Máquinas de soporte vectorial y redes neuronales artificiales en la predicción del movimiento USD/COP *spot* intradiario

Nicolás Sánchez Anzola*

* Magíster en Finanzas (Enfoque Mercado de Capitales), Universidad Externado de Colombia. Bolsa de Valores de Colombia. nicosancheza@hotmail.com
Fecha de recepción: 04 de abril de 2015.
Fecha de aceptación: el 01 de mayo de 2015.
Forma de citar
Sánchez Anzola, N. (2015). Máquinas de soporte vectorial y redes neuronales artificiales en la predicción del movimiento USD/COP spot intradiario. *ODEON*, 9, pp. 113-172. DOI: http://dx.doi.org/10.18601/17941113.n9.04

Introducción

La predicción del precio y movimiento de las tasas de cambio se considera un importante problema económico. A pesar de que se han desarrollado numerosas investigaciones empíricas asociadas a la predicción del movimiento de divisas, acciones, índices accionarios, materias primas, entre otros, la mayoría de los hallazgos están asociados con mercados desarrollados. Sin embargo, existe un limitado número de investigaciones en la literatura que intentan predecir el movimiento de las tasas de cambio en los mercados emergentes.

En la última década, se han venido desarrollando nuevas metodologías y modelos de inteligencia artificial y modelos estadísticos, contemplados en un campo multidisciplinar conocido como "el aprendizaje de máquina". Estos modelos han demostrado ser efectivos y aplicables en campos como la medicina, la biología, el campo financiero y económico, los sectores empresarial y bancario, el medioambiente, el sector público, entre otros. Más recientemente, estos modelos de inteligencia computacional se han empezado a implementar en la predicción de series de tiempo, una tarea más compleja en la medida que los datos disponibles tienen ruido, características no estacionarias y dimensionalidad compleja (Kim, 2003a).

Dado que el mercado de capitales es esencialmente dinámico, no lineal, complicado, no paramétrico y caótico en la naturaleza (Kara, Acar Boyacioglu y Baykan, 2011b; Karazmodeh, Nasiri y Hashemi, 2013; Tan, Quek y Ng, 2004), la predicción del movimiento del precio *spot* intradiario del USD/COP (dólar americano/peso colombiano) resulta ser una tarea retadora y desafiante en el análisis de series de tiempo. Los precios se ven afectados por variedad de componentes fundamentales y macroeconómicos como son factores políticos, políticas de empresas y entes nacionales, condiciones económicas del país, expectativas de los inversionistas, factores psicológicos de los inversionistas, movimiento de otro mercado ante la coyuntura económica global, entre otras.

Durante las últimas décadas, la representación de relaciones económicas no lineales ha sido un problema de investigación muy importante, lo que ha llevado a que la econometría haya acogido variedad de modelos y técnicas, cuyos orígenes se encuentran en la estadística y la inteligencia artificial. Tanto los modelos de redes neuronales artificiales (RNA) como las máquinas de soporte vectorial (SVM) han venido ganando terreno, debido a su capacidad para clasificar datos y representar relaciones desconocidas a partir de los mismos datos (relaciones no lineales), en aquellos casos donde la economía no brinda suficientes elementos para establecer las relaciones. Por otra parte, la predicción de tasas de cambio ha sido reconocida

como un importante problema económico, en especial por la dificultad para realizar predicciones tanto de corto como de largo plazo, entendiendo una predicción como la pretensión de encontrar aquellos modelos que permitan pronosticar con mayor precisión el valor futuro del activo, en periodos intradiarios o mayores.

El mercado cambiario colombiano presenta muchas oportunidades para inversiones especulativas por su alta volatilidad y continuo movimiento en las tasas de cambio. Por tal motivo, la implementación de modelos de inteligencia computacional (p. ej. redes neuronales artificiales y máquinas con vectores de soporte) que han resultado ser exitosos en la predicción de otros mercados cambiarios como los del EUR/USD (Fletcher y Shawe-Taylor, 2010; Papadimitriou, Gogas y Plakandaras, n.d.; Theofilatos y Karathanasopoulos, 2012), USD/GBP (Cao, Pang y Bai, 2005), AUD/USD, AUD/GBP AUD/JPY, AUD/NZD, AUD/CHF (Kamruzzaman, Sarker y Ahmad, n.d.), USD/JPG USD/CHF y USD/DM (Cao *et al*., 2005), es una tarea trascendente y aplicable en Colombia.

El objetivo principal del presente trabajo es diseñar, implementar y comparar un modelo de red neuronal artificial (red multicapa) y un modelo de máquina de soporte vectorial (máquinas de Kernels), en la predicción (para intervalos de 10 minutos) de los movimientos alcistas y bajistas del mercado *spot* intradiario del USD/COP. La mayor contribución de este trabajo es demostrar y verificar la tasa de predicción de la dirección del movimiento intradiario del mercado *spot* USD/COP aplicando RNA y SVM y comparando el rendimiento de estas dos técnicas.

El presente artículo se divide en cinco secciones, siendo esta introducción la primera. Los capítulos 2 y 3 presentan los modelos teóricos de las redes neuronales artificiales (RNA) y las máquinas de soporte vectorial (SVM). En el capítulo 4 se plasman las bases para desarrollar los experimentos y finaliza con los resultados, las principales observaciones y las comparaciones de los modelos. Por último, el capítulo 5 recoge las conclusiones finales.

1. Redes neuronales artificiales

Si se tuviera que definir la principal característica que separa al ser humano del resto de animales, seguramente la respuesta sería su capacidad de raciocinio. Por tal motivo, esta capacidad ha hecho que la tecnología se haya orientado a descubrir su origen: ¿Cómo funciona el cerebro humano? ¿Qué es la memoria y donde se produce? ¿Se pueden construir modelos artificiales que lo emulen? ¿Se pueden crear máquinas inteligentes? Este cuestionamiento llevó a los científicos e investigadores a desarrollar y entender durante muchos años el campo de la inteligencia artificial.

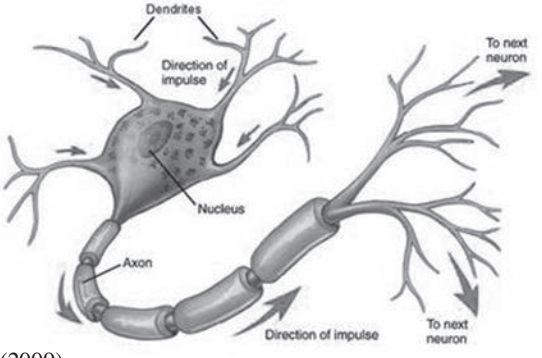
"Una red neuronal artificial es un conjunto de algoritmos matemáticos que procesan información y encuentran relaciones no lineales entre el conjunto de datos, y cuya unidad básica de procesamiento está inspirada en la célula fundamental del sistema nervioso humano: la neurona" (Caicedo y López, 2009).

La capacidad del cerebro humano de pensar, recordar, relacionar hechos y resolver problemas ha inspirado a muchos científicos a intentar modelar su funcionamiento. Es así que las RNA tratan de emular ciertas características propias de los cerebros humanos, como la capacidad de memorizar y de asociar hechos. Es importante tener en cuenta que todos aquellos problemas que no pueden expresarse a través de un algoritmo, el hombre es capaz de resolverlos acudiendo a algo denominado la experiencia. Por tanto, las redes neuronales son modelos artificiales y simplificados del cerebro humano, un sistema que es capaz de adquirir conocimiento a través de la experiencia (Caicedo y López, 2009).

1.1. La neurona biológica

La teoría y el modelo de las redes neuronales artificiales están inspirados en la estructura y el funcionamiento del sistema nervioso y, en particular, de la neurona biológica. Una neurona es una célula viva y contiene los mismos elementos que forman parte de todas las células biológicas. En general, una neurona consta de un cuerpo celular relativamente esférico, de entre 5 y 80 micrómetros de longitud (millonésima parte de un metro), del que surge un denso árbol de ramificaciones (árbol dendrítico) compuesto por las dendritas, y del cual parte una fibra tubular denominada axón (cuya longitud varía desde los 100 micrómetros), que también se ramifica en su extremo final para conectar con otras neuronas (Del Brio y Sanz, 2006) (figura 1).

Figura 1. Modelo de neurona artificial



Fuente: Caicedo y López (2009).

Una de las características que diferencian a las neuronas del resto de células vivas, es su capacidad de comunicación. Las dendritas y el cuerpo celular reciben señales de entrada, el cuerpo celular las combina e integra y luego se emiten señales de salida. El axón transporta esas señales a sus terminales, los cuales se encargan de distribuir la información a un nuevo conjunto de neuronas. La unión entre dos neuronas se denomina sinapsis. La sinapsis es direccional y la información fluye en un solo sentido.

Las señales en la neurona biológica son de naturaleza eléctrica y química. La señal generada por medio de la neurona y que es transportada a lo largo del axón es de categoría eléctrica, mientras que la señal que se transmite entre los terminales del axón de una neurona y las dendritas de las neuronas siguientes es de origen químico (Hilera, 1994).

Para establecer las similitudes directas entre la actividad de una neurona biológica y una artificial se deben abordar los siguientes aspectos funcionales (Caicedo y López, 2009):

a. **Las neuronas reciben señales de entrada:** una de las características principales de las neuronas es que están altamente conectadas con otras neuronas de las cuales reciben un estímulo de cualquier evento que está ocurriendo o de señales eléctricas con la información aprendida. Este proceso hace que tengan una gran capacidad de procesamiento de información y realización de tareas de alta complejidad.

b. **Las señales pueden ser modificadas por los pesos sinápticos:** la comunicación entre neuronas no se hace por contacto directo sino por algo denominado sinapsis. La sinapsis es un espacio que está ocupado por sustancias químicas denominadas neurotransmisoras, las cuales se encargan de bloquear o dejar pasar las señales que provienen de las otras neuronas. Las neuronas que envían la información se conocen como presinápticas y las neuronas que reciben la señal pos-sinápticas.

c. **Los elementos de proceso suman las entradas afectadas por las sinapsis:** las neuronas reciben señales eléctricas de otras neuronas con las que tienen contacto, y estas señales se acumulan en el cuerpo de la neurona para definir qué hacer.

d. Bajo una circunstancia definida, la neurona transmite una señal de salida: si el total de la señal eléctrica que recibe una neurona es suficientemente grande, se puede vencer el potencial de acción permitiendo que la neurona se active o, por el contrario, permanezca inactiva.

e. La salida puede ir a muchas neuronas: cuando se activa una neurona, esta se encuentra en capacidad de transmitir un impulso eléctrico a las otras neuronas con las cuales tiene contacto. El nuevo impulso actúa ahora como entrada para otras neuronas o como estímulo en algún músculo.

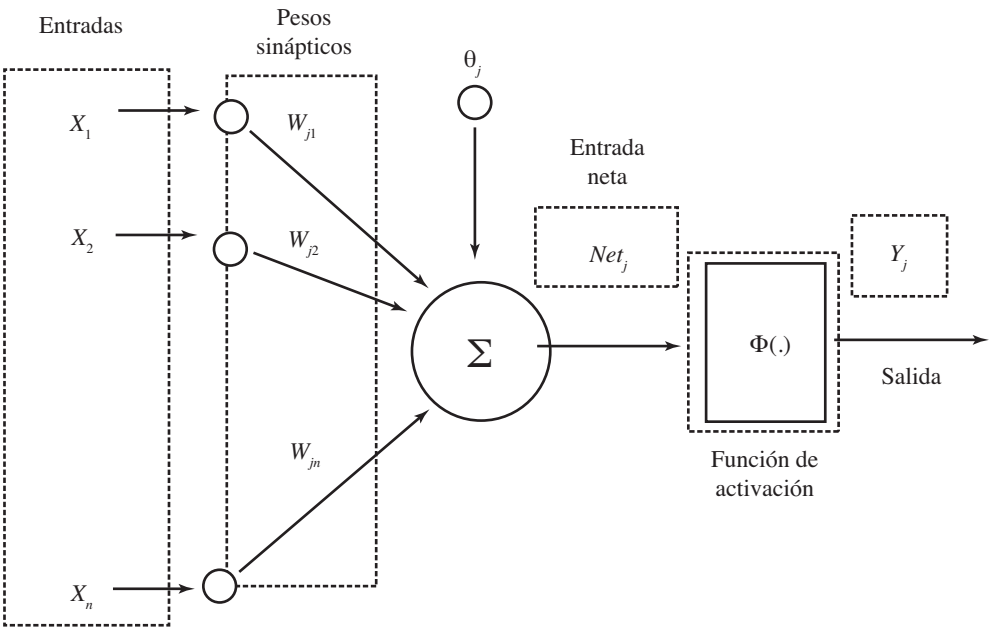
1.2. La neurona artificial

Como en el caso de la neurona biológica, la neurona artificial recibe unas entradas de estímulo que pueden venir del exterior o de la conexión con otras neuronas. La información de entrada que recibe la neurona se puede definir con el vector de entradas:

$$X = [x_1, x_2, \ldots x_n]$$

Las entradas que recibe la neurona son modificadas por un vector *w* de pesos sinápticos, cuyo papel es el de simular la sinapsis que realizan las neuronas biológicas. Sea el parámetro θ_j el umbral de una neurona, los diferentes valores que recibe la neurona modificada por los pesos sinápticos, se suman para producir lo que se denomina la entrada neta, la cual determina si la neurona se activa o se mantiene inactiva. La activación o no de esta depende de lo que se denomina función de activación; para ello, la entrada neta se evalúa en esta función y se obtiene la salida de la red. La salida y_j de la neurona se genera al evaluar la entrada neta en la función de activación y esta se puede propagar hacia otras neuronas o resultar la salida final de la red. El uso de la función de activación es una creación matemática para poder aplicar las RNA a diversidad de problemas reales. Por ejemplo, si se define la función como un escalón unitario, la salida será 1, si la entrada neta es mayor que 0, en caso contrario, la salida será 0. Dicho proceso se ilustra en la figura 2.

Figura 2. Modelo de neurona artificial



Fuente: Caicedo y López (2009).

En la neurona artificial, el cálculo de la entrada se puede representar de la siguiente manera: ecuación 1, o en forma vectorial, ecuación 2.

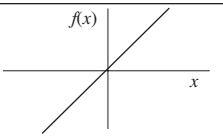
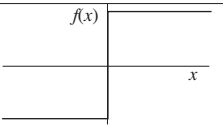
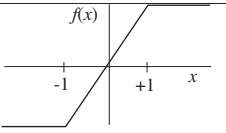
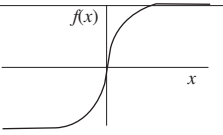
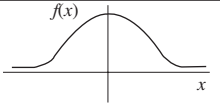
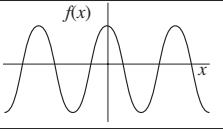
$$Net_j = \sum_{i=1}^{N} x_i\, w_{ji} + \theta_j \quad (1)$$

$$+ w_2 x_{j2} + \ldots + w_i x_{ji} + \ldots + w_N x_{jN} + \theta_j$$

$$Net_j = w^T X_j + \theta_j \quad (2)$$

La salida de la neurona artificial está determinada por la función de activación (figura 3):

Figura 3. Principales funciones de activación

	Función	Rango	Gráfica
Identidad	$y = x$	$[-\infty,+\infty]$	
Escalón	$y = sign(x)$ $y = H(x)$	$\{-1,+1\}$ $\{0,+1\}$	
Lineal a tramos	$y = \begin{cases} -1, si\ x < -l \\ x, si\ +l \le x \le -l \\ +1, si\ x > +l \end{cases}$	$[-1,+1]$	
Sigmoidea	$y = \frac{1}{1+e^{-x}}$ $y = tgh(x)$	$[0,+1]$ $[-1,+1]$	
Gaussiana	$y = Ae^{-Bx^2}$	$[0,+1]$	
Sinusidal	$y = A\ sen(\omega x + \varphi)$	$[-1,+1]$	

Fuente: Hilera (1994).

Generalmente, la función de activación puede ser de tipo escalón, lineal, sigmoidal y gaussiana. La figura 3 ilustra las principales funciones de activación, sus intervalos y las gráficas correspondientes.

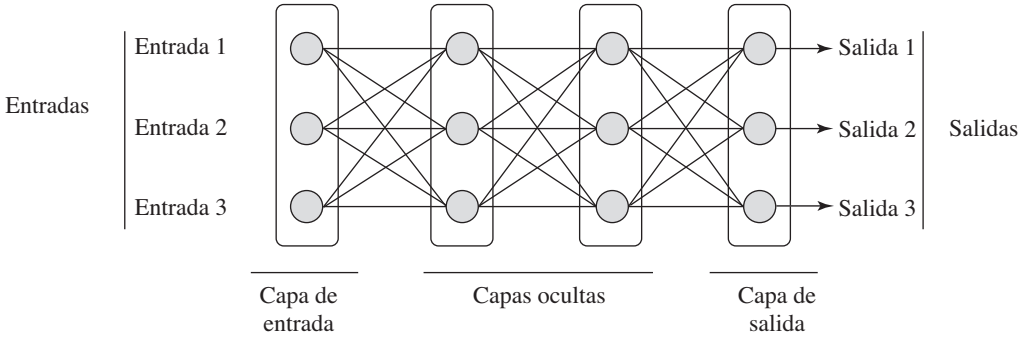
1.3. Red neuronal artificial

La neurona artificial por sí sola posee baja capacidad de procesamiento y su nivel de aplicabilidad es bajo. El verdadero potencial de las neuronas radica en la interconexión entre ellas, tal como sucede con las neuronas biológicas en el cerebro humano. Dadas estas premisas, diferentes investigadores han propuesto variedad de estructuras de conectividad de las neuronas artificiales dando lugar a lo que se conoce como redes neuronales artificiales. En términos generales, una red neuro-

nal artificial es un conjunto de algoritmos matemáticos que procesan información y encuentran relaciones no lineales entre el conjunto de datos, en un intento por emular el comportamiento de las redes neuronales biológicas. La distribución de neuronas dentro de una red neuronal artificial se realiza formando niveles o capas de un número determinado de neuronas cada una (figura 4). A partir de su situación dentro de la red, se pueden distinguir tres tipos de capas:

– **Capas de entrada:** es la capa que recibe directamente la información de las fuentes externas de la red. Capa que recibe las variables de entrada.
– **Capas ocultas:** son internas a la red y no tienen contacto directo con el entorno exterior. El número capas ocultas puede ser mayor a uno y las neuronas de las capas ocultas pueden estar interconectadas de distintas maneras, lo que determina, junto con su número, las tipologías de redes.
– **Capas de salida:** es el conjunto de neuronas que transfieren la información que la red ha procesado hacia el exterior.

