
FACULTAD DE ECONOMÍA

TESIS DE MAESTRÍA

# Adaptación de una red neuronal para la negociación en el mercado de divisas

Tesis presentada por Darwin Javier Fonseca Lemus para el título de Magister en Finanzas Cuantitativas de la Universidad del Rosario

---

Supervisada por:
Hugo Eduardo Ramírez Jaime

Julio 2018

# Resumen

En esta tesis se exploran varias técnicas de machine learning, las cuales son aplicadas al problema de decisión de inversión. Se utilizan una serie de indicadores de análisis técnico como entradas de una red neuronal, entrenada para determinar con cada vector de entrada una señal de compra, venta o permanencia. Asimismo, mediante la aplicación de un algoritmo genético se optimiza la estructura de la red con el fin de determinar su profundidad adecuada. Con este trabajo, se establecen algunas bases para realizar distintos estudios empíricos que mejoren y profundicen las temáticas desarrolladas.

**Palabras claves** Inteligencia artificial, machine learning, red neuronal artificial, algoritmo genético, perceptron, análisis técnico, reglas de decisión.

# Abstract

In this thesis, several machine learning techniques are explored and applied to the investment decision problem. Some indicators of technical analysis are used as inputs of a neural network which is trained to determine, with each input vector, a buy, sell or hold signal. Likewise, the structure of the network is optimized by applying a genetic algorithm, in order to determine its adequate depth. With this work, some bases are established to perform different empirical studies that improve and deepen the analyzed topics.

**Keywords** Artificial intelligence, machine learning, artificial neural network, genetic algorithm, perceptron, technical analysis, decision rules.

# Índice general

# Índice de figuras

# Índice de cuadros

# Capítulo 1

# Introducción

Las herramientas de inteligencia artificial abarcan una gran variedad de metodologías, por lo cual se desarrollará un modelo de trading algorítmico estructurado sobre la metodología más adecuada para la dinámica de un determinado portafolio, de tal forma que se pueda maximizar la rentabilidad esperada a través de una estrategia de negociación particular, con parámetros claramente definidos que permitan realizar la ejecución automática de las decisiones de negociación.

Las redes neuronales tienen una gran capacidad para la clasificación y reconocimiento de patrones, debido a que son capaces de aprender y generalizar de la experiencia, por lo cual se han empleado en diferentes campos como en finanzas, ciencias, industria, etc.

Asimismo, en la actualidad los distintos participantes de los mercados financieros como inversionistas, traders, especuladores, etc., se enfrentan a un número cada vez mayor de fuentes y volúmenes de información, lo que dificulta aún más generar análisis adecuados y determinar la veracidad y pertinencia de dicha información a la hora de tomar decisiones.

Debido a esto, la presente investigación pretende articular los beneficios que ofrece un sistema de aprendizaje como las redes neuronales con las fuentes de información disponibles en el mercado financiero, de manera que sea posible realizar un análisis adecuado de las señales del mercado para la negociación estructurada y sistemática en una fracción de tiempo de acuerdo a un conjunto de instrucciones.

Una metodología de uso generalizado para comprender la información que aportan los mercados financieros se basa en aplicar el análisis técnico, el cual puede entenderse como una colección de algoritmos y reglas mecánicas que intentan ayudar a los inversores a pronosticar futuros movimientos de mercado. En este trabajo se busca diseñar un sistema capaz de vincular la información que aportan distintas herramientas y variables con sistemas de

aprendizaje automático, de tal forma que permita generar un resultado para la toma de decisiones de inversión en el mercado de divisas USD-COP.

Igualmente, se utilizan técnicas de análisis genético para realizar la optimización de los parámetros de la red neuronal, determinando la estructura más adecuada para obtener mejores combinaciones de las señales del mercado.

Esta tesis está organizada en seis capítulos, siendo la introducción el primero de ellas, en el segundo capítulo se realizará un revisión de la literatura que será la base de esta investigación, en el tercer capítulo se explicarán las características generales de una red neuronal, en el cuarto se muestran los principios fundamentales para realizar un algoritmo genético, en el quinto capítulo se explica el diseño de la herramienta de negociación, y en el capítulo sexto se presentan algunas consideraciones finales y recomendaciones para desarrollo futuro.

# Capítulo 2

# Revisión de Literatura

Esta tesis desarrolla un algoritmo de aprendizaje automático para la negociación en el mercado de valores por medio de la predicción de la dirección futura del precio de activos financieros que conforman el portafolio objeto de estudio. Por lo anterior en este capítulo se realizará una síntesis de algunos estudios previos que utilizan inteligencia artificial para la construcción de estrategías de negociación, igualmente se presenta una revisión de las teorías y metodologías que sirven como base para la estructuración del modelo de negociación automático.

## 2.1 Machine learning

El aprendizaje automático (Machine Learning) se puede definir como una rama de la inteligencia artificial AI (por sus siglas en inglés, Artificial Intelligence) relacionada con la construcción de programas que aprenden de la experiencia como se puede encontrar en [9], en general estas técnicas están orientadas a resolver problemas complejos, prediciendo el comportamiento de un sistema particular que obedece a situaciones que pueden o no ser observadas (o se puede emplear con el fin de comprender un comportamiento ya observado).

Ahora bien, dentro del ambiente de AI es posible identificar dos enfoques principales: AI simbólica (convencional) y AI sub simbólica (inteligencia computacional). La AI convencional hace uso de operaciones lógicas y reglas para tomar decisiones, es decir, pretenden deducir consecuencias a partir del conocimiento del fenómeno empleando análisis formal y estadístico. Los sistemas expertos y las redes bayesianas son algunos ejemplos de técnicas convencionales de AI, también conocidos como enfoques de top-down porque a partir de la información y el conocimiento del fenómeno intentan realizar

una representación simbólica (una analogía del funcionamiento del sistema).

La inteligencia computacional (computación blanda) se inspira en los mecanismos biológicos y utiliza un enfoque ascendente, es decir, a partir de señales que se identifiquen en la información de un fenómeno particular buscan generar un sistema inteligente imitando el funcionamiento de los cerebros biológicos. Las redes neuronales, algoritmos genéticos, sistemas difusos, etc., son algunos ejemplos de técnicas de inteligencia computacional (ver más en [11]).

De igual forma, las finanzas computacionales se caracterizan por ser una combinación tanto de infraestructura computacional como de herramientas de aprendizaje automático, y las emplea con el fin de dar soluciones a distintas problemáticas concernientes al ámbito financiero.

Este tipo de técnicas se emplean en la presente investigación con el fin de encontrar una herramienta factible que facilite la toma de decisiones de inversión y gestión de portafolios (en este caso el enfoque se basa en el mercado de Divisas USD-COP).

Siguiendo a [5], tanto las herramientas de finanzas computacionales como las derivadas de la AI pueden ser utilizados para abordar distintos problemas planteados comunmente en el entorno financiero, tales como predicción de series financieras, comportamiento de los mercados bursátiles, administración de riesgos, etc.

Un ejemplo de este tipo de aplicaciones expuesto por [4], consiste en alternar tendencias ascendentes y descendentes, identificando niveles que representan máximos o mínimos locales de los precios de un instrumento financiero. Uno de los objetivos más importantes de las técnicas de pronóstico del mercado financiero es detectar estos niveles de manera consistente y correcta. Dichos movimientos de precios no son regulares y la aplicación de herramientas comunes de pronóstico no proporcionan resultados satisfactorios.

Ahora bien, en el aprendizaje automático existen tres paradigmas diferentes que permiten abordar las distintas problemáticas de acuerdo a sus características particulares; supervisado, no supervisado, y de refuerzo. El primero de estos ha sido el más utilizado y desarrollado, en términos generales son algoritmos que buscan descubrir y evaluar hipótesis generales sobre las observaciones suministradas externamente (ver [9]). En otras palabras, con este tipo de técnicas se busca deducir la función que más se ajuste al comportamiento de los datos a partir de una muestra determinada (conjunto de entrenamiento), en donde tanto los valores de entrada y salida son conocidos.

El análisis no supervisado, por su parte, cuenta con datos de entrada pero no con realizaciones para alguna variable de respuesta, por lo cual requiere de algoritmos capaces de aprender de la estructura de datos suministrada.

Finalmente, el aprendizaje de refuerzo se emplea con el fin de que el sistema sea capaz de clasificar diferentes situaciones para tomar decisiones.

La técnica empleada en este trabajo está basada en el aprendizaje supervisado puesto que es el que más se ajusta a las características del fenómeno que se pretende abordar.

## 2.2 Análisis técnico

Esta investigación se basa en el uso eficiente de la información disponible en el mercado de divisas USD-COP, recogida en una serie de indicadores que utilizan el análisis técnico para comprender de una mejor forma su comportamiento. Por lo anterior, es relevante revisar las características fundamentales que estructuran este tipo de análisis y la forma en la que se pueden utilizar para generar herramientas valiosas en la negociación financiera.

El análisis técnico se concentra en el estudio de las variaciones que se presentan en el mercado, mediante el uso de herramientas gráficas que ilustren el comportamiento histórico de los activos financieros con el propósito de prever las tendencias de su conducta futura (ver [16]).

Alternativamente existe otro enfoque que se suele emplear como complemento al análisis técnico: el análisis fundamental. Este tipo de análisis se basa en las fuerzas macroeconómicas de la oferta y la demanda que afectan el comportamiento de los mercados. El análisis fundamental busca determinar el valor intrínseco de los activos bajo la ley de oferta y demanda para compararlo con su valor de mercado y tomar decisiones de inversión.

Así, es claro que con ambos enfoques se busca resolver el mismo cuestionamiento; la dirección en que se moverán los precios, pero utilizan enfoques totalmente diferentes. Aunque en muchos casos ambos enfoques se complementan dando sentido a las realizaciones en el mercado, existen momentos de discrepancia y disociación que reflejan que posiblemente es el precio de mercado el que desencadena el comportamiento de los fundamentos conocidos, actuando como un indicador de los fundamentos y no siendo el resultado de los conocimientos convencionales.

### 2.2.1 Indicadores de análisis técnico

Existe un amplio conjunto de *indicadores* que facilitan la comprensión de la información del mercado financiero, los cuales se pueden clasificar en dos grandes grupos: indicadores de *confirmación (o divergencia)* e indicadores de *momentum*.

#### Indicadores de confirmación

También se conocen como indicadores de tendencia porque permiten confirmar o no la tendencia de un activo, sin embargo cuando el indicador presenta un comportamiento que no se corresponde con el presentado por la serie de precios específica (en dirección opuesta), se encuentra una señal de divergencia que permite advertir cambios en el comportamiento de la serie.

Los indicadores de este tipo más comunes son: los indicadores de volumen, rendimiento relativo y promedios móviles.

#### Indicadores de momentum

Este tipo de indicadores miden la velocidad de los movimientos de los precios a través de la tasa de cambio, es decir, representan (de manera absoluta o relativa) los cambios ocurridos en el precio de un activo durante un periodo de tiempo determinado. Dentro de los indicadores de momentum que suelen utilizarse con mayor frecuencia se tienen: el índice de fuerza relativa, los promedios móviles de convergencia/divergencia y los osciladores estocásticos.

## 2.3 Revisión teórica

Las finanzas se estructuran sobre un conjunto de teorías y principios en los que se suele asumir un comportamiento racional de los inversionistas, a continuación se hace un repaso de las principales hipótesis y teorías que constituyen un importante soporte para el desarrollo de la presente investigación, igualmente se presenta un breve recuento de algunos de los trabajos que han tratado temas relacionados con aplicaciones de AI en series de tiempo financieras y que servirán de base y guía para el desarrollo de este documento.

### 2.3.1 Hipótesis de Mercado Eficiente

La hipótesis de mercado eficiente sostiene que el precio de mercado de un determinado activo constituye una aproximación o estimación confiable al reflejar su precio teórico, debido a que se basa en el supuesto de que los inversionistas o participantes del mercado actuan de forma racional. Esto implica que los precios de mercado recogen toda la información disponible en el mercado.

Sin embargo, si bien es un principio ampliamente conocido en el mercado, es igualmente criticado al suponer que los precios del mercado reflejan toda la información disponible, puesto que no existe un mercado perfectamente eficiente en el que el comportamiento de sus partipantes sea estrictamente

racional, de ser así no se presentarían oportunidades de beneficios financieros, y no habría incentivo alguno para generar metodologías que traten de entender el comportamiento de los mercados y así inferir las tendencias futuras que puedan presentar los activos financieros con base en las señales obtenidas de su comportamiento histórico [20].

### 2.3.2 Hipótesis de mercado adaptativo

Esta hipótesis adopta algunos principios de la hipótesis de mercado eficiente, pero abordando y proponiendo una hipótesis alternativa que pretende explicar y entender de una manera más fiel el comportamiento real de los participantes del mercado financiero, introduciendo principios conductuales biológicamente irracionales presentes en las finanzas.

De hecho y de acuerdo con lo establecido por [14]

> *"debido a que las emociones son la base para el correcto funcionamiento del sistema de recompensa-castigo, sobre el cual se basa la selección evolutiva del comportamiento, no es posible, al menos no sin que el análisis resulte incompleto, estudiar el comportamiento de los agentes de los mercados financieros, y en general de los seres humanos, tan solo desde la perspectiva del comportamiento racional"*

Esta connotación particular permite entender la dinámica del mercado de una forma más compleja en presencia de ciclos, tendencias, pánico, burbujas, etc., análogo al comportamiento biológico de un ecosistema, entendiendo al agente económico participante del mercado como un agente con racionalidad limitada e imperfecta abandonando la idea de que los individuos en el mercado actuan bajo un principio de expectativas racionales.

### 2.3.3 Estado del arte

Actualmente la negociación en el mercado financiero empleando distintas herramientas (como algoritmos, procesos, etc.) ha cobrado mayor atención por parte no solo de la academia sino también de los agentes del mercado que buscan generar mecanismos cada vez más automatizados para el procesamiento de la información y la gestión de las carteras de inversión. En este trabajo se han tomado como referencia dos investigaciones en las cuales se realiza una integración de dos técnicas de AI: algoritmo genético y redes neuronales.

El primer trabajo que se toma como referencia es una investigación en la que se utilizan sistemas inteligentes para aplicarlos a la negociación de divisas, específicamente se emplean redes neuronales y análisis genético para la

optimización de las reglas de decisión generadas por el análisis técnico para crear un sistema de negociación en el mercado de diversas divisas, como por ejemplo el EUR, CHF, ZAR, CAD, AUD, entre otras (se puede consultar en [19]).

