por un lado se encuentra el **principio de estabilidad**, el cual propone que la población solo cambia su comportamiento como grupo cuando uno de los miembros que lo integran encuentran una posición mejor que la históricamente visitada, y está se actualiza con la nueva. Por otro lado la población debe ser capaz de modificar su comportamiento y movimiento cuando existe alguna señal que asi lo recomiende, en PSO esto se consigue cuando alguna partícula alcanza una solución global que mejora el resultado, en ese instante la población cambia de rumbo, a esto se le conoce como **principio de adaptabilidad**.

4.3. Algoritmo tradicional de PSO.

El algoritmo comienza con la inicialización de manera aleatoria de las partículas, esta a su vez, se considera como un punto en el espacio d-dimensional de búsqueda, donde cada dimensión corresponde a una incógnita del problema, PSO optimiza sus soluciones a medida que aumentan las iteraciones. El enjambre está representado por $X = (x_{i1}, x_{i2}, ..., x_{id})$ donde $i = 1,2,....,n$ corresponde a la i-ésima partícula y $d = 1,2,....,D$ a sus dimensiones.

Una vez que las partículas han sido inicializadas, se da inicio a las iteraciones, en éstas las partículas se actualizan en base a dos valores importantes, el primero, llamado **mejor personal** p_{best}, que corresponde a la mejor posición encontrada por la partícula, el pbest de una i-ésima partícula se representa por $P_i = (p_{i1}, p_{i2}, ..., p_{id})$. El segundo valor, llamado **mejor global** p_{gbest}, que corresponde a la mejor posición encontrada por el enjambre. Ambos valores se Irán actualizando a medida que aumenten las iteraciones y tomaran una gran importancia, ya que con ellos se calculara la velocidad con que las partículas se moverán en cada una de las dimensiones, además con el valor de la velocidad, la partícula podrá calcular su nueva posición para la próxima iteración.

La velocidad de una partícula en un tiempo determinado está dado por:

$$v_{id}(t+1) = w \bullet v_{id}(t) + c_1 r_1 \left(p_{id}(t) - x_{id}(t) \right) + c_2 r_2 \left(p_{gd}(t) - x_{id}(t) \right) \qquad (4.3.1)$$

donde w corresponde al factor inercial, que según [Kennedy y Eberhart, 2001], se utiliza para balancear la velocidad de la partícula respecto a la búsqueda local y global. Por otro lado $c_1 \quad y \quad c_2$ son variables para controlar la influencia de las componentes cognitivas y social.

Finalmente los parámetros r_1 y r_2 son valores independientes que se asignan aleatoriamente dentro del intervalo [0,1].

El parámetro w es el compensa la capacidad de exploración global y local del enjambre, por lo que se considera critico en la convergencia del algoritmo. Si w toma un valor grande sería favorable para la exploración global, por el contrario, si el valor es pequeño facilita la exploración local. Según [Abraham y Liu, 2006] es recomendable asignarle un valor grande al factor inercial $(w = 1)$, para facilitar la exploración global y gradualmente disminuirlo para refinar las soluciones (disminuirlo gradualmente hasta w=0).

Si bien los valores c_1 y c_2 no son críticos para lo convergencia del algoritmo, si podrían acelerar la convergencia y aliviar al proceso respecto de los mínimos locales. Es aconsejable que ambos parámetros tomen el valor 2, pero algunos estudios señalan que el valor 1.49 también se podría utilizar. Últimamente se ha establecido que para asignar un valor a estos parámetros basta sólo con que cumplan con la restricción $c_1 + c_2 \leq 4$. La nueva posición de la partícula se puede calcular en base a la velocidad ya calculada como se puede ver en la siguiente ecuación:

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \qquad (4.3.2)$$

Para evitar que las velocidades de las partículas se disparen infinitamente, se incorpora el parámetro V_{max}, el que delimita entre $\left[-V_{\max}, V_{\max}\right]$ el valor que puede tomar la velocidad de una partícula. Si $(v_{ij} > V_{\max})$ o $(v_{ij} < -V_{\max})$ se aplica la siguiente ecuación:

$$v_{ij} = sign\left(v_{ij}\right) \bullet V_{\max} \qquad (4.3.3)$$

Existen dos términos claves a la hora de definir los parámetros del algoritmo, estos son **exploración** y **explotación.** Se define como exploración a la libertad de las partículas para que se muevan por un amplio espacio durante la búsqueda de la solución del problema, por el contrario se entiende como explotación a la búsqueda de la solución el problema en un solo espacio. Por lo tanto, lo ideal es encontrar un equilibrio entre estos dos conceptos, es decir, que en un principio los parámetros tomen valores que faciliten la exploración, para luego cambiar a un modo de explotación. En la literatura se revela que los valores altos de **factor inercial**, $\mathbf{V_{max}}$ y **coeficiente de cognición**, como los que se muestran a continuación, facilitan la exploración de PSO

- $0{,}75 < w < 1$
- $2 < c_1 \text{ y } c_2 < 4$

Mientras que, valores pequeños de estos parámetros, facilitan la explotación del PSO

- $0{,}4 < w < 0.75$
- $0{,}1 < c1 \text{ y } c2 < 2$

Existen otros factores claves que se deben considerar, como es la definición de la función **fitness**, la que se utiliza para evaluar la aptitud de cada partícula, este proceso es clave

para asegurar el éxito del algoritmo, pues es esta función la que guía la búsqueda durante la optimización. Por otro lado se encuentra la selección del **tamaño del enjambre**, este proceso también debe ser riguroso, ya que valores muy altos pueden explorar de manera minuciosa el espacio de búsqueda, pero el costo computacional es bastante elevado debido al número de evaluaciones que se debe realizar. Comúnmente se utilizan enjambres entre 10 y 50 y para problemas complejos entre 100 y 200.

Ya definidos y analizados los parámetros que intervienen en el algoritmo, es necesario conocer los criterios existentes para terminar con las iteraciones, a continuación se muestran los criterios mas usados:

- **Fitness alcanzado:** Se preestablece un valor deseado al fitness, una vez alcanzado las iteraciones culminan.
- **Definir máximo de iteraciones:** El proceso se da por terminado una vez alcanzado un número de iteraciones previamente establecidas.
- **Definir iteraciones sin mejoras:** El proceso termina cuando se alcanza un número fijo de iteraciones sin mejoras.

4.4. Algoritmo Narrativo de PSO.

Dado un problema determinado y considerando los elementos vistos anteriormente un algoritmo de PSO esta dado por las siguientes líneas:

Generar *un enjambre de tamaño n e inicializar las velocidades y las posiciones de manera aleatoria en un espacio d-dimensional.*

Hacer

> ***Evaluar*** *el fitness de cada partícula en base a algún criterio de optimización.*
>
> ***Comparar*** *el fitness actual de cada partícula con el fitness de la mejor posición personal (p_{best}) encontrada hasta el momento.*
>
> > ***Si****, el fitness actual es mejor que el anterior, se actualiza la mejor posición personal de la partícula p_{id} al valor de la posición actual $x_{id.}$*
>
> ***Comparar*** *el fitness actual de cada partícula con el mejor fitness encontrado por el enjambre.*
>
> > ***Si****, el fitness actual es mejor que el anterior, se actualiza la mejor posición global p_{dg} al valor de la posición actual $x_{id.}$*
>
> ***Actualizar*** *las velocidades y posiciones de las partículas de acuerdo a las ecuaciones vistas en la sección 4.3, verificando que le velocidad no sobrepase la velocidad máxima definida.*

Mientras *No cumpla con algún criterio de termino.*

4.5. Actualizaciones síncronas y asíncronas en PSO.

El instante en el que se realiza la actualización de la memoria de cada partícula y el conocimiento social del grupo es la diferencia entre una actualización síncrona y una asíncrona.

En el modelo síncrono las partículas se mueven en paralelo, se evalúa en cada iteración el fitness de cada partícula, actualizando su memoria p_{id} y el conocimiento social p_{gd}. En cambio, en la actualización asíncrona cada partícula aprovecha la información actualizada por sus predecesores al momento de desplazarse, entonces, durante cada iteración, la partícula i-ésima se mueve hacia un punto nuevo usando la información de los vectores p_{id} y p_{gd}, actualizado por todas las partículas previas. Luego la partícula evalúa el fitness del nuevo punto y, de ser necesario, actualiza las variables p_{id} y p_{gd}. El modelo asíncrono acelera la optimación, pues esta información se transmite a las demás partículas.

4.6. Modelos de óptimo local y óptimo global.

En el modelo de óptimo local una partícula solo puede ser influenciada por ciertos miembros del enjambre que son adyacentes a ella, con esto convergen lentamente a las soluciones del problema. Mientras que en el modelo de óptimo global la trayectoria de búsqueda de cada partícula esta influenciada por la mejor posición encontrada por cualquiera de los miembros del enjambre, lo que le permite converger rápidamente a las soluciones del problema, pero puede quedar atrapado en un óptimo local. Estos modelos fueron propuestos por [Kennedy y Eberhart, 1995].

Según algunos estudios, es recomendable utilizar el modelo de óptimos locales para funciones multimodales, mientras que el modelo de óptimos globales para funciones unimodales.

4.7. Variantes del Algoritmo PSO Tradicional.

Hoy en día existen una gran cantidad de variaciones del algoritmo tradicional de PSO, se ha tratado de incorporar nuevos parámetros, nuevos métodos de actualización e híbridos con otros algoritmos, todo esto para mejorar la optimización y la velocidad de convergencia de este algoritmo. A continuación se describen algunas de estas variaciones:

4.8. PSO con parámetros de tiempo de vida.

En el año 2003 se introdujo un nuevo parámetro al algoritmo PSO tradicional, al que se le llamo **tiempo de vida.** A cada partícula del enjambre se le asignará este nuevo parámetro, el cual irá disminuyendo su valor a medida que la partícula no rinda en alguna iteración. Se asumen que la partícula que presente el peor valor del fitness será eliminada o relevada, y para reemplazarla se debe generar una nueva partícula en la vecindad de las restantes [Khosla y otros, 2003]. Los autores postulan que la decisión de destruir la peor partícula es tomada solamente una vez cumplido el número definido de oportunidades de mejora.

La incorporación del parámetro de tiempo de vida permite que el algoritmo de PSO rinda de mejor manera en términos de la cantidad de iteraciones para la convergencia y la calidad de la solución.

4.8.1. PSO con restricción de velocidad mínima.

Se incorpora al algoritmo de PSO tradicional un nuevo parámetro, el cual permite controlar la velocidad minima a la que puede llegar una partícula, dicho parámetro se denominó V_{min} . Una de las ventajas del algoritmo tradicional, es la no convergencia hacia un mínimo global, ya que si las partículas poseen velocidades muy altas pueden escaparse del espacio de búsqueda, y por el contrario, si presentan velocidades muy bajas pueden quedar atrapas en un optimo local [Pu, Fang y Liu, 2007]. Con la presencia de V_{min}, se puede decir que: Si $(v_{id} > -V_{\min})$ o $(v_{id} < -V_{\min})$ se aplica la siguiente ecuación:

$$v_{id} = sign\left(v_{id}\right) \bullet \bar{v} \tag{4.8.1.1}$$

dond el promedio de las velocidades de las partículas $\bar{v}$, está dado por:

$$\bar{v} = \frac{1}{N \bullet D} \sum_{i=1}^{N} \sum_{d=1}^{D} \left(\left|v_{id}\right|\right) \tag{4.8.1.2}$$

donde N representa el número de partículas el enjambre, y D el número de dimensiones. Con esta variante se ha mejorado la calidad del óptimo encontrado, pero se podría hacer una modificación en el promedio de velocidad empleado, en lugar de utilizar todas las velocidades, se podrían utilizar solo las que pertenecen a la dimensión donde se encontró la velocidad mínima.