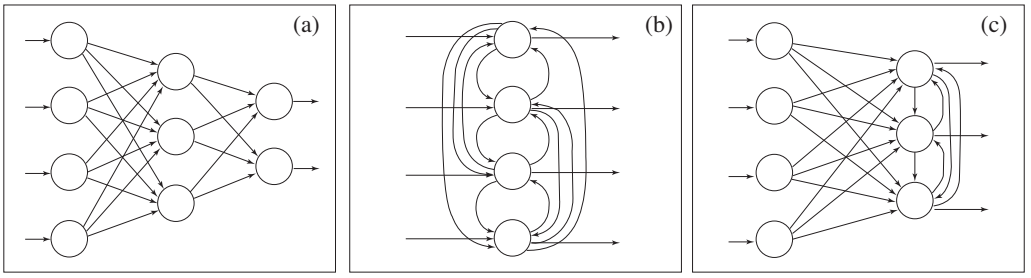
Figura 4. Modelo de red neuronal artificial



Fuente: gráfica ilustrativa realizada por el autor.

La conexión entre las neuronas está relacionada con la forma en que las salidas de estas se encuentran conectadas para ser entradas de otras nuevas neuronas. La señal de salida de una neurona puede ser la entrada para otra o una entrada a sí misma. Cuando ninguna salida de las neuronas es entrada de neuronas del mismo nivel o de niveles siguientes, la red se describe como propagación hacia adelante (*feedforward*). Sin embargo, cuando las salidas se conectan como entradas de neuronas de niveles previos o del mismo nivel, incluyéndose ellas mismas, la red es de propagación hacia atrás (feedback) (Del Brio y Sanz, 2006). La figura 5 ilustra las conexiones hacia adelante (a), laterales (b) y hacia atrás (c) en una red neuronal.

Figura 5. Tipos de conexiones en una red neuronal



Fuente: Hilera (1994).

La arquitectura de una RNA es la forma como se organizan las neuronas en su interior y está ligada al algoritmo de aprendizaje utilizado para entrenar la red neuronal. El número de capas de la red define si son de tipo monocapa o multicapa. Por tanto, los parámetros principales de una red neuronal son: el número de capas, el número de neuronas por capa, el grado de conectividad y el tipo de conexiones entre neuronas. En las redes monocapa, se establecen conexiones entre las neuronas que pertenecen a la única capa que constituye la red. El nombre de "red monocapa" se debe a que en este tipo de arquitectura solo se dispone de una capa de procesamiento de los datos hasta llegar a la salida de la red. Las redes multicapa, por el contrario, son aquellas que disponen de conjuntos de neuronas agrupadas en varios niveles o capas, normalmente entre 2 capas o más a la capa de salida, teniendo conectividad total entre neuronas.

1.3.1. Mecanismos de aprendizaje en las RNA

Biológicamente, la información memorizada en el cerebro está más relacionada con los valores sinápticos de las conexiones entre las neuronas que con las mismas neuronas, es decir, el conocimiento se encuentra en la sinapsis entre neuronas. En las redes neuronales artificiales, el conocimiento de la red está representado en los pesos de las conexiones entre las neuronas y todo proceso de aprendizaje requiere un cierto cambio en la ponderación de estas, por lo cual la red obtiene su proceso de aprendizaje modificando los pesos sinápticos de las neuronas. Para modelar el proceso de aprendizaje en las RNA, se puede plantear la siguiente ecuación:

$$w(t+1) = w(t) + \Delta w(t) \quad (3)$$

Donde:

$w(t + 1)$: valor actualizado del peso sináptico
$w(t)$: valor actual del peso sináptico
$\Delta w(t)$: variación del peso sináptico

Un aspecto importante y que se puede utilizar para diferenciar las reglas de aprendizaje se basa en los aprendizajes *on line* y *off line*. Cuando el entrenamiento es *off line*, se distingue entre una fase de entrenamiento y una fase de funcionamiento, existiendo un set de datos de entrenamiento y un conjunto de datos de test o prueba que serán utilizados en la correspondiente fase. En este tipo de aprendizaje, los pesos de las conexiones son fijas después de que termina la etapa de entrenamiento de la red. Estos sistemas no presentan problemas de estabilidad en su funcionamiento debido a su carácter estático. Por otro lado, en el aprendizaje *on line* no se distingue entre la fase de entrenamiento y prueba, por lo cual los pesos varían siempre que se presente nueva información al sistema (Ruiz y Matich, 2001).

Otro aspecto fundamental en el aprendizaje de las RNA es conocer cómo se modifican los valores de los pesos en la red y qué criterios se toman para que se realicen modificaciones a las conexiones cuando ingresa nueva información. Estos criterios son la base de lo que se denomina regla de aprendizaje de la red. Existen dos tipos de reglas: las redes que responden a un aprendizaje supervisado y las que responden a uno no supervisado. La diferencia fundamental entre ambos es si existe o no un agente externo supervisor que controle el proceso de aprendizaje de la red (Del Brio y Sanz, 2006).

El aprendizaje supervisado se caracteriza porque el proceso de aprendizaje se realiza mediante un entrenamiento controlado por un agente externo denominado supervisor externo que determina la respuesta que debería generar la red a partir de unas entradas. El supervisor comprueba la salida de la red y, en el caso que esta no coincida con la salida deseada, se procede a modificar los pesos de las conexiones, con el fin de conseguir que la salida obtenida se ajuste o se aproxime a la deseada. El aprendizaje por corrección de error (aprendizaje supervisado) consiste en ajustar los pesos de las conexiones de la red en función de la diferencia entre los valores deseados y los obtenidos en la salida de la red. Una regla simple de aprendizaje por corrección de error es la siguiente:

$$\Delta w_{ji} = \alpha y_i (d_j - y_j) \quad (4)$$

Donde:

Δw_{ji}: variación en el peso de la conexión entre las neuronas *i* y *j*

$\Delta w_{ji} = w_{ji\ actual} - w_{ji\ anterior}$

y_i: Valor de salida de la neurona *i*
d_j: Valor de salida deseado por la neurona *j*
y_j: Valor de salida obtenido en la neurona *j*
α: Factor de aprendizaje ($o < \alpha \leq 1$) que regula la velocidad del aprendizaje

Por lo general, la red neuronal debe aprender todo el conjunto de patrones de entrenamiento y por ello no debe entrenarse utilizando un error local, sino que el aprendizaje se hace en términos de un error global. El error global se define como el error cuadrático medio. Este se entiende como el error que produce la red en sus diferentes neuronas de salida ante todos los patrones de aprendizaje que se estén utilizando para el proceso de entrenamiento (Caicedo y López, 2009).

$$Error\ Global = \frac{1}{2P}\sum_{K=1}^{p}\sum_{j=1}^{N}\left(y_j^{(k)} - d_j^{(k)}\right)^2 \quad (5)$$

Donde:

N: número de neuronas en la capa de salida
P: número de patrones de entrenamiento (información que debe aprender la red)

$\frac{1}{2}\sum_{j=1}^{N}\left(y_j^k - d_j^k\right)^2$: Error cometido en el aprendizaje de la información *k* - ésima

Por tanto, el proceso anterior busca encontrar unos pesos para las conexiones de la red que minimicen la función de error. El ajuste de los pesos se puede hacer de forma proporcional a la variación relativa del error que se obtiene al variar el peso correspondiente:

$$\Delta w_{ji} = k\frac{\partial Error\ Global}{\partial w_{ji}} \quad (6)$$

El otro tipo de aprendizaje es el no supervisado. Este se diferencia del supervisado simplemente por la eliminación del agente externo en el proceso de aprendizaje. Este tipo de aprendizaje no necesita influencia externa para ajustar los pesos de las conexiones entre las neuronas. Es decir, no hay un agente externo que le diga si la salida generada en respuesta a una entrada determinada es o no correcta. Este tipo de redes deben encontrar las características, correlaciones y regularidades que se le presentan en los datos de entrada. Para este tipo de aprendizaje, la salida de la red representa el grado de similitud entre la información que se le presenta a la entrada y la información que se le ha presentado en el pasado. En general, se pueden considerar dos tipos de algoritmos de aprendizaje no supervisado: aprendizaje hbbiano y aprendizaje competitivo y cooperativo (Del Brio y Sanz, 2006).

El Perceptrón fue el primer modelo de RNA presentado a la academia por el psicólogo Frank Rosenblatt en 1958. Este tipo de red causó interés en la década de los sesenta, debido a su capacidad para aprender a reconocer patrones sencillos. Un Perceptrón, formado por varias neuronas lineales que reciben las entradas a la red y una neurona de salida, era capaz de decidir cuándo una entrada que era presentada a la red pertenecía a una de las dos clases que era capaz de reconocer. El Perceptrón es una red monocapa, posee conectividad total y, por lo general, se le implementan umbrales para que la superficie de separación no se quede anclada en el origen del espacio n-dimensional en donde se esté realizando la separación lineal de los datos, ya que existen algunos problemas de clasificación de patrones que sin el umbral no tendrían solución. La única neurona de salida del Perceptrón realiza la suma ponderada de las entradas, resta el umbral y pasa el resultado a una función de transferencia de tipo escalón. La regla de decisión es responder +1 si el patrón presentado pertenece a la clase A, o -1 si el patrón pertenece a la clase B. La salida depende de la entrada neta (Sumatoria de las entradas ponderadas) y del valor del umbral (Hilera, 1994).

El Perceptrón aplica un algoritmo de aprendizaje supervisado, lo cual significa que sus resultados deben ser evaluados y se realizan las modificaciones respectivas al sistema, si es necesario. Los valores de los pesos de la red pueden determinar como tal el funcionamiento y aprendizaje de la red, por lo que esos valores pueden fijarse o adaptarse utilizando diferentes algoritmos de entrenamiento. Los perceptrones pueden ser usados como máquinas de aprendizaje pero no pueden aprender a realizar todo tipo de clasificaciones. Esta limitación se debe a que el Perceptrón usa un separador lineal como método de decisión, con lo cual no es posible realizar sino una sola separación lineal, por medio del hiperplano. Uno de los principales

inconvenientes que tiene el Perceptrón es su incapacidad para separar regiones que no sean linealmente separables (Hilera, 1994).

1.4. Perceptrón multicapa y algoritmo *backpropagation*

La imposibilidad que tenían las redes tipo Perceptrón de solucionar problemas de clasificación no lineales, hizo que el interés por las redes neuronales artificiales decayera. Rosenblatt ya intuía la manera de solucionar estos problemas y planteó que un Perceptrón multicapa podía hacerlo, dado que de esa manera era posible obtener regiones más complejas, debido a una estructura de red que incluye, además de las capas de entrada y salida, una o más capas internas u ocultas. Muchos años pasaron para poder responder a las preguntas planteadas por Rosenblat de cómo incorporar una nueva capa a la red neuronal, y la tarea más compleja fue la de encontrar la manera de calcular el error de la nueva capa intermedia (capa oculta) para modificar los pesos sinápticos.

A mediados de los años setenta, el científico norteamericano Paul Werbos, en su tesis doctoral, propuso el algoritmo *backpropagation*, el cual permite entrenar al Perceptrón multicapa y posibilita su aplicación a la solución de problemas complejos. A diferencia de las anteriores arquitecturas de RNA, la estructura de red multicapa (Multi Layer Perceptrón - MLP), posee al menos tres niveles de neuronas: una capa de entrada, una capa oculta y una capa de salida. Existen algunas MLP que tienen más de una capa oculta en la red, pero no es muy recomendado dado que aumenta la complejidad computacional del algoritmo de aprendizaje. Una MLP tiene conectividad total y sus conexiones son *feedforward*. Por lo general, se implementan umbrales con el objetivo de hacer que la superficie de separación no se quede anclada al origen del espacio n-dimensional en donde se realiza la clasificación. La función de activación para las MLP son en la mayoría de casos sigmoidales o lineales.

El algoritmo *backpropagation* funciona de la siguiente manera: el algoritmo tiene dos fases, una hacia adelante y una hacia atrás. En la primera fase, el patrón de entrada se presenta a la red y se propaga a través de las capas hasta llegar a la capa de salida, obteniéndose los valores de salida de la red. La segunda fase inicia cuando se comparan los valores obtenidos con los valores de salida esperados para así obtener el error. La segunda fase transmite hacia atrás el error, a partir de la capa de salida, hacia todas las neuronas de la capa intermedia que contribuyan directamente a la salida, recibiendo el porcentaje de error aproximado a la partici-

pación de la neurona intermedia en la salida original. Este proceso se repite, capa por capa, hasta que todas las neuronas de la red hayan recibido un error que describa su aportación relativa al error total. Basándose en el valor del error recibido, se reajustan los pesos de conexión de cada neurona, de manera que la siguiente vez que se presente el mismo patrón, la salida esté más cercana a la deseada y, por tanto, disminuya el error (Caicedo y López, 2009).

El algoritmo finaliza cuando se verifica su condición de parada, ya sea porque el error calculado de la salida es inferior al permitido, o porque se ha superado el número de iteraciones, por lo cual se considera que se deberán hacer ajustes al diseño de la red, pues la solución no converge, o se debe ampliar el número de iteraciones. Generalmente, la función que se utiliza es sigmoidal.

A continuación se define la notación utilizada para revisar la formulación matemática del algoritmo *backpropagation* (Caicedo y López, 2009; Del Brío y Sanz, 2006; Hilera, 1994). Siendo,

x_p: patrón o vector de entrada

x_{pi}: entrada i-ésima del vector de entrada x_p

N: dimensión del vector de entrada

P: número de ejemplos, vectores de entrada y salidas diferentes

L: número de neuronas de la capa oculta: h

M: número de neuronas de la capa de salida, dimensión del vector de salida

w_{ji}^{h}: peso de interconexión entre la neurona i-ésima de la entrada y la j-ésima de la capa oculta

θ_j^h: término de tendencia de la neurona j-ésima de la capa oculta

$Neta_{pj}^{h}$: entrada neta de la j-ésima neurona de la capa oculta

i_{pj}: salida de la j-ésima neurona de la capa oculta

f_j^h: función de activación de la j-ésima unidad oculta

w_{kj}^{0}: peso de interconexión entre la j-ésima neurona de la capa oculta y la k-ésima neurona de la capa de salida.