Similarmente en el segundo trabajo se utilizan tanto redes neuronales como análisis genético pero aplicadas a la minimización del valor en riesgo de un portafolio de activos financieros, en este caso el algoritmo genético se utiliza para la minimización de la función objetivo (VaR) más no para el mejoramiento de los parámetros de reglas de decisón o la optimización de la estructura neuronal (ver más en [15]).

# Capítulo 3

# Redes Neuronales

## 3.1 Introducción

En este capítulo se presentan las generalidades y nociones básicas relacionadas con la estructuración de las redes neuronales, especificando los fundamentos teóricos y matemáticos que serán la base de la metodología empleada en la presente investigación.

## 3.2 Generalidades

Las redes neuronales artificiales ANN (por sus siglas en inglés Artificial Neural Network) se inspiran en el comportamiento de los sistemas biológicos [2] que intenta simular los principios operativos de la red de neuronas del sistema nervioso biológico como se observa en la figura 3.1 [1], en la que se esboza la estructura elemental de una neurona biológica.

La principal ventaja de utilizar ANN es su capacidad para capturar la no

[1]Derechos de autor http://cs231n.github.io/neural-networks-1/

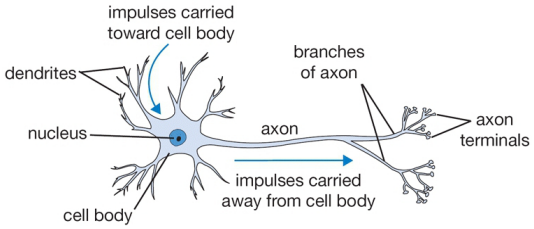



Figura 3.1: Esquema neurona biológica

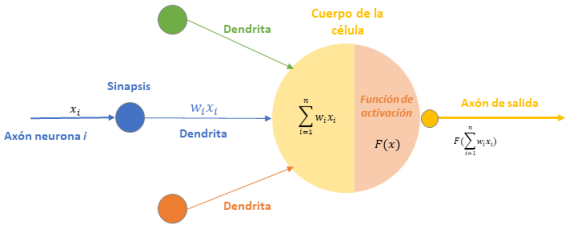



Figura 3.2: Esquema del modelo de representación de la neurona.
***Fuente:*** *elaboracón propia.*

linealidad sin conocimiento previo sobre las relaciones funcionales entre las variables. Las ANN operan como sistemas de aproximación universal con la capacidad de imitar casi cualquier dependencia de funciones. Son resistentes a valores atípicos y no requieren una distribución específica. En comparación con los modelos econométricos ANN dan resultados en un período de tiempo más corto.

El incoveniente de la aplicación de las ANN es la ausencia de un paradigma estándar para diseñar la red. Para las ANN, la principal desventaja en la utilización se describe por el concepto de "caja negra". Es difícil identificar la influencia de una variable de entrada seleccionada sobre la salida de la red, igualmente cuando se usan ANN, se requiere un conjunto de datos de cierta longitud para evitar sobreajustar la red durante una fase de aprendizaje (ver más en [4]).

Ahora bien, la estructura de una ANN se basa en un conjunto de nodos (neuronas) interconectadas distribuidas en una o más capas ocultas, las cuales tienen asociados unos pesos determinados $w_i$ que representan la fuerza de la señal pasada por cada input a cada nodo.

La fuerza de estas conexiones se van mejorando con base en las características escenciales de los datos de entrada a la red, y se van ajustando por medio de algoritmos de aprendizaje, los cuales mediante el empleo de procesos iterativos, permiten actualizar el valor de los pesos de las conexiones entre cada una de las unidades ocultas de la red, optimizando el valor de cada uno de los pesos $w_i$, como se puede observar en la figura 3.2.

### 3.2.1 Perceptron

A continuación introducimos el concepto del *perceptron* el cual constituye el elemento más simple pero fundamental para comprender el diseño de una red

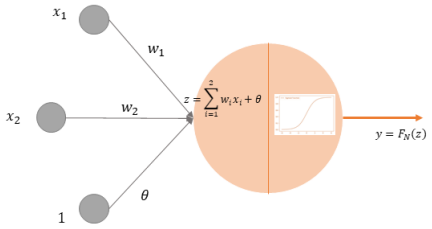



Figura 3.3: Modelo esquemático de un perceptron, sin capas ocultas y con una única salida.
***Fuente:*** *elaboracón propia.*

neuronal compleja.

El perceptron es la representación del modelo matemático más simple que ejemplifica el funcionamiento de un neurona biológica, esta compuesto por un número determinado (que puede ser indefinido) de *inputs* y una salida, esto significa que la unidad de salida estará directamente conectada con las unidades de entrada, tal como lo muestra la figura 3.3.

Como se ilustra en la figura 3.3 en este caso el perceptron está compuesto por un vector de entrada de dos elementos $[x_1, x_2]$, un término de error $\theta$ y una única salida (puede tener más de una salida). El algoritmo de aprendizaje del perceptron vendría definido de la siguiente manera (ver [2]):

1. Inicializar los pesos (interconexiones) aleatoriamente.

2. Elegir un vector de entrada inicial dentro de la muestra de entrenamiento y estimar la salida de la red con los pesos establecidos y las entradas inciales así definidas.

3. Contrastar la respuesta obtenida con la salida real del sistema, si ésta es $\hat{y} \neq y$ se deben ajustar las conexiones establecidas inicialmente $w_i$ con el factor de ajuste $\Delta(w_i) = yx_i$.

4. Repetir el proceso desde el paso 1.

El nodo del perceptron realiza la combinación lineal por medio de una sumatoria de cada entrada y su peso correspondiente $w_i$, y el resultado de esta suma ponderada pasa por una función de activación para obtener la salida del sistema (ver más en [8]). la figura 3.3 ilustra el sistema descrito.

Una red neuronal se puede entender como un red de perceptrones, es decir, es la combinación de varios perceptrones que se interconectan, en donde

la salida de la sumatoria que se denota como $z_i$, y que realiza la combinación lineal de los inputs $(x_{li}...x_{ni})$ y sus pesos respectivos para el *i-ésimo* perceptron, viene definida por la siguiente relación:

$$z_i = \sum_{j=1}^{m} w_{ij} x_{ij} \tag{3.1}$$

Donde $w_{ij}$ representa el peso (optmizable) de la *j-ésima* entrada al *i-ésimo* nodo (recuerde que una red es una combinación de perceptrones por lo tanto se tendrán varios nodos). Cada una de éstas salidas de las sumatorias $z_i$ se pasan por una función de activación para obtener la salida de la capa final de la red, este proceso se conoce como *forward propagation*.

## 3.3 Algoritmo back propagation

Rumelhart, Hinton y Williams [18] propusieron el algoritmo de Back Propagation (BP) que permite establecer los pesos adecuados para cada una de las ramas de la red y por ende permite implementar el entrenamiento de la ANN.
En primer lugar se desarrollará el algoritmo BP para la capa de salida de la red, debido a que (como se explicará más adelante) para esta etapa se puede estimar el error de las salidas de la ANN.

Para el entrenamiento de la ANN se establece una función de costo o de error que permitirá establecer el error en cada simulación y con esta información rebalancear los pesos que interconectan las capas de la red. Se trabajará con la función de error cuadrático medio, definida así:

$$\varepsilon = \frac{1}{2} \sum_{k} (d_k - \hat{y}_k)^2 = \frac{1}{2} \sum_{k} e_k^2 \tag{3.2}$$

Para $k = 1, 2, 3, ..., N$, donde $d_k$ es el valor de $y$ observado, es el único valor disponible para el que se tiene observaciones (para el caso específico de este trabajo solo se tendrá una salida que representa la señal de compra, venta o mantenimiento. $k = 1$), $\hat{y}$ es el valor estimado que se obtiene al aplicar el proceso de forward propagation con el cual se llega a la salida de la ANN para el primer conjunto de entrenamiento y $k$ representa el número de neuronas que se encuentran en la capa de salida de la ANN, es decir es el número de salidas que tiene la ANN.

Una vez estimado el valor de $\hat{y}$ se calcula el error del modelo para la primera iteración y se aplica el procedimiento BP propagando la optimización (en este caso la minimización de la función de error) a través de todas las capas ocultas (y neuronas) hasta llegar a la primera capa de la ANN.

Dado que el objetivo del algoritmo de BP es entrenar la red para que se genere el proceso de aprendizaje de la misma, y teniendo en cuenta que cada peso de la red aporta información relevante para la configuración del error, se procede a calcular el gradiente de $\varepsilon$ respecto a cada uno de los pesos, esto servirá para rebalancear los pesos que interconectan la red:

$$\bigtriangledown \varepsilon_k = \frac{\partial \varepsilon}{\partial w_{kj}} \tag{3.3}$$

Donde $w_{kj}$ denota el peso que va del *j-ésimo* input a la *k-ésima* neurona de la capa de salida de la red. Ahora bien para poder actualizar los valores de los pesos interconectores y de esta forma optimizar el sistema de la red, se empleará la técnica de steepest gradient descent, la cual estipula que:

$$w_{kj}(m+1) = w_{kj}(m) + \Delta(w_{kj}(m)) \tag{3.4}$$

El peso $w_{kj}$ que conecta el input $j$ con la neurona $k$ en la iteración (m+1) equivale al peso $w_{kj}$ en la iteración (m) más el cambio o variación de $w$ representada por el término $\Delta(w_{kj})$.

La variación de $w$ en $m$ viene dada por la relación que existe con el gradiente de $\varepsilon$:

$$\Delta(w_{kj}) = -\eta \frac{\partial \varepsilon}{\partial w_{kj}} \tag{3.5}$$

Donde $\eta$ representa la tasa de aprendizaje, y el gradiente de $\varepsilon$ ($\bigtriangledown \varepsilon$) está definido por la relación expuesta en la ecuación (3.3). De la definición de *perceptron* en la sección anterior se tiene que:

$$\hat{y_k} = F_N(z_k) \tag{3.6}$$

$$z_k = \sum_j w_{kj} x_j \tag{3.7}$$

Donde ($F_N$) representa la función de activación, esta función permite mantener la salida de cada neurona dentro de determinados límites e igualmente, como su nombre lo indica, determina si la neurona activa o no una señal de acuerdo al impulso que reciba. Existen varias funciones que pueden cumplir con estas propiedades, dentro de las que se encuentran, la función

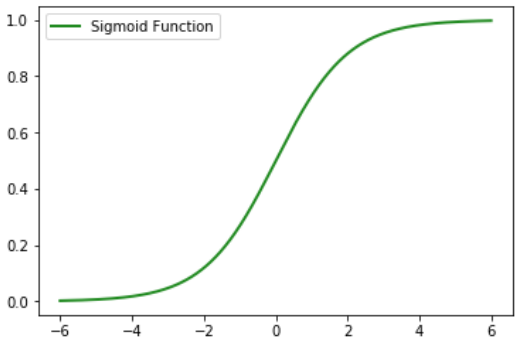



Figura 3.4: Función de activación sigmoidea
***Fuente:*** *elaboracón propia.*

tangente, la función unipolar, la función de activación binaria (0,1), la función sigmoidea, entre otras (para ampliar ver [8]).

Se utiliza como función de activación ($F$) para la ANN la función sigmoidea debido a que es una función continuamente diferenciable, lo cual será de gran utilidad para la optimización de la red, e igualmente porque este tipo de funciones permite representar más fielmente el comportamiento biológico, puesto que la función permanecerá en cero y se irá incrementando gradualmente de acuerdo al impulso que ésta reciba llegando a ser asintótica (ver figura 3.4, y que puede representarse por la siguiente relación:

$$F_N(z_k) = \frac{1}{1 + exp(-z_k)} = \hat{y}_k \tag{3.8}$$

Como se puede observar:

$$z_k \to -\infty \Leftrightarrow \hat{y}_k \to 0; z_k = 0 \Leftrightarrow \hat{y}_k = 0{,}5; z_k \to \infty \Leftrightarrow \hat{y}_k \to 1$$

El gradiente de $\varepsilon$ consiste en hallar la derivada parcial de la función de costo con respecto a cada peso $w_{kj}$, para lo cual será útil emplear la regla de la cadena:

$$\frac{\partial \varepsilon}{\partial w_{kj}} = \frac{\partial \varepsilon}{\partial \hat{y}_k} \frac{\partial \hat{y}_k}{\partial z_k} \frac{\partial z_k}{\partial w_{kj}} \tag{3.9}$$

Por la ecuación (3.7) la derivada de $z_k$ respecto a $w_{kj}$, puede expresarse así:

$$\frac{\partial z_k}{\partial w_{kj}} = x_j(p) \tag{3.10}$$

Donde $p$ hace referencia a la capa de salida de la red (última capa). Notese que en la ecuación (3.7) $x_j$ esta definido como el *j-ésimo* imput que llega a la *k-ésima* neurona de la capa de salida, por lo cual se puede enteder que $x_j$ es igual a la salida de la *j-ésima* neurona de la capa inmediatamente anterior a la capa de salida (es decir de la capa $p-1$), por lo tanto la ecuación (3.10) puede verse así:

$$\frac{\partial z_k}{\partial w_{kj}} = \hat{y_j}(p-1) \tag{3.11}$$

Si definimos un término $\Phi_k$ de la siguiente forma:

$$\Phi_k(p) = -\frac{\partial \varepsilon}{\partial \hat{y}_k}\frac{\partial \hat{y}_k}{\partial z_k} \tag{3.12}$$

Entonces se puede expresar la ecuación (3.9) de la siguiente manera:

$$\frac{\partial \varepsilon}{\partial w_{kj}} = -\Phi_k(p)\hat{y_j}(p-1) \tag{3.13}$$

Y tomando la ecuación (3.13) y (3.5), se obtiene que:

$$\Delta(w_{kj}) = \eta\Phi_k(p)\hat{y_j}(p-1) \tag{3.14}$$

Ahora bien, la única incógnita que queda por despejar es el término $\Phi_k(p)$, recordando la ecuación (3.12) y la ecuación (3.2), se tiene que:

$$\frac{\partial \varepsilon}{\partial \hat{y}_k} = -\left(d_k - \hat{y}_k\right) = \hat{y}_k - d_k \tag{3.15}$$

Por otro lado, teniendo en cuenta la ecuaciones (3.12) y (3.8), se tiene que:

$$\frac{\partial \hat{y}_k}{\partial z_k} = \hat{y_k}(1-\hat{y_k}) \tag{3.16}$$

Así pues con la ecuaciones (3.16), (3.15) y (3.12), obtenemos que $\Phi_k$ queda definido como:

$$\Phi = \hat{y_k}(1-\hat{y_k})(d_k - \hat{y}_k) \tag{3.17}$$

Utilizando las ecuación (3.13) y recordando la ecuación (3.5), se obtiene:

$$\begin{aligned}\Delta(w_{kj})(p) &= \eta\Phi_k(p)\frac{\partial z_k}{\partial w_{kj}} \\ &= \eta\Phi_k(p)\hat{y}_j(p-1)\end{aligned} \tag{3.18}$$

Donde $p$ representa la capa de salida de la red. Así se completa la derivación de la configuración de los pesos de esta última capa. El mismo principio debe aplicarse para obtener la configuración de los pesos para las capas ocultas de la red, com se verá a continuación.