En lo últimos años la popularidad de PSO se ha incrementado, pues se ha utilizado en un gran cantidad de aplicaciones y ha recibido una amplia atención por parte de los investigadores, a continuación se mencionan alguna un listado de aplicaciones del algoritmo.

- **Problema del cajero viajante:** Se implementaron algunas propuestas de enfoques de PSO para problemas multiobjetivos (MOPSO) y se realizó una comparación entre los resultados conseguidos por estos enfoques y los resultados obtenidos al aplicar algoritmos de Optimización de Colonias de Hormigas (ACO) para problemas multiobjetivos (MOACO). Como resultado, quedó demostrado que el uso de técnicas MOPSO junto con algunas adaptaciones aplicadas constituyen una buena alternativa para la resolución del problema [Baran y Lima, 2005].
- **Síntesis de Arrays:** Se compara PSO y algoritmos genéticos, llegando a la conclusión de que la aplicación de ambos métodos aplicados a la síntesis de alimentaciones en agrupaciones lineales de antenas, demuestra que la rapidez de convergencia, y la facilidad de implementación en PSO están por sobre los AG [Correa y otros, 2005].

5. SIMULACIÓN Y DISCUSIÓN DE RESULTADOS.

5.1. Introducción.

Como se mencionó en la sección 2.4. los problemas de SVM para regresión se resuelven frecuentemente por medio de programación cuadrática, formular un problema de este tipo con unos pocos cientos de datos, lo cual es habitual en problemas prácticos, demanda muchos recurso de cómputo, esto plantea un problema en el uso de recursos al momento de entrenar una SVM.

Debido a lo anterior se propone utilizar máquinas de soporte vectorial de mínimos cuadrados (LS-SVM) para la resolución del problema planteado, ya que este planteamiento permite simplificar algunos aspectos de las SVM sin perder sus ventajas.

Los distintos modelos fueron implementados en Java y Matlab, donde los algoritmos evolutivos AG y PSO fueron codificados en Java, mientras que el cálculo de la matriz pseudo-inversa necesario para resolver el problema de LS-SVM se implemento en Matlab, debido a su mejor capacidad al momento de trabajar con matrices.

Se utilizó la versión de Matlab R2008a, el ToolBox LS-SVMlab 1.5, que proporciona las funciones necesarias para trabajar con máquinas de soporte vectorial y mínimos cuadrados, además se debió utilizar la librería JMatLink para poder acceder a las funciones Matlab desde Java. Los modelos fueron testeados en un equipo Dell, Centrino Duo 1,6 Ghz. y 2 Gb. RAM. En el Anexo A se muestran el código fuente de los principales métodos de los algoritmos evolutivos.

El número de datos utilizados tanto para el entrenamiento como testeo de los modelos es de 400, de los cuales 350 se utilizaron para entrenamiento y 50 para testeo. Estos datos corresponden el precio de cierre de 400 días de las **acciones de LAN,** las cuales fueron elegidas aleatoriamente y corresponden a los días entre el 26-06-2007 y el 30-07-2008.

En la etapa de testing, se realizaron entre 4 y 5 iteraciones, utilizando en cada una distintos desfases de 2,3 y 4 días, las métricas usadas fueron RMSE, MAPE y R^2, las cuales permites medir la calidad de los distintos modelos.

Durante este capítulo se muestran las métricas utilizadas para la evaluación del desempeño de los modelos propuestos, se describe detalladamente la implementación de los modelos LS-SVM con AG, LS-SVM con PSO, el híbrido de los modelos anteriores LS-SVM AG+PSO, y los resultados de cada uno de estos.

5.2. Métricas de Evaluación.

Para la estimación de los parámetros de la LS-SVM se utilizaron un conjunto de métricas de exactitud calculadas entre los datos observados (valores reales) y los datos pronosticados (entregados por LS-SVM). A continuación se describen las métricas utilizadas, donde d_i corresponde al valor observado en el día i, d_i al valor pronosticado en el día i, $\overline{d}$ como la media de la data observada y N como el número total de días:

- Error Cuadrático Medio (RMSE). Consiste en la suma de las diferencias entre los datos observados y los datos proyectados por el modelo.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}\left(d_i - y_i\right)^2}{N}} \qquad (5.2.1)$$

- Coeficiente de Determinación (R^2). Mide la dependencia entre los datos reales y los pronosticados. El 0 muestra independencia y el 1 lo contrario.

$$R^2 = 1 - \frac{\sum_{i=1}^{N}\left(d_i - y_i\right)^2}{\sum_{i=1}^{N}\left(d_i - \overline{d}\right)^2} \qquad (5.2.2)$$

- Porcentaje de Error Medio Absoluto (MAPE). Proporciona una indicación de que tan grandes son los errores de pronóstico comparados con los valores reales de la serie. También, correspondiente a la operación interna de la sumatoria se encuentra el Porcentaje de Error Absoluto (APE).

$$MAPE = \frac{\sum_{i=1}^{N}\left|\frac{d_i - y_i}{d_i}\right|}{N} x100 \qquad (5.2.3)$$

Es necesario señalar que también se considerará el tiempo de computo para cada prueba realizada a los modelos.

5.3. Tendencia de RMSE.

Como se mencionó anteriormente, un punto importante es el número de iteraciones de cada algoritmos. Se estableció un número máximo de 20 iteraciones, pues tanto en AG como en PSO el RMSE, que fue la métrica que se utilizó como función fitness en todos los modelos que aquí se describen, tiene la tendencia que se muestra en la figura 5-1, donde se presentan los resultados de 6 pruebas, 3 midiendo la tendencia de RMSE con AG, y 3 midiendo la tendencia de dicha métrica con PSO.

Se puede apreciar que en las 6 pruebas RMSE converge a un valor antes de la iteración número 20.

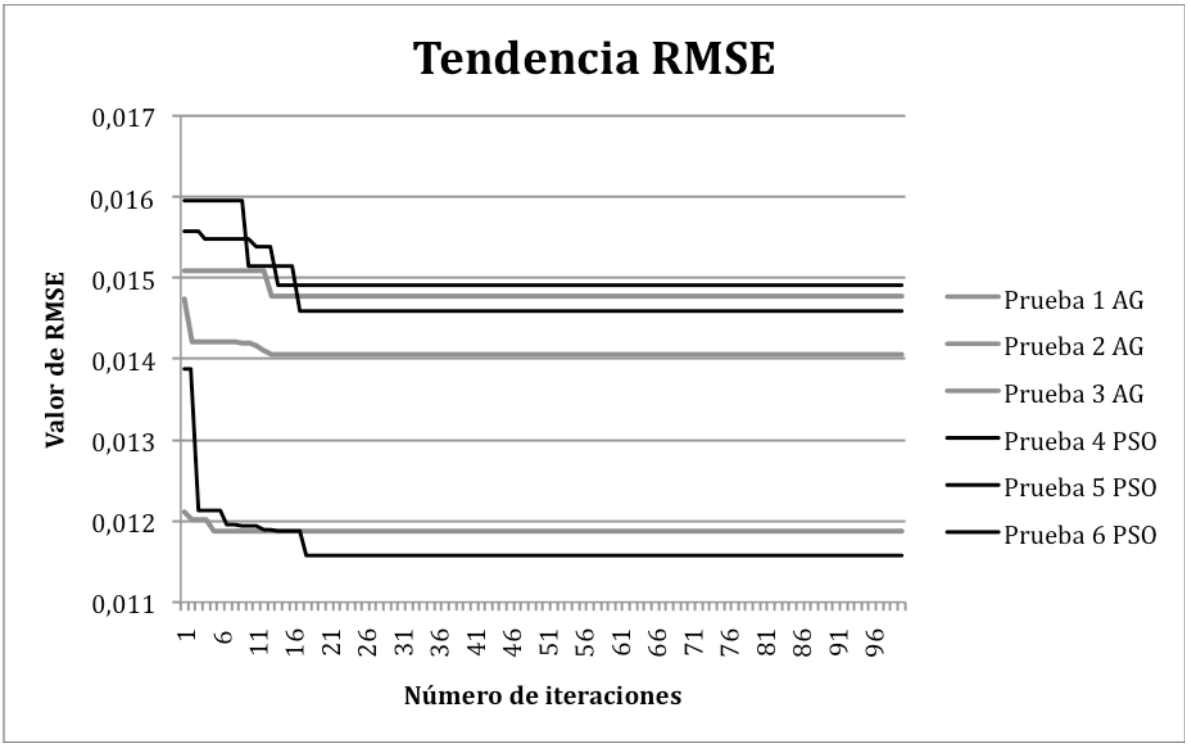

Figura 5- 1. Tendencia RMSE.

5.4. Modelo LS-SVM AG.

Este modelo consiste en una máquina de soporte vectorial de mínimos cuadrados (LS-SVM), la cual utiliza una función Kernel RBF. Los parámetros C, correspondiente al parámetro de regularización de la máquina y N perteneciente al kernel definido, son estimados mediante algoritmos genéticos.