θ_k^0: término de tendencia de la k-ésima neurona de la capa de salida

$Neta_{pk}^{0}$: Entrada neta de la k-ésima neurona de la capa de salida

y_{pk}: salida de la k-ésima unidad de salida

f_k^0: función de activación de la k-ésima unidad de salida

d_{pk}: valor de salida deseado para la k-ésima neurona de la capa de salida

e_p: valor del error para el p-ésimo patrón de aprendizaje

α: tasa o velocidad de aprendizaje

$\delta_{pk}^{\ 0}$: término d error para la k-ésima neurona de la capa de salida

$\delta_{pj}^{\ h}$: término de error para la j-ésima neurona de la capa oculta h

$f'_j{}^h$: derivada de la función de activación de la j-ésima neurona de la capa oculta

$f'_k{}^0$: derivada de la función de activación de la k-ésima neurona de la capa de salida

Los pasos que sigue el algoritmo *backpropagation* son los siguientes:

1. Se inicializan los pesos del MLP.
2. Mientras la condición de parada sea falsa se ejecutan los pasos 3 al 12.
3. Se aplica el vector de entradas a la red $x_p = [x_{p1}, x_{p2}, \ldots\ldots x_{pi} \ldots\ldots x_{pN}]^T$.
4. Se calculan los valores de las entradas netas para la capa oculta:

$$Neta_{pj}{}^h = \sum_i^N w_{ji}{}^h \, x_{pi} + \theta_j{}^h$$

5. Se calcula la salida de la capa oculta:

$$i_{pj}{}^h = f_j{}^h(Neta_{pj}{}^h)$$

6. Se calculan los valores netos de entrada para la capa de salida:

$$Neta_{pk}{}^0 = \sum_{j=1}^{L} w_{kj}{}^0 \, i_{pj}{}^0 + \theta_k{}^0$$

7. Se calculan las salidas de la red:

$$y_{pk} = f_k{}^0(Net_{pk}{}^0)$$

8. Se calculan los términos de error para las unidades de salida:

$$\delta_{pk}{}^0 = (d_{pk} - y_{pk}{}^0) \, f_k{}^{0'}(Neta_{pk}{}^0)$$

9. Se estiman los términos de error para las unidades ocultas:

$$\delta_{pj}{}^{h} = f_j{}^{h'}(Neta_{pj}{}^{h}) \sum_{k=1}^{M} \delta_{pk}{}^{0} w_{kj}{}^{0}$$

10. Se actualizan los pesos en la capa de salida:

$$w_{kj}{}^{0}(t+1) = w_{kj}{}^{0}(t) + \alpha \delta_{pk}{}^{0} i_{pj}{}^{h}$$

11. Se actualizan los pesos de la capa oculta:

$$w_{ji}{}^{h}(t+1) = w_{ji}{}^{h}(t) + \alpha \delta_{pj}{}^{h} x_{pi}$$

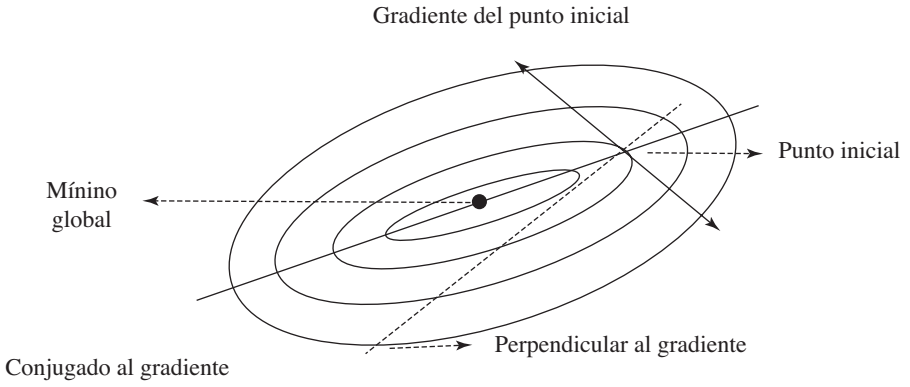
12. Se verifica si el error global cumple con la condición de finalizar:

$$E_p = \frac{1}{2} \sum_{p=1}^{P} \sum_{k=1}^{M} (d_{pk} - y_{pk})^2$$

El algoritmo *backpropagation* encuentra su valor mínimo de error local o global mediante la aplicación de pasos descendentes (algoritmo del gradiente descendente). Con el gradiente descendente, cada vez que se realizan cambios a los pesos de la red, se asegura el descenso por la superficie del error hasta encontrar el valle más cercano, lo que puede hacer que el proceso de aprendizaje se detenga en un mínimo local de error. Uno de los problemas que presenta este algoritmo de entrenamiento es que busca minimizar la función de error, pudiendo caer en un mínimo local o en algún punto estacionario, con lo cual no se llega a encontrar el mínimo global de la función de error. Dado el inconveniente que presenta el algoritmo de gradiente descendente en la red *backpropagation* de tener el parámetro de aprendizaje α fijo, se revisa el mismo algoritmo pero con un alfa variable. Este valor también podría variar en el proceso de aprendizaje con el fin de modificar el tamaño de la variación de los pesos Δ_{wi} y acelerar la convergencia del algoritmo de aprendizaje (Hilera, 1994).

En la misma vía, el algoritmo de aprendizaje del gradiente conjugado es otro de los más famosos algoritmos de optimización multivariable para el aprendizaje de redes tipo MLP. Este algoritmo es un método avanzado de aprendizaje supervisado y usualmente funciona mejor que el *backpropagation*, y puede ser utilizado en las mismas aplicaciones. En el algoritmo del gradiente conjugado se busca minimizar el error sobre una dirección conjugada (el cambio en el gradiente de la función es perpendicular), lo que produce una convergencia más rápida que si la búsqueda se hiciese en la dirección del gradiente descendente (figura 6).

Figura 6. Cálculo del error - Gradiente conjugado



Fuente: Caicedo y López (2009).

Como se muestra en la gráfica, si en vez de calcular una dirección perpendicular a la dirección previa, se calcula una dirección conjugada y se minimiza a lo largo de esta, se llega más rápido al mínimo global de la función (Caicedo y López, 2009).

2. Máquinas de soporte vectorial (SVM)

Las máquinas de soporte vectorial (SVM, por sus siglas en inglés) son un conjunto de algoritmos de aprendizaje supervisado desarrollados por Vapnik y Cortés (1995) y su equipo AT&T, que han surgido como métodos relacionados con problemas de clasificación y regresión. Su buen desempeño ha llevado a su uso en una gran variedad de problemas, algunos investigadores (Fletcher, Hussain y Shawe-taylor, 2010; Huang, Nakamori y Wang, 2005; Kara *et al*., 2011a; Kim, 2003b; Masoud, 2014; Moein Aldin, Dehghan Dehnavi y Entezari, 2012; Net- y Pino, 2012; Velásquez, Olaya y Franco, 2010; Yuan, 2011), han utilizado SVM para solucionar problemas de clasificación y regresión relacionados a la predicción de series de tiempo, mostrando ser mejores en comparación a otras metodologías tradicionales como modelos econométricos y en algunos casos a modelos de aprendizaje de máquina.

2.1. Generalidades de las SVM

La construcción de las máquinas de soporte vectorial (SVM) se basa en la idea de transformar o proyectar un conjunto de datos pertenecientes a una dimensión *n*

dada, hacia un espacio de dimensión superior aplicando una función kernel - Kernel Trick (Alpaydın, 2010). A partir del nuevo espacio creado, se operarán los datos como si se tratase de un problema de tipo lineal, resolviendo el problema sin considerar la dimensionalidad de los datos (Abe, 2005; Alpaydın, 2010; Chalup y Mitschele, 2006; Vapnik y Cortes, 1995; Riobó Otero, n.d.; Karatzoglou, Smola, Hornik y Zeileis, 2004; Alexander, Peter, Bernhard y Dale, 1999).

Las SVM se empezaron a emplear para resolver problemas de clasificación y reconocimiento de patrones para luego extenderse en el estudio de predicción de series de tiempo. Los problemas de clasificación se emplean para obtener resultados de tipo cualitativo, por ejemplo, determinar la clase de un dato de entrada o características, mientras que las de tipo regresión son más útiles en problemas cuantitativos, cuando se trata de obtener una salida numérica al dato de entrada (Camps y Bruzzone, 2009; Fernando y Gutiérrez, 2011; Lember, 2012). Un punto a favor de utilizar este tipo de modelos es que el desempeño de las SVM no depende del tamaño de la muestra que se va utilizar para el problema, por lo que puede ser utilizado para una cantidad limitada de datos en contraste con otras metodologías que presentan mejor desempeño cuando el tamaño de la muestra es grande. Asimismo, el algoritmo detrás de las SVM se puede ajustar a problemas no lineales y la solución se realiza bajo programación cuadrática, lo cual hace que su solución sea única y generalizable (Abe, 2005; Cristianini, 2000; Velásquez *et al*., 2010).

La idea detrás de las SVM es que a partir de unos *inputs* de entrada al modelo, se etiquetan las clases y se entrena una SVM construyendo un modelo que sea capaz de predecir la clase de los nuevos datos que se introduzcan al modelo. La SVM representa en un eje de coordenadas los vectores de entrenamiento, separando las clases por un espacio lo más grande posible. Cuando nuevos datos son introducidos al modelo, estos se colocan sobre el mismo eje y en función de la cercanía de los grupos antes separados, los cuáles serán clasificados en una u otra clase.

2.2. SVM – Caso linealmente separable[1]

Para entender cómo funcionan las SVM, el caso más sencillo que se puede analizar es aquel donde el conjunto de datos de entrenamiento o *inputs* del modelo son

[1] Para el caso separable, tomado de: Alpaydın (2010); Canales (2009); Chalup y Mitschele (2006); Cristianini (2000); Fernando y Gutiérrez (2011); García (2005); Riobó Otero (n.d.); Vapnik y Cortes (1995)

linealmente separables. Un ejemplo de este caso se puede representar a partir de un problema con dos variables de entrada y una salida. Para un problema de pronóstico de series de tiempo las dos características de entrada que podrían utilizarse son la media móvil exponencial de 10 días (*EMA 10 días*) y un indicador de análisis técnico como el RSI. Lo que se quiere analizar es el impacto que tienen estas dos variables en la predicción del movimiento del precio del activo al siguiente instante ($t+1$), implementando una SVM. La salida del modelo sería la dirección que tomará la acción al siguiente día (alcista o bajista). La figura 7 ilustra la relación hipotética entre las dos variables, donde cada punto es el vector de características en la muestra que además refleja el comportamiento a partir del movimiento alcista o bajista que toma el precio en la predicción (triángulos o rombos, respectivamente).

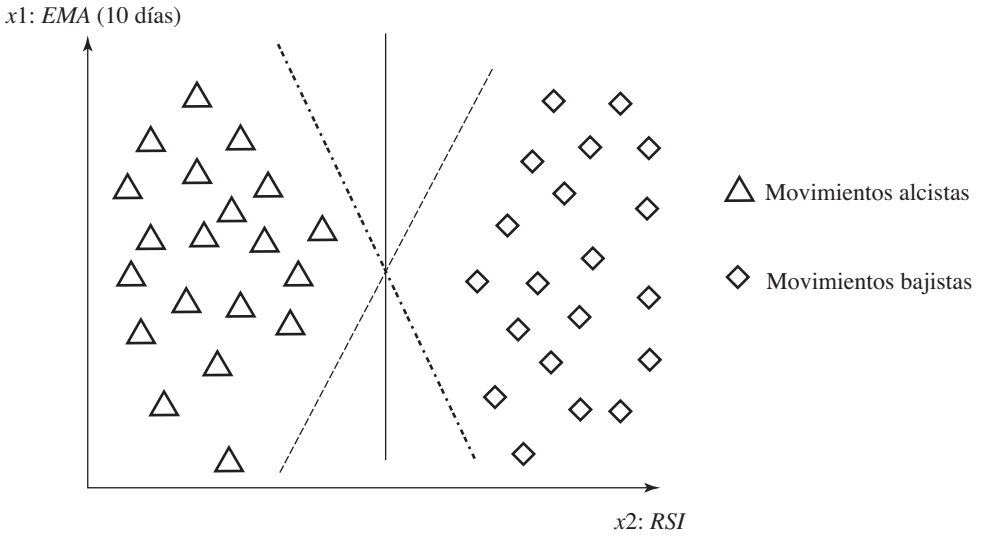
La figura 7 muestra la relación entre el indicador técnico RSI, la media móvil exponencial de diez días y el comportamiento que tendrá la acción al siguiente día. Los triángulos representan los movimientos alcistas y los rombos los movimientos bajistas de la acción. Las líneas representan las posibles clasificaciones de los datos según el comportamiento que tomen los precios.

El objetivo principal es encontrar el hiperplano que maximice el margen que separa las dos clases (alcistas y bajistas). Para ilustrar este problema, las figuras 8 y 9 ilustran dos posibles hiperplanos que separan las dos clases (movimientos alcistas de los movimientos bajistas). Aunque la figura 9 tiene un margen mayor que el de la figura 8, no se puede asegurar que aquel es el mayor respecto a todos los posibles hiperplanos que pueden separar las dos clases. Por tanto, el problema se centra en encontrar dicho hiperplano que maximiza la distancia que separa las dos clases.

Aunque las figuras 8 y 9 muestran dos posibles hiperplanos de separación para el conjunto de datos, la figura 9 representa el margen con mayor franja de separación de los datos, el cual se podría considerar el margen de separación óptimo para la clasificación de los datos.

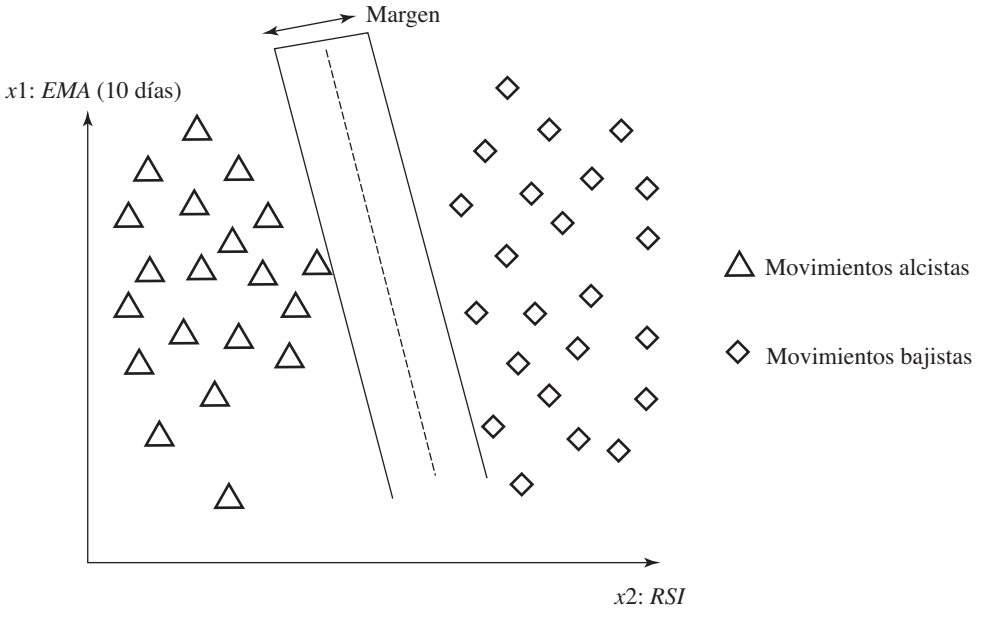
Para encontrar una solución única al problema, Vapnik sugería recurrir a una formulación matemática que relacionara el hiperplano que maximiza el margen de separación con un determinado conjunto de entrenamiento. Inicialmente, antes de poder clasificar las clases, se realiza una etapa de aprendizaje la cual consiste en encontrar el hiperplano $h(x) = 0$ que mejor separe el conjunto de datos según la clase $Y \in \{-1,1\}$ a la que pertenecen.

Figura 7. Separación de dos clases por medio de un hiperplano



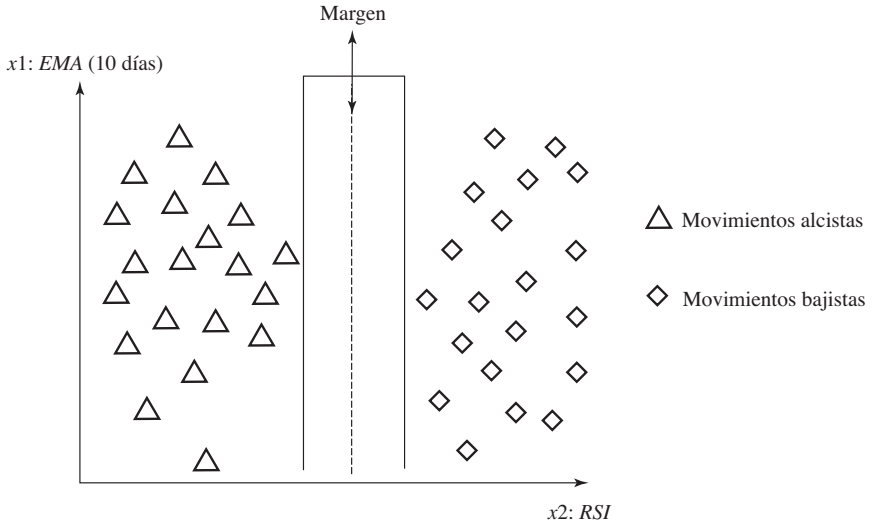
Fuente: gráfica ilustrativa realizada por el autor.

Figura 8. Comparación de hiperplano de separación



Fuente: gráfica ilustrativa realizada por el autor.

Figura 9. Comparación de hiperplano de separación (2)



Fuente: gráfica ilustrativa realizada por el autor.

Dicho hiperplano es el que maximiza la distancia al punto más próximo de cada clase. Por tanto, en el ejemplo, si se cuenta con el conjunto de entrenamiento* que contiene las características y el comportamiento del precio respecto a si va subir o bajar, entonces es posible encontrar un hiperplano que separe aquellos eventos en los que la predicción del precio de cierre al siguiente día es mayor o menor a la del día anterior.

$$* \textit{ Conjunto de entrenamiento} = \{(x_k, y_k) | x_k \in \{-1,1\}\}\ k = 1,.....,N$$

Aquellos puntos x que se encuentran en el hiperplano de separación satisfacen la siguiente relación:

$$x * w + b = 0 \ (7)$$

$$w = \textit{Vector normal y perpendicular al hiperplano}$$

$$x * w = \textit{Producto punto entre los dos vectores}$$

El margen es la distancia entre las proyecciones perpendiculares del punto a la izquierda y a la derecha más cercanos al hiperplano de separación. Por tanto, los datos que pertenecen al conjunto de entrenamiento satisfacen las siguientes restricciones:

$$x_k * w + b \geq +1 \; para \; y_k = +1 \; (8)$$
$$x_k * w + b \leq -1 \; para \; y_k = -1 \; (9)$$

Que se puede expresar de la siguiente manera:

$$x_k \, (x_k * w + b) - 1 \geq 0 \; \forall k \; (9)$$

Adicionalmente, se deben considerar los puntos para los que se cumple la igualdad en (8), que se encuentran en el hiperplano:

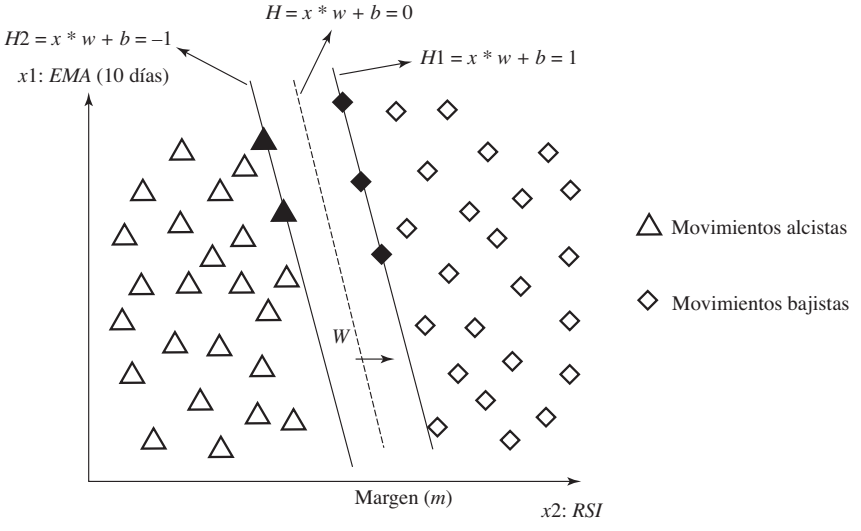
$$H1 = x_k * w + b = 1 \; (11)$$

También, se toman los puntos en los que se cumple la igualdad en (9), que se encuentran en el hiperplano:

$$H2 = x_k * w + b = -1 \; (12)$$

Los anteriores conceptos plantean el problema que encuentra una solución óptima y única para w y b, maximizando como tal el margen en el hiperplano de separación. En el caso de dos dimensiones, la figura 10 ilustra los anteriores conceptos.