**Algoritmo de back propagation para las capas ocultas de la red** De forma similar a la derivación anterior, para la *r-ésima* capa oculta se tiene:

$$\Delta(w_{ji}) = -\eta\frac{\partial\varepsilon}{\partial w_{ji}} \tag{3.19}$$

Donde para cada *r-ésima* capa oculta, $w_{ji}$ representa el peso para de la *i-ésima* rama en la *j-ésima* neurona. Si se supone que:

$$\begin{aligned}\Phi_k &= -\frac{\partial\varepsilon}{\partial z_k} \\ &= -\frac{\partial\varepsilon}{\partial\hat{y}_k}\frac{\partial\hat{y}_k}{\partial z_k}\end{aligned} \tag{3.20}$$

Teniendo en cuenta el resultado obtenido en la ecuación (3.11), la ecuación (3.19) puede expresarse como sigue:

$$\begin{aligned}\Delta w_{ji} &= -\eta\frac{\partial\varepsilon}{\partial z_j}\hat{y}_i(r-1) \\ &= \eta\Phi_j(r)\hat{y}_i(r-1)\end{aligned} \tag{3.21}$$

Teniendo en cuenta la relación mencionada en la ecuación (3.20), se puede ver la relación de $\Delta(w_{ji})$ de la siguiente manera:

$$\Delta w_{ji} = -\eta\left[\frac{\partial\varepsilon}{\partial\hat{y}_j(r)}\frac{\partial\hat{y}_j}{\partial z_j}\right]\hat{y}_i(r-1) \tag{3.22}$$

Tenga en cuenta que el término $\frac{\partial\varepsilon}{\partial\hat{y}_j}$, representa la derivada parcial del error con respecto a la salida de la *j-ésima* neurona de la *r-ésima* capa oculta, por tal razón no es posible calcularlo directamente puesto que no existe ninguna observación deseada (realización) para ninguna capa oculta.

Sin embargo, dadas las interconexiones que existen con la capa de salida de la red, para la cual el término de error $\varepsilon$ se puede estimar, la expresión $\frac{\partial \varepsilon}{\partial \hat{y}_j}$ puede entenderse de la siguiente manera:

$$\begin{aligned}\frac{\partial \varepsilon}{\partial \hat{y}_j(r)} &= \sum_{k=1}^{k_{(r+1)}} \frac{\partial \varepsilon}{\partial z_k(r+1)} \left[\frac{\partial z_k(r+1)}{\partial \hat{y}_j(r)}\right] \\ &= \sum_{k=1}^{k_{(r+1)}} \frac{\partial \varepsilon}{\partial z_k(r+1)} \left[\frac{\partial}{\partial \hat{y}_j(r)} \sum_{m=1}^{m_r} w_{km}(r+1)\hat{y}_m(r)\right]\end{aligned} \tag{3.23}$$

La suma sobre $k$ se realiza sobre las neuronas de la siguiente capa $(r+1)$ que se conecta con $\hat{y}_j(r)$, es decir la sumatoria definida así: $\sum_k^{k_{(r+1)}}$, se realiza desde $k = 1$ (donde $k$ representa una neurona en la capa $(r+1)$), hasta $k_{(r+1)}$, donde $k_{(r+1)}$ representa el número total de neuronas $k$ que se encuentran en la capa $(r+1)$ y que se conectan con la salida de la neurona $j$ en la capa $(r)$, $\hat{y}_j$.

Por otro lado la suma sobre $m$ se realiza sobre todas las entradas a cada *k-ésima* neurona de la capa (r+1), es decir la sumatoria se realiza desde $m = 1$ que representa una neurona perteneciente a la capa $(r)$, hasta $m_r$ que representa el número total de neuronas que hay en la capa $(r)$.

El funcionamiento de este tipo de ANN se puede observar en la figura 3.5 donde se muestra como está compuesta una red neuronal con capas ocultas intermedias y con varias neuronas por capa (engranaje de perceptrones).

La derivada definida por la expresión $\left[\frac{\partial z_k(r+1)}{\partial \hat{y}_j(r)}\right]$ se puede calcular, por lo que se obtiene que la ecuación (3.23) se puede expresar como:

$$\frac{\partial \varepsilon}{\partial \hat{y}_j(r)} = \sum_{k=1}^{k_{(r+1)}} \frac{\partial \varepsilon}{\partial z_k(r+1)} w_{kj} \tag{3.24}$$

Partiendo de la ecuación (3.23) y teniendo en cuenta la definición dada en la ecuación (3.20), se obtiene que:

$$\frac{\partial \varepsilon}{\partial \hat{y}_j(r)} = -\sum_{k=1}^{k_{(r+1)}} \Phi_k(r+1) w_{kj}(r+1) \tag{3.25}$$

Puesto que solo $w_{kj}(r+1)$ está conectado con $\hat{y}_j(r)$.

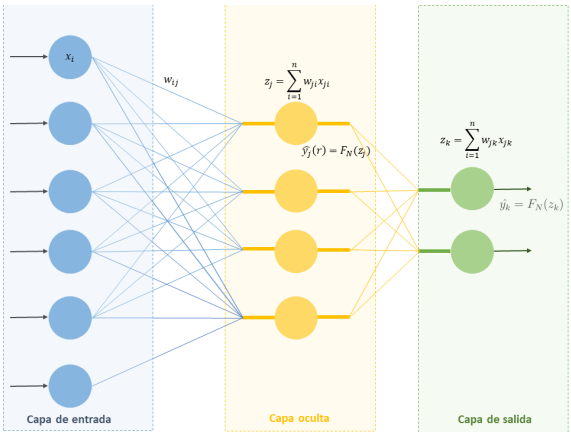



Figura 3.5: Modelo esquemático de una red neuronal artificial, con capas ocultas intermedias.
***Fuente:*** *elaboracón propia.*

Ahora bien, considerando la ecuación (3.25), (3.20) y (3.16), se tiene que $\Phi_j(r)$ es igual a:

$$\begin{aligned}\Phi_j(r) &= \frac{\partial \hat{y}_j}{\partial z_j} \sum_{k=1}^{k_{(r+1)}} \Phi_k(r+1) w_{kj}(r+1) \\ &= \hat{y}_j(r)\left[1-\hat{y}_j(r)\right] \sum_{k=1}^{k_{(r+1)}} \Phi_k(r+1) w_{kj}(r+1)\end{aligned} \tag{3.26}$$

De esta ecuación (3.26) y recordando las ecuaciones (3.21) y (3.19), se obtiene:

$$\Delta(w_{ji})(r) = \eta \Phi_j(r) \hat{y}_i(r-1) \tag{3.27}$$

De esta forma se obtiene el algoritmo de back-propagation para cualquier *r-ésima* capa oculta (ver [8]), partiendo del hecho que no se puede estimar derivadas parciales de $\varepsilon$ con respecto a las capas ocultas consideradas, por lo cual es necesario calcular derivadas parciales de $\varepsilon$ con respecto a las variables que ascienden por la red hacia la capa de salida que son las que efectivamente están afectando el comportamiento de $\varepsilon$.

# Capítulo 4

# Algoritmo Genético

## 4.1 Introducción

Los algoritmos genéticos GA (por sus siglas en inglés, Genetic Algorithm) son técnicas que nos permiten dar solución a un determinado problema empleando los principios básicos de la evolución biológica, utilizan la reproducción para generar descendencia apta que pueda sobrevivir en el entorno.

Análogamente el GA selecciona los individuos basado en el principio de supervivencia del más apto para mejorar la calidad de las soluciones al problema deseado (ver [13]). A continuación se presentan las consideraciones fundamentales del funcionamiento del algoritmo genético, brindando al lector una comprensión de la flexibilidad disponible dentro de esta metodología y de los aspectos que se deben tener en cuenta para utilizarlos en conjunto con la red neuronal.

## 4.2 Descripción

Los algoritmos genéticos se pueden utilizar para realizar la selección adecuada de características, aunque también se puede emplear para buscar el ajuste óptimo de los parámetros dentro de un espacio de parámetros multidimensional que se emplean dentro de una determinada función.

En la reproducción sexual de la descendencia, se seleccionan dos padres y el cromosoma de la descendencia se construye mediante la combinación de secciones de los cromosomas de los padres. Hay una pequeña probabilidad de que pueda producirse una mutación en la descendencia antes de que se establezca su genotipo. Este genotipo produce un fenotipo para la descendencia. La "supervivencia del más apto"de Darwin generalmente determina qué

padres se usan para cruce y si la descendencia es lo suficientemente viable como para convertirse en padre (ver más en [13]).

Para implementar un GA, se deben imitar los principios básicos del comportamiento de la evolución biológica, [13] propone seguir la siguiente serie de pasos:

1. Crear una población inicial de vectores genéticos y calcular su ajuste (o aptitud).

2. Elegir dos miembros de esta población en función de su aptitud para convertirse en padres.

3. Utilizar un operador de cruce para construir un nuevo vector genético de los padres.

4. Usar un operador de mutación para cambiar probabilísticamente el vector genético.

5. Calcule la aptitud de esta descendencia y pídale que reemplace al miembro más débil de la población.

6. Repetir desde el paso 2 hasta que se haya producido un número suficiente de descendientes.

Este algoritmo es relativamente rápido, sin embargo es probable que no converja en la solución óptima global, debido principalmente a que el procedimiento realiza una búsqueda local en torno a las soluciones más adecuadas en la población actual. Si estos miembros en buen estado se acercan a una solución buena, pero no la mejor, el problema se llama engañoso y no se encontrará la mejor solución (ver [13]).

### 4.2.1 Función de aptitud

Los GA utilizan un procedimiento de selección en el cual se elige a los padres en una cantidad proporcional a su estado físico. Esto significa que si se usan métodos de selección basados en la aptitud física, las mejores soluciones deben tener una mejor forma física y cada miembro (vector genético) en la población debe tener una aptitud física no negativa.
Esto también significa que el GA se puede entender desde este punto de vista como un algoritmo de maximización ya que selecciona las características de tal manera maximizar la aptitud de la descendencia y la población en general.

El objetivo del GA puede ser tanto maximizar el valor de la solución, representada por ejemplo por una función $J(X)$ donde $X$ representa un determinado vector genético con elementos $x_i$ ($X = x_i$), como minimizar el valor de una función objetivo, todo depende de la naturaleza del problema que se quiera trabajar. En el presente estudio, el empleo del GA tiene como objetivo minimizar la función de costo $\varepsilon(X)$ de la ANN en cuyo caso se pretende emplear el GA para seleccionar las características o parámetros que minimicen dicha función.

### 4.2.2 Vectores genéticos

Los vectores genéticos representan los cromosomas del organismo y consisten en una serie de genes. A continuación se describirá el esquema de codificación que se empleará en la presente investigación.

Existen varios métodos de codificación, basada en genes, basada en nodos o la codificación delta, en este trabajo se usará la codificación basada en genes, en la cual siguiendo a [13] existe una correspondencia uno a uno entre el número de genes y una característica particular del problema que se pretende solucionar.

Así mismo es posible emplear más de un tipo de codificación para un mismo problema en particular, sin embargo una consideración adicional que debe ser tenida en cuenta es la forma del alfabeto genético que depende tanto del problema a solucionar como del esquema de codificación empleado, como se puede encontrar en [7] se suele emplear la codificación binaria puesto que resulta más fácil construir un esquema.

La codificación binaria es la más común, sin embargo se debe tener en cuenta que a pesar de su amplio uso, se puede incurrir en errores que pueden afectar la solución óptima del algoritmo. Los cuales se relacionan con el efecto del operador de mutación, esto significa que si este operador simplemente cambia un bit y resulta ser el bit de orden superior (el que se encuentra más a la izquierda y puede producir las variaciones más grandes), la nueva cadena de bits podría representar en su valor entero un valor muy diferente al valor base, por lo cual resultaría muy complejo controlar la magnitud del cambio causado por una sola mutación.

Si se utiliza una probabilidad de mutación muy grande, los resultados obtenidos con el GA pueden ser diferentes aún cuando se este empleando el mismo conjunto de datos, debido a que en cada generación se podría tener un número relativamente alto (por ejemplo si esta probabilidad es mayor a 5 %, en promedio más del 5 % de los individuos mutarían en cada población) de individuos que van a mutar sus características, lo cual impide tener

consistencia suficiente en la optimización.

Esta característica se conoce como *Hamming cliff* (ver [13]), debido principalmente a esta razón se va emplear la codificación Gray para solucionar el problema, esta codificación es un sistema de numeración binario en el que dos valores consecutivos difieren solamente en uno de sus dígitos, en otras palabras representa una asignación uno a uno no única entre cadenas de bits, de modo que si los números enteros resultantes difieren en uno, sus cadenas de bits codificadas Gray solo difieren en un bit (es decir, su distancia de Hamming es 1).

## 4.3 Población

Para implementar el GA después de definir las connotaciones pertinentes al vector genético se debe determinar la población inicial que va a conformar la primera generación,

Existen varios métodos empleados para determinar el tamaño de la población, uno de los más conocidos fue establecido por Goldberg en 1989, quien determinó que el tamaño óptimo de la población para longitudes de bit de codificación binaria $n_b$ crece exponencialmente de acuerdo al tamaño de $n_b$ (ver en [7]).