A continuación, en la figura 5-2, se muestra un diagrama que describe el proceso de entrenamiento:

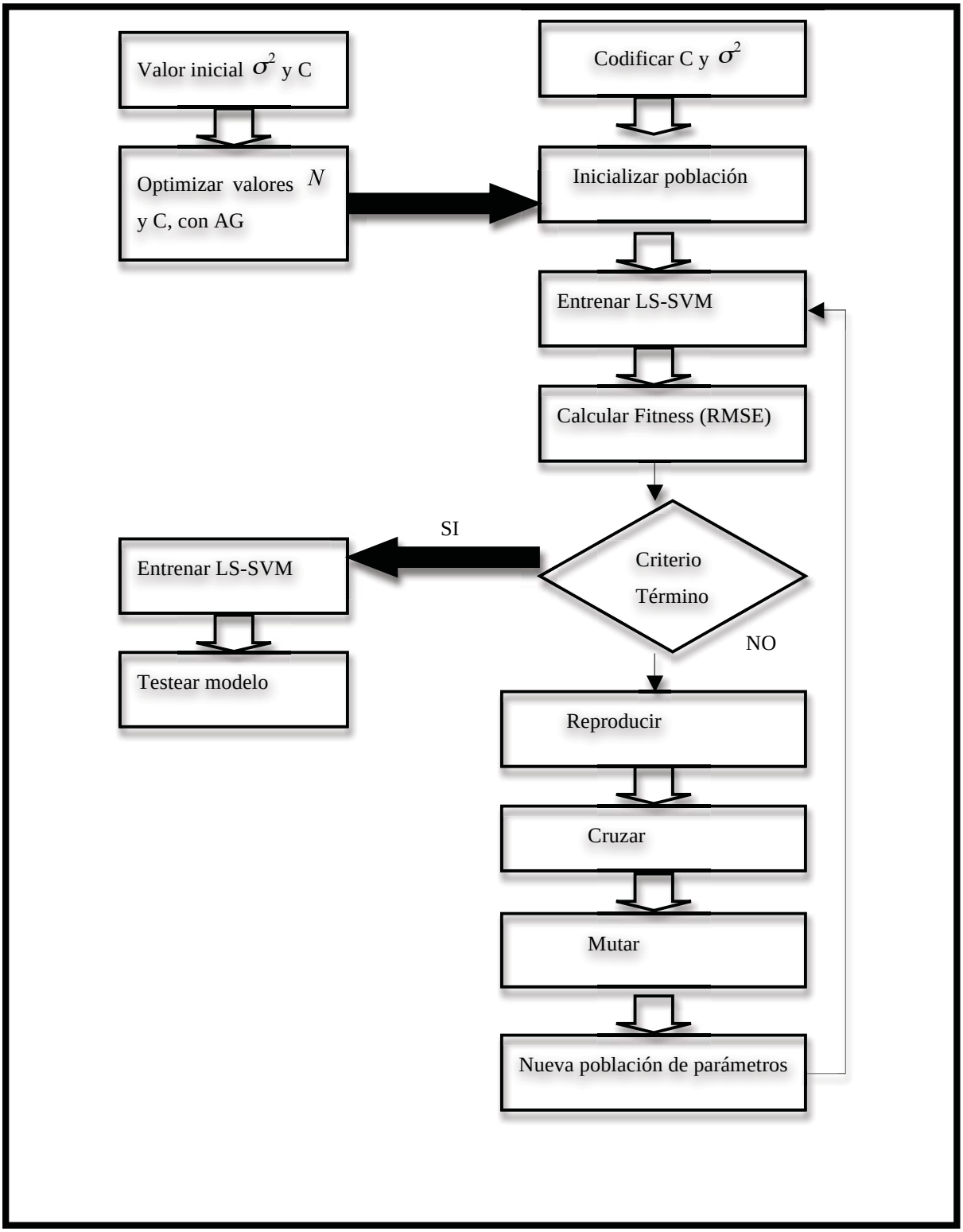

Figura 5- 2. Diagrama algoritmo LS-SVM AG.

5.4.1. Características del Entrenamiento.

En la utilización de Algoritmos Genéticos se deben tomar decisiones claves respecto a elementos para el correcto funcionamiento de la técnica. Los elementos utilizados y seleccionados se detallan a continuación.

La cantidad de individuos de la población utilizada, corresponde a un número de 40 individuos. Cada uno es representado por un cromosoma que posee una codificación binaria, Cada gen va a corresponder a un valor para 1 ó 0 para cada valor de entrenamiento. El tamaño del cromosoma es de 20 genes, donde:

- 13 genes son utilizados para definir el parámetro C, pudiendo tomar valores entre 0 y 1023.
- 7 genes son utilizados para definir el parámetro σ^2, pudiendo tomar valores ente 0 y 127

La estructura del cromosoma es la que se muestra en la figura

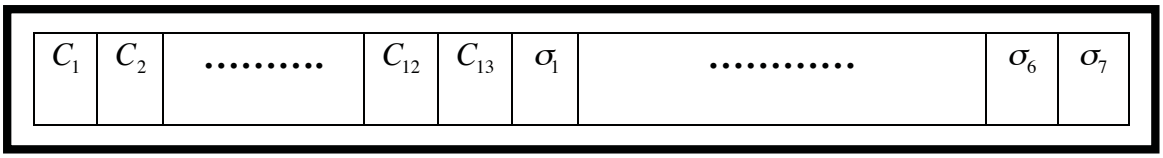

Figura 5- 3. Codificación del cromosoma.

Para medir el desempeño de cada cromosoma se utilizó la métrica RMSE. Una de las decisiones claves durante la técnica de AG, es elegir algún criterio para la selección de los padres para generar la nueva generación. En esta etapa de la investigación se optó por seleccionar al cromosoma que presente un menor RMSE.

El procedimiento para generar la nueva generación de individuos, consistió primero en ordenar a los cromosomas de menor a mayor rendimiento. La primera mitad de cromosomas mejor evaluada fueron seleccionados para ser padres, y los cromosomas restantes se reemplazaron por la siguiente nueva generación. Posteriormente, se seleccionaron padre y madre para realizar el cruzamiento y mutación, los cuales se detallan a continuación:

- Cruzamiento: Se definió de manera aleatoria un punto de cruce.
- Mutación: Consiste en cambiar el valor del gen ya sea de 1 a 0 o viceversa.

El entrenamiento a través de Algoritmo Genético necesita un criterio de término para finalizar la búsqueda de los vectores de soporte. El criterio elegido fue un número fijo **20 de iteraciones**.

La siguiente tabla muestra el resumen de los parámetros utilizados en el AG durante la optimización de los parámetros de la LS-SVM:

Tabla 5- 1. Parámetros algoritmo LS-SVM AG.

Parámetros GA	
Tamaño población	40
Tipo selección	Ranking
Probabilidad cruce	1.0
Tipo cruce	1 punto
Probabilidad mutación	0.8
Tipo de mutación	1 punto
Número iteraciones	10

En la siguiente tabla se muestran los mejores resultados de 5 iteraciones del algoritmo con distintos desfase en los datos

Tabla 5- 2. Resultados LS-SVM AG.

Iteración	Desfase	RMSE	R^2	C	σ^2	Tiempo Testing
1	**2**	**0,0093**	**0,9293**	**1023**	**0,02**	**3 min. 50 seg.**
2	2	0,0105	0,9111	1019	0,01	3 min. 57 seg.
3	3	0,0120	0,8992	1023	0,02	4 min. 25 seg.
4	3	0,0127	0,8670	998	0,01	4 min. 25 seg.
5	4	0,0150	0,8249	1020	0,01	8 min. 23 seg.

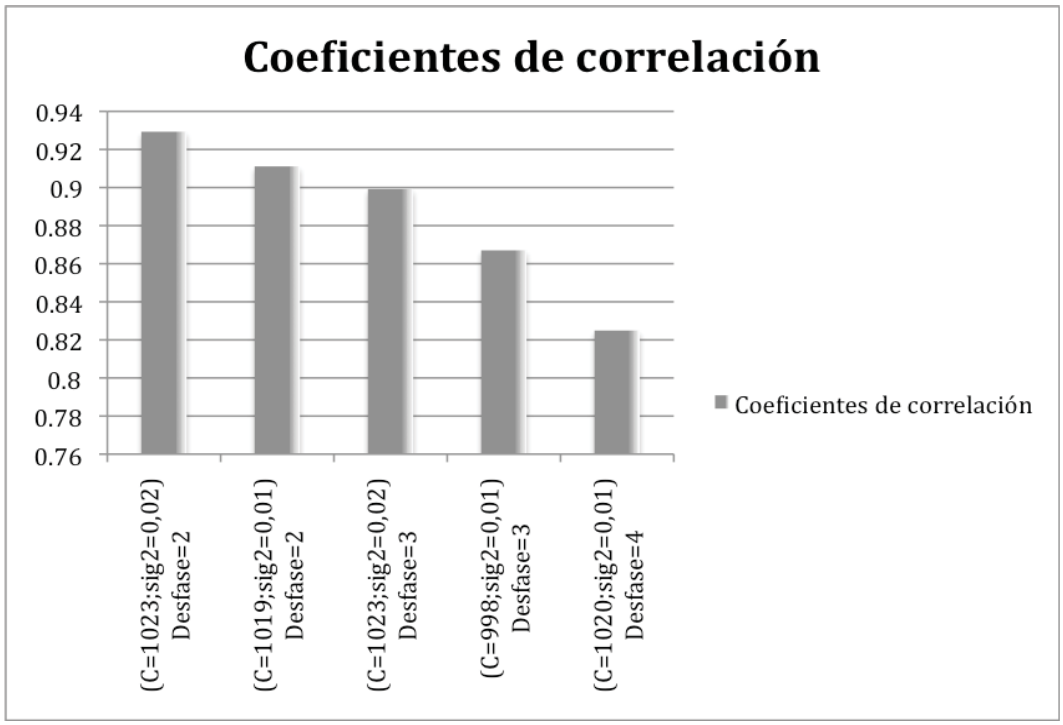

Figura 5- 4. Coeficiente de Correlación LS-SVM AG.

Basándose en los resultados de las pruebas realizadas, que se aprecian resumidamente en la Tabla 5-2 y Figura 5-4, se puede apreciar que los mejores resultados se obtienen con el parámetro C cercano a 1000 y σ^2 cercano a 0.1, a demás, los mejores resultados se obtienen con un desfase de 2 en los datos, comprobando que a medida que este desfase se incrementa, el porcentaje de la varianza explicada disminuye en prácticamente 10 puntos porcentuales.

Debido a que los mejores resultados se obtienen con el desfase 2, el tiempo de cómputo es menor, siendo de 3 minutos y 50 segundos llegando a presentar un porcentaje del 92,93% de la varianza explicada.

En este punto es necesario comentar que, a medida que aumente el valor del R^2 entregado por un modelo, mayor será el porcentaje de la varianza explicada por éste, es decir, la línea que grafica la data pronosticada deberá seguir la misma trayectoria que la data observada. A continuación se presenta el gráficos de estimación del modelo LS-SVM AG

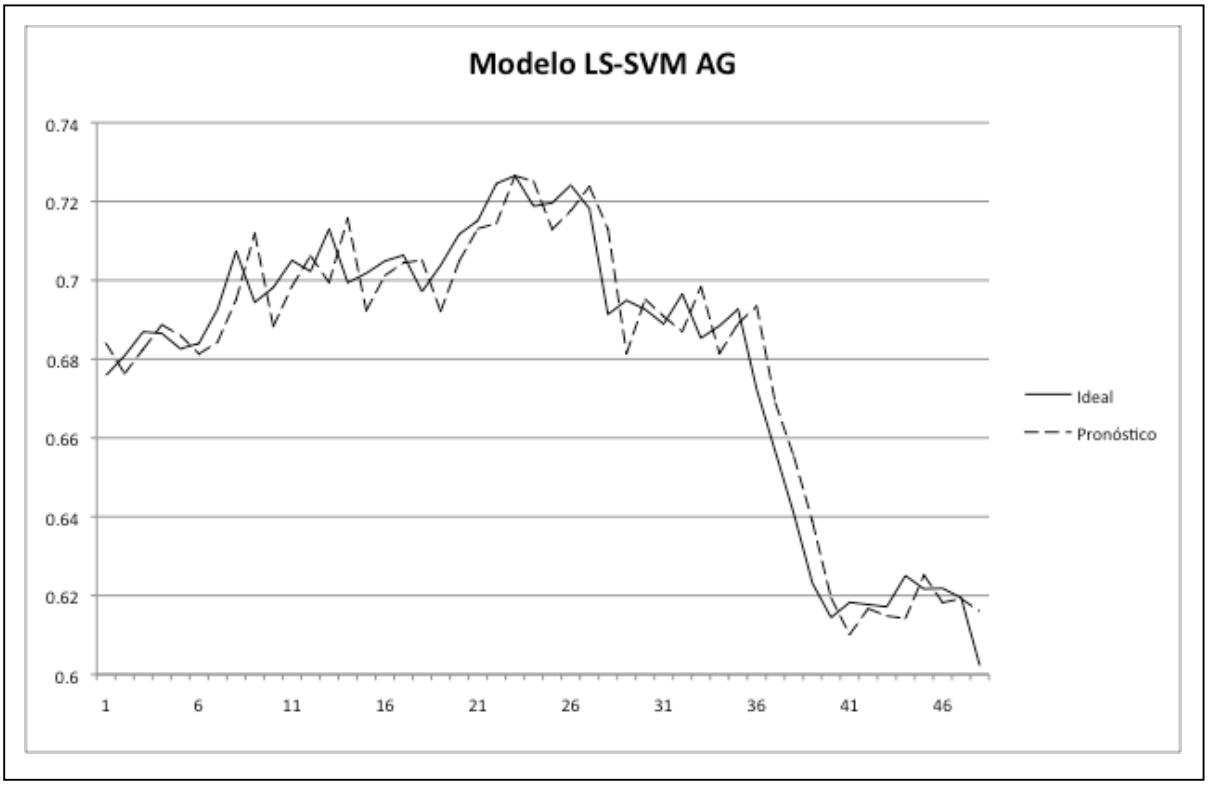

Figura 5- 5. Mejor estimación modelo LS-SVM AG.