Figura 10. Problema de clasificación - Hiperplano



Fuente: gráfica ilustrativa realizada por el autor.

La figura 10 se puede entender de la siguiente manera: la distancia entre el hiperplano de separación y el dato de entrenamiento más cercano al hiperplano, se denomina "margen" (*m*). La habilidad de generalización depende de la localización del hiperplano de separación, y el hiperplano con máximo margen es llamado hiperplano de separación óptimo (H). La habilidad de generalizar se maximiza cuando el hiperplano de separación óptimo es seleccionado como aquel de separación. Optimizar el margen geométrico significa minimizar la norma del vector de pesos. El problema se resuelve bajo programación cuadrática tratando de encontrar el hiperplano óptimo y dos hiperplanos (H1 y H2) paralelos.

Las distancias entre H1 y H2 se maximizan y se restringen a que ningún dato exista entre los dos hiperplanos. Cuando la distancia entre H1 y H2 es maximizada, algunos puntos de datos pueden estar sobre H1 y algunos sobre H2. Estos puntos son llamados vectores soporte, ya que participan de forma directa en definir el hiperplano de separación; los otros puntos pueden ser removidos sin cruzar los planos H1 y H2, y no modificarán de alguna forma la habilidad de generalización del clasificador. La solución de una SVM está dada únicamente por los vectores de soporte. Cualquier hiperplano puede ser representado mediante w, x y b, donde w es un vector perpendicular al hiperplano.

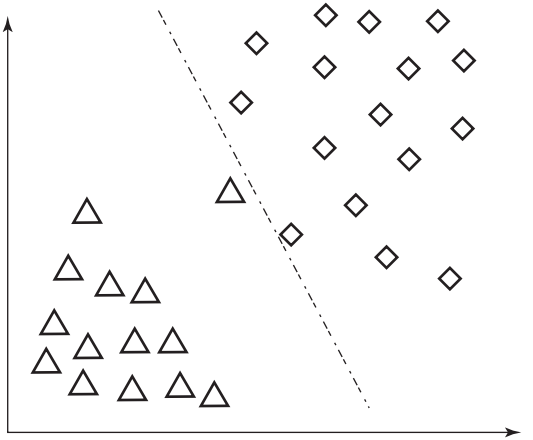
2.3. SVM con margen suavizado – Caso linealmente no separable[2]

El problema de aprendizaje que se presentó en la anterior sección es válido para el caso donde los datos son linealmente separables, es decir, los datos de entrenamiento no tienen intersecciones. Sin embargo, este tipo de problemas no se presentan con regularidad en la práctica. Las soluciones de programación cuadrática, como se mostraron anteriormente, no pueden ser utilizadas en el caso de intersecciones ya que la condición $y_i((w * x_i) + b) \geq 1 \ \forall i$ no se satisface cuando se presentan intersecciones, dado que los puntos que se encuentran en la intersección no pueden ser correctamente clasificados. Hay casos de datos linealmente separables, en los que puede existir ruido debido a errores en la medida de los datos o por la presencia de algún dato atípico o extremo (figura 11).

---

[2] Para el caso no separable con margen suavizado, tomado de: Alpaydın (2010); Canales (2009); Chalup y Mitschele (2006); Cristianini (2000); Fernando y Gutiérrez (2011); García (2005); Riobó Otero (n.d.); Vapnik y Cortes (1995).

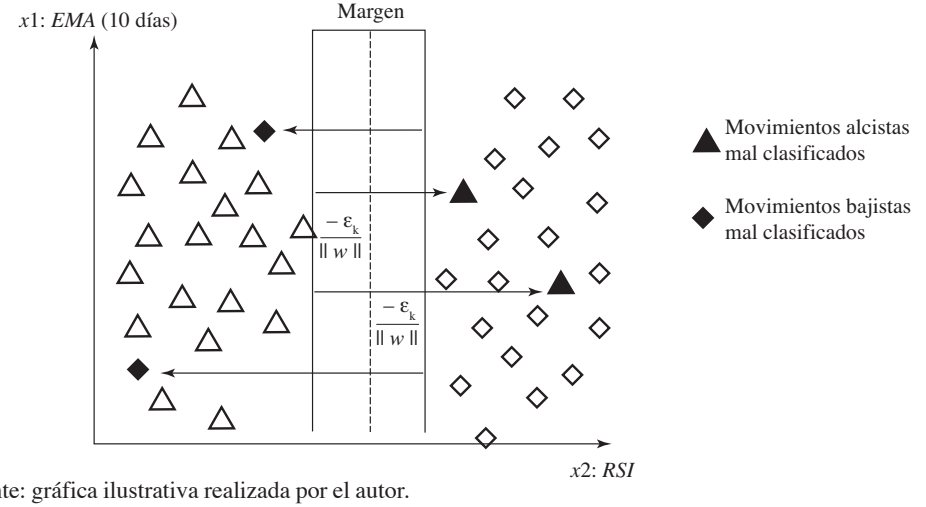
Figura 11. Datos atípicos en el proceso de clasificación



Fuente: gráfica ilustrativa realizada por el autor.

En dichos casos, no es conveniente que la SVM se ajuste totalmente a los datos. En la figura 11 se observan dos conjuntos de datos y la frontera de decisión que se obtendría con la SVM de margen máximo. Cada conjunto está agrupado excepto por un punto que se encuentra próximo a los datos de la otra clase. Este se puede considerar como un dato atípico o que ha sido clasificado por error. Este punto no debería ser considerado para hallar la frontera de decisión ya que podría alterar los resultados deseados y llevaría a clasificaciones incorrectas (figura 12).

Figura 12. Separación de dos clases - Datos linealmente no separables



Fuente: gráfica ilustrativa realizada por el autor.

Existen otros casos en los que los datos no son linealmente separables. Los triángulos y rombos negros representan la submuestra que impide la separación.

Con el fin de extender la metodología de SVM para casos que no son linealmente separables, se deben introducir variables *flojas* o de *holgura* a las ecuaciones 9 y 10, introduciendo una variable no negativa $\varepsilon_k \geq 0$. Por tanto, para encontrar un clasificador con margen máximo, el algoritmo presentado anteriormente en la sección 2.1 se debe modificar permitiendo un margen blando.

$$x_k * w + b \geq +1 - \varepsilon_k \ para\ y_k = +1 \quad (13)$$

$$x_k * w + b \leq -1 + \varepsilon_k \ para\ y_k = -1 \quad (14)$$

Por tanto,

$$y_k(x_k * w + b) - 1 + \varepsilon_k \geq 0 \ donde\ \varepsilon_k \geq 0 \forall k \quad (15)$$

Este tipo de SVM se denomina máquinas de vectores de soporte con margen suavizado. Este procedimiento penaliza a aquellos puntos que generan la no linealidad; por ejemplo, los cuadrados y círculos grises de la figura 12; ε_k se puede interpretar como una penalización proporcional a la distancia entre los puntos mal clasificados y los hiperplanos H1 y H2. Mediante las variables ε_k, la solución factible siempre existe. Para los datos de entrenamiento x_i, si $0 < \varepsilon_k < 1$, los datos no poseen el margen máximo, pero pueden ser correctamente clasificados. Por otro lado, el ancho de este margen blando puede ser controlado por el parámetro de penalización C.

A medida que C aumenta, se penaliza más el error de clasificación. Por ende, un C grande proporciona un pequeño número de errores de clasificación y, tomando $C = \infty$ requiere que el número de datos mal clasificados sea cero. Sin embargo, en este caso no es posible, ya que el problema puede ser factible únicamente para algún valor $C < \infty$.

Por ende, la diferencia entre el caso linealmente separable y el no linealmente separable, es que ahora α_k tiene un límite superior C. A partir de las condiciones (KKT), explicadas en el anexo 4, es posible encontrar b^*, además, deducir que $\varepsilon_k = 0$ para todo $\alpha_k < C$. El problema de optimización cuadrática es prácticamente el mismo que en el caso separable, con la única diferencia de las cotas modificadas de los multiplicadores de Lagrange α_k. El parámetro C es determinado por el usuario. La selección de una C apropiada es realizada experimentalmente usando alguna

técnica de validación cruzada (ver anexo 1: SVM – Cálculo del hiperplano óptimo para el caso lineal y no linealmente separable).

2.4. SVM no lineales – Kernel Machines[3]

Cuando las ecuaciones 10 y 15 no son funciones lineales (no es posible separar linealmente los datos en su espacio de entrada), es necesario emplear un método que permita generalizar el uso de SVM a este tipo de problemas.

$$x_k(x_k * w + b) - 1 \geq 0 \; \forall k$$

$$y_k(x_k * w + b) - 1 + \varepsilon_k \geq 0 \; donde \; \varepsilon_k \geq 0 \; \forall k$$

Los anteriores problemas de optimización que fueron abordados en las secciones 2.2 y 2.3, dependen únicamente de los datos a través del producto punto $x_k * x$.

La metodología para generalizar SVM se basa en un procedimiento denominado Kernel Trick (Aizerman *et al*., 1964), que consiste en que cuando los datos de entrada no son linealmente separables, existe la posibilidad de transformar los datos a un espacio euclidiano de mayor dimensión (el espacio de características) en el que los puntos sí pueden ser separados por un hiperplano, y en el que aproximadamente se tiene una estructura lineal (Alpaydın, 2010; Fernando y Gutiérrez, 2011).

Sea ϕ una función tal que:

$$\Phi: R^n \rightarrow \varphi$$

El algoritmo ahora depende de los datos por medio del producto punto en el espacio φ; es decir, de las funciones de la forma $K(x_k * x) = \Phi(x_k) * \Phi(x)$. Si se utiliza esta función K, denominada función kernel, no será necesario especificar a ϕ. Por tanto, el caso no lineal es equivalente a aplicar una función a los datos en el espacio de entrada y el algoritmo de aprendizaje es utilizado en el nuevo espacio de llegada de la función como se observa en la figura 13. Esta metodología permite encontrar vectores de soporte en el espacio de llegada φ sin importar el algoritmo utilizado. Por consiguiente, todas las consideraciones hechas en las anteriores secciones

[3] Para el caso de SVM no lineales, tomado de: (Alpaydın, 2010; Canales, 2009; Chalup y Mitschele, 2006; Cristianini, 2000; Fernando y Gutiérrez, 2011; García, 2005; Riobó Otero, n.d.; Vapnik y Cortes, 1995).

aplican en la medida en que se realiza una separación lineal, pero en un espacio superior, diferente al inicial. Por ejemplo, si se tiene en cuenta el caso linealmente separable, el hiperplano óptimo para SVM no lineales está dado por:

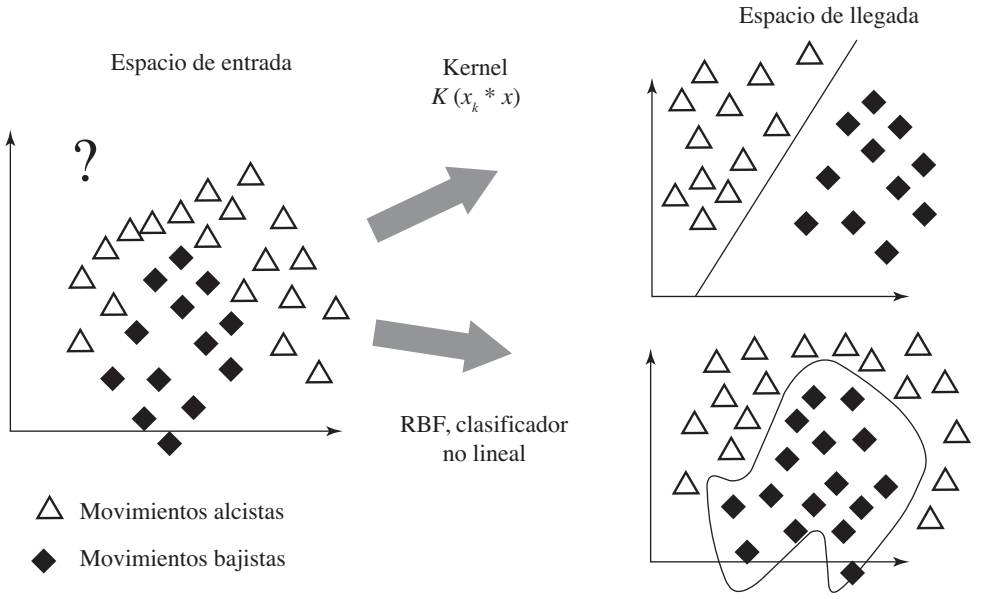
$$f(x, \alpha^* b^*) = \sum_{k \in SV}^{N} y_k \alpha_k^* K(x_k, x) + b^*$$

Sin embargo, para que este procedimiento pueda llevarse a cabo es necesario que la función kernel utilizada satisfaga las condiciones de Mencer (Vapnik y Cortés, 1995):

$$K(x_i, x_j) = \sum_{k}^{\infty} a_k \phi_k(x_i) \phi_k(x_j), a_k \geq 0 \; y$$

$$\iint K(x_i, x_j) g(x_i) g(x_j) dx_i dx_k > 0$$

Figura 13. Transformación de los datos a través del kernel



Fuente: gráfica ilustrativa realizada por el autor.

Las funciones kernel frecuentemente utilizadas por medio de SVM son:

Kernel lineal: este se recomienda cuando hay vectores de datos dispersos.

$$K(x_i, x_j) = x_i * x_j$$

Kernel polinomial: este es un kernel muy utilizado para modelar relaciones no lineales. Sin embargo, a medida que aumenta el parámetro *d* (grado del polinomio) la superficie de clasificación se hace más compleja.

$$K(x_i, x_j) = (1 + x_i * x_j)^d$$

Kernel gaussiano: kernel de base radial o gaussiano es uno de los más utilizados. El parámetro σ controla la forma del hiperplano de separación y este puede optimizarse a partir de métodos de validación cruzada:

$$K(x_i, x_j) = exp\left(-\frac{\left|\left|x_i - x_j\right|\right|^2}{2\sigma^2}\right)$$

Otras funciones Kernel:

Neural (Sigmoid, Tanh) Kernel:

$$K(x_i, x_j) = \tanh\left(ax_i * x_j + b\right)$$

Anova Kernel:

$$K(x_i, x_j) = \left(\sum_i \exp\left(-\gamma(x_i - x_j)\right)\right)^d$$

Fourier Series Kernel:

$$K(x_i, x_j) = \frac{sen(N + \frac{1}{2})(x_i - x_j)}{sen(\frac{1}{2}(x_i - x_j))}$$

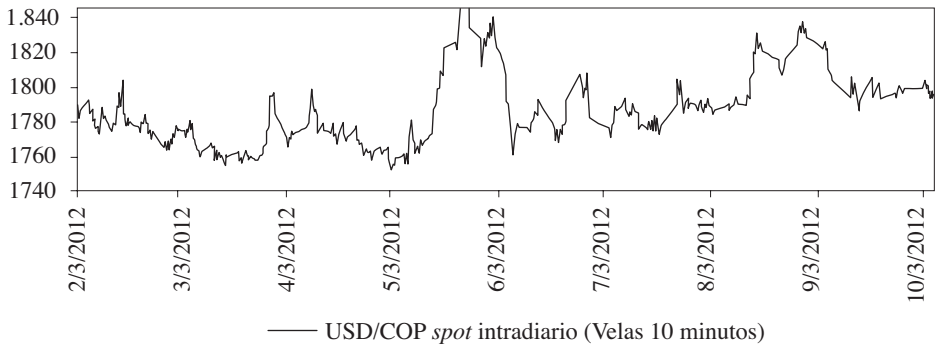
Spline Kernel:

$$K(x_i, x_j) = \sum_{r=0}^{k} x_i^r x_j^r + \sum_{s=1}^{N} (x_i - t_s)^k + (x_j - t_s)^k$$

3. Experimento: aplicación de RNA y SVM en la predicción de la dirección del USD/COP *spot* intradiario

El objetivo del experimento se centró en diseñar, implementar y comparar un modelo de Red Neuronal Multicapa con aprendizaje *backpropagation* con gradiente conjugado y un modelo de Máquina de Soporte Vectorial aplicando funciones kernel lineal, polinomial y gaussiana, para la predicción en intervalos de 10 minutos del movimiento alcista o bajista del *spot* intradiario del USD/COP (figura 14). La base de datos estudiada cubre el periodo entre el 2 de febrero y el 9 de octubre de 2012. El número total de días fue de 179 y el número de registros fue de 15.120 (precios máximo, mínimo, apertura y cierre para intervalos de 10 minutos de la cotización del *spot* USD/COP). De los 3.780 precios de cierre analizados, el 50,7 % de las observaciones tuvieron una dirección alcista y el 49,3 % una dirección bajista.