Sin embargo, la propuesta de Goldberg presenta mucha sensibilidad a la estructura utilizada, dependiendo de la longitud de bit que se emplee se podrían obtener tamaños de población relativamente grandes generando un costo computacional alto. Debido a esto, surgieron métodos alternativos para dar solución al problema de encontrar un tamaño de población adecuado.

Este es el caso de [1] en donde a través de pruebas y evidencias empíricas se determinó que tamaños de población definidos como lo muestra la ecuación (4.1) se ajustan adecuadamente a los problemas considerados.

$$n_b \leq S_{opt}(n_b) \leq 2n_b \tag{4.1}$$

Donde $S_{opt}(n_b)$ representa el tamaño óptimo de la población.

Sin embargo, no se tienen reglas fijas para determinar el tamaño de la población inicial para el GA, en la práctica se suelen emplear tamaños de población pequeños (de 20 a 100 individuos), una vez establecido el tamaño de la población inicial se generan (por lo general aleatoriamente) los vectores genéticos de cada individuo.

Utilizando la función de aptitud escogida (en este trabajo se trabaja con la función de error descrita en la ecuación (3.2)) se debe evaluar a cada individuo para realizar la selección, sin embargo dado que de la población inicial se deben escoger los mejores individuos para iniciar el proceso evolutivo

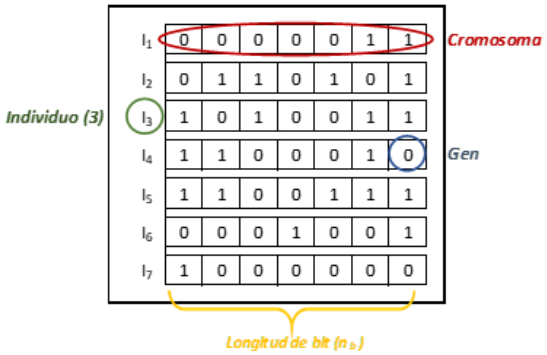



Figura 4.1: Representación de la población en un algortimo genético

de la siguiente generación, se pueden utilizar algunas técnicas para mejorar la aptitud promedio de la población inicial por medio de optimizaciones locales. Sin embargo se debe ser juicioso al implementar estas técnicas de mejora, puesto que se puede encontrar que la busqueda y el proceso de selección se realice dentro de un pequeño número de mínimos locales.

Esto significa que la busqueda de un mejor optimización para mejorar la aptitud de los individuos de la primera generación puede conducir a que se genere una busqueda dentro de una región que puede no contener el resultado más óptimo. Por lo cual sería recomendable en estos casos utilizar varias poblaciones iniciales para realizar la simulación de los individuos.

La figura 4.1 ilustra un poco mejor la descripción realizada hasta aquí sobre la población, los individuos (cromosomas) y los genes, en donde el tamaño del cromosoma representa la longitud de bit antes mencionada.

## 4.4 Selección y cruce

Una vez se ha establecido la población inicial, se debe elegir el método por medio del cual se seleccionarán los individuos que serán los padres de la siguiente generación así como los descendientes que la constituirán.

Como ya se mencionó en la introducción de este capítulo, se suele utilizar la aptitud de los individuos para seleccionar a los mejores, sin embargo también es posible realizar esta seleción eligiendo algunas soluciones de población y seleccionando aleatoriamente de este subgrupo los dos individuos que serán padres, así mismo existe la selección de torneo la cual consiste en seleccionar dos cromosomas de la población, y aleatoriamente generar un número entre [0,1], si este número es menor a un nivel predeterminado conocido como ta-

maño de torneo, se elige aquel individuo con el mejor ajuste, de lo contrario se seleccionaría el individuo más débil.

En este trabajo se utiliza como criterio de selección la aptitud de los individuos medida por la función de error (ver ecuación 3.2). Por lo cual en cada generación se medirá la aptitud de cada individuo y se escogeran los mejores (es decir, los que presenten menor error). El número de individuos seleccionado para ser padres de la siguiente generación se determina estableciendo un parámetro adicional: la *"tasa de cruce"*, $p_c$, siguiendo a [7] este parámetro suele ser del 20 %, por lo tanto, si se multiplica el tamaño de la población por la tasa de cruce se obtiene el número o cantidad esperada de cromosomas que serán padres de la siguiente generación y sobre los cuales se podrán realizar los cruces.

Ahora bien, los operadores de cruce toman información genética de dos padres y pueden crear uno o más descendientes, y así como existen varios y diversos métodos de selección también existen diferentes métodos para realizar el cruce de individuos, donde los segmentos de dos cromosomas de cruce se eligen al azar y se intercambian para producir una o más crías.

Igualmente es necesario tener en cuenta que dependiendo de los vectores genéticos que se empleen se utilizará una u otra metodología de cruce. A continuación se describen las metologías más comunes para realizar el proceso de cruce. La primera se conoce como cruce de punto único, y la segunda es el cruce multipunto.

**Cruce de punto único** Para realizar el cruce con está metodología se debe seguir el siguiente procedimiento:

1. Seleccionar dos cromosomas padres para la reproducción.
2. Seleccionar la posición del punto de cruce aleatoriamente.
3. Copiar aquellos genes que se ubiquen al lado izquierdo del punto de cruce del primer padre al cromosoma del hijo 1.
4. Copiar aquellos genes que se ubiquen al lado derecho del punto de cruce del segundo padre y agregarlos al cromosoma del hijo 1.

De esta forma nace el hijo 1, para el hijo 2 se sigue un procedimiento similar:

1. Utilizar los mismos cromosomas seleccionados para ser padres del hijo 1.
2. Intercambiar los roles del padre 1 y el padre 2 y repetir los pasos del 2 al 4 enunciado anteriormente para producir el hijo 2.


Figura 4.2: Método de cruce de único punto.
***Fuente:*** *elaboracón propia.*

La figura 4.2 muestra un ejemplo de como se realizaría un cruce de este tipo.

**Cruce multipunto** Como su nombre lo indica se puede realizar el proceso de reproducción utilizando más de un punto de cruce, a continuación se ilustra el procedimiento para el caso de dos puntos de cruce:

1. Como en el método anterior, se debe seleccionar dos cromosomas padres para la reproducción.

2. Seleccionar aleatoriamente la posición de dos (o más) puntos de cruce.

3. Copiar aquellos genes que esten afuera de los puntos de cruce (seleccionados en el paso anterior) del padre 1 y pegarlos al cromosoma del hijo 1.

4. Copiar los genes que se encuentren dentro de los dos puntos de cruce del padre 2 y agragarlos (en las mismas posiciones) al cromosoma del hijo 1.

De esta forma nace el hijo 1, para producir el hijo 2 se debe intercambiar el rol de cada padre seleccionado para la reproducción del hijo 1 y repetir los pasos ya explicados, como se puede observar en la figura 4.3 donde se esboza un ejemplo de este tipo de cruce (en este caso con solo dos puntos de cruce).

## 4.5 Operadores de mutación

La mutación dentro del proceso del GA es muy importante puesto que permite añadir nuevos y diferentes componentes al vector genético de la población nueva con el fin de encontrar la mejor solución. Al igual que en el proceso de cruce, la mutación depende del método de codificación que se emplee.


Figura 4.3: Método de cruce multipunto.
***Fuente:*** *elaboracón propia.*

El proceso de mutación reemplaza aleatoriamente algunos genes de un cromosoma por nuevos valores, en el caso de que el vector genético contenga una cadena de bits (es decir codificación basada en genes con un alfabeto genético binario) una de los formas más utilizadas para realizar este proceso es invirtiendo el valor que toma un determinado bit, así pues si el valor que toma el bit a mutar es 1 su nuevo valor será 0, y viceversa.

La mutación depende de un parámetro $p_m$ (*probabilidad de mutación*), antes de iniciar el proceso de mutación se genera un número aleatorio entre [0,1] y si resulta que este número es menor a la tasa de mutación, el proceso de mutación se realizará, de lo contrario este proceso no se llevará a cabo. Con este proceso se añade diversidad a la nueva población que conforma la siguiente generación, lo que permite ayudar a reducir el riesgo de que el algoritmo de optimización se quede atrapado en algún mínimo local y no encontrar la solución óptima.

Alternativamente, se puede aplicar la mutación a cada elemento del vector genético (es decir a cada gen), sin embargo si decide emplearse esta alternativa es recomendable utilizar una probabilidad de mutación ($p_m^l$) más baja que la que se emplea en la metodología anterior, puesto que puede generar poblaciones en las que varios individuos hayan presentado mutaciones (así sea en uno solo de sus genes).

Para realizar este proceso se debe generar aleatoriamente un número para cada elemento del vector genético y en los casos en el que éste sea menor a $p_m^l$ el bit es intercambiado por su opuesto, para que el proceso de mutación no sea relativamente frecuente, se espera que $p_m^l$ sea menor que $p_m$.

Existen otras metodologías para realizar el proceso de mutación, por ejemplo para vectores genéticos binarios puede realizarse por medio del operador de desplazamiento o de traslocación. En el operador de desplazamiento un bloque del vector genético se desplazará una posición, y este desplazamiento se puede llevar a cabo elimiando el bit que se encuentre antes del bloque, es decir, se desplaza el bloque una posición hacia la izquierda y se agrega un

bit seleccionado aleatoriamente después del bloque.

Sin embargo también se puede ubicar un bit aleatorio antes del bloque y éste se desplazaría una posición hacia la derecha y el primer bit que se encuentre después del bloque se elimina. La traslocación por su parte se seleccionan dos bloques del mismo tamaño y se intercambian de posición.

## 4.6 Descendencia

Definiendo el tamaño de la población como $\mu$ durante el proceso de simulación este parámetro debe permanecer invariante, lo que significa que el tamaño de población en cada generación debe ser el mismo. Hasta ahora se ha descrito como generar la población inicial para comenzar con el GA, como seleccionar el vector genético, el proceso de cruce y la mutación; de esta forma dos padres se utilizan para generar uno o más descendientes, y la pregunta es qué hacer con la descendencia.

La descendencia puede compararse con uno o ambos padres y las mejores soluciones se colocan en la población, de esta forma se conservaría posiblemente buena información genética, y esta comparación puede ser determinista o probabilística. La idea básica es que la aptitud de la descendencia se compare con la capacidad física de uno o ambos padres y posteriormente descartar la solución con la capacidad física más baja.

Generalmente se intenta conservar la mayor diversidad genética posible en la población principal de la generación siguiente, la población principal estará compuesta por los individuos seleccionados para ser padres, lo que supondría que si el cruce produce una descendencia razonablemente adecuada, toda la información genética de ambos padres se transferirá a la siguiente generación.

La figura 4.4 representa el procedimiento que sigue el algoritmo genético para la optimización de un sistema particular.

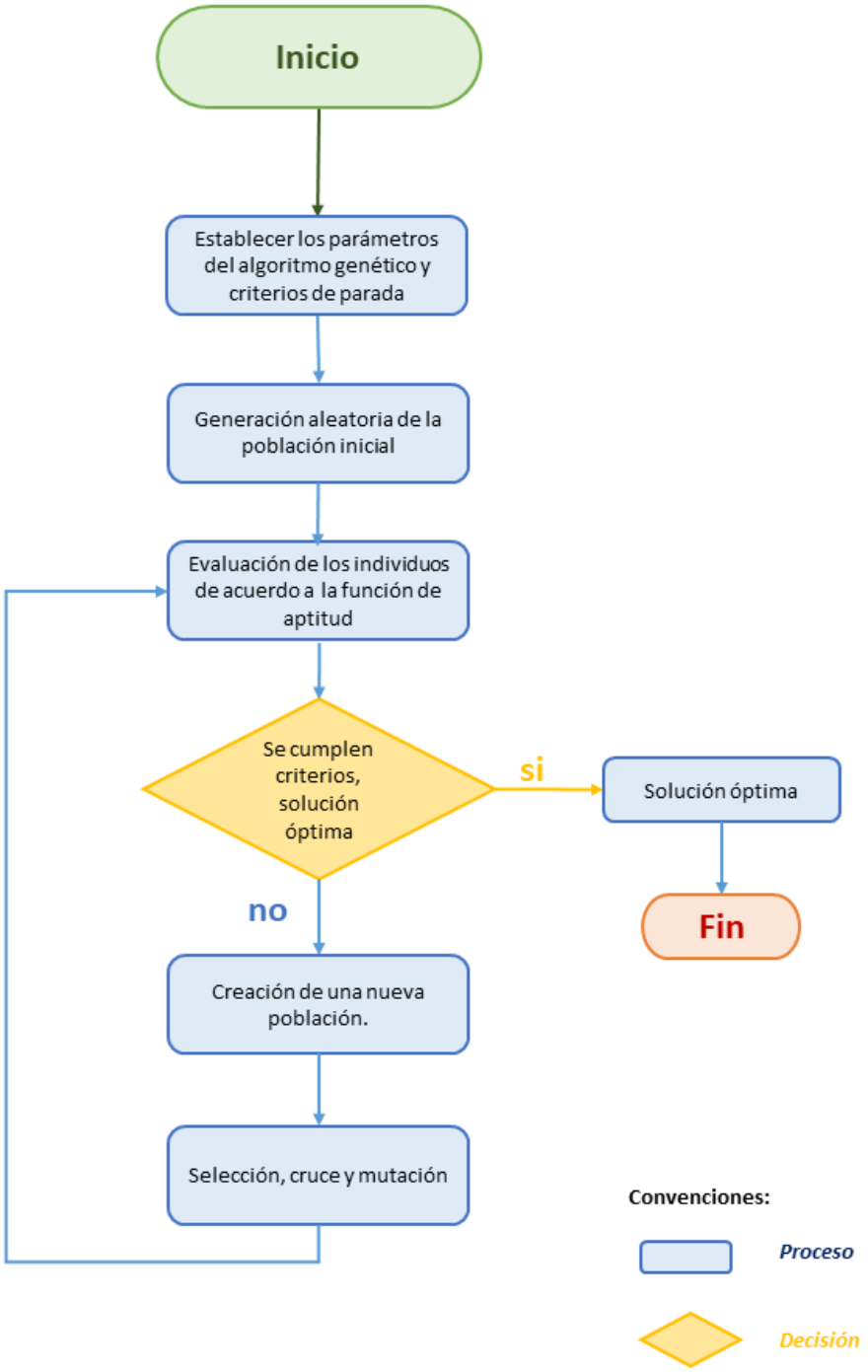



Figura 4.4: Diagrama de flujo para un algoritmo genético.
***Fuente:*** *elaboracón propia.*

# Capítulo 5

# Diseño e Implementación

Este cápitulo constituye la aplicación práctica de la tesis, empleando conjuntamente redes neuronales y algoritmos genéticos con el fin de resolver el problema de "la aplicación de *machine learning* al mercado financiero para la toma de decisiones de inversión en el mercado de divisa USD-COP".