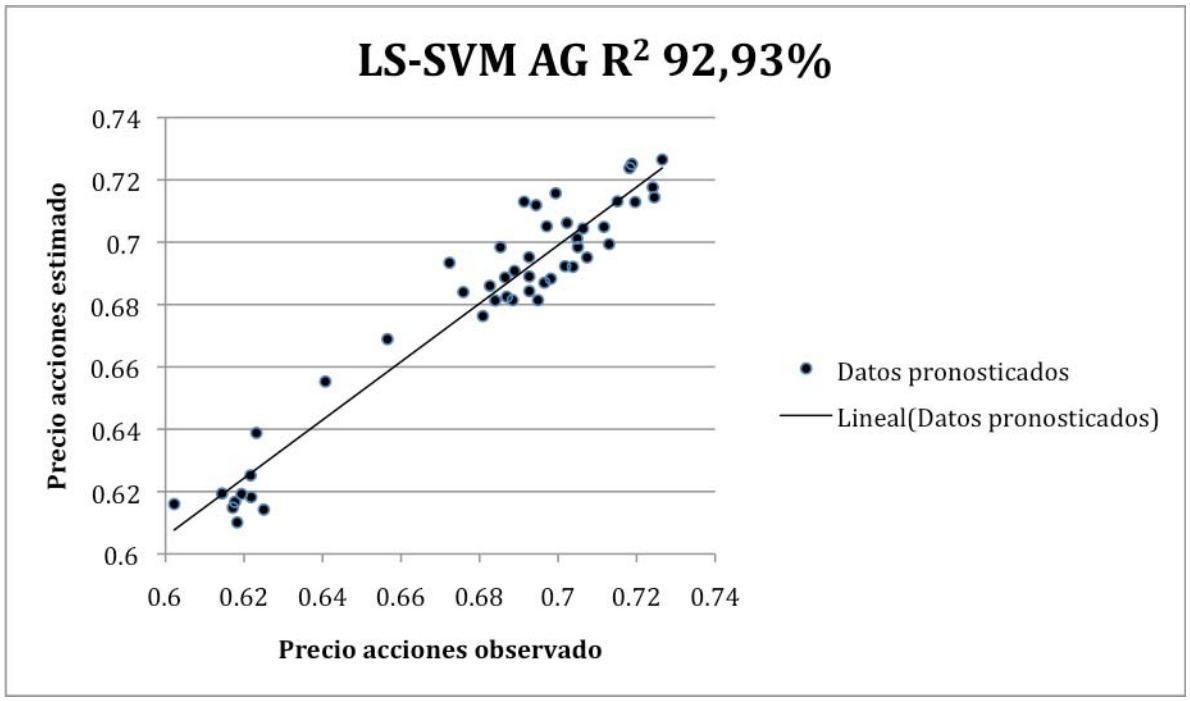

Figura 5- 6. Modelo LS-SVM AG R^2 92,93%.

5.5. Modelo LS-SVM PSO.

Este modelo es una propuesta análoga a LS-SVM AG, pues consiste en una máquina de soporte vectorial de mínimos cuadrados (LS-SVM), que utiliza una función Kernel RBF. Pero los parámetros C, correspondiente al parámetro de regularización de la máquina y el valor σ^2 perteneciente al kernel definido, son estimados mediante PSO.

La figura 5-7 muestra un diagrama que describe el proceso de entrenamiento del modelo.

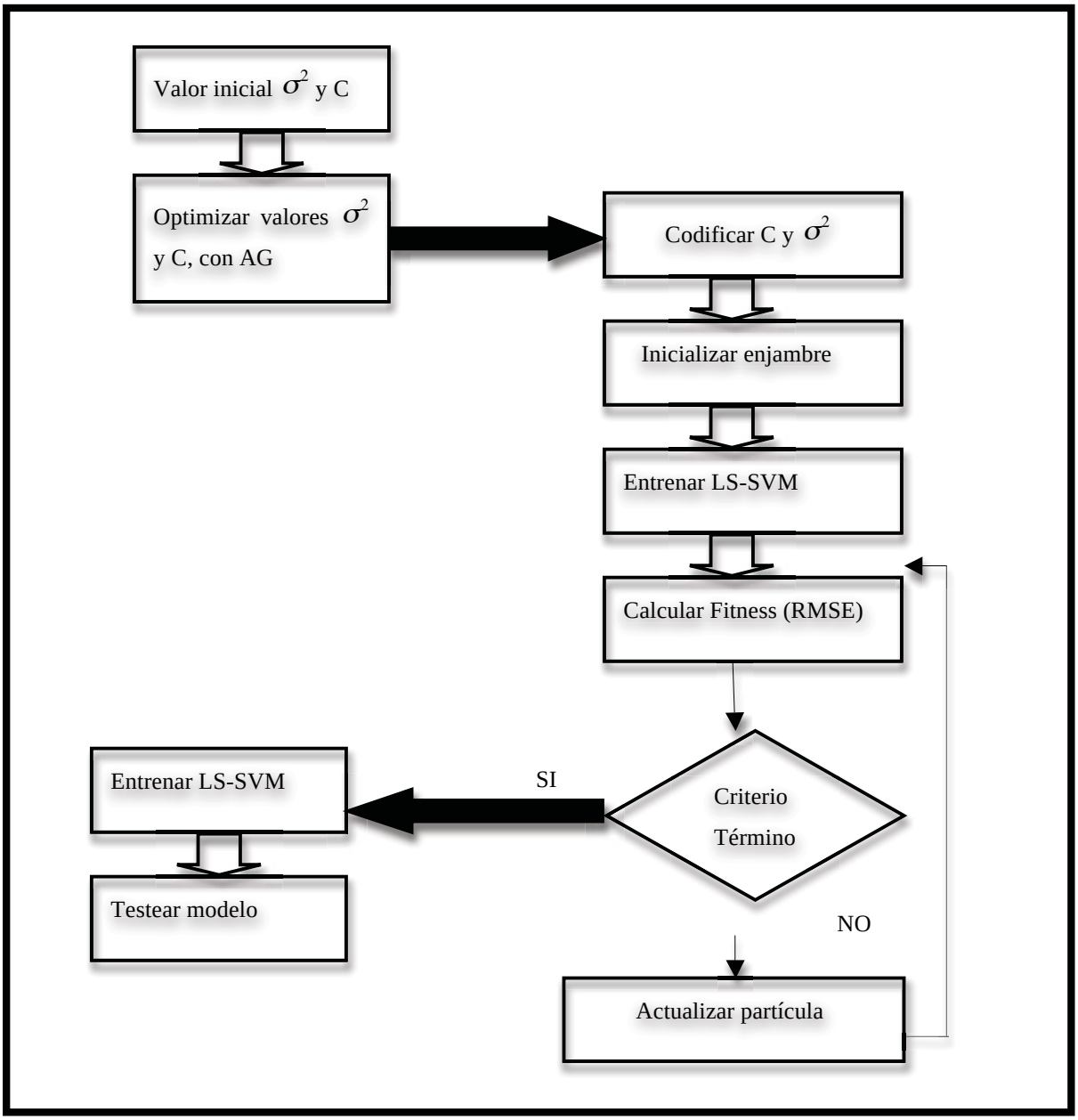

Figura 5- 7. Diagrama Algoritmo LS-SVM PSO.

5.5.1. Características del Entrenamiento.

Al igual que en AG, en la utilización de PSO se deben tomar decisiones claves respecto a elementos para el correcto funcionamiento de la técnica. Estos elementos se detallan a continuación. La cantidad de partículas del enjambre, corresponde a un número de 40 individuos. Cada uno es representado por una partícula que posee una codificación real.

El proceso de entrenamiento comienza con la creación aleatoria de un enjambre inicial de 40 partículas, donde cada una de estas contendrá como posición (al igual que el cromosoma en el caso de AG) C, correspondiente al parámetro de regularización de la máquina, σ^2 perteneciente al kernel RBF, una velocidad generada de aleatoriamente y una memoria personal (P_{best}) con el mismo valor que la posición. Una vez definida la posición, velocidad y memoria de la partícula se procede al cálculo inicial del error. Finalmente se tendrá un enjambre inicial de 40 partículas definiendo como mejor posición global (P_{gbest}) al menor error encontrado. El criterio de término utilizado fue, al igual que en LS-SVM AG un número de iteraciones máximo de 20, 10 iteraciones de carácter exploratorio y 10 de carácter de explotación.

El proceso de actualización de las partículas se basa en el algoritmo descrito en la sección 4.3, específicamente aplicando las fórmulas (4.3.1) y (4.3.2) para la actualización de la velocidad y posición respectivamente. Luego se procedió a la actualización del error de cada partícula, P_{best} y P_{gbest}. Esto se hace reemplazando las nuevas posiciones (meta-parámetros) obtenidas por la máquina de soporte vectorial y el cálculo del RMSE obtenido al procesar la data de entrenamiento y compararla con la data esperada.

Tabla 5- 3. Parámetros algoritmo LS-SVM PSO.

Parámetros PSO	
Tamaño enjambre	40
Número iteraciones	20
Fitness	RMSE
Explotación	
Coeficiente inercial (w)	0.8
Componente cognitiva (C_1)	1.5
Componente social (C_2)	2.0
Exploración	
Coeficiente inercial (w)	0.4
Componente cognitiva (C_1)	2.0
Componente social (C_2)	1.49

A continuación, en la siguiente tabla se muestran los mejores resultados de 5 iteraciones del algoritmo, con un desfase igual a 2:

Tabla 5- 4. Resultados LS-SVM PSO.