Figura 14. Comportamiento intradiario *spot* USD/COP - Año 2012



Fuente: gráfica ilustrativa realizada por el autor.

El set de datos está comprendido en un horario entre las 8:20 a.m. hasta las 12:40 p.m. para cada día. Se eliminaron los primeros 20 minutos de la serie (8:00 a.m.-8:20 a.m.) y los últimos 20 minutos (12:40 p.m.-1:00 p.m.) con el fin de eliminar el alto ruido en la cotización de divisas por temas de volatilidad en la apertura y

cierre del mercado, lo cual no se pretendía tener en cuenta para el modelo. Posteriormente, la base de datos se dividió en 7 meses, cada mes compuesto por 540 precios de cierre en intervalos de 10 minutos. Para cada mes se utilizó el 80 % de los precios de cierre (432 observaciones), y para la etapa de entrenamiento y validación de los modelos el 20 % restante (108 observaciones) para el *testing* o prueba del modelo. Cada nuevo mes el modelo iniciaba la etapa de aprendizaje con nueva información y realizaba el mismo proceso de predicción.

3.1. *Inputs* de los modelos

Se experimentó con tres subconjuntos de datos de entrada diferentes a los modelos con el fin de evaluar cuál subconjunto de características se comportaba mejor ante el nivel de predictibilidad de los modelos propuestos. A continuación se explican los tres subconjuntos de datos de entrada.

3.1.1. Indicadores de análisis técnico

Los gestores de fondos, *traders* e inversionistas en el mercado de valores en general aceptan y utilizan ciertos criterios para la incorporación de indicadores técnicos en sus estrategias de inversión (Kim, 2003a). Existe una variedad de indicadores técnicos disponibles en el mercado. Algunos indicadores son eficaces en mercados con tendencia y otros se desempeñan mejor bajo mercados cíclicos o laterales. Para el experimento se seleccionaron once indicadores técnicos como variables de entrada. Con el fin de calcular los indicadores técnicos fue necesario realizar una transformación de los datos, para lo cual se llevaron a cabo las operaciones matemáticas a fin de calcular —a partir de los datos de Apertura, máximo, mínimo y cierre (vela japonesa)—, un único dato denominado oscilador o indicador técnico.

En la figura 15 se resumen los indicadores técnicos seleccionados para el experimento, los cálculos que se realizaron y sus respectivas fórmulas.

Figura 15. *Inputs* de los modelos: 10 Indicadores de análisis técnico

Nombre de los indicadores técnicos	Fórmulas
Simple 10 day moving average	$\frac{C_t + C_{t-1} + \cdots + C_{t-10}}{10}$
Weighted 10 day moving average	$\frac{((n) * C_t + (n-1) * C_{t-1} + \cdots + C_{10})}{(n + (n-1) + \cdots . + 1)}$
Momentum	$C_t - C_{t-n}$
Rate of Change (ROC)	$\frac{C_t - C_{t-n}}{C_{t-n}} * 100$
Commodity Channel Index (CCI)	$\frac{M_t - SM_t}{0{,}015 * D_t}$
RSI – Relative Strength Index	$100 - \frac{100}{1 + (\sum_{i=0}^{n-1} \frac{Up_{t-i}}{n}) / (\sum_{i=0}^{n-1} \frac{Down_{t-i}}{n})}$
Stochastic Oscillator %K	$\frac{C_t - LL_{t-n}}{HH_{t-n} - LL_{t-n}} * 100$
Stochastic Oscillator %D	$\frac{\sum_{i=0}^{n-1} K_{t-i}\%}{n}$
Williams %R	$\frac{H_n - C_t}{H_n - L_n} * 100$
MACD – Moving Average Convergence/ Divergence	$MACD(n)_{t-1} + \frac{2}{n} + 1 * (DIFF_t - MACD(n)_{t-1})$
Average Directional Index (ADX)	$ADX = Average\ DX14$ $DX = 100 * (DI14\ Diff / DI\ 14\ Sum)$

C_t: *es el precio de cierre*, L_t = *es el precio mínimo en t*, H_t = *es el precio máximo en t*, $DIFF$: $EMA(12)_t - EMA(26)_t$, EMA: *Media móvil exponencial*, $EMA(k)_t$: $EMA(k)_{t-1} + \alpha * (C_t - EMA(k)_{t-1})$, α *factor que suaviza los datos*: $^2/_1 + k$. k *es el periodo de tiempo de la media móvil exponencial hace k días*, LL_t *y* HH_t: *son el mínimo más bajo y el máximo más alto en los* últimos t *días respectivamente*, M_t: $H_t + L_t + \frac{C_t}{3}$; SM_t: $(\frac{\sum_{i=1}^{n} M_{t-i+1})}{n}$, D_t: $\frac{(\sum_{i=1}^{n} |M_{t-i+1} - SM_t|)}{n}$, UP_t*es el cambio del precio hacia el alza en tiempo* t, $DOWN_t$ *es el cambio en el precio hacia abajo en tiempo* t.

3.1.2. Los componentes de una vela japonesa: precio apertura, máximo, mínimo y precio de cierre

Se tuvieron en consideración como datos de entrada a los modelos los retornos de los últimos tres periodos del precio máximo, mínimo, apertura y cierre del par de divisas. Es decir, las variables de entrada fueron los retornos de las últimas 3 velas japonesas para un rango de 10 minutos cada una.

3.1.3. Los retornos del precio de cierre desde C_{t-1} hasta C_{t-5}

Se tuvieron en consideración los retornos de los precios de cierre. Se utilizaron los últimos 5 retornos (intervalos de 10 minutos) para predecir la dirección del precio de cierre en los siguientes 10 minutos.

Normalización de los datos de entrada: dado que se está considerando un problema en el que se pretende estimar el valor futuro de un activo, y los valores de entrada al modelo tienen magnitudes diferentes, esto puede generar problemas en la etapa de aprendizaje y entrenamiento de los modelos. Para evitar esta situación, fue necesario normalizar los datos de entrada, realizando una técnica de preprocesamiento de los datos, la cual buscaba que los datos de entrada y de salida quedaran en un intervalo [-1, +1]. Existen dos formas comunes para normalizar los datos. La primera normaliza el dato, a partir del valor máximo del conjunto de datos y del valor mínimo. El segundo, utiliza la técnica de normalización gaussiana.

$$x_n = 2\left(\frac{x - x_{min}}{x_{max} - x_{min}}\right) - 1$$

$$x_n = \left(\frac{x - \mu}{\sigma}\right)$$

Donde:

μ: *valor medio de los datos*

σ: *desviación estándar de los datos*

El dato de salida de los modelos es la dirección del cambio cada 10 minutos del precio del *spot* USD/COP, el cual se clasifica como "0" o "1". Si el precio en el *spot* en el tiempo *t* es mayor o igual que en el momento *t* – 1, la dirección *t* es "1". Si el

precio en el *spot* en el tiempo *t* es menor que en el tiempo *t* – 1, la dirección *t* es "0''. Finalmente, se evaluarán las tres propuestas de *inputs* (11 indicadores técnicos, 3 retornos de la vela japonesa y 5 retornos pasados del precio cada 10 minutos) para determinar cuál de las tres propuestas de entradas al modelo permite pronosticar y modelar de una mejor manera el comportamiento de la dirección del precio *spot* intradiario del USD/COP.

3.2. Experimento 1: red neuronal artificial (RNA)

Entre las consideraciones para el diseño de la red MLP se tuvieron que tener en cuenta los siguientes aspectos: ¿Cómo entrenar la red?, ¿cómo sería su arquitectura?, ¿cuántas neuronas se deben definir por capa?, ¿cómo saber si la red tuvo un buen aprendizaje?, entre otras. A continuación se explican las principales consideraciones del experimento:

i. Para solucionar el problema se dispuso de una base de datos representativa (3.780 datos). Para el experimento se realizaron 4 pruebas con diferentes datos de entrada al modelo. La primera prueba utilizó 11 neuronas en la capa de entrada, en la segunda prueba 6 neuronas, la tercera 12 neuronas y la cuarta 5 neuronas. La capa de salida siempre estuvo compuesta por una única neurona la cual es la predicción (1 o 0), predicción del movimiento alcista o bajista. La dificultad se encontró en definir el número de neuronas en la capa oculta de la red. Existen dos tendencias, empezar desde un alto número de neuronas y evaluar su funcionamiento, y luego, paulatinamente, ir decreciendo el número de neuronas hasta el momento en el que el error de aprendizaje y verificación se mantengan en el mínimo requerido. La otra manera es empezar con un número pequeño de neuronas en la capa oculta, y luego empezar a incrementar el tamaño de neuronas verificando el desempeño de mejora, hasta el punto de que al subir más neuronas no haya un decrecimiento en el error mínimo requerido. Para el experimento se inició desde un número pequeño de neuronas (10 neuronas) en la capa oculta, y se empezó a incrementar de 10 en 10 hasta llegar a 50 neuronas en la capa oculta. Se realizaron 140 simulaciones con el objetivo de encontrar el mejor desempeño de la RNA frente al cambio de las neuronas en la capa oculta.
ii. Otro punto que se tuvo en cuenta fue el de determinar cuántas capas debía tener la RNA. Para este caso se utilizaron tres capas, una de entrada, una capa oculta y una capa de salida. El utilizar más de una capa oculta en la red aumenta la

carga computacional y se optó por aumentar el tamaño de la capa oculta y no incrementar su número.

iii. El algoritmo *backpropagation* se utilizó para entrenar la red de tres capas con conexiones hacia adelante (*feedforward*). Para evaluar el desempeño de la RNA, se implementó el indicador de error cuadrático medio o Root Mean Square Error (MSE %). Por otro lado, el método del gradiente conjugado se utilizó para minimizar el MSE %.

iv. La función de activación por utilizar en las capas ocultas fue de tipo sigmoidal, dado que garantiza la capacidad de procesamiento no lineal. De igual manera, para la capa de salida, se utilizó la función de activación sigmoidal en su salida con el fin de garantizar una salida selectiva, similar a una binaria.

v. En algunas ocasiones, las RNA pueden caer en un problema que se conoce como sobreentrenamiento, el cual trae como consecuencia que el error de test o verificación de la red sea mucho mayor que el de entrenamiento. Para solucionar esto, la técnica de regularización se tuvo en cuenta a fin de solucionar dicho fenómeno del sobreentrenamiento bajo la técnica de regularización por parada temprana. En este caso, el modelo es capaz de parar cuando no encuentra un punto donde no sea capaz de mejorar su entrenamiento o validación. Es así que se tiene en cuenta el término *epochs* o épocas, el cual es el número de veces que el modelo utilizó el set de datos para encontrar el menor error cuadrático medio.

vi. La salida de la capa de salida de la red sigue dos patrones (0 o 1) para la dirección del movimiento del precio. La capa de salida de la red consiste en una única neurona que representa la dirección que toma el movimiento del precio (al alza o a la baja).

vii. Los experimentos se realizaron implementando un código y apoyándose en el Toolbox de Redes neuronales de Matlab.

A continuación (figuras 16-19), se ilustran las estructuras de los modelos de RNA propuestos bajo los cuatro subconjuntos de datos de entradas:

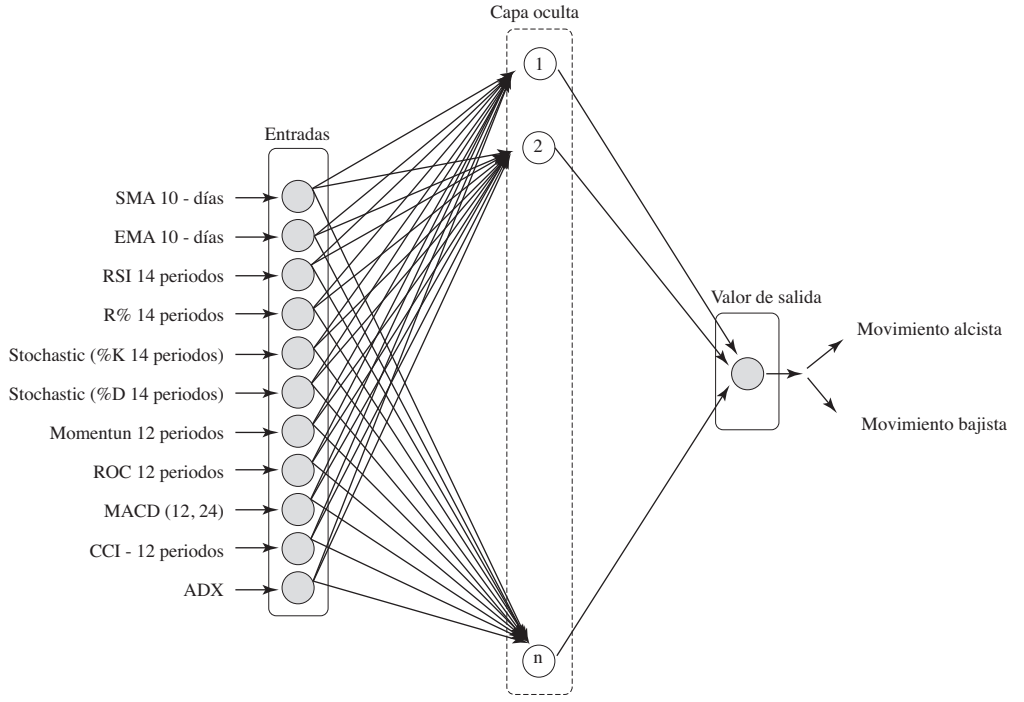
– 11 indicadores de análisis técnico (11 neuronas).

– 6 indicadores de análisis técnico (6 neuronas).

– 5 últimos retornos del precio de cierre (5 neuronas).

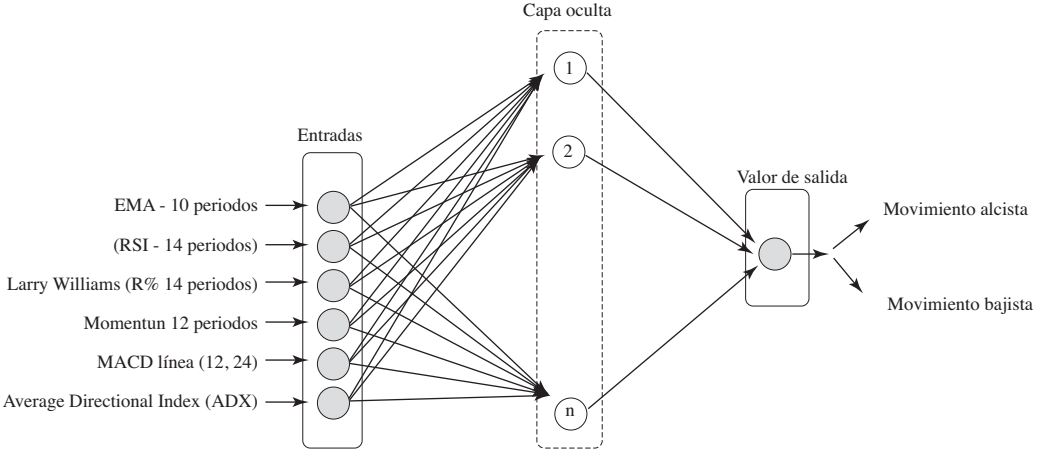
– 3 últimos retornos de las velas japonesas (Precios de apertura, máximo, mínimo y cierre (12 neuronas).

Figura 16. Modelo de red neuronal multicapa (Propuesta No. 1)



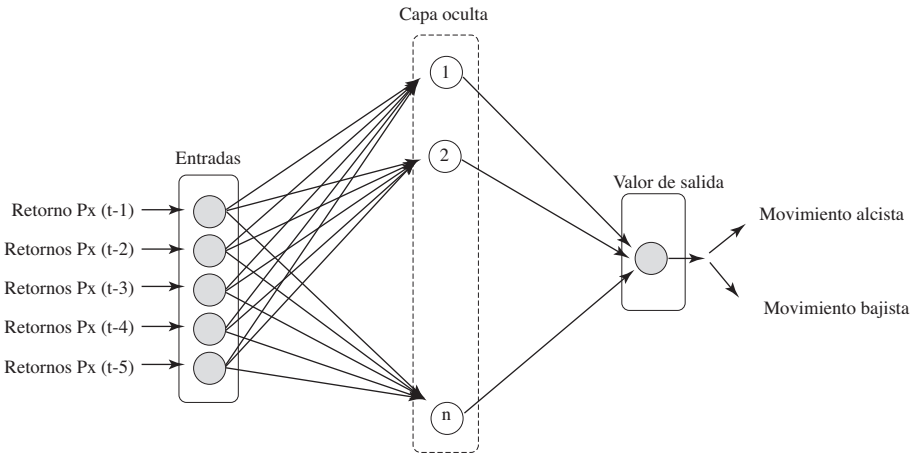
Fuente: gráfica ilustrativa realizada por el autor.