La primera parte de este capítulo se enfoca en la descripción del modo de entrenamiento del algoritmo, en la segunda parte explica como se implementa en el ámbito financiero y finalmente se describen los resultados obtenidos con el modelo diseñado.

## 5.1 Algoritmo de negociación

La ANN se encargará de combinar las diferentes señales de análisis técnico (señales de filtro y de momentum, ver sección 5.2) provenientes de las reglas de negociación (trading rules). Estas señales se han generado dentro del ambiente de prueba, el cual permite realizar el entrenamiento de la red neuronal, ajustando sus parámetros y estructura (en el Anexo 1 se encuentra el código en Python para la implementación de la ANN).

El entrenamiento de la ANN implica definir el número adecuado de capas ocultas, el número de neuronas por capa, la función de activación, la función de costo, realizar la optimización de los pesos interconectores, etc.

Siguiendo a [10] a pesar de que en los últimos años ha ido aumentando el uso de las distintas técnicas de AI, el diseño y entrenamiento de una ANN aún sigue siendo tanto un arte como una ciencia, y la definición de sus parámetros (función de activación, patrón de interconexión y sistema de aprendizaje para la actualización de los pesos) y de su estructura (es decir el número de capas ocultas en la red y el número de neuronas en cada una de ellas) generalemente sigue estando sujeta a un trabajo de ensayo y error.

Es por esta razón que se emplea un algoritmo genético para optimizar la estructura de la ANN, de esta forma, el problema de diseño de la red queda sujeto a un problema de optimización para el GA, específicamente a la minimización de la función de costo definida en la ecuación (3.2).

### 5.1.1 Optimización del sistema algorítmico

El algoritmo genético determinará que configuración de la ANN es más adecuada teniendo como criterio de selección la estructura de ANN que genere el menor error de predicción. Para realizar la optimización se han empleado diferentes tamaños de población que van desde $N = 10$ hasta $N = 100$, sin embargo, es relevante resaltar que si bien mayores tamaños de población pueden generar mejores resultados ya que el proceso de optimización podría encontrar un mejor máximo, también se puede caer en un problema de sobreajuste de la ANN a los datos de entrenamiento, lo cual disminuiría su capacidad de predicción para el conjunto de prueba [19].

El proceso de mutación de los individuos a través de las generaciones se realiza utilizando la mutación uniforme según la cual todos los elementos de la cadena de bits tienen la misma probabilidad de ser seleccionados para mutar de acuerdo a una probabilidad definida. Asimismo, y siguiendo los resultados obtenidos por [19] se emplea un número de generaciones que van desde 100 hasta 150, ya que se encontró que este intervalo es suficiente para encontrar resultados estables para la esructura de la ANN.

En términos de la ANN, la *población inicial* se entiende como el número de estructuras distintas (número de capas ocultas y número de neuronas por capa) que van a ser utilizadas por el GA para desarrollar el proceso evolutivo a través de las disitnas generaciones:

$$\begin{aligned} K_1 &= [h_1^{k_1}, h_2^{k_1}, ..., h_{m_1}^{k_1}] \\ &\vdots \\ K_n &= [h_1^{k_n}, h_2^{k_n}, ..., h_{m_n}^{k_n}] \end{aligned} \tag{5.1}$$

Donde el subíndice $m_1$ representa el número de elementos que contiene el vector $K_1$, y el subíndice $m_n$ indica el número de elementos que contiene el vector $K_n$ (que pueden ser diferentes). Cada vector $K_i$ representa una estructura específica para la ANN, es decir es un individuo de la población

(la cual tiene un tamaño $n$). La estructura particular de cada individuo está representada por los distitnos vectores: $[h_1^{k_i}, h_2^{k_i}, ..., h_{m_i}^{k_i}]$, el tamaño de cada vector $K_i$ indica el número de capas ocultas que conforman la estructura de esa red. Mientras que cada elemento $h_j^{k_i}$ representa el número de neuronas que tendrá cada capa oculta del individuo $K_i$.

El tamaño de cada vector $K_i$ y de cada uno de los elementos que lo componen ($h_j^{k_i}$) toman valores dentro del conjunto de los números naturales ($K_i, h_j^{k_i} \in \mathbb{N}$), esto significa que el menor número de capas ocultas que puede tener cualquier individuo es 1, e igualmente que el número mínimo de neuronas por capa que puede tener un individuo es 1.

Cada individuo de la población se genera siguiendo los siguientes pasos: en primer lugar se genera un número aleatorio dentro del intervalo [1, número máximo de capas], éste representa el número de capas ocultas que tendrá ese individuo (es decir indica el tamaño del *K-ésimo* vector); posteriormente para cada posición dentro de este vector se genera un número aleatorio dentro del intervalo [1, número máximo de neuronas por capa], que indica el número de neuronas que tendrá cada capa del vector, es decir es el valor que toma cada elemento $h_m^{k_n}$. De esta forma se genera un individuo, posteriormente se repite este proceso hasta generar el número total de individuos de la población.

El GA realizará el entrenamiento de cada uno de los individuos (recuerde que cada individuo representa un estructura particular de ANN) obteniendo una predicción individual y evaluará su desempeño y capacidad (por medio de la función de error). Así se procede con cada individuo de la población inicial.

Una vez se ha realizado el entrenamiento de cada una de las ANN que conforman la primera generación (población inicial) se eligen los individuos con mejor desempeño (aquellos que generen el menor error) para ser los padres de la siguiente generación, y se realiza el proceso de crossover empleando el método de punto único. Igualmente se lleva a cabo la mutación y demás etapas que componen el proceso de evolución explicado en el capítulo anterior (ver figura 4.4).

El proceso descrito hasta aquí se debe realizar para cada una de las generaciones que se han establecido en la definición de parámetros (en el Anexo 2 se presenta el código con el que se realiza está implementación). La figura 5.1 resume el esquema que va a seguir el proceso de optimización de la ANN con la implementación del algortimo genético hasta aquí explicado.

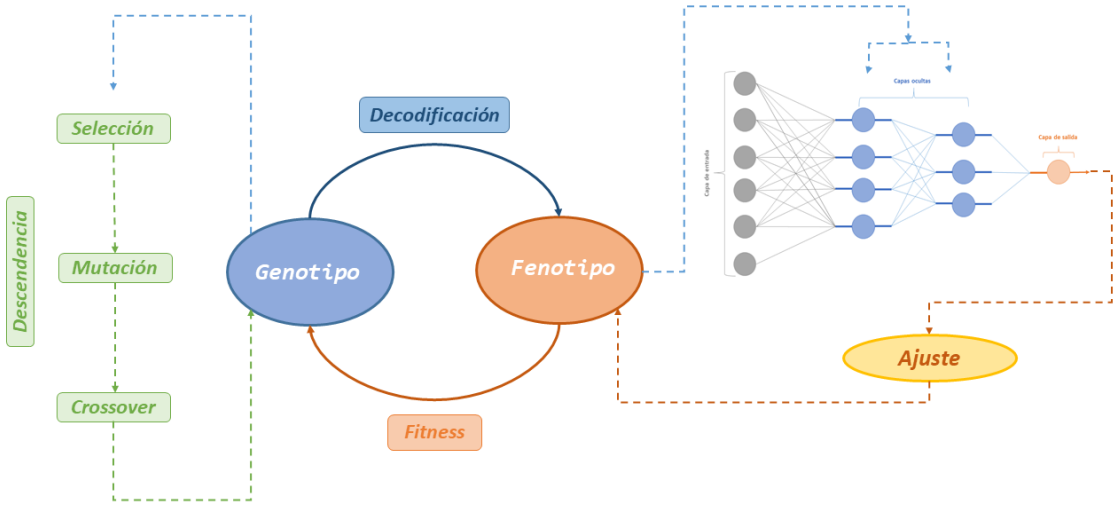



Figura 5.1: Proceso de optimización de la ANN utilizando algoritmo genético.
***Fuente:*** *elaboracón propia.*

## 5.2 Análisis técnico e implementación

Las reglas de análisis técnico permiten analizar los distintos mercados financieros y las connotaciones particulares del comportamiento del precio de un determinado activo, siguiendo a [6], el análisis técnico es la ciencia de registrar, generalmente en forma gráfica, la historia real de la negociación (cambios de precio, volumen de transacciones, etc.) en una determinada acción o en los promedios y luego deducir de esa historia la probable tendencia futura.

Este tipo de análisis es ampliamente usado por los participantes del mercado financiero, a continuación se exponen los fundamentos más importantes que soportan su enfoque.

1. **Comportamiento tendencial de los precios**

   En muchos análisis se asume que en el comportamiento de los precios de diferentes activos se pueden presentar tendencias, que se pueden identificar y estudiar para determinar en que momento las tendencias se pueden revertir.

2. **Relación con el comportamiento pasado**

De acuerdo con [16], los patrones gráficos identificados en este tipo de análisis están basados en el estudio de la psicología humana, por lo cual se espera que si en el pasado han permitido identificar las tendencias alcistas o bajistas de los mercados, también lo hagan en el futuro.

Esto implica que si se estudia el comportamiento pasado se puede comprender e inferir el comportamiento futuro de un activo financiero lo que permitiría proteger el capital en épocas de crisis y potenciar los beneficios en el caso contrario.

### 5.2.1 Indicadores de análisis técnico

Se van a utilizar cuatro indicadores, el índice de fuerza relativa RSI (por sus siglas en inglés, relative strength index), el indicador de Convergencia/Divergencia de Medias Móviles MACD (por sus siglas en inglés, Moving average convergence divergence), el índice de fuerza verdadera TSI (por sus siglas en inglés, True Strength Index) y el indicador TRIX (por sus siglas en inglés, Triple Exponential Average), adicionalmente se tiene en cuenta un indicador de volatilidad medido a través de la desviación estándar mensual.

Para el análisis y la aplicación al problema planteado se emplea un serie histórica para la divisa USD-COP que abarca desde el 01/01/2010 hasta 14/06/2018, obtenida del sitio web Yahoo Finance [1], de la página de internet del Banco de la República [2] y de la terminal de Bloomberg [3].

La serie total de USD-COP es dividida en 2 conjuntos, el conjunto de entrenamiento y el conjunto de prueba. El primero de ellos abarca el periodo comprendido entre el 01/01/2010 y el 19/03/2014, mientras que el conjunto de prueba recoge el periodo restante entre el 20/03/2014 y el 14/06/2018. A continuación se describen brevemente los indicadores utilizados en el análisis resaltando sus principales características así como las reglas de decisión propias de cada uno de ellos.

#### ÍNDICE DE FUERZA RELATIVA

Este indicador fue desarrollado por J. Welles Wilder en 1978, es un indicador de tipo oscilador que pretende resolver el problema que se se presenta por las variaciones bruscas de los precios de los activos en la construcción de líneas de momento (ver [16]), la fórmula para su cálculo esta definida por la siguiente expresión:

[1] Recuperado de https://finance.yahoo.com/
[2] Recuperado de http://www.banrep.gov.co/es/trm
[3] Bloomberg Anywhere

$$RSI = 100 - \frac{100}{1 + RS}$$

$$Donde: \tag{5.2}$$

$$RS = \frac{Media\ de\ cierres\ al\ alza\ para\ X\ dias}{Media\ de\ cierres\ a\ la\ baja\ para\ X\ dias}$$

Se suele utilizar un periodo de 14 días ($X = 14$) por recomendación del autor, la media de cierres al alza para X días calcula utilizando una media móvil exponencial EMA (por sus siglas en inglés, exponential moving average); análogamente se sigue el mismo procedimiento para encontrar el valor de la media de cierres a la baja para X días. De esta forma el cociente entre ambas da como resultado la fuerza relativa (RS), cabe resaltar que en ocasiones se suele usar un periodo de 9 días, con el fin de que el oscilador sea más sensible, por lo que sus fluctuaciones serán más pronunciadas.

**Interpretación.** El RSI se representa sobre una escala vertical de 0 a 100 (Ver Figura 5.2), si el RSI alcanza un nivel por encima de 70 se consideran que el valor está sobrecomprado, mientras que los movimientos por debajo de 30 indicarían que el valor está sobrevendido.