(**Iteración**	**Desfase**	**RMSE**	**R^2**	**C**	σ^2	**Tiempo Testing**
1	2	0,0089	0,9314	284	0,55	4 min. 0 seg.
2	**2**	**0,0087**	**0,9356**	**1041**	**0,2**	**4 min. 2 seg.**
3	3	0,0091	0,9230	801	0,26	8 min. 2 seg.
4	3	0,0090	0,9318	784	0,62	8 min. 5 seg.
5	4	0,01	0,9021	821	0,43	12 min. 8 seg.

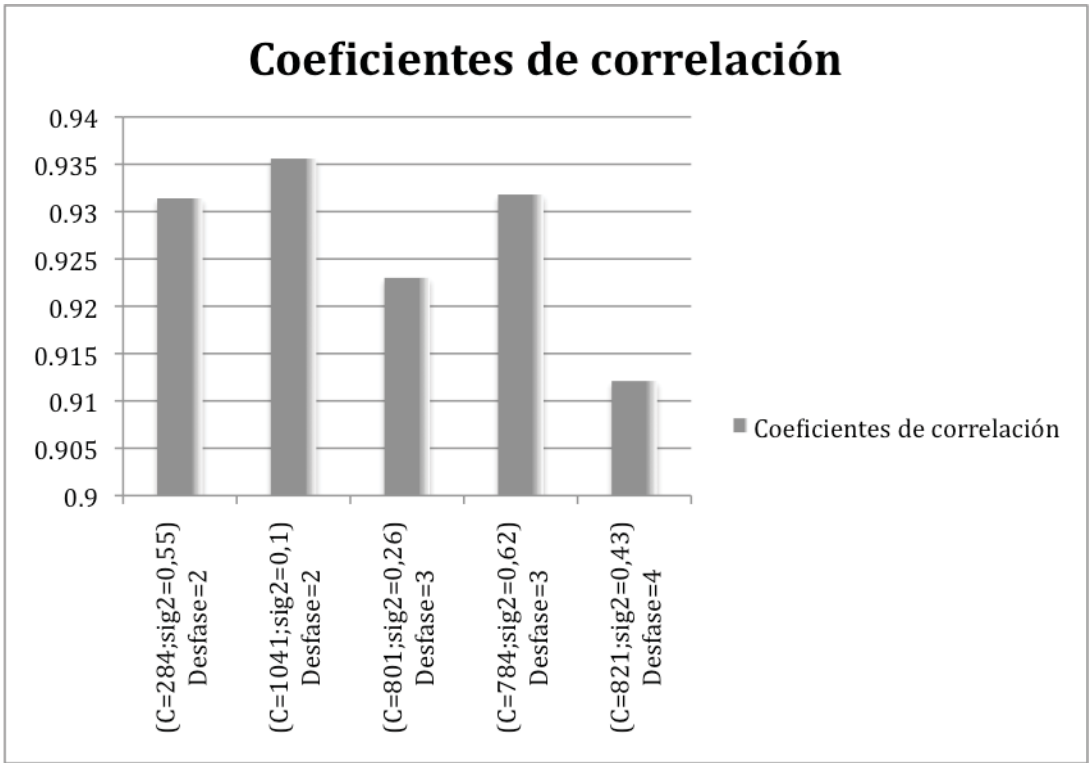

Figura 5- 8. Coeficiente Correlación LS-SVM PSO.

Basándose en los resultados de las pruebas realizadas, que se aprecian resumidamente en la Tabla 5-4 y Figura 5-8, se puede apreciar que los mejores resultados se obtienen con el parámetro C= 1041 y σ^2 cercano a 0.2, a demás, los mejores resultados se obtienen, al igual que con el modelo LS-SVM PSO, con un desfase de 2 en los datos, comprobando que a medida que este desfase se incrementa, el porcentaje de la varianza explicada prácticamente no disminuye en comparación al modelo anterior.

Como se mencionó anteriormente, mientras menor es el desfase, menor es el tiempo de cómputo, el mejor resultado presentó un porcentaje del 94% de la varianza explicada, en 4 minutos.

Como ya se mencionó, a medida que aumente el valor del R^2 entregado por un modelo, mayor será el porcentaje de la varianza explicada por éste, es decir, la línea que grafica la data pronosticada deberá seguir la misma trayectoria que la data observada. A continuación se presenta el gráficos de estimación del modelo LS-SVM AG

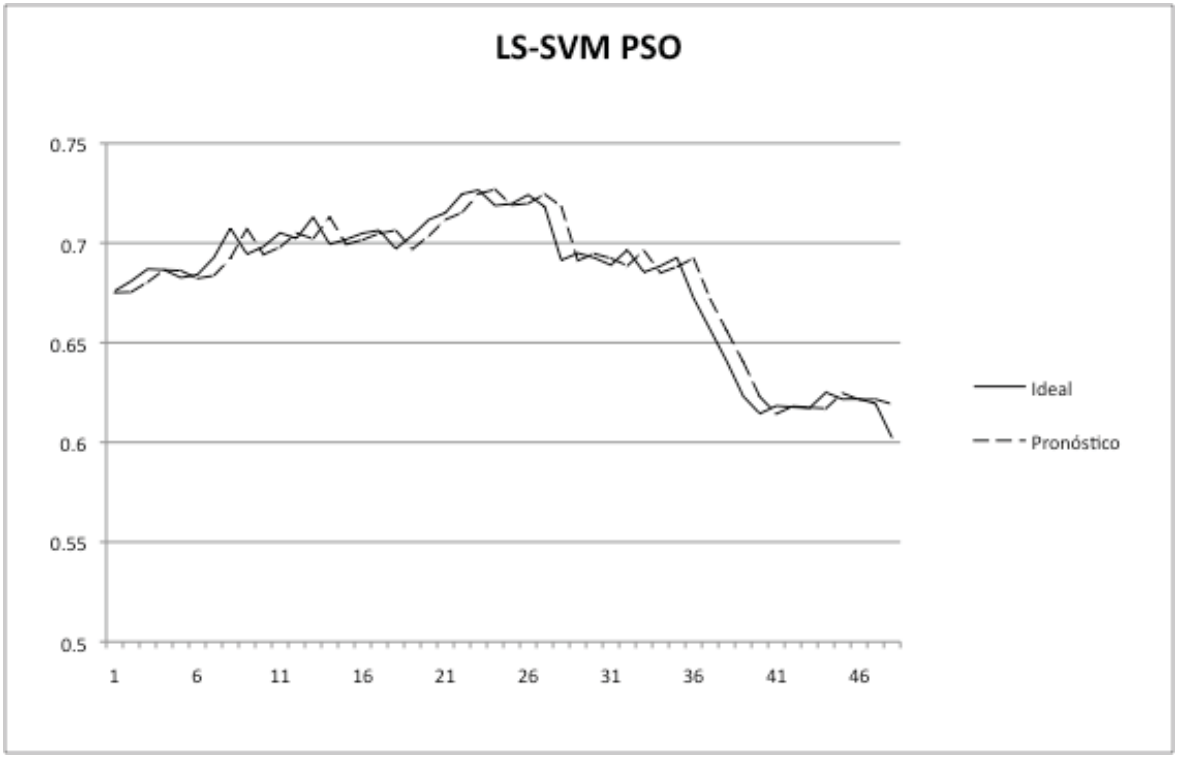

Figura 5- 9. Mejor estimación modelo LS-SVM PSO.

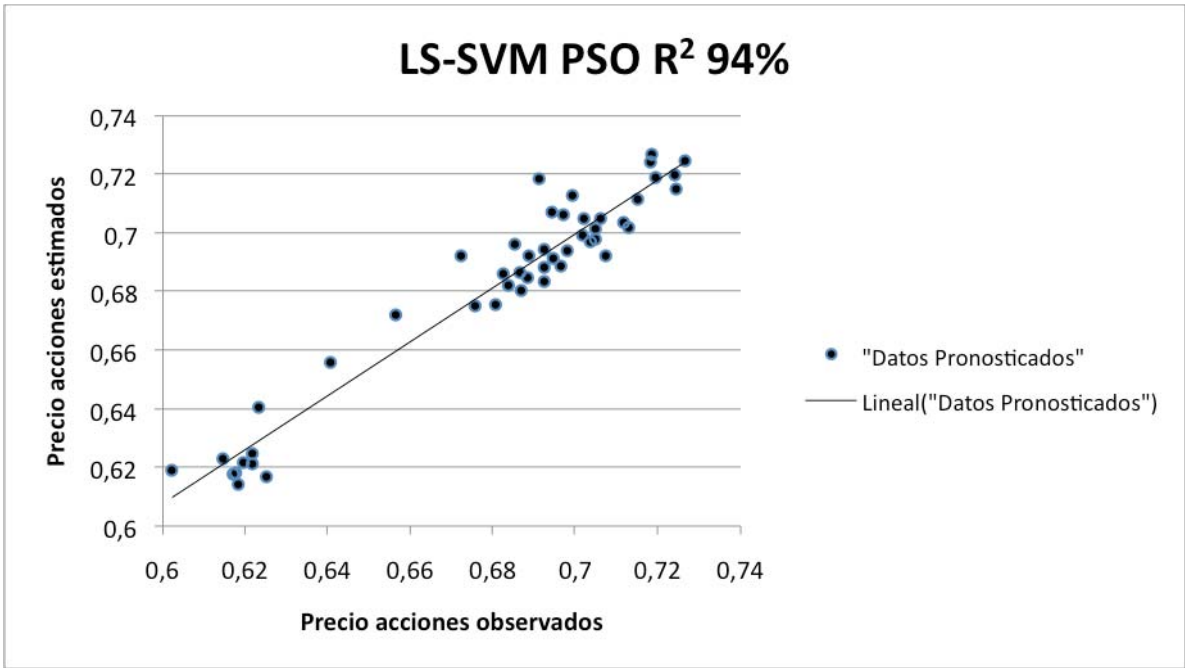

Figura 5- 10. Modelo LS-SVM PSO R^2 94%.

5.6. Modelo LS-SVM AG+PSO.

Este modelo corresponde a la LS-SVM utilizada en los modelos anteriores, con un algoritmo de entrenamiento que consiste en la búsqueda de los meta-parámetros C y σ^2 por parte de AG, dando así un carácter inicialmente exploratorio y continuando con PSO el cual presenta un carácter de explotación del espacio de búsqueda. En la figura 5-14, se muestra el diagrama que describe el proceso de entrenamiento híbrido de AG y PSO.

5.6.1. Características del Entrenamiento.

Como se mencionó en los modelos anteriores, tanto en la utilización de AG como PSO deben tomar decisiones claves respecto a elementos para el correcto funcionamiento de la técnica. En este caso se utilizaron 20 iteraciones, 10 para AG y 10 para PSO.

La única diferencia, con relación a los modelos presentados en las secciones 5.3 y 5.4, se aprecia en la inicialización del enjambre de PSO, dado que ahora no es de carácter aleatorio, sino que consiste en la obtención de las posiciones en base a la población de cromosoma que presento GA en la última iteración.

Los parámetros utilizados en cada una de las recurrencias y algoritmos correspondientes se muestran en las siguientes tablas.