Figura 17. Modelo de red neuronal multicapa (Propuesta No. 2)



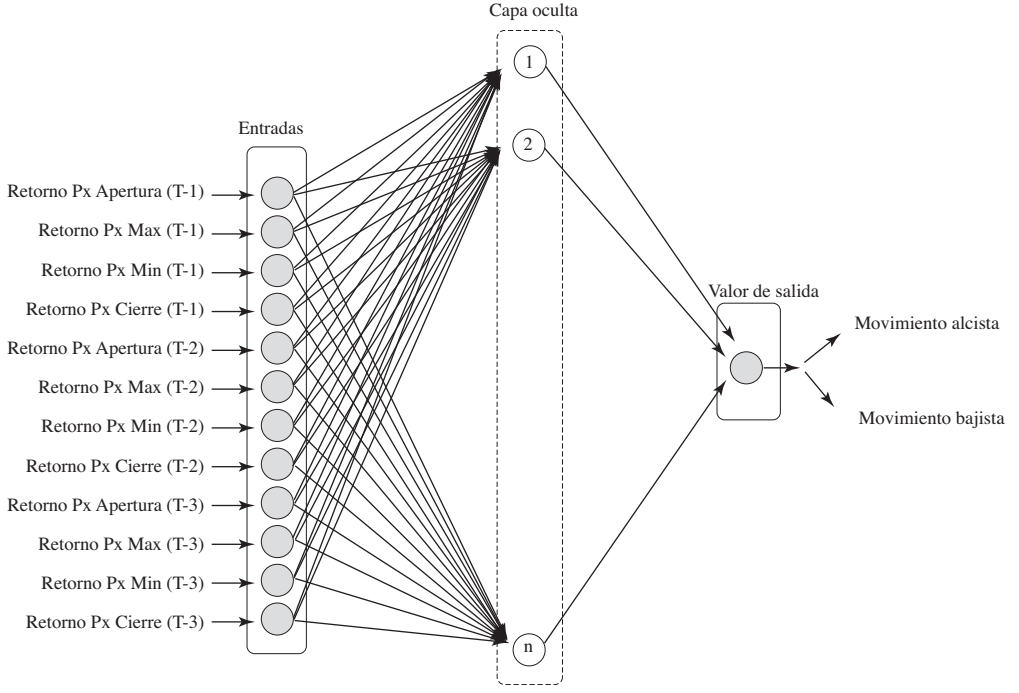
Fuente: gráfica ilustrativa realizada por el autor.

Figura 18. Modelo de red neuronal multicapa (Propuesta No. 3)



Fuente: gráfica ilustrativa realizada por el autor.

Figura 19. Modelo de red neuronal multicapa (Propuesta No. 4)



Fuente: gráfica ilustrativa realizada por el autor.

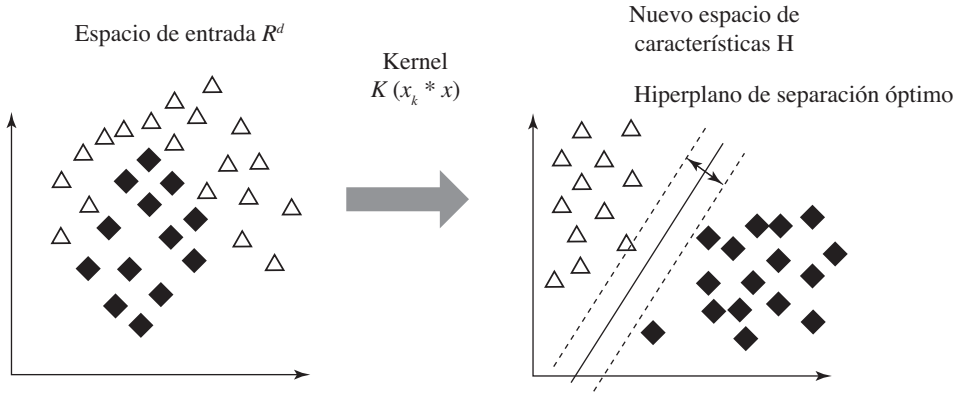
3.3. Experimento 2: máquinas de soporte vectorial

El objetivo principal para el experimento 2 es construir un hiperplano como medida de decisión para que el margen de separación entre las clases positivas y negativas se maximice. Para un set de datos de entrenamiento, con vectores de entrada $x_i \in R^d$ y clases $y_i \in \{+1,0\}$, la SVM aprende a clasificar los objetos en dos clases, en este caso +1 y 0, indicando los movimientos alcistas y bajistas del precio.

La SVM construye un clasificador binario o una función de decisión de los datos disponibles para que esta tenga una probabilidad muy baja de clasificar erradamente un dato futuro. Debido a que los datos no son linealmente separables en su espacio de entrada, para la estimación del SVM se utiliza el algoritmo presentado en el capítulo 3, en el cual se transforman los datos mediante una función Kernel.

La SVM mapea los vectores de entrada $x_i \in R^d$ hacia otro espacio dimensional superior $\Phi(x_i) \in H$ y construye un hiperplano de separación óptimo, el cual maximiza el margen o la distancia entre el hiperplano y los datos más cercanos de cada clase en el espacio H. En la figura 22 se ilustra el hiperplano de separación en el espacio de características que corresponde a un espacio no lineal en el espacio de entradas.

Figura 20. Implementación de kernels



Fuente: gráfica ilustrativa realizada por el autor.

El mapeo de ϕ se realiza bajo una función denominada kernel $K(x_i, x_j)$ la cual define el producto punto en el espacio H. El clasificador se basa en la función de decisión y el problema de programación cuadrática que se describen a continuación:

$$f(x) = \left(\sum_{i=1}^{N} y_i \alpha_i * K(x, x_i) + b \right)$$

El problema cuadrático se basa en:

$$Maximizar \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j * y_i y_j * K(x_i, x_j)$$

$$Sujeto\ a\ que\ 0 \leq \alpha_i \leq c$$

$$\sum_{i=1}^{N} \alpha_i y_i = 0\ i = 1,2,3, \ldots\ldots, N$$

C es un parámetro de regularización que controla el error de clasificación en el margen de separación. Para el experimento se utilizan tres funciones Kernel principales, el gaussiano o de base radial, polinómico y el lineal. Los kernels se muestran a continuación:

$$Función\ polinómica: K(x_i, x_j) = (x_i * x_j + 1)^d$$

$$Función\ de\ base\ radial: K(x_i, x_j) = exp\left(- \frac{\left| \left| x_i - x_j \right| \right|^2}{2\sigma^2} \right)$$

$$Función\ lineal: K(x_i, x_j) = x_i * x_j$$

Donde:

d: grado de la función polinómica

y: constante de la función de base radial

La decisión de escoger el kernel es crítica para la eficiencia de la predicción. Los kernel polinomial, de base radial y lineal son usados en el experimento. Se tienen en cuenta diferentes niveles del grado de la función polinómica (*d*) de la constante sigma (σ), de la función de base radial y del parámetro de regularización (*c*), para ser evaluados en el experimento. La calibración de los parámetros de los kernel

se lleva a cabo mediante búsqueda directa. Se selecciona aquella combinación de parámetros que genera el menor error. Los parámetros y sus niveles se resumen en la tabla 1:

Tabla 1. Niveles de los parámetros de la SVM en la selección de parámetros

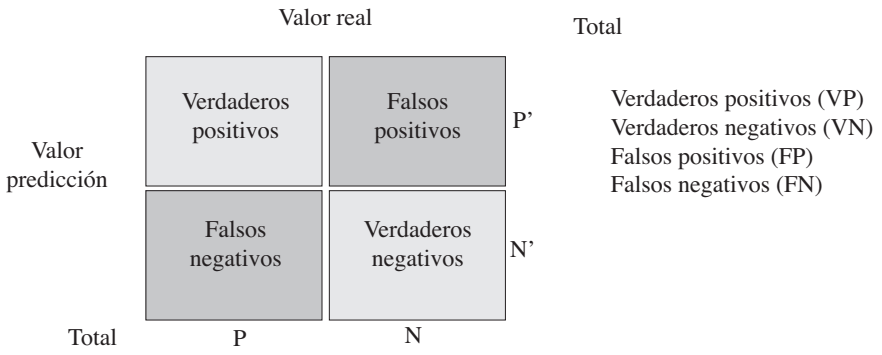
Parámetros	**Niveles (kernel base radial)**	**Niveles (kernel polinomial)**	**Niveles (kernel lineal)**
Grado del kernel función *(d)*	-	*1, 2, 3*	-
Sigma en la función kernel *(σ)*	*0.1, 0,5, 1, 3, 5*	-	-
Parámetro de regularización *(c)*	*1, 10, 20*	*1, 10, 20*	*1, 10, 20*

Los niveles de los parámetros evaluados totalizaron 476 tratamientos para la SVM. Cada combinación de los parámetros se aplicó tanto al set de datos de entrenamiento como al de pronóstico para luego evaluar la *ratio* de predicción de los modelos. El rendimiento del entrenamiento y del pronóstico fue calculado para cada combinación de parámetros. La combinación de los parámetros que resultasen con los mejores rendimientos, tanto de entrenamiento como de pronóstico, se seleccionó como óptima para los correspondientes modelos. Asimismo, se concluye cuál función kernel se comporta mejor para el modelo propuesto de SVM.

3.4. Metodología de validación de los resultados

Para validar los resultados de los experimentos 1 y 2 se utilizan medidas que cuantifican el desempeño del pronóstico para los dos modelos. Para ello se tiene en cuenta el concepto de matriz de confusión, una herramienta de visualización que se emplea en aprendizaje supervisado, en la que en cada columna de la matriz se representa el número de predicciones de cada clase, mientras que en cada fila se representan las instancias de la clase real. Uno de los beneficios de las matrices de confusión es que facilitan identificar si el sistema está confundiendo las clases y proporciona herramientas para seleccionar los modelos posiblemente óptimos y descartar modelos no tan buenos.

Figura 21. Terminología – Matriz de confusión



Fuente: gráfica ilustrativa realizada por el autor.

Los verdaderos positivos (VP), corresponden a la cantidad de movimientos del precio que fueron al alza y fueron clasificados hacia el alza: los falsos positivos (FP), corresponden a la cantidad de movimientos del precio que fueron al alza y fueron clasificados como movimientos a la baja: los verdaderos negativos atañen a la cantidad de movimientos del precio que fueron a la baja y se clasificaron hacia la baja, y los falsos negativos (FN), a la cantidad de movimientos que fueron a la baja y se clasificaron hacia el alza. Para los experimentos de RNA y SVM se tienen en cuenta principalmente los siguientes indicadores de precisión de la clasificación total del modelo:

$$Precisión\ en\ la\ clasificación\ total = \frac{VP + VN}{VP + FP + VN + FN}$$

$$Precisión\ en\ la\ clasificación\ de\ los\ precios\ que\ van\ al\ alza\ = \frac{VP}{VP + FP}$$

$$Precisión\ en\ la\ clasificación\ de\ los\ precios\ que\ van\ a\ la\ baja = \frac{VN}{VN + FN}$$

Por otro lado, para validar la ratio de predicción tanto de las RNA como de las SVM, se implementa otro concepto muy utilizado: el concepto de la curva ROC (Receiver Operating Characteristic, o Característica Operativa del Receptor). La curva ROC es una representación gráfica de la razón de verdaderos positivos frente a la razón de falsos positivos, según varía el umbral de discriminación (valor a partir del cual decidimos que un caso es un positivo o negativo).

Cada resultado de predicción o instancia de la matriz de confusión representa un punto en el espacio ROC. El mejor método posible de predicción se situaría en un punto en la esquina superior izquierda, o coordenada (0,1) del espacio ROC, representando un 100 % de sensibilidad (ningún falso negativo) y un 100 % también de especificidad (ningún falso positivo). A este punto (0,1) también se le llama una clasificación perfecta.

3.5. Resultados de los experimentos

Para el experimento de RNA (Experimento 1), se realizaron 140 combinaciones al modelo (4 x 5 x 7), teniendo en cuenta los 4 subconjuntos de datos de entrada, 5 niveles diferentes de capas ocultas y las 7 semanas de validación de los pronósticos. Los resultados del experimento (tabla 2) evidencian que el modelo propuesto de redes neuronales tuvo un promedio de acierto del 57,8 % utilizando los 6 indicadores técnicos como datos de entrada al modelo, del 56,9 % con los 11 indicadores técnicos, del 55,3 % con los 3 últimos retornos de la vela japonesa y del 54,8 % con los retornos de los últimos 5 precios de cierre. Dados estos resultados, se comprueba que los indicadores técnicos pueden resultar favorables como variables de entrada al momento de implementar modelos de aprendizaje de máquina, como ya lo habían establecido otros investigadores aplicándolos a otros mercados cambiarios y accionarios del mundo (Fletcher y Shawe-taylor, 2010; Papadimitriou *et al*., n.d.; Theofilatos y Karathanasopoulos, 2012; Cao *et al*., 2005; Kamruzzaman *et al*., n. d.)

Tabla 2. Tasa de predicción RNA - Promedio 28 semanas

	***Inputs*: 6 indicadores técnicos (%)**	***Inputs*: 11 indicadores técnicos (%)**	***Inputs*: 3 rezagos (Retornos vela japonesa) (%)**	***Inputs*: 5 rezagos (retornos precio cierre) (%)**
Semana 4	63,0	58,6	61,1	58,3
Semana 8	60,1	59,1	58,3	56,5
Semana 12	57,5	57,1	54,6	53,7
Semana 16	55,6	56,3	55,3	56,5
Semana 20	55,6	54,6	51,3	52,8
Semana 24	56,4	54,6	53,2	52,8
Semana 28	57,4	58,2	53,5	52,9
Promedio total	**57,8**	**56,9**	**55,3**	**54,8**

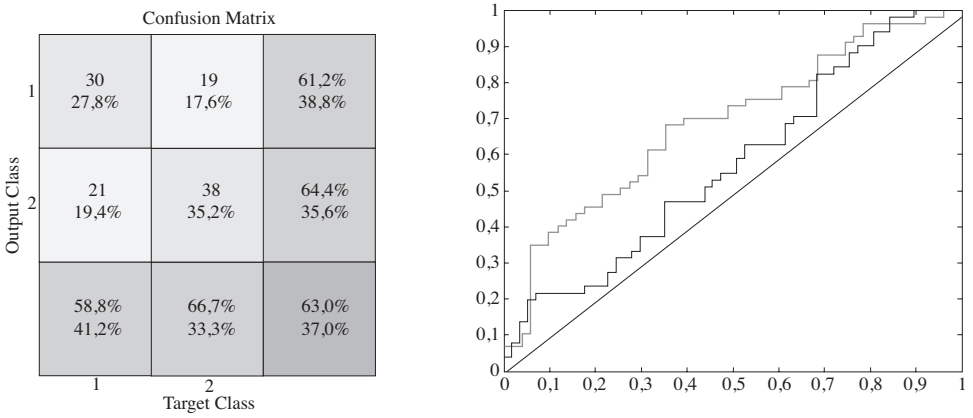
La tabla 3 ilustra las 9 mejores combinaciones de parámetros utilizados en los experimentos de las redes neuronales multicapa. En general, las predicciones se mantuvieron en un rango entre el 57,5 y el 63 %. La mejor combinación de parámetros fue utilizada como datos de entrada de los 6 indicadores técnicos, 20 neuronas en la capa oculta y 24 *epochs*, logrando así una tasa de predicción del 63 % y un MSE de 0,2483.

Tabla 3. Mejor combinación de parámetros para modelos con RNA

No.	*Inputs*	*Neuronascapa oculta*	*MSE*	*epochs*	PTP (%)
1	6 indicadores técnicos	20	0,2483	24	63,00
2	3 Retornos - Velas Japonesas	50	0,2563	15	61,10
3	6 indicadores técnicos	50	0,2673	25	60,10
4	3 Retornos - Velas Japonesas	50	0,2563	15	59,10
5	11 indicadores técnicos	50	0,2501	10	58,60
6	5 retornos - Precio Cierre	50	0,2533	25	58,30
7	3 Retornos - Velas japonesas	10	0,2578	14	58,30
8	11 indicadores técnicos	10	0,2602	24	58,20
9	6 indicadores técnicos	50	0,2673	25	57,50

La matriz de confusión muestra que el modelo propuesto tuvo una tasa de acierto total del 63 %. Una precisión en la clasificación de precios al alza (PPA %) del 61,2% y una precisión en la clasificación de precios a la baja (PPB %) de 64,4 %. Por otro lado, el ROC se encuentra por encima de la diagonal central, lo que supone que el modelo resultó positivo en cuanto a la relación entre verdaderos positivos y falsos positivos. El modelo obtuvo su mejor desempeño de validación en el *epoch* 19, encontrando el menor error cuadrático medio. Por otro lado, el algoritmo *back-propagation* con gradiente conjugado encontró en el *epoch* 25 el menor error del modelo (0,031052).

Figura 22. Matriz de confusión y curva ROC – Modelo con RNA



En el segundo experimento se implementó un modelo de máquinas de soporte vectorial aplicando tres funciones Kernel (base radial, polinomial y lineal). Se realizaron 476 simulaciones teniendo en cuenta las diferentes combinaciones de parámetros entre los kernels. Se ajustaron los parámetros del grado *(d)* del kernel polinomial, la función sigma (σ) del kernel base radial y el parámetro de regularización *(c)* para los 3 kernels, incluyendo el kernel lineal. La tabla 4 ilustra la combinación de parámetros aplicados en el modelo de SVM (experimento 2).