### INDICADOR DE CONVERGENCIA/DIVERGENCIA DE MEDIAS MÓVILES

Este indicador se basa en el análisis de promedios de los precios de un determinado activo para periodos de distinta duración, fue desarrollado por Gerald Appel en la década de 1970. Es un indicador de tipo oscilador que utiliza dos medias móviles exponenciales suavizadas de los precios de cierre del activo [17]. Generalmente suele utilizarse un periodo de 12 y 26 días para calcular los EMA, respectivamente. Si la resta entre estos dos EMA es positivo significa que el precio del activo en los últimos días presentó un incremento inportante con respecto a los días precedentes, esto se debe a que mayores niveles de precio recientes van a generar un impacto mayor sobre la EMA de más corto plazo y viceversa.

$$MACD = EMA_{12} - EMA_{26} \tag{5.3}$$

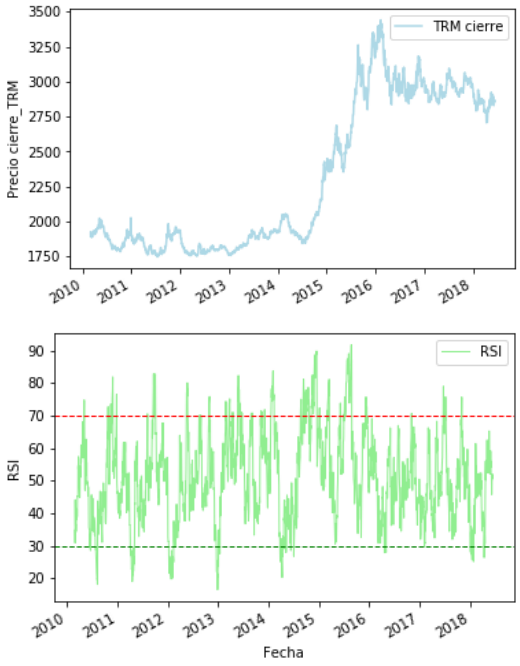



Figura 5.2: En este gráfico se muestra el índice RSI construido con base en la serie de la TRM de cierre, igualmente se presentan los límites de decisión.
***Fuente:*** *elaboracón propia.*

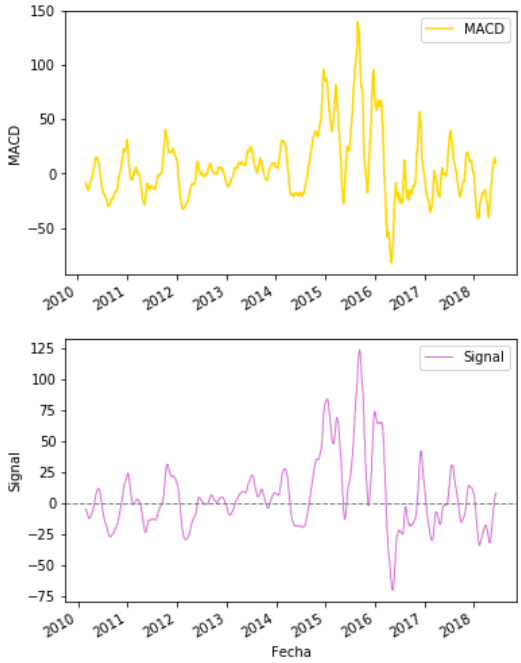



Figura 5.3: La primera gráfica representa el indicador MACD con parámetros 12, 26 y 9. La segunda gráfica muestra la señal construida con base en el MACD (ecuación (5.4)). ***Fuente:*** *elaboracón propia.*

Sin embargo este indicador también viene acompañado de una señal que se usa en conjunto con el MACD para la toma de decisiones, la cual consiste en estimar un promedio movil MA (por sus siglas en inglés, moving average) estimado a 9 días del MACD:

$$Signal = MA(MACD, 9) \tag{5.4}$$

**Interpretación.** Con este indicador las señales de compra y venta se darán cuando se produzcan cruces entre ambas series (MACD y Signal) [16]. Por ejemplo un cruce de la línea MACD por encima de la línea de señal se puede interpretar como un indicador de compra y permanecerá mientras se mantenga en niveles superiores a la señal; mientras que un cruce de la línea MACD por debajo de la señal será considerado como un indicador de venta, la figura 5.3 ilustra el comportamiento de este indicador y la señal para el caso de estudio.

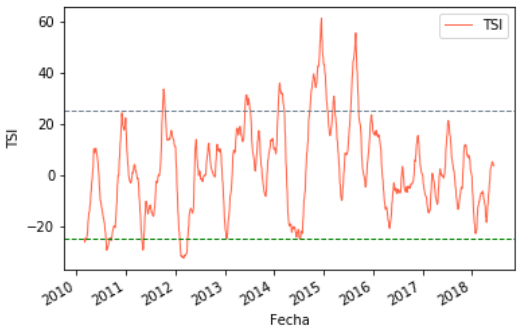



Figura 5.4: El índice de fuerza verdadera está construido con parámetros $r = 25$ y $s = 13$, las líneas punteadas representan los límites de decisión.
***Fuente:*** *elaboracón propia.*

## ÍNDICE DE FUERZA VERDADERA

Consiste en un indicador doblemente suavizado de *momentum* que permite identificar las fluctuaciones de los precios a corto plazo mientras se negocia en dirección a la tendencia [3]. Se suele emplear una ventana de tiempo de 25 días para estimar la EMA sobre la diferencia del precio entre los tiempos $t$ y $t-1$. Por otro lado se emplea un periodo de 13 días para estimar un EMA sobre el resultado, de esta forma, esta medida se convierte en un indicador más sensible a las condiciones prevalecientes del mercado. El indicador se calcula resolviendo la siguiente ecuación:

$$TSI(c_0, r, s) = 100 * \frac{EMA * (EMA(m, r), s)}{EMA * (EMA(|m|, r), s)} \tag{5.5}$$

Donde $m$ representa la diferencia entre los precios de cierre de hoy y de ayer ($t$, y $t-1$), $r$ es el periodo suavizado para el EMA, generalmente a $r = 25$ días, mientras que $s$ es el periodo suavizado para el impulso, usualmente $s = 13$ días.

**Interpretación.** El resultado del indicador de fuerza verdadera está determinado en el intervalo $[-100,100]$, sin embargo la mayoría de valores se van a encontrar dentro del intervalo $[-25, 25]$ (ver figura 5.4), por lo cual se suele interpretar estos dos límites como las señales de sobrecompra y sobreventa (respectivamente).

## TRIX

El *Triple Exponential Average* como su nombre lo sugiere es un indicador formado por una triple media exponencial, desarrollado por Jack Huston

en 1984 [12], que puede encajar dentro de dos categorías, puede entenderse como oscilador el cual permite tomar decisiones de compra, y asimismo se puede entender como un indicador de tiempo, identificando la fuerza de la tendencia en la que se encuentre el precio del activo. Este indicador se suele utilizar para medir el grado de cambio en el precio de cierre del activo, un incremento del indicador puede mostrar un tendencia alcista, mientras que una disminución del mismo puede representar una señal de que los precios se enmarcan dentro de un escenario bajista. De esta forma las variaciones positivas del TRIX indican que la tendencia del precio se está volviendo más fuerte, y se infiere que la tendencia perdería fuerza ante cambios negativos del indicador.

Como indicador de oscilación se compara en torno a su relación con el valor cero (0), entre más lejos se encuentre del 0, la señal será más fuerte, un valor positivo (TRIX mayor a 0) indica que el activo está sobrecomprado, mientras que un valor negativo indica que el activo presenta un exceso de venta. A continuación se presenta la fórmula para su cálculo:

$$\begin{aligned} EMA_1 &= EMA_n \\ EMA_2 &= EMA(EMA_1, n) \\ EMA_3 &= EMA(EMA_2, n) \\ TRIX &= \frac{EMA_3(t) - EMA_3(t-1)}{EMA_3(t-1)} \end{aligned} \tag{5.6}$$

Como se puede observar para realizar el cálculo del indicador TRIX se debe estimar una EMA del precio del activo ($EMA_1$), posteriormente se cálcula un EMA sobre el resultado del paso anterior, finalmente se debe estimar una tercera EMA sobre el resultado obtenido ($EMA_2$). El indicador TRIX será el resultado de calcular la variación de la $EMA_3$ entre el tiempo $t$ y $t-1$ (Ver figura 5.5).

**Interpretación.** Al ser un indicador basado en el cálculo de varias EMA, la serie de precios del activo es suavizada al incoporarase la estimación de 3 EMA, el TRIX representa un indicador que permite inferir el comportamiento de la tendencia de la serie sin tener en cuenta el ruido de las variaciones a corto plazo de la misma, e igualmente es un indicador que brinda información sobre el mometum de la serie indicando la velocidad en el cambio de los precios.

Se ha decidido tener en cuenta esta gama de indicadores debido a que el uso generalizado de los indicadores basados en el análisis técnico ha dis-

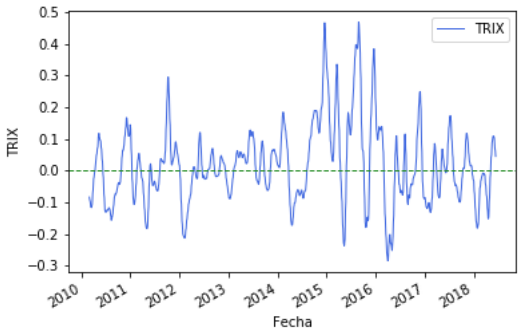



Figura 5.5: En esta figura se muestra el el indicador TRIX para la divisa USD-COP respecto a la línea de referencia (recta del cero).
***Fuente:*** *elaboracón propia.*

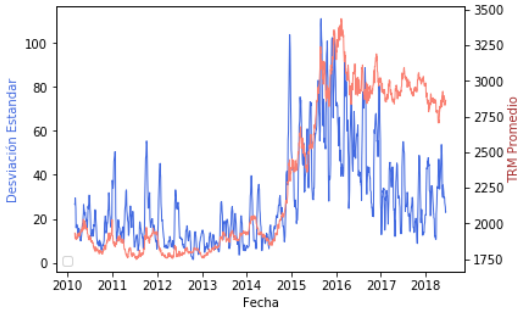



Figura 5.6: En esta figura se muestra el comportamiento de la medida de volatilidad utilizada (desviación estándar de un mes) y de la TRM promedio.
***Fuente:*** *elaboracón propia.*

minuido en cierta medida la rentabilidad de los mismos, razón por la cual se utilizan dos indicadores cuya aplicación es ampliamente conocida, y otros dos de menor uso para generar una mejor combinación de la información disponible para obtener una mejor aproximación a la decisión eficiente de inversión.

### 5.2.2 Entrenamiento e implementación

Cada uno de los indicadores explicados en la sección anterior son calculados para el conjunto de entrenamiento así como la medida de volatilidad (en este caso representada por la desviación estándar con ventana de un mes), como se observa en la figura 5.6.

De esta forma se va tener un vector de *inputs* que representan cada uno de los indicadores hasta aquí explicados (dimensión $5 \times 1$) para cada día del periodo comprendido dentro del conjunto de prueba $n$:

$$\begin{pmatrix} x_{1,1} & \dots & x_{1,n} \\ x_{2,1} & \dots & x_{2,n} \\ x_{3,1} & \dots & x_{3,n} \\ x_{4,1} & \dots & x_{4,n} \\ x_{5,1} & \dots & x_{5,n} \end{pmatrix} \tag{5.7}$$

Cada término $x_{m,n}$ representa el valor del indicador $m$ para la fecha $n$, por ende se tiene una matríz de dimensión $5 \times n$ que resume la información disponible para el conjunto de entrenamiento, y que servirán de inputs para la ANN. Así pues, para el proceso de optimización del sistema mediante el empleo del GA, cada individuo de la población (es decir cada estructura particular de ANN) realizará este proceso de entrenamiento para cada una de las generaciones consideradas.

### Regla de negociación

Solo resta conocer cual es la variable objetivo del sistema de optimización, dado que lo que se pretende lograr con la implemetación de este algoritmo (en adelante ANN + GA) es generar una señal de decisión de inversión para un periodo de 5 días, la señal puede ser de comprar, vender o mantener la posición. Para deteminar el valor de la señal dentro del conjunto de entrenamiento se va a generar un indicador que adopte valores dentro del intervalo $[-1, 1]$, de la siguiente forma:

$$Zscore = -1 + \left( \frac{(x_i - x_{min}) \times 2}{x_{max} - x_{min}} \right) \tag{5.8}$$

Donde $x_i$ es la diferencia en el valor de la TRM con un rezago de 5 días (horizonte de inversión), a saber:

$$x_{i,t} = TRM_t - TRM_{t-5} \tag{5.9}$$

Mientras que $x_{max}$ y $x_{min}$ representan el valor máximo y mínimo (respectivamente) del conjunto de entrenamiento. Para determinar si se debe comprar, vender o mantener la posición se establecen unas bandas o límites que permitirán determinar si la señal es lo suficientemente fuerte para tomar la decisión (Ver figura 5.7). Como se observa en la figura 5.7, para esta tesis se escogen bandas de 0,5 y $-0{,}5$.

De tal forma que si el *Zscore* esta por fuera de las bandas establecidas se generará una señal de compra o venta (según corresponda), y si éste perma-

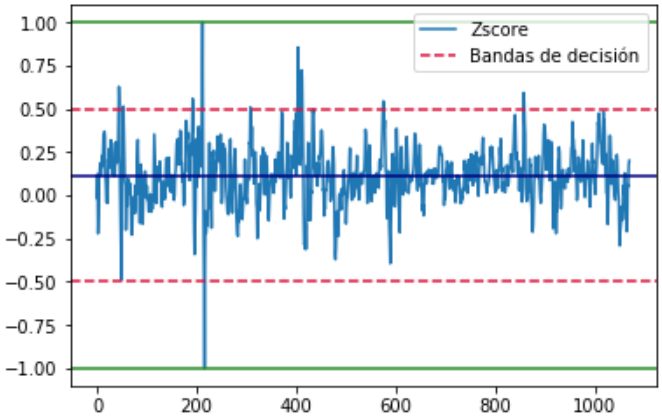



Figura 5.7: Esta figura presenta el comportamiento del Zscore definido en la ecuación 5.8, así como las bandas de decisión definidas.
***Fuente:*** *elaboracón propia.*

nece dentro de las bandas, la señal que se genera es de mantener la posición que se traía.

Así pues, teniendo un capital inicial, la estrategía consiste en invertir todo el capital de acuerdo a la señal generada, es decir, si se genera una señal de compra la decisión de inversión consistirá en adquirir la posición correspondiente en la divisa USD-COP, y cuando se genere una señal de venta se deberá vender la posición acumulada a la tasa de mercado vigente en ese momento.

## 5.3 Resultados

En esta sección se presentan los resultados obtenidos con la aplicación de la ANN + GA diseñada, igualmente se muestran los resultados que se obtendrían con la negociación basada en las herramientas de análisis técnico.

### 5.3.1 Sistema optimizado

Al realizar la aplicación del GA para la optimización de la red, se encontró que el resultado para la estructura de la ANN que arroja el algoritmo está compuesta por dos capas ocultas, la primera con 13 neuronas y la segunda con 15 neuronas, este resultado se ha obtenido promediando la solución óptima de varias iteraciones del algoritmo entrenado para el mismo conjunto de datos.
Esta estructura se ha determinado utilizando un tamaño de población de 30

individuos, debido a que si se utiliza un número mayor de individuos se obtienen mejores reultados en el conjunto de entrenamiento, pero los resultados en el conjunto de prueba generan errores de ajuste relativamente altos (de más de 15 %) en comparación con el tamaño de población escogido (30) que presenta un error de predicción similar al obtenido en el conjunto de prueba (al rededor de 5 %).