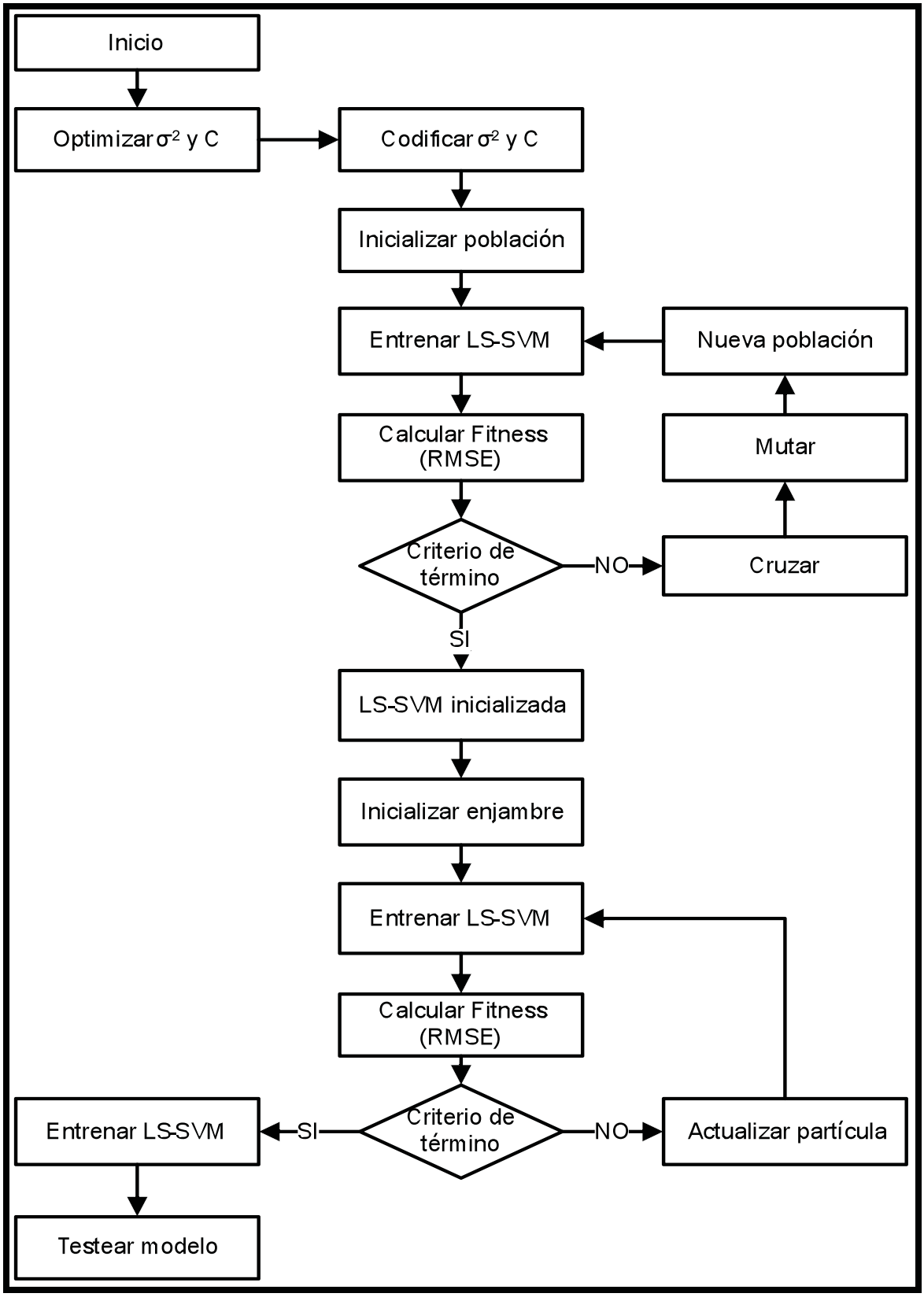

Figura 5- 11. Mejor estimación modelo LS-SVM AG+PSO.

Tabla 5- 5. Parámetros algoritmo LS-SVM AG+PSO.

Modelo LS-SVM AG+PSO			
Parámetros AG		**Parámetros PSO**	
Tamaño población	50	Tamaño enjambre	50
Tipo selección	Ranking	Peso inercial	0.4
Probabilidad cruce	0.8	Componente cognitiva	2.0
Tipo cruce	1 punto	Componente social	1.49
Probabilidad mutación	0.5	Velocidad máxima	0.04
Tipo de mutación	1 punto	Posición máxima	1200
Número iteraciones	10	Número iteraciones	10

Tabla 5- 6. Resultados algoritmo LS-SVM AG+PSO.

Iteración	**Desfase**	**RMSE**	**R^2**	**C**	σ^2	**Tiempo Testing**
1	**2**	**0,0088**	**0,9333**	**983**	**0,01**	**14 min. 7 seg.**
2	2	0,0096	0,9212	1041	0,2	14 min. 2 seg.
3	3	0,0103	0,9120	831	0,36	20 min. 2 seg.

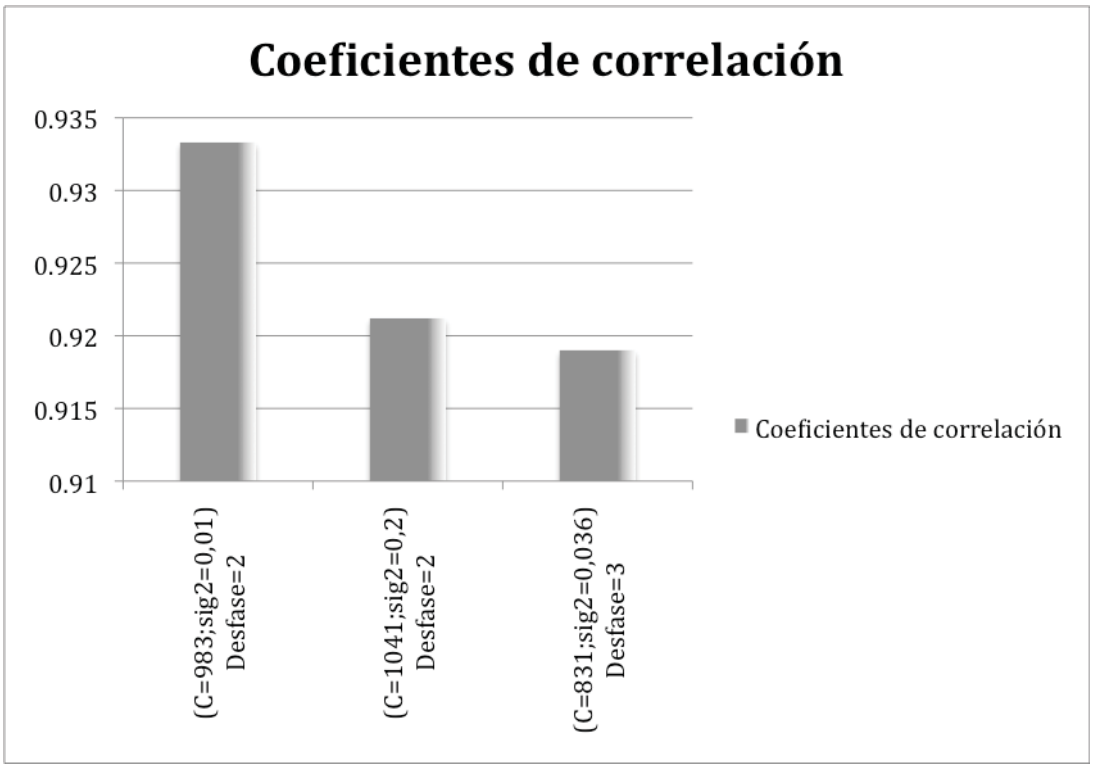

Figura 5- 12. Coeficiente correlación modelo LS-SVM AG+PSO.

A partir de los resultados de las pruebas realizadas, los resultados presentados en la Tabla 5-6 y la Figura 5-8 se aprecia que los mejores resultados se obtienen con el parámetro C= 983 y σ^2 cercano a 0.01, nuevamente el mayor valor de R^2 se obtiene con un desfase de 2 en los datos, pero en el presente modelo, el costo en tiempo se dispara por sobre los 14 minutos.

El mejor resultado presentó un porcentaje del 93,36% de la varianza explicada, en un tiempo de 14 minutos y 7 segundos.

A continuación, en la figura 5-13, se presenta el gráfico de estimación del modelo LS-SVM AG+PSO.

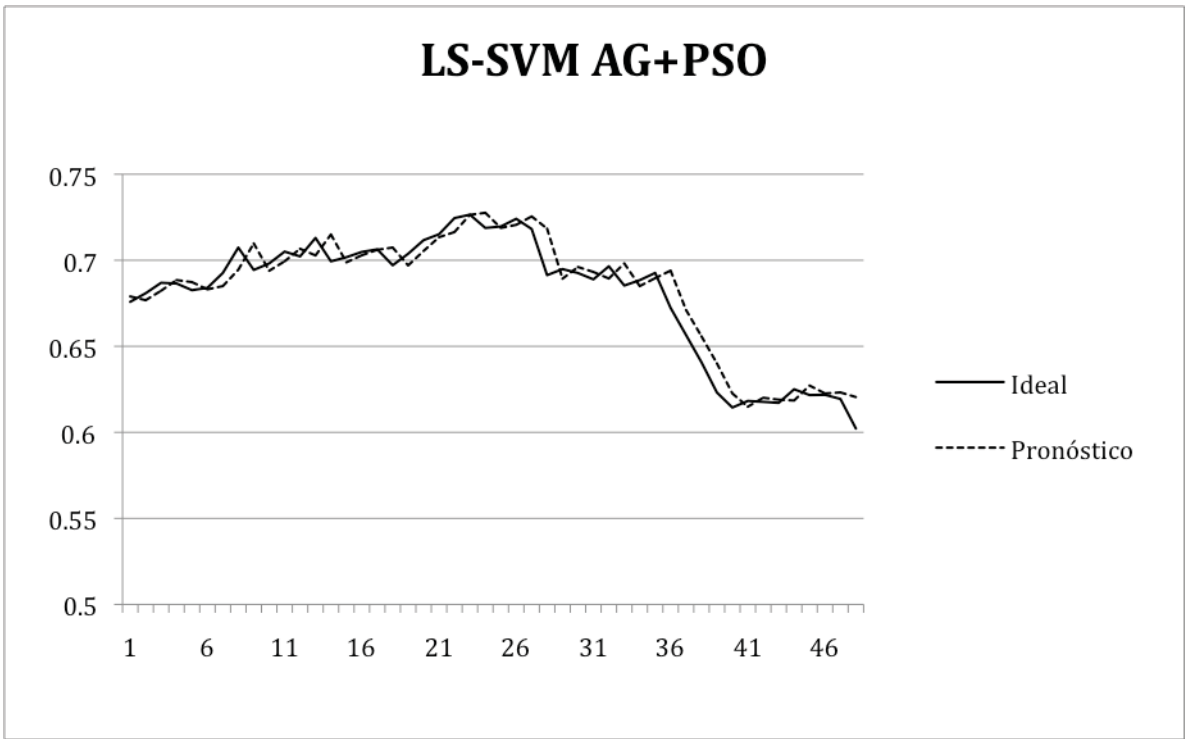

Figura 5- 13. Mejor pronóstico Modelo LS-SVM AG+PSO.