Tabla 4. Combinación de parámetros - SVM

Parámetros	**Niveles (Kernel base radial)**	**Niveles (Kernel polinomial)**	**Niveles (Kernel lineal)**
Grado del kernel función *(d)*	-	1, 2, 3	-
Sigma en la función kernel (σ)	0,1, 0,5, 1,3, 5	-	-
Parámetro de regularización *(c)*	1, 10, 20	1, 10, 20	1, 10, 20

Los resultados del experimento 2 ilustrados en la tabla 5, evidencian que el modelo propuesto de SVM tuvo un promedio de acierto del 55,6% durante los siete meses de estudio utilizando 6 indicadores técnicos como datos de entrada al modelo: del 55,2% con 11 indicadores técnicos, del 54,8% con los retornos de los últimos 5 precios de cierre y del 53,4% con los 3 últimos retornos de la vela japonesa. Los resultados comprueban que el modelo de SVM encontró una tasa de acierto mayor implementando indicadores técnicos, frente a los datos de retornos del precio y de la vela japonesa.

Tabla 5. Tasa de predicción SVM - Promedio 28

	6 indicadores técnicos (%)	11 indicadores técnicos (%)	5 retornos precio de cierre (%)	3 retornos velas japonesas (%)
Semana 4	63,9	62,0	56,5	54,6
Semana 8	53,7	50,9	57,4	53,7
Semana 12	55,6	52,8	49,1	48,1
Semana 16	51,9	59,3	61,1	57,4
Semana 20	51,1	50,9	54,6	53,7
Semana 24	53,7	49,1	53,7	50,0
Semana 28	59,3	61,1	51,0	56,6
Promedio total	**55,6**	**55,2**	**54,8**	**53,4**

La figura 24 evidencia el mejor comportamiento por parte de los indicadores de análisis técnico en los modelos de SVM. Los mejores desempeños se presentaron en la semana 4, alcanzando un 63,9 y 62,0 % de predicción total respectivamente. Sin embargo, los peores desempeños se presentaron en las semanas 20 y 24. La tabla 6 ilustra las 9 mejores combinaciones de parámetros utilizados en los experimentos de SVM aplicando los kernels RBF, polinomial y lineal para los 4 subconjuntos de datos de entrada al modelo.

Tabla 6. Mejor combinación de parámetros para modelos con SVM

No.	Función kernel	Inputs	*D*	*Σ*	*c*	*RVP* (%)	*RVN* (%)	*PPA* (%)	*PPB* (%)	*PTP* (%)
1	RBF	6 Indicadores técnicos	-	1	1	62,7	64,9	61,5	66,1	63,9
2	RBF	11 indicadores técnicos	-	3	10	74,5	50,9	57,6	69,0	62,0
3	RBF	11 indicadores técnicos	-	3	10	60,0	62,3	62,3	60,0	61,1
4	Polino-mial	6 Indicadores técnicos	1	-	1	49,0	67,8	55,8	61,5	59,3
5	Polino-mial	11 Indicadores técnicos	2	-	1	41,2	73,7	58,3	58,3	58,3
6	Polino-mial	11 Indicadores técnicos	1	-	10	49,0	64,4	53,3	60,3	57,4

No.	Función kernel	Inputs	*D*	Σ	*c*	*RVP* (%)	*RVN* (%)	*PPA* (%)	*PPB* (%)	*PTP* (%)
7	Lineal	11 Indicadores técnicos	-	-	10	49,0	67,8	55,8	61,5	59,3
8	Lineal	6 Indicadores técnicos	-	-	10	90,9	26,4	56,2	73,7	59,3
9	Lineal	11 Indicadores técnicos	-	-	10	90,7	25,0	55,7	72,2	58,3

PPA%: precisión en la clasificación de precios al alza

PPB%: precisión en la clasificación de precios a la baja

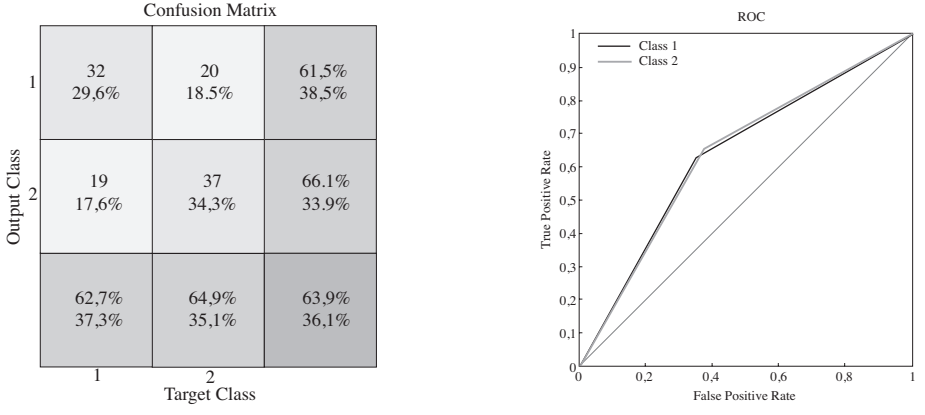
PTP%: precisión total de pronóstico

RVP%: razón de verdaderos positivos

RVN%: razón de verdaderos negativos

La mejor combinación de parámetros se mantuvo en un rango de precisión total entre el 58,3 y 63,9 %. Se encuentra que la mejor combinación de parámetros tuvo su mejor desempeño pronosticando los precios a la baja (64,7 %) frente a los precios al alza (57,4 %). La mejor combinación de parámetros se presentó utilizando como datos de entrada los 6 indicadores técnicos, un kernel de base radial con parámetros sigma (1) y parámetro de regularización (1).

Figura 23. Matriz de confusión y curva ROC – Modelo con SVM

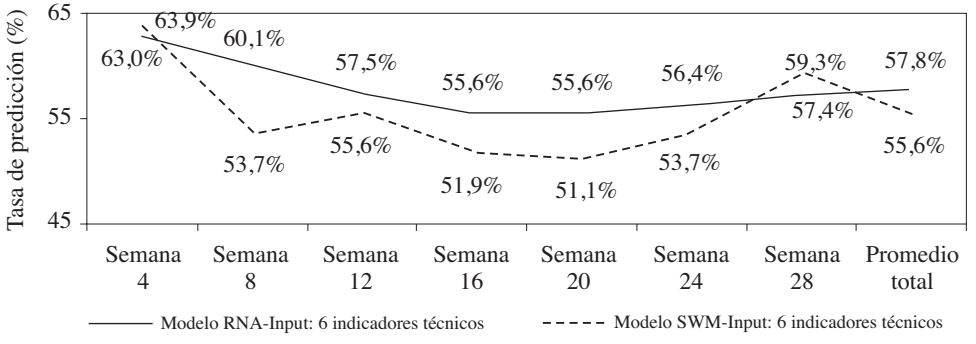


La matriz de confusión en la figura 24 ilustra el resultado para la mejor predicción del modelo de SVM obtenido en la semana 4 del experimento. El modelo propuesto logró una tasa de acierto total del 63,9 %, una precisión en la clasificación de precios al alza (PPA%) del 61,5 % y una precisión en la clasificación de precios a la baja (PPB%) de 66,1 %. Asimismo, el ROC se encuentra por encima de la diagonal central, lo que supone que el modelo resultó positivo en cuanto a la relación entre verdaderos positivos y falsos positivos. Por otro lado, se analizaron los resultados para cada kernel. Los resultados mostraron que el mejor desempeño se obtuvo aplicando kernels de base radial (55,1 %), frente a kernels polinomiales (52,7 %) y lineales (52,2 %) en la SVM.

4. Conclusiones

El objetivo principal del artículo estuvo enfocado en predecir la dirección del movimiento del *spot* intradiario USD/COP. Se construyeron dos modelos predictivos y sus resultados se compararon para la información comprendida entre el 2 de febrero y el 9 de octubre 2012. Basado en los resultados del experimento, algunas conclusiones importantes fueron analizadas. En primera instancia, se puede enfatizar que los 2 modelos —las redes neuronales artificiales (RNA) y las máquinas de soporte vectorial (SVM)— mostraron resultados satisfactorios al predecir la dirección del movimiento cada 10 minutos del mercado *spot* intradiario del USD/COP. Por tanto, se puede concluir que tanto las RNA como las SVM son herramientas útiles para predecir problemas de series de tiempo y, en particular, problemas de predicción de monedas. Los resultados evidenciaron que la tasa de predicción promedio para el modelo de RNA fue de 57,8 %, y para el modelo de SVM fue de 55,6 %.

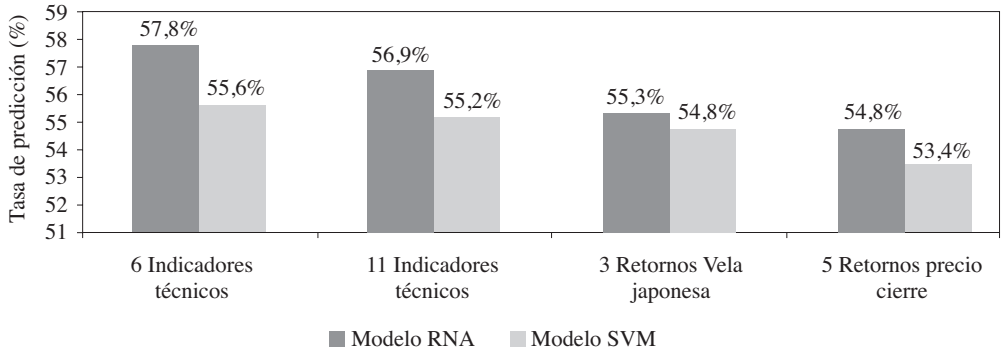
Figura 24. Tasa de predicción promedio - Modelos RNA y SVM



Fuente: gráfica ilustrativa realizada por el autor.

Se utilizaron como datos de entrada a los modelos de RNA y SVM tres subconjuntos de datos principales: indicadores de análisis técnico, los últimos 3 retornos de la vela japonesa y los últimos 5 retornos del precio de cierre. Los resultados del experimento concluyen que los indicadores de análisis técnico fueron significativamente superiores frente a los otros subconjuntos de datos de entrada, tanto para el modelo de RNA como para el de las SVM. La tasa de acierto de los modelos, del 57,8 y 55,6 % respectivamente, se obtuvo utilizando 6 indicadores técnicos como datos de entrada al modelo.

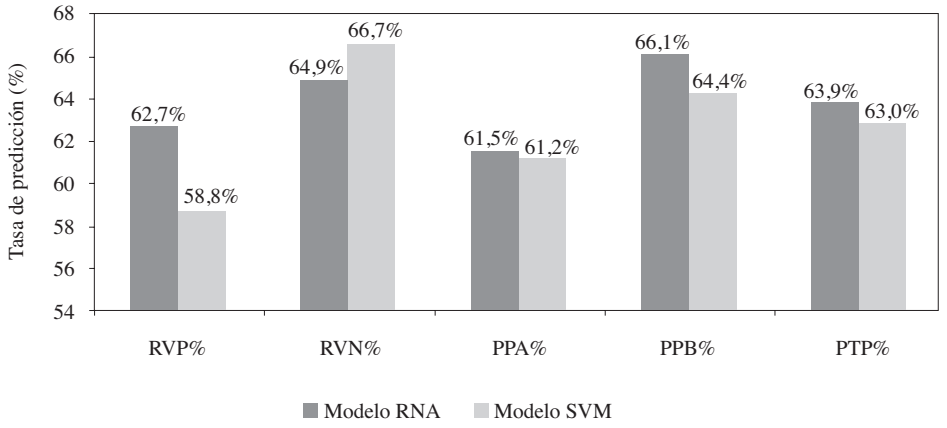
Figura 25. Tasa de predicción - *Inputs* del modelo



Fuente: gráfica ilustrativa realizada por el autor.

La mejor precisión de pronóstico (PTP %) para las RNA y las SVM fue de 63,9 y 63 % respectivamente. Se evidenció que los modelos tuvieron mejor comportamiento en los pronósticos de los precios que fueron a la baja (PPB %) frente a los precios que fueron al alza (PPA %). El mismo fenómeno se observa en la relación de verdaderos negativos (RVN %) frente a la de verdaderos positivos (RVP %).

Figura 26. Matriz de confusión - Mejor predicción RNA y SVM



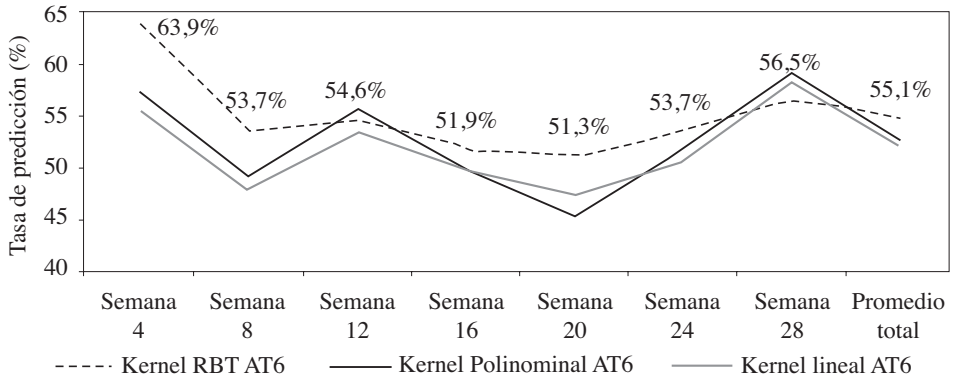
PPA%: Precisión en la clasificación de precios al alza
PPB%: Precisión en la clasificación de precios a la baja
PTP%: Precisión total de pronóstico
RVP%: Razón de verdaderos positivos
RVN%: Razón de verdaderos negativos

Fuente: gráfica ilustrativa realizada por el autor.

Indiferentemente de los resultados del experimento, se debe tener en cuenta un punto a favor de utilizar modelos de SVM, saber, el algoritmo detrás que tiene el modelo permite ajustarse a problemas no lineales y la solución se realiza bajo programación cuadrática, lo cual hace que su solución sea única y generalizable.

Otro aspecto relevante que se obtuvo del experimento fue determinar qué funciones kernel eran las que mejor se adaptaban al modelo de SVM. Se implementaron tres funciones kernel (base radial, polinomial y lineal), a partir de la realización de 476 simulaciones, teniendo en cuenta las diferentes combinaciones de parámetros para las funciones: ajustes a los parámetros del grado *(d)* del kernel polinomial, la función sigma (σ*)* del kernel de base radial y el parámetro de regularización *(c)* para los 3 kernels incluyendo el kernel lineal. Los resultados revelaron que el mejor desempeño de las SVM se obtuvo aplicando kernels de base radial - RBF (55,1%), para obtener una tasa de predicción promedio superior frente a la aplicación de kernels polinomiales y lineales. El kernel lineal fue el de peor desempeño, lo que evidencia la no linealidad de los datos.

Figura 27. Comportamiento kernel - Modelo SVM



Fuente: gráfica ilustrativa realizada por el autor.

Otro asunto que se tiene en consideración son las diferencias entre las tasas de predicción para los 7 meses examinados. Se observa que los peores resultados se obtuvieron entre mayo y junio, y agosto y septiembre de 2012, meses en los que se presentaron niveles de alta volatilidad en el mercado y cambios bruscos de tendencia en la cotización.

Anexo 1. SVM – Cálculo del hiperplano óptimo para el caso lineal y no linealmente separable[4]

Para desarrollar el problema es necesario definir el margen m, en donde m es una función de la norma euclidiana de w, $\|w\|$, tal como sigue:

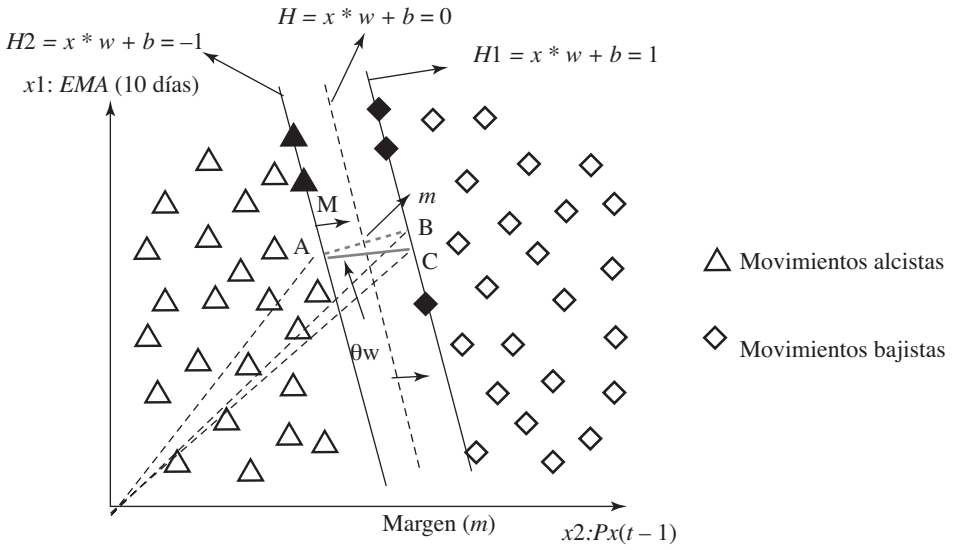
$$m = \frac{2}{||w||} \quad (15)$$

La ecuación 15 representa el margen de separación que se explica a continuación:

Sea A un punto de coordenadas $\{x1; y1\}$ *que hace parte del hiperplano H2, B y C puntos de coordenadas* $\{x2; y2\}$ *y* $\{x3; y3\}$, *respectivamente, que hacen parte de H1 y A, B y C que forman el triángulo rectángulo* $\triangle ABC$ *cuyo ángulo BAC es* θ. *Lo anterior se puede observar en la figura 29.*

[4] Tomado de Fernando y Gutiérrez (2011); Vapnik y Cortés (1995).