Igualmente se encontró que con tamaños de población más grandes, se obtienen estructuras de ANN con mayor profundidad (es decir con un mayor número de capas y en algunos casos con un número mayor de neuronas por capa). Al probar estas estructuras de ANN, se observa que se ajustan mejor al conjunto de entrenamiento, lo que puede ser una razón para suponer que se genera un sobrajuste a los datos de etrenamiento y por tanto, puede ser debido a este sobreajuste que se pierde precisión en el conjunto de prueba.

Así mismo, se realizaron pruebas con distintos valores para la tasa de mutación del GA, encontrando que tasas de mutación por individuo de 5 % pueden ser muy grandes y representar variaciones importantes en la estructuras de ANN resultantes, mientras que tasas de mutación más pequeñas como de 1 %, es decir que en cada generación haya una probabilidad de que el 1 % de la población presente alguna mutación, genera resultados más consistentes en cuanto a la estructura de ANN que se obtiene con la optimización.

La figura 5.8, muestra el comportamiento del retorno obtenido para una inversión inicial de COP $1,000,000$, empleando la estructura de ANN descrita inicialmente. Es importante resaltar que el error esperado con la implementación de la red, indica que el retorno que se puede obtener de la aplicación del ANN + GA puede variar en alrededor de 5 %, es decir, que el retorno obtenido puede tener una variación de mas o menos 5 %.

Para el proceso de negociación, se ha establecido un sencillo conjunto de reglas de negociación, es importante resaltar que para esta aplicación financiera no se están considerando costos de transacción, los cuales pueden generar una considerable variación en los retorno obtenidos con la estrategia, dependiendo la cantidad de señales de compra o venta que se generen:

- Para entrar por primera vez a la negociación se debe esperar hasta que el ANN + GA genere una señal de compra.

- Se toman decisiones de inversión en un periodo semanal, es decir cada 5 días hábiles se revisan las señales de compra, venta o permanencia de la posición que arroje el ANN + GA.

- La ejecución, ya sea de una señal de compra o venta que se generen, se realiza al precio de apertura de la TRM del día en que se efectúe


Figura 5.8: Esta figura presenta el comportamiento del rendimiento obtenido con la implementación de la herramienta de negociación diseñada.
***Fuente:*** *elaboracón propia.*

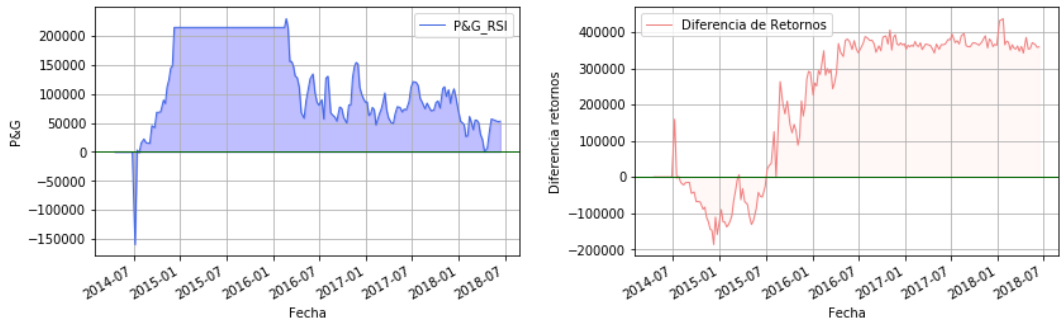
la operación (es importante resaltar que no se tiene en cuenta la interacción entre oferta y demanda del mercado, es decir no se toman los precios de *bid* y *ask* para cada nivel de TRM).

El periodo de inversión comienza el 3 de julio de 2014, sin embargo la primera señal de compra se produce solo hasta el mes de noviembre de ese año momento en el cual la TRM rompe la barrera de los 2,200 $COP/USD$ (enviando una señal suficientemente fuerte para el ANN + GA).
De aquí en adelante el comportamiento del retorno obedece mucho al comportamiento de la divisa, resaltando la volatilidad presentada a finales de 2015, cuando la TRM promedio en algunos periodos alcanzó a presentar variaciones (5 días) de +/- 5 % (como se muestra en la figura 5.6), lo cual se refleja en la variabilidad del retorno obtenido.
Finalmente en los últimos meses del periodo de prueba, dado el nivel que la TRM alcanzó (cercano a 3,000 $COP/USD$), se obtienen retornos positivos con las señales generadas por la red, las cuales son de compra y de permanencia de la posición comprada.

A continuación se revisará el comportamiento obtenido de haber invertido basando las decisiones en las herramientas de análisis técnico.

## 5.4 Machine learning vs. análisis técnico

En esta sección se comparan los resultados obtenidos con la aplicación del ANN + GA desarrollado en esta investigación a la luz de los que podrían




(a) Indicador RSI (b) Diferencia ANN-RSI

Figura 5.9: (a) Esta figura presenta el comportamiento del retorno obtenido utilizando el indicador RSI; (b) Representa la diferencia entre el retorno obtenido por la ANN y el indicador RSI, periodo Abril 2014 - Junio 2018.
***Fuente:*** *elaboracón propia.*

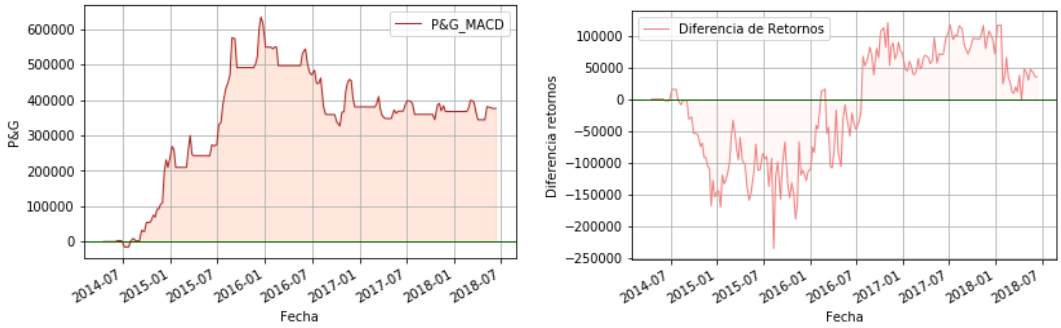
haberse obtenido utilizando únicamente las herramientas del análisis técnico con base en la información del mercado de la divisa USD-COP para el periodo prueba.

**Índice RSI** Si se hubiesen tomado las decisiones de inversión para la negociación de de la divisa USD-COP durante el periodo de prueba utilizando las señales recibidas del indicador RSI, se obtendría un retorno de la inversión como se representa en la figura 5.9.

Recuerde que las señales de decisión de este indicador sugieren que:

- Si $RSI > 70$ se genera una señal de venta, es decir si el $RSI$ cruza desde abajo el límite de 70 se debe vender.
- Las señales de venta se producen cuando $RSI < 30$.
- Y se debe mantener la posición en otro caso.

Como se observa en la figura 5.9 (b), en el primer año de inversión el índice RSI genera mayores retornos que los obtenidos con el ANN + GA, sin embargo a partir de julio de 2015, el ANN + GA logra generar mejores señales de negociación. Precisamente la diferencia en los retornos obtenidos se debe a que el indicador RSI genera un número mayor de señales de negociación debido a la volatidad de la TRM durante este periodo, las cuales en algunos casos generan menores retornos de los que se hubiesen podido obtener.




(a) Indicador MACD (b) Diferencia ANN-MACD

Figura 5.10: (a) Ilustra el comportamiento del retorno obtenido utilizando el indicador MACD en conjunto con la señal que se obtiene a partir del mismo; (b) representa la diferencia de retornos entre la ANN y el indicador MACD, periodo Abril 2014 - Junio 2018.

***Fuente:*** *elaboracón propia.*

**Índice MACD** Las señales de decisión para comprar, vender o mantener la posición con este indicador se basan en los siguientes aspectos:

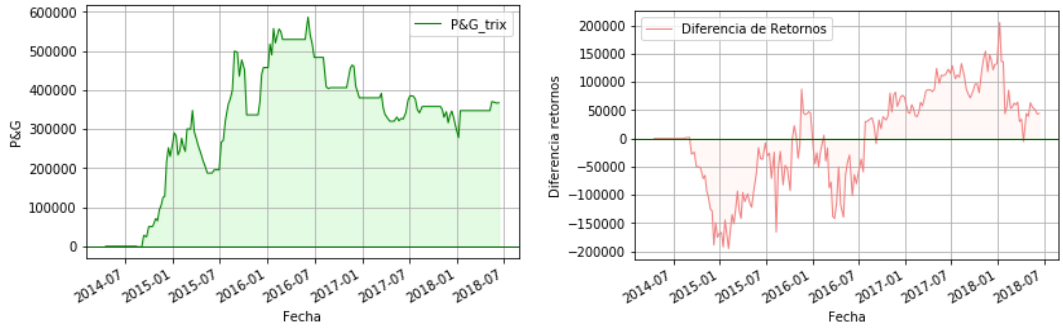
- Cuando $MACD > Signal$, la señal que se genera es de compra, es decir si el $MACD$ cruza desde abajo la "señal"se debe comprar.
- Si $MACD < Signal$ se crea una señal de venta, es decir si el $MACD$ cruza desde arriba la "señal"se debe vender.
- Y se debe mantener la posición en otro caso.

La figura 5.10 (a), representa el retorno obtenido para el caso de la negociación con base en este indicador.

Se puede observar en la figura 5.10a (b), que el indicador MACD genera mayores retornos que los que se obtienen con el ANN + GA en los dos primeros años, en este caso la diferencia al comienzo del periodo de prueba se debe a que el MACD genera señales tempranas de compra, lo que le permitío acumular una buena posición en USD, que se valorizó con el incremento del precio del dólar durante el año 2015.

**Índice TRIX** El indicador TRIX que permite entender la tendencia subyacente del activo dejando de lado el ruido producido por las oscilaciones de corto plazo, genera un nivel de retorno similar al obtenido por el MACD como se muestra en la figura 5.11.

Este indicador permite tomar decisiones de inversión con base en las siguientes reglas:




(a) Indicador TRIX (b) Diferencia ANN-TRIX

Figura 5.11: (a) Esta figura presenta el comportamiento del retorno obtenido utilizando el indicador TRIX; (b) diferencia entre los retornos obtenidos con ANN e indicador TRIX, periodo Abril 2014 - Junio 2018.
***Fuente:*** *elaboracón propia.*

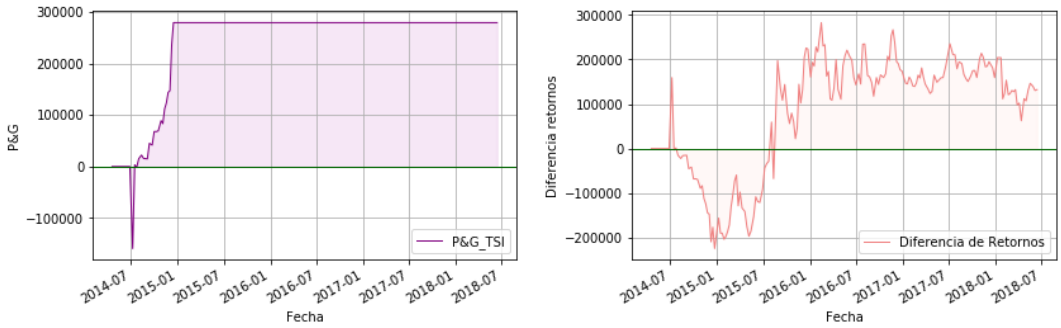
- $TRIX > 0$ indica una señal de compra, es decir si el $TRIX$ cruza desde abajo el límite de 0 se debe comprar o abrir la posición.
- La señal de venta se genera cuando $TRIX$ cruza desde arriba el límite de 0.
- Y se debe mantener la posición en otro caso.

La figura 5.11 (b), esboza un comportamiento similar al observado en la figura 5.10 (b), en este caso el TRIX genera señales tempranas de compra lo que le permite obtener retornos positivos desde el comienzo, mientras que el ANN + GA no genera señales tempranas de entrada, en efecto si se observa la figura 5.8 se puede evidenciar que hasta finales del año 2014 se empiezan a obtener retornos positivos.

**Índice TSI** La negociación con base en el indicador TSI a diferencia de todos los demás indicadores empleados genera un retorno con características que difieren claramente de los obtenidos y explicados anteriormente como se muestra en la figura 5.12.

Es probable que el comportamiento particular del retorno de obtenido con este indicador obedezca a la parametrización que se emplea para la generación de las señales, las cuales indican que:

- Una señal de venta se produce si $TSI > 25$.
- Si $TSI < -25$ se genera una señal de compra, es decir si el $TSI$ cruza desde arriba el límite, se debe comprar.




(a) Indicador TSI (b) Diferencia ANN-TSI

Figura 5.12: (a) Representa el retorno obtenido con la implementación del indicador TSI; (b) ilustra la diferencia entre el retorno que se obtuvo con la ANN y el que se obtuvo con el indicador TSI, periodo Abril 2014 - Junio 2018.
***Fuente:*** *elaboracón propia.*

- Y se debe mantener la posición en otro caso.

De hecho, desde enero de 2015 no se generaron señales de negociación diferentes a la de mantener la posición, por esta razón la figura 5.12 (b) obedece prácticamente al comportamiento presentado por el retorno obtenido con el ANN + GA.

Ahora bien, el cuadro 5.1 resume el rendimiento obtenido mediante la utilización de cada una de estas herramientas de análisis técnico así como el obtenido con la implementación del ANN + GA diseñado. Como se puede observar el rendimiento promedio anual que se obtuvo de la negociación con el ANN + GA inteligente es mayor al que se hubiese podido obtener con cualquier otra herramienta de análisis técnico.

Esto implica que es posible adaptar una red neural para generar una combinación eficiente de las herramientas de anális técnico para la negociación en el mercado financiero, la cual a su vez, permite obtener un beneficio económico (aclarando que no se tienen en cuenta los costos de transacción).

Sin embargo, quizas uno de los grandes aciertos del ANN + GA diseñado en este trabajo consitió en generar señales de negociación en periodos de tiempo más apropiados, es decir el algortimo generó un menor número de señales en comparación con las señales generadas por las herramientas de análisis técnico, lo cual permitió que con una determinada posición en USD se obtuviera un mayor retorno por valoración del mercado.