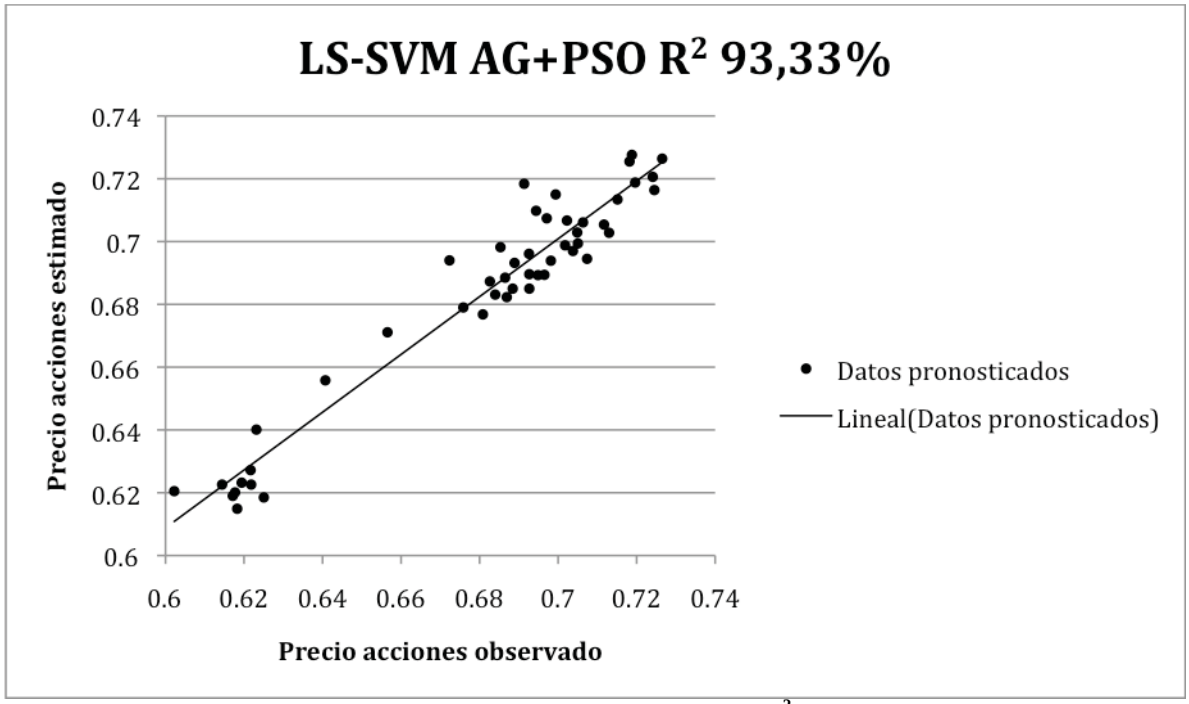

Figura 5- 14. Modelo LS-SVM AG+PSO R^2 93,33%.

5.7. Análisis de resultados.

Al comparar los resultados obtenidos por los tres modelos anteriormente presentados, se puede apreciar que LS-SVM AG, LS-SVM PSO y LS-SVM AG+PSO, presentan resultados similares, presentando, en los tres casos, un porcentaje superior al 90% de la varianza explicada.

Los dos últimos modelos, LS-SVM PSO y LS-SVM AG+PSO presentan resultados superior al 93% de la varianza explicada, donde el primero destaca por tener una raíz del error cuadrático medio menor y un coeficiente de correlación mayor, pero la característica que más resalta de la comparación de los dos modelos ya mencionados, es el menor tiempo de la solución basada en PSO en comparación a la solución híbrida AG+PSO, 4 minutos 2 segundos frente a 14 minutos 7 segundos.

Al LS-SVM PSO presentar un RMSE menor y un R^2 mayor, debería suponer que la línea que grafica la data pronosticada debería sobreponerse a la trayectoria de la data observada, pero en la figura 5-7 se aprecia un cierto desfase entre ambas curvas, lo que implica que la curva de pronostico sigue la misma trayectoria de la curva observada pero con un retardo en el tiempo.

Los gráficos correspondientes a las figuras 5-6, 5-10 y 5-14 presentan la correlación existente entre la data pronosticada y la observada en los modelos AG, PSO y AG+PSO respectivamente. Aquí PSO refleja su 94% de varianza explicada presentando los datos pronosticados más apegados a la recta de tendencia que los demás. Es decir, al tener AG y AG+PSO datos pronosticados más dispersos que PSO se da a entender que los datos que estos pronosticas no varían tan acorde con la data real observada como lo hace PSO.

6. CONCLUSIONES.

En el presente proyecto de título se presentó el estudio del estado del arte de los principales componentes que se utilizaron en el desarrollo de los modelos basados en máquinas de soporte vectorial para la predicción del precio de las acciones LAN Chile.

Se describió la técnica de las SVM y SVMR, además se presentaron los fundamentos teóricos de LS-SVM, lo que permitió simplificar aspectos en la formulación de las SVMR, sin perder sus ventajas.

Luego se describieron los fundamentos de los AG y PSO, se definieron sus principales elementos y se presentó el algoritmo de cada una de las técnicas mencionadas.

Se propusieron tres modelos de pronóstico, LS-SVM AG, LS-SVM PSO y LS-SVM AG+PSO, que optimizaron dos parámetros, el valor C, que compensa la maximización del margen y la reducción del error de la SVM, y el valor σ^2, que corresponde a una variable del kernel Gaussiano utilizado en los modelos.

De los 400 datos utilizados, que corresponden al precio de cierre diario de las acciones de LAN Chile, 350 fueron usados para el entrenamiento de los modelos, y 50 para el testing, este testing fue realizado con desfases de 2, 3 y 4 días.

En el modelo LS-SVM AG, el mejor resultado obtenido fue de 92,93% de la varianza explicada utilizando un desfase de 2 días en un tiempo de cómputo de 3 minutos 50 segundos, por otro lado el peor resultado fue de un 82,49% de la varianza explicada, con un desfase de 4 días en 8 minutos 23 segundos. Con esto se puede concluir que con 2 días de desfase se mejora el rendimiento del modelo en un 10,44% y es 4 minutos 33 segundos mas rápido que con un desfase de 4 días.

Para el modelo LS-SVM PSO, el mejor resultado que se obtuvo fue de un 94% de la varianza explicada, con un desfase de 2 días en 4 minutos 2 segundos, mientras que, el peor resultado fue de un 90,21%, con un desfase de 4 días en un tiempo de 12 minutos 3 segundos. Con lo que el mejor desempeño supera en un 3,79% y es 8 minutos mas rápido que el peor resultado.

El modelo híbrido LS-SVM AG+PSO obtuvo como mejor resultado un 93,33% de la varianza explicada con un desfase de 2 días en un tiempo de 14 minutos 7 segundos, por otro lado, el peor resultado es de 91,20% con un desfase de 3 días en 20 minutos 2 segundos. Por lo tanto, se puede concluir que la diferencia de desempeño es de un 2,13% y la diferencia de tiempo de cómputo es aproximadamente de 6 minutos entre el mejor y peor resultado obtenido por el modelo.

Es necesario destacar, en base a los resultados obtenidos, que independiente del modelo utilizado, los mejores desempeños y el menor tiempo de cómputo se obtuvieron con un desfase de 2 días en los datos.

Posterior al análisis de los distintos modelos propuestos, es posible concluir que, las Máquinas de soporte vectorial son capaces de presentar una buena calidad de pronóstico frente

a la necesidad de predecir el precio de las acciones de LAN Chile, presentándose el modelo LS-SVM PSO el mejor desempeño con un 94% de la varianza explicada.

Finalmente, se puede decir que los objetivos definido en la sección 1.2 se han cumplido cabalmente y se espera que esta investigación sirva como base para futuros estudios relacionados con el pronóstico de acciones, y que permita ampliar el campo de estudio a otras acciones del mercado local.

7. BIBLIOGRAFÍA.

[Abraham y Liu, 2006] A. Abraham, H. Guo, H. Liu, "Swarm Intelligence: Foundations, Perspectives and Applications", *Studies in Computational Intelligence (SCI)*, Springer-Verlag, vol. 26, pages 3-25, November 2006.

[Burges, 1998] Burges, C. "A Tutorial on support vector machines for pattern recognition", *Data Mining and Knowledge Discovery*, 1998.

[del Mar y Urdaneta, 2009] A. Del Mar, A. Urdaneta, P. Guillen, F. Hidrobo, "*Estimación de parámetros petrofísicos y máquinas soporte vectorial*", Cuarto Congreso Colombiano de Computación, 2009.

[Goldberg, 1989] D. Goldberg, "*Genetic Algorithms in search, Optimization, and machine learning*", Addison-Wesley, 1989.

[Haupt y Haupt, 1998], Haupt, "Practical Genetic Algorithms", second edition. John Wiley & Sons, 1998.

[http://www.esat.kuleuven.ac.be/sista/lssvmlab/] disponible via web. Revisada por última vez el 19 de septiembre de 2009.

[Kennedy y Eberhart, 1995] J. Kennedy, R.C. Eberhart, "Particle Swarm Optimization", *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pages 1942-1948, December 1995.

[Kennedy y Eberhart, 2001] J. Kennedy, R.C. Eberhart, Y. Shi, "Swarm Intelligence", *The Morgan Kaufmann Series in Artificial Intelligence*, Morgan Kaufmann, San Francisco - California, pages 3-80, April 2001.

[Korn y Korn, 2005] Korn, Elke. y Korn, Ralf. "*Modelo para el precio de las acciones*" , Universidad Técnica de Kaiserslautern, Facultad de matemáticas, 2005.

[Khosla y otros, 2006] A. Khosla, S. Kumar, K.K. Aggarwal, J. Singh, "Particle Swarm for Fuzzy Models Identification", *Studies in Computational Intelligence (SCI)*, Springer-Verlag, vol 26, pages 149-173, November 2006.

[Liu, Sun y XU, 2006]]J. Liu, J. Sun, W. Xu, "Quantum-Behaved Particle Swarm Optimization with AdaptiveMutation Operator", *Lecture Notes in Computer Science (LNCS)*, Springer-Verlag, vol. 4221, pages 959-967, September 2006.

[Miranda, Rey, Weber, 2005], J. Miranda, P. Rey, R. Weber, "Prediccion de fugas de clientes para una institución financiera mediante Support Vector Machine", Revista Ingeniería de sistemas, vol. 19, 2005.

[Pedroza, 2007] J. Pedroza, "*Aplicación de las máquinas de soporte vectorial al reconociemiento de hablantes",* Tesis maetría en ciencias y tecnologias de información, Universidad Autónoma Metropolitana, 2007.

[Pu, Fang y Liu, 2007] X. Pu, Z. Fang, Y. Liu, "*Multilayer Perceptron Networks Training Using Particle SwarmOptimization with Minimum Velocity Constraints*", *Lecture Notes in Computer Sciencie (LNCS)*, Springer-Verlag, vol. 4493, pages 237-245, Jule 2007.