Figura 28.



Fuente: gráfica realizada por el autor.

Por otro lado, sea M tal que $||M|| = m$ y $M = \alpha w$ de forma que M es paralela a w, entonces mediante propiedades trigonométricas se tiene que:

$$\mathrm{M} = \cos(\theta)\,||\mathrm{C}-\mathrm{A}|| \text{Dado que } \cos(\theta) = \frac{\mathrm{M}*(\mathrm{C}-\mathrm{A})}{||\mathrm{M}||*||\mathrm{C}-\mathrm{A}||}, \text{se tiene que}$$

$$\mathrm{m} = \frac{\mathrm{M}*(\mathrm{C}-\mathrm{A})}{||\mathrm{M}||*||\mathrm{C}-\mathrm{A}||}\,||\mathrm{C}-\mathrm{A}||$$

$$\mathrm{m} = \frac{\alpha \mathrm{w}}{||\alpha \mathrm{w}||}(\mathrm{C}-\mathrm{A})$$

$$\mathrm{m} = \frac{\mathrm{w}}{||\mathrm{w}||}(\mathrm{C}-\mathrm{A})$$

Entonces, la resta entre las ecuaciones (8) y (9)

$$\mathrm{x_k}*\mathrm{w}+\mathrm{b} \geq +1 \text{ para } \mathrm{y_k} = +1 \quad (8)$$

$$\mathrm{x_k}*\mathrm{w}+\mathrm{b} \leq -1 \text{ para } \mathrm{y_k} = -1 \quad (9)$$

evaluadas en las cotas que definen la igualdad para dos puntos A y C (figura 29) es igual a:

$$\mathrm{w}\,(\mathrm{C}-\mathrm{A}) \;=\; 2$$

Finalmente, reemplazando en:

$$\mathrm{m} = \frac{\mathrm{w}}{||\mathrm{w}||}\,(\mathrm{C}-\mathrm{A})$$

Se tiene que:

$$\boxed{\mathbf{m} = \frac{\mathbf{2}}{||\mathbf{w}||}}$$

Entonces, el problema radica en maximizar el margen *m* definido en la ecuación (15).

$$argmín_{w,b}\,[w * w] = \left|\left|w\right|\right|^2 \quad (16)$$

$$sujeto\ a{:}\ y_k(x_k * w + b) - 1 \;\geq 0\ \forall k$$

Entonces, el hiperplano tiene un margen máximo (geométrico) $\gamma = \frac{1}{||w||}$.

La expresión (16) se puede transformar a partir de su lagrangiano asociado.

$$L(w,b,\alpha) = \frac{1}{2}||w||^2 - \sum_{k=1}^{N} \alpha_k[y_k\,(x_k * w + b) - 1] \quad (17)$$

$$Donde\ los\ \alpha_k \geq 0\ son\ los\ multiplicadores\ de\ Lagrange$$

El dual de las ecuaciones se encuentra en dos pasos. Primero diferenciando la ecuación con respecto a *w* y *b*:

$$\frac{\partial L(w,b,\alpha)}{\partial w} = w - \sum_{k=1}^{N} y_k \alpha_k x_k = 0$$

$$\frac{\partial L(w,b,\alpha)}{\partial b} = w - \sum_{k=1}^{N} y_k \alpha_k = 0$$

Por tanto,

$$w = \sum_{k}^{N} y_k \, \alpha_k x_k \; (18)$$

$$0 = \sum_{k}^{N} y_k \, \alpha_k \; (19)$$

Segundo, restituyendo las relaciones obtenidas en el Lagrangiano original. Reemplazando 18 y 19 en 17 obtenemos lo siguiente:

$$L(w,b,\alpha) = \frac{1}{2}(w * w) - \sum_{k=1}^{N} \alpha_k \left[y_k * \left(x_k \left(\sum_{l=1}^{N} y_l \alpha_l x_l \right) + b \right) - 1 \right] =$$

$$= \frac{1}{2}\left(\sum_{k=1}^{N} y_k \alpha_k x_k * \sum_{l=1}^{N} y_l \alpha_l x_l \right) - \sum_{k=1}^{N} \sum_{l=1}^{N} y_k y_l \alpha_k \, \alpha_l (x_k * x_l) - b \sum_{k}^{N} y_k \alpha_k + \sum_{k}^{N} \alpha_k$$

$$= \frac{1}{2} \sum_{k,l=1}^{N} y_k y_l \alpha_k \alpha_l (x_k * x_l) - \sum_{k,l=1}^{N} y_k y_l \alpha_k \alpha_l (x_k * x_l) + \sum_{k=1}^{N} \alpha_k$$

$$W(\alpha) = \sum_{k}^{N} \alpha_k - \frac{1}{2} \sum_{k,l=1}^{N} y_k y_l \alpha_k \alpha_l (x_k * x_l) \; (20)$$

El cambio de notación de $L(w, b, \alpha)$ por $W(\alpha)$ refleja la nueva representación del problema. Ahora, es necesario encontrar el hiperplano óptimo a partir de los coeficientes α_k^* que maximizan la ecuación (20). El anterior problema se puede formular como:

$$argmáx_{\alpha 1, \ldots, \alpha N} \sum_{k}^{N} \alpha_k - \frac{1}{2} \sum_{k,l=1}^{N} y_k y_l \alpha_k \alpha_l (x_k * x_{l)} \; (21)$$

$$s.a. \sum_{k}^{N} y_k \alpha_k = 0$$

$$\alpha_k \geq 0, k = 1, \ldots., N.$$

Reemplazando en la ecuación 18 los coeficientes:

$$\alpha_k^*, k = 1, \ldots\ldots, N \text{ se obtiene}:$$

$$w^* = \sum_k^N y_k \alpha_k^* x_k$$

El valor de *b* no está definido en la ecuación 21, por lo que se debe encontrar por medio de (8) y (9) cuando se satisface la igualdad:

$$b^* = -\frac{2x_k * w^*}{2}$$

El problema de optimización anteriormente planteado sugiere que la solución óptima para *w** y *b** debe satisfacer las condiciones Karush-Kuhn-Tucker (KKT). Las condiciones de Karush-Kuhn-Tucker son un componente esencial en la teoría de optimización, dado que brindan las condiciones para obtener una solución óptima a un problema de optimización general (Canales, 2009; Vapnik y Cortés, 1995).

$$\alpha^*(y_k(x_k * w^* + b^*) - 1) = 0, k = 1, \ldots\ldots, N \ (22)$$

Cuando α_k^* es diferente de cero, los valores corresponden a los vectores x_k, denominados los vectores de soporte, que satisfacen la igualdad:

$$y_k(x^k * w^* + b^*) = 1 \ (23)$$

En la figura 10, los vectores de soporte son los símbolos que hacen parte de los hiperplanos H1 y H2. Estos vectores son fundamentales en la construcción del algoritmo, dado que el vector *w** está definido a partir de ellos:

$$w^* = \sum_k^N y_k \alpha_k^* x_k \text{ por lo que el hiperplano óptimo es}:$$

$$f(x, \alpha^*, b^*) = \sum_{k \in SV}^N y_k \alpha_k^* (x_k * x) + b^* \ (24)$$

Caso linealmente no separable: teniendo en cuenta que el número de vectores mal clasificados debe ser mínimo, el problema descrito en (16) se puede presentar de la siguiente manera:

$$argmín_{w,b,\varepsilon} ||w||^2 + C \sum_{k=1}^{N} \varepsilon_k \quad (25)$$

$$sujeto\ a: y_k(x_k * w + b) - 1 + \varepsilon_k \geq 0$$

El problema se resuelve de la misma manera en que se solucionó el problema 3.10, por medio del Lagrangiano y de su representación dual: el dual es encontrado en dos pasos: de la misma manera que en el caso linealmente separable, primero diferenciando con respecto a *w* y *b*, y después restituyendo en el Lagrangiano original:

$$L(w, b, \varepsilon, \alpha) = \frac{1}{2}||w||^2 + C \sum_{k=1}^{N} \varepsilon_k - \sum_{k=1}^{N} \alpha_k [y_k(x_k * w + b) - 1 + \varepsilon_k] - \sum_{k=1}^{N} \mu_k \varepsilon_k \quad (26)$$

Las condiciones de primer orden serían:

$$\frac{\partial L(w, b, \varepsilon, \alpha)}{\partial w} = w - \sum_{k=1}^{N} y_k \alpha_k x_k = 0$$

$$\frac{\partial L(w, b, \varepsilon, \alpha)}{\partial b} = w - \sum_{k=1}^{N} y_k \alpha_k = 0$$

$$\frac{\partial L(w, b, \varepsilon, \alpha)}{\partial \varepsilon_k} = C - \alpha_k - \mu_k = 0$$

Por tanto,

$$w = \sum_{k}^{N} y_k \alpha_k x_k \quad (27)$$

$$0 = \sum_{k}^{N} y_k \alpha_k \quad (28)$$

$$C = \alpha_k \mu_k \quad (29)$$

Reemplazando en 26 se obtiene:

$$L(w,b,\varepsilon,\alpha) = \frac{1}{2}\sum_{k,l=1}^{N} y_k y_l \alpha_k \alpha_l (x_k * x_l) + C\sum_{k=1}^{N}\varepsilon_k - \sum_{k,l=1}^{N} y_k y_l \alpha_k \alpha_l (x_k * x_l)$$

$$- b\sum_{k=1}^{N}\alpha_k y_k + \sum_{k=1}^{N}\alpha_k - \sum_{k=1}^{N}\alpha_k \varepsilon_k - \sum_{k=1}^{N}\mu_k \varepsilon_k$$

$$= \sum_{k}^{N}\alpha_k - \frac{1}{2}\sum_{k,l=1}^{N} y_k y_l \alpha_k \alpha_l (x_k * x_l) + (C - \mu_k - \alpha_k)\sum_{k=1}^{N}\varepsilon_k$$

$$W(\alpha) = \sum_{k}^{N}\alpha_k - \frac{1}{2}\sum_{k,l=1}^{N} y_k y_l \alpha_k \alpha_l (x_k * x_l) \quad (30)$$

Ahora, la ecuación 21 queda bajo otras restricciones:

$$argmáx_{\alpha 1,\ldots..,\alpha N}\sum_{k}^{N}\alpha_k - \frac{1}{2}\sum_{k,l=1}^{N} y_k y_l \alpha_k \alpha_l (x_k * x_{l)} \quad (31)$$

$$s.a.\sum_{k}^{N} y_k \alpha_k = 0$$

$0 \le \alpha_k \le C, k = 1, \ldots., N.$ *La solución es dada por*:

$$w^* = \sum_{k\in SV}^{N} y_k \alpha_k^* x_k$$

Referencias

Abe, S. (2005). Support Vector Machines for Pattern Classification. London: Springer.

Alexander, J. S., Peter, B., Bernhard, S. y Dale, S. (1999). *Advances in Large Margin Classifiers*. London: The MIT Press.

Alpaydın, E. (2010). *Introduction to Machine Learning* (2 ed.). London: The MIT press.

Appel, G. (2005). Technical analysis: power tools for active investors. Recuperado de http://dl.acm.org/citation.cfm?id=1408581.

Caicedo, Eduardo y López, Jesús. (2009). *Una aproximación práctica a las Redes Neuronales Artificiales*. Cali: Universidad del Valle.

Camps, G. y Bruzzone, L. (2009). *Kernel Methods for Remote Sensing Data Analysis*. UK: Wiley.

Canales, J. C. (2009). *Clasificación de grandes conjuntos de datos vía Máquinas de Vectores Soporte y aplicaciones en sistemas biológicos*. Tesis doctoral: México D.F.

Cao, D., Pang, S. y Bai, Y. (2005). Forecasting exchange rate using support vector machines. Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou, 18-21 August 2005.

Chalup, S. y Mitschele, A. (2006). *Kernel Methods in Finance*. Handbook on Information Technology in Finance. London: Springer.

Chen, W. y Shih, J. (2006). Comparison of support-vector machines and back propagation neural networks in forecasting the six major Asian stock markets Soushan Wu. Int. J. Electronic Finance, 1 (1).

Cristianini, N. J. S.-T. (2000). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge UK: Cambridge University Press.

Del Brio, M. y Sanz, S. (2006). *Redes neuronales y sistemas borrosos*. Zaragosa: Alfaomega

Fernando, J. y Gutiérrez, M. (2011). *Pronóstico de incumplimiento de pago mediante máquinas de vectores de soporte*. Bogotá: Banco de la República

Fletcher, T., Hussain, Z. y Shawe-Taylor, J. (2010). *Currency Forecasting using Multiple Kernel Learning with Financially Motivated Features*. London: Centre for Computational Statistics and Machine Learning, Department of Computer Science.

Fletcher, T. y Shawe-taylor, J. (2010). Multiple Kernel Learning on the Limit Order Book. JMLR: Workshop and Conference Proceedings 11. London: Workshop on Applications of Pattern Analysis.

García, E. E. (2005). *Boosting Support Vector Machines*. Bogotá: Facultad de los Andes.

Hilera, G. (1994). *Redes neuronales artificiales. Fundamentos, modelos y aplicaciones*. Serie Paradigma, RaMa.

Huang, W., Nakamori, Y. y Wang, S. Y. (2005). Forecasting stock market movement direction with support vector machine. *Computers & Operations Research*, 32 (10), 2513-2522. doi:10.1016/j.cor.2004.03.016

Kamruzzaman, J., Sarker, R. A. y Ahmad, I. (n.d.). SVM based models for predicting foreign currency exchange rates. *Third IEEE International Conference on Data Mining*, 557-560. doi:10.1109/ICDM.2003.1250976

Kara, Y., Acar Boyacioglu, M. y Baykan, Ö. K. (2011a). Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange. *Expert Systems with Applications*, 38 (5), 5311-5319. doi:10.1016/j.eswa.2010.10.027

Kara, Y., Acar Boyacioglu, M. y Baykan, Ö. K. (2011b). Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange. *Expert Systems with Applications*, 38 (5), 5311-5319. doi:10.1016/j.eswa.2010.10.027

Karatzoglou, A., Smola, A., Hornik, K. y Zeileis, A. (2004). kernlab – An S4 Package for Kernel Methods in R. *Journal of Statistical Software*, 11 (9), 1-20. Recuperado de http://www.jstatsoft.org/v11/i09/paper

Karazmodeh, M., Nasiri, S. y Hashemi, S. M. (2013). Stock Price Forecasting using Support Vector Machines and Improved Particle Swarm Optimization. *Journal of Automation and Control Engineering*, 1 (2), 173-176. doi:10.12720/joace.1.2.173-176

Kim, K. (2003a). Financial time series forecasting using support vector machines. *Neurocomputing*, 55 (1-2), 307-319. doi:10.1016/S0925-2312(03)00372-2

Lember, J. (2012). *Introduction to statistical learning*. London: Springer

Masoud, N. (2014). Predicting Direction of Stock Prices Index Movement Using Artificial Neural Networks: The Case of Libyan Financial Market. *British Journal of Economics, Management & Trade*, 4 (4), 597-619. doi:10.9734/BJEMT/2014/5519

Moein Aldin, M., Dehghan Dehnavi, H. y Entezari, S. (2012). Evaluating the Employment of Technical Indicators in Predicting Stock Price Index Variations Using Artificial Neural Networks (Case Study: Tehran Stock Exchange). *International Journal of Business and Management*, 7 (15), 25-34. doi:10.5539/ijbm.v7n15p25

Murphy, J. J. (2000). *Análisis técnico de los mercados financieros*. Gestión 2000. Recuperado de http://dialnet.unirioja.es/servlet/libro?codigo=49652

Net-, N. y Pino, R. (2012). Predicción del índice IBEX-35 aplicando Máquinas de Soporte Vectorial y Redes. 6th International Conference on Industrial Engineering and Industrial Management. XVI Congreso de Ingeniería de Organización. Vigo, July 18-20.

Nison, S. (1994). *Más allá de las velas*. Madrid: Gesmovasa.

Riobó Otero, V. (n.d.). *Reconocimiento de localizaciones mediante máquinas de soporte vectorial*. Universidad Carlos III: Madrid.

Tan, T. Z., Quek, C. y Ng, G. S. (2004). Brain-inspired genetic complementary learning for stock market prediction. Singapore: Nanyang Technological University.

Theofilatos, K. y Karathanasopoulos, A. (2012). Modeling and Trading the EUR / USD Exchange Rate Using Machine Learning Techniques. ETASR - Engineering, Technology & Applied Science Research, 2 (5), 269-272.

Vapnik, V. y Cortés, C. (1995). Support-Vector Networks, 297, 273-297.

Velásquez, J. D., Olaya, Y. y Franco, C. J. (2010). Time series prediction using support vector machines, 18, 64-75.

Wilder, W. (1988). *New concepts in technical trading systems*. Winston: Hunter Publishing Company.

Yuan, C. (2011). *Predicting S & P 500 Returns Using Support Vector Machines: Theory and Empirics*. Washington: University in St. Louis - Center for Research in Economics and Strategies Fellowship.