Este hecho puede deberse a distintas causas, en primer lugar la señal di-

| RENTABILIDAD PROMEDIO | | |
|---|---|---|
| HERRAMIENTA | PESOS ($) | (%) |
| RSI | $ 120,725.91 | 12% |
| MACD | $ 349,144.52 | 35% |
| TSI | $ 256,220.08 | 26% |
| TRIX | $ 351,611.96 | 35% |
| ALGORITMO | $ 382,546.71 | 38% |
| CAPITAL INICIAL | $ 1,000,000.00 | |

Cuadro 5.1: En esta tabla se muestran los rendimientos promedios semanales obtenidos con las diferentes herramientas de negociación.
***Fuente:*** *elaboracón propia.*

señada para la toma de decisiones con el ANN + GA (ver ecuación (5.8)) y las bandas establecidas, afectan la generación de las señales, con una pequeña variación de dichas bandas el número de ordenes de negociación podría ser muy diferente y así mismo los retornos obtenidos (igualmente si se utiliza una metodología alterna para la generación de la señal, por ejemplo regresiones). En segundo lugar, las reglas establecidas para las herramientas de análisis técnico (recuerde que en este trabajo se utilizó la parametrización recomendada por sus autores) podrían modificarse (ser más flexible o exigente) lo cual generaría comportamientos de inversión (y rendimientos) muy diferentes a los obtenidos con esta aplicación.

# Capítulo 6

# Conclusiones y trabajo futuro

A continuación se presentan algunas conclusiones y comentarios finales sobre el trabajo realizado, y se mencionan algunos aspectos que se pueden mejorar y que pueden analizarse en trabajos futuros.

## 6.1 Conclusiones

Se demostró que se pueden generar mejores resultados utilizando una adaptación de una red neuronal al mercado de divisas en comparación con los que se obtendrían utilizando las herramientas del análisis técnico, aún empleando una señal de respuesta relativamente simple como la ilustrada en este trabajo.
La optimización con el algortimo genético generó mejores resultados utilizando tamaños de población moderados, se encontró para este estudio que con un tamaño de población igual a 30 se obtienen estructuras de ANN menos complejas pero con mejor desempeño en el conjunto de prueba, en comparación con los resultados obtenidos con tamaños de población más grandes.

Es importante que al momento de definir los parámetros del GA se prueben distintas tasas de mutación, puesto que se pueden generar afectaciones significativas al resultado final de la optimización de la estructura de las redes. Se comprobó que con tasas de mutación mayores a 5 % se generaron distintas estructuras óptimas utilizando el mismo conjunto de datos, sin embargo con tasas de mutación menores (en este caso 1 %) la estructura óptima para la red resultaba ser más estable en las distintas iteraciones del GA.

Adicionalmente, emplear un algoritmo genético puede ser una alternativa factible para resolver el problema de encontrar la estructura más idónea para la red neuronal.

## 6.2 Trabajo futuro

Como se demostró en el desarrollo de este trabajo, con herramientas sencillas se pueden obtener resultados importantes, sin embargo existen varios aspectos que se pueden mejorar y probar para dar solución al problema planteado e incluso para abordar distintas temáticas, a continuación se mencionan algunas de las más importantes:

- Se pueden emplear distintas técnicas para la optimización de los pesos interconectores de la ANN, como el gradiente descendente estocástico (Stochastic Gradient Descent), el Batch gradient descent, e incluso, se puede emplear el GA para realizar dicha optimización.

- Ampliar la aplicación financiera a carteras o portafolios, en donde se pueda incluir activos de distinta naturaleza (por ejemplo de renta fija o renta variable, o una combinación de ambos), empleando diferentes herramientas para generar las señales de respuesta para la toma de decisiones de inversión, por ejemplo regresiones, análisis de VaR, etc.

- Es importante utilizar la información aportada por la desviación estándar o por alguna otra medida de volatilidad de la serie financiera, en este caso se empleó como un insumo para la ANN, sin embargo también puede utilizarse como una señal de negociación, igualmente se pueden tener en cuenta los costos de transacción dentro la negociación con el fin de tener una aproximación más realista de los retornos obtenidos.

- Desarrollar una metodología que integre tanto enfoques ascendentes como descendentes de la inteligencia artificial para el análisis del comportamiento de los activos financieros.

# Anexos

# Anexos A

# Red Neuronal

A continuación se presentan las principales funciones utilizadas para la programación de la ANN, se realiza la explicación para dos capas ocultas, puesto que basta con extender el código para obtener las demás capas que se quieran agregar.

```
1
- Se determinan aleatoriamente los pesos iniciales de la ANN
    teniendo en cuenta su estructura, para una, dos o mas capas
    ocultas:
3
        if hLayerLength == 1:
5           self.inputLayerSize = 6
            self.outputLayerSize = 1
7           self.hiddenLayerSize = hLayerSize1
            self.W1 = np.random.randn(self.inputLayerSize, self.
    hiddenLayerSize)
9           self.W2 = np.random.randn(self.hiddenLayerSize, self
    .outputLayerSize)

11      if hLayerLength == 2:
            self.inputLayerSize = 6
13          self.outputLayerSize = 1
            self.hiddenLayerSize = hLayerSize1
15          self.hiddenLayerSize2 = hLayerSize2
            self.W1 = np.random.randn(self.inputLayerSize, self.
    hiddenLayerSize)
17          self.W2 = np.random.randn(self.hiddenLayerSize, self
    .hiddenLayerSize2)
            self.W3 = np.random.randn(self.hiddenLayerSize2,
    self.outputLayerSize)
19

21- La funcion forward permite realizar el proceso de forward
```

propagation, el cual tomando los inputs y pesos genera la salida de la red:

```
    def forward(self, X, hLayerLength = '1'):
        if hLayerLength == 1:
            self.z2 = np.dot(X, self.W1)
            self.yHat2 = self.sigmoid(self.z2)
            self.z3 = np.dot(self.yHat2, self.W2)
            yHat = self.sigmoid(self.z3)
            return yHat

        if hLayerLength == 2:
            self.z2 = np.dot(X, self.W1)
            self.yHat2 = self.sigmoid(self.z2)
            self.z3 = np.dot(self.yHat2, self.W2)
            self.yHat3 = self.sigmoid(self.z3)
            self.z4 = np.dot(self.yHat3, self.W3)
            yHat = self.sigmoid(self.z4)
            return yHat
```

- La funcion sigmoidea, calcula la el valor de respuesta de la funcion sigmoidea, tiene como argumentos el valor de entrada a la funcion y deriv, que en caso de ser True devuelve el valor de la derivada de la funcion sigmoidea evaluada en el valor sumistrado:

```
    def sigmoidea(self, z,deriv=False):
        if deriv:
            return np.exp(-z)/((1+np.exp(-z))**2)
        return 1/(1+np.exp(-z))
```

- La FuncionCosto evalua la funcion de error cuadrado medio, recibe como entradas la matriz de inputs X, y el valor deseado de y:

```
    def FuncionCosto(self, X, y, hLayerLength):

        self.yHat = self.forward(X,hLayerLength)
        if hLayerLength == 1:
            J = 0.5*sum((y-self.yHat)**2)
            return J

        if hLayerLength == 2:
            J = 0.5*sum((y-self.yHat)**2)
            return J
```

```
63
- DerivFuncionCosto, calcula las derivadas de la funcion de
    costo respecto a cada peso de la ANN, recibe como entradas
    lamatriz de inputs X, y el valor deseado de y; devuelve las
    derivadas parciales de la funcion de error con respecto a los
     pesos:
65
    def DerivFuncionCosto(self, X, y, hLayerLength):
67
        self.yHat = self.forward(X,hLayerLength)
69      if hLayerLength == 1:

71          delta3 = np.multiply(-(y-self.yHat), self.sigmoid(
    self.z3,deriv=True))
            dJdW2 = np.dot(self.yHat2.T, delta3)
73          delta2 = np.dot(delta3, self.W2.T)*self.sigmoid(self
    .z2,deriv=True)
            dJdW1 = np.dot(X.T, delta2)
75
            return dJdW1, dJdW2
77
        if hLayerLength == 2:
79
            delta4 = np.multiply(-(y-self.yHat), self.sigmoid(
    self.z4,deriv=True))
81          dJdW3 = np.dot(self.yHat3.T, delta4)

83          delta3 = np.dot(delta4, self.W3.T)*self.sigmoid(self
    .z3,deriv=True)
            dJdW2 = np.dot(self.yHat2.T, delta3)
85
            delta2 = np.dot(delta3, self.W2.T)*self.sigmoid(self
    .z2,deriv=True)
87          dJdW1 = np.dot(X.T, delta2)

89          return dJdW1, dJdW2, dJdW3

91
- computeGradients, utiliza la funcion DerivFuncionCosto para
    generar el gradiente de cada peso y utiliza la funcion ravel
    de la libreria numpy con el fin de devolver un arreglo
    contiguo:
93
    def computeGradients(self, X, y, hLayerLength):
95
        if hLayerLength == 1:
97          dJdW1, dJdW2 = self.DerivFuncionCosto(X, y,
    hLayerLength)
            return np.concatenate((dJdW1.ravel(), dJdW2.ravel())
```

```
    )
99
        if hLayerLength == 2:
101         dJdW1, dJdW2, dJdW3 = self.DerivFuncionCosto(X, y,
    hLayerLength)
            return np.concatenate((dJdW1.ravel(), dJdW2.ravel(),
     dJdW3.ravel()))
103


105
- Entrenamiento, es una funcion que ayuda a realizar el
    entrenamiento de la ANN
107
     def Entrenamiento(self, trainX, trainY, testX, testY,
    hLayerLength):
109
        self.X = trainX
111     self.y = trainY

113     self.testX = testX
        self.testY = testY
115     self.hLayerLength = hLayerLength

117     self.J = []
        self.testJ = []
119
- Se utiliza la funcion optimize.minimize de la libreria scipy,
    que permite minimizar la funcion de error por medio de un
    metodo de optimizacion numerica, se utiliza el algoritmo
    Broyden-Fletcher-Goldfarb-Shanno (BFGS) el cual tiene en
    cuenta la segunda derivada de la funcion objetivo para
    realizar de forma mas rapida el descenso y busqueda de la
    minimizacion. Esta funcion requiere como entrada, la funcion
    objetivo, datos de entrada y de salida (X, y) y devuelve el
    costo y los gradientes.
121
        params0 = self.N.getParams(self.hLayerLength)
123
        options = {'maxiter': 200, 'disp' : True}
125     _res = sp.optimize.minimize(self.costFunctionWrapper,
    params0, jac=True, method='BFGS', \
                                    args=(trainX, trainY), options=
    options, callback=self.callbackF)
127
        self.N.setParams(_res.x, self.hLayerLength)
129     self.optimizationResults = _res
```

# Anexos B

# Algoritmo Genético

En este anexo se presentan las funciones utilizadas para desarrollar el algoritmo genético.

```
class GANN(object):

- Se determinan los parametros del GA para realizar la
    optimizacion:

    def __init__(self, tam_poblacion=30, punto_crossover=[0, 1],
                 tasa_mutacion=.01, funcion_ajuste=None,
    swap_rate=.2,
                 generaciones=100,encoding='binary', step_size
    =2, max_layers=10, max_nodes=80):

        if fitness_function is None:
            raise Exception('Ingrese funcion de ajuste.')

        self.tam_poblacion = tam_poblacion
        self.punto_crossover = punto_crossover
        self.tasa_mutacion = tasa_mutacion
        self.funcion_ajuste = funcion_ajuste
        self.swap_rate = swap_rate
        self.generaciones = generaciones
        self.encoding = encoding
        self.step_size = step_size
        self.max_layers = max_layers
        self.max_nodes = max_nodes


- Se genera la poblacion inicial para el algoritmo:

        self.ajuste_poblacion = []

```

```
        initial_population = []
        for individual in range(0, tam_poblacion):

            individual_arr = []
            num_layers = random.randint(1, max_layers)
            for layer in range(0, num_layers):
                num_nodes = random.randint(1, max_nodes)
                individual_arr.append(num_nodes)
            if num_layers < max_layers:
                for iteration in range(0, max_layers -
    num_layers):
                    individual_arr.append(0)


- Se van guardando tanto la estructura generada como su valor de
     ajuste, la funcion vector devuelve el vector que representa
    la estructura de la red:

            individuo = Individual(self.vector(individual_arr),
                                                self.fitness_function)
            initial_population.append(individuo)


- la fucion calc_ajuste devuelve el ajuste de cada individuo,
    como se explica en la siguiente ecuacion.

        self.ajuste_poblacion= self.calc_ajuste(
    initial_population)


- vector permite eliminar (si se llegan a generar) ceros extras
    del vector seleccionado

     def vector(self, vector):
        try:
            return vector[:vector.index(0)]
        except ValueError:
            return vector


- calc_ajuste recibe como argumento la poblacion para la que se
    va a calcular el ajuste, y devuelde :

     def calc_ajuste(self, individuals):
        valor_ajuste = [indiv.fitness() for indiv in individuals
    ]
        genotipos = [self.genotipo(indiv) for indiv in
    individuals]
        ajuste_poblacion = zip(genotipos, valor_ajuste)
```

```
        return ajuste_poblacion
70
72 - genotipo, permite obtener la estructura de la red en codigo
     binario:

74      def genotipo(self, indiv):
            hidden_layers = indiv.phenotype # quita el input y el
     output layer
76          if self.encoding == 'binary':
                genotype = str()
78              for layer in hidden_layers:
                    genotype += format(layer, '07b')
80              return genotype
            else:
82              return hidden_layers

84
   - Con estas dos funciones convertimos nuevamente a enteros
     nuestra estructura
86
        def cadenabit(self,genotype):
88          num_chars = 7
            return [genotype[i:i + num_chars] for i in
90                  range(0, len(genotype), num_chars)]

92      def vector_cadenabit(self, genotype):
            vector = [int(el, 2) for el in self.cadenabit(genotype)]
94          return self.vector(vector)

96
   - La funcion crossover permite seleccionar los mejores
     individuos para realizar los cruces
98
        def crossover(self):
100
            self.ajuste_poblacion = sorted(self.ajuste_poblacion,
102                                        key=operator.itemgetter
     (1))
            slice_to = (self.tam_poblacion -
104                     int(self.swap_rate * self.tam_poblacion))
            return self.ajuste_poblacion[:slice_to]
106

108 - Aplicar_cruce es la funcion que realiza el cruce de los padres
     .

110     def Aplicar_cruce(