[Smola, 1996] Smola, A. "*Regression Estimation with Support Vector Learning Machines"*, Technische Universitat Munchen, 1996.

[Vapnik, 1995] Vapnik, V. "*The Nature of Statistical Learning Theory"*, second edition, Springer, New York, 1995.

[Zhang y Xie, 2003] X. Xie, W. Zhang, "DEPSO: Hybrid Particle Swarm with Differential Evolution Operator", *IEEE Internat. Conf. on System*, vol. 4, pages 3816-3821, Octuber 2003.

[Suykens, J., A., Van Gestel, K., T., De Brabanter, J., De.Moor, B., Vandewalle, J.] "Least

Squares Support Vector Machines". World Scientific, Singapore, ISBN 981-238-151-1

(2002).

ANEXOS

A: Código fuente.

En este anexo se muestra el código fuente de los principales metodos utilizados en los algoritmos evolutivos, partiendo con los usados en AG y finalizando con los de PSO.

AG-Estructura y constructor clase Cromosoma:

```
public class Cromosoma {
//Los genes asociados a este cromosoma
    protected double [] genes;
    //Error asociado al cromosoma
    protected double errorDelCromosoma;
    //El porcentaje de mutación
    protected double porcentajeMutacion=0.8;
    //Cantidad de material genético a cortar
    protected int tamañoCorte;
    public JMatLink conMatlab;

//Constructor de la clase Cromosoma
Cromosoma(int numeroGenes){
        //generar al azar los genes del cromosoma
        genes = new double[numeroGenes];
        for(int i=0;i<genes.length;i++){
            if(Math.random()<0.5)
                genes[i]=0.0;
            else
                genes[i]=1.0;
        }

        errorDelCromosoma = 0.0;
        tamañoCorte = genes.length/2;
        }
```

AG-Reproducción (cruzamiento y mutación de individuos):

```
/**
* Este cromosoma sería la "madre" para realizar la reproducción
* @param padre Corresponde al cromosoma padre para realizar el cruzamiento
* @param descendiente1 Es el cromosoma a reemplazar de la generación anterior
* @param descendiente2 Es el cromosoma a reemplazar de la generación anterior
* @return La cantidad de mutaciones realizadas
*/
int reproduccion(Cromosoma padre, Cromosoma descendiente1, Cromosoma
descendiente2) {
        int puntoCorte = (int)
(0.999999*Math.random()*(double)(genes.length-tamañoCorte));
        double desc1 [] = new double[genes.length];
        double desc2 [] = new double[genes.length];
       //Cruzamiento 1
       for(int j=0;j<puntoCorte;j++){
           desc1[j]=this.obtenerGen(j);
           desc2[j]=padre.obtenerGen(j);
       }
       for(int j=puntoCorte;j<genes.length/2;j++){
           desc1[j]=padre.obtenerGen(j);
           desc2[j]=this.obtenerGen(j);
       }
```

```
       puntoCorte = (int) (0.999999*Math.random()*(double)(genes.length-
tamañoCorte))+genes.length/2;
       //Cruzamiento 2
       for(int j=genes.length/2;j<puntoCorte;j++){
           desc1[j]=this.obtenerGen(j);
           desc2[j]=padre.obtenerGen(j);
       }
       for(int j=puntoCorte;j<genes.length;j++){
           desc1[j]=padre.obtenerGen(j);
           desc2[j]=this.obtenerGen(j);
       }
       int mutate=0;
       //Mutación
       //Descendiente 1
      if ( Math.random() < porcentajeMutacion ) {
              //Primera mutación
              int iswap = (int)
(0.999999*Math.random()*(double)(genes.length/2));
              if(genes[iswap]==0)
                    desc1[iswap]=1.0;
              else
                    desc1[iswap]=0.0;

              //Segunda mutación
              iswap = (int)
(0.999999*Math.random()*(double)(genes.length/2))+genes.length/2;
              if(genes[iswap]==0)
                    desc1[iswap]=1.0;
              else
                    desc1[iswap]=0.0;
              mutate++;
        }

        //Mutacion
        //Descendiente 2
        if ( Math.random() < porcentajeMutacion ) {
              //Primera mutación
              int iswap = (int)
(0.999999*Math.random()*(double)(genes.length/2));
              if(genes[iswap]==0)
                    desc2[iswap]=1.0;
              else
                    desc2[iswap]=0.0;

              //Segunda mutación
              iswap = (int)
(0.999999*Math.random()*(double)(genes.length/2))+genes.length/2;
              if(genes[iswap]==0)
                    desc2[iswap]=1.0;
              else
                    desc2[iswap]=0.0;
              mutate++;
        }

        // copiar resultados
        descendiente1.setGenes(desc1);
        descendiente2.setGenes(desc2);
        return mutate;
```

```
}
```

AG- Inicio Algoritmo Genético:

```
//Inicio del algoritmo genético
  public void start(){
        cromosomas = new Cromosoma[TAMAÑO_POBLACION];
        arregloRuleta= new int[TAMAÑO_RULETA];
        for (int i=0;i<TAMAÑO_POBLACION;i++) {
            //Cantidad de genes del cromosoma
            System.out.println("Creando cromosoma numero: "+i);
            cromosomas[i] = new Cromosoma(20);
            cromosomas[i].set_jmatlink(this.conMatlab);
            cromosomas[i].calcularErrorCromosoma(entrada,ideal);
            Cromosoma c = cromosomas[i];
            System.out.println("RMSE Cromosoma: "+i+" es:
"+cromosomas[i].obtenerErrorCromosoma());
        }
        Cromosoma.ordenarCromosomas(cromosomas,cromosomas.length);
        for(int i=0;i<TAMAÑO_POBLACION;i++){
            cromosomas[i].calcularErrorCromosoma(entrada,ideal);
            double error = cromosomas[i].obtenerErrorCromosoma();
            System.out.println("Cromosoma ordenado #"+i+" : "+error);
        }
  }
```

AG-Calcular error asociado al cromosoma:

```
public void calcularErrorCromosoma(double entradas[][],double ideal[][]){

        double pronostico[][]=new double[entradas.length][1];
        INFO rmseCromosoma=new INFO();
        //Se transformar el parametro C y ro de binario a real
        String genBinario = Integer.toString((int)Math.floor(genes[0]));
        String aux;
        for(int x=1;x<=9;x++){
            aux = Integer.toString((int)Math.floor(genes[x]));
            genBinario = genBinario+aux;
        }
        int C = Integer.parseInt(genBinario,2);
        genBinario = Integer.toString((int)Math.floor(genes[10]));
        String aux2;
        for(int x=11;x<=15;x++){
            aux = Integer.toString((int)Math.floor(genes[x]));
            genBinario = genBinario+aux;
        }
        double ro = Integer.parseInt(genBinario,2);
        ro = ro/100;
        //llamar LS-SVMlab para entrenar máquina y luego calcular error del
cromosoma
        conMatlab.engEvalString("gam = "+C+";");
        conMatlab.engEvalString("sig2 = "+ro+";");
        conMatlab.engEvalString("type = 'function estimation';");
        conMatlab.engEvalString("[alpha,b] =
trainlssvm({X,Y,type,gam,sig2,'RBF_kernel','original'});");
```

```
        conMatlab.engEvalString("Yt =
simlssvm({X,Y,type,gam,sig2,'RBF_kernel','original'},{alpha,b},X);");
        pronostico = conMatlab.engGetArray("Yt");
        errorDelCromosoma= rmseCromosoma.RMSE(ideal, pronostico);
}
/**Obtener el error que permitirá evaluar el rendimiento del cromosoma
* @return El error asociado al cromosoma
*/
```

PSO-Estructura, constructor e inicialización clase Enjambre:

```
public class Enjambre {
    private int N;
    private double w;
    private double c1;
    private double c2;
    private double Vmax;
    private double Xmax;
    private Particula[] P;
    private double[] Gbest;
    private double E; //error asociado al Gbest
    private final JMatLink conMatlab;
    protected double entrada[][];//Datos de Entrenamiento
    protected double ideal[][]; //resultados ideales para entrenamiento
    Enjambre(JMatLink conMatlab, int N, double w, double c1, double c2,
double Vmax, double Xmax){
        this.conMatlab = conMatlab; //parametro para hacer oncexion con
Matlab
        this.N = N; //numero de particulas
        this.w = w; //peso de inercia
        this.c1 = c1; //factor cognitivo
        this.c2 = c2; //factor social
        this.Vmax = Vmax; //velocidad maxima
        this.Xmax = Xmax; //posicion maxima
        this.E = 1; //error inicial enjambre
        P = new Particula[N]; //particulas del enjambre
    }

public void inicializar(){
    double[] parametros = new double [2];
        for(int i=0; i<N; i++){
            parametros[0]=Math.random();
            parametros[1]=Math.random()*4000;
            P[i] = new Particula();
            P[i].inicializar(parametros,Vmax);
            //calculo error inicial
            //Se entrena la máquina con los parámetros obtenidos por PSO
            //error inicial partícula
            P[i].setError(calculoError(P[i],entrada,ideal));
            //mejor error inicial partícula
            P[i].setBestError(P[i].getError());
        }
        //primer mejor error default es P[0
        Gbest = P[0].getPosicion();]
        E = 1;

    }
```

PSO-iterar:

```
    public void iterar() {
        double Eit = 0;
        for(int i=0; i<N; i++){
            P[i].actualizar(w,c1,c2,Vmax,Xmax,Gbest); //actualiza X,V
            //calcula error de particula
            Eit = calculoError(P[i],entrada,ideal);
            //nuevamente busca error
            if(Eit < P[i].getBestError()){
                P[i].setPbest(P[i].getPosicion());
                P[i].setBestError(Eit);
            }
            if(Eit < E){
                this.Gbest = P[i].getPosicion();
                this.E = Eit;
            }
        }
    }
```

PSO-Estructura, constructor e inicialización clase Partícula:

```
public class Particula {

    private double[] X;
    private double[] V;
    private double[] Pbest;
    private double E;
    private double Ebest;

    public Particula(){
    }
    public void inicializar(double[] X, double Vmax){

        double rand;
        this.X = X; //posiciones iniciales
        this.Pbest = X; //mejor posicion inicial
        V = new double[X.length]; //se genera vector de velocidad
        //velocidades iniciales
        for(int i=0; i<X.length; i++){
            rand = Math.random();
            V[i] = rand*Vmax + (1-rand)*(-Vmax);
        }
    }
```

PSO-Actualiza posición y velocidad de partícula:

```
public void actualizar(double w, double c1, double c2, double Vmax, double
Xmax, double[] Gbest){

        for(int i=0; i<X.length; i++){

            V[i] = w*V[i] + c1*Math.random()*(Pbest[i] - X[i]) +
c2*Math.random()*(Gbest[i]-X[i]);

            if(V[i] > Vmax) V[i] = Vmax;
            if(V[i] < -Vmax) V[i] = -Vmax;
            X[i] = X[i] + V[i];
            if(X[i] > Xmax) X[i] = Xmax;
```

```
            if(X[i] < -Xmax) X[i] = -Xmax;
        }
    }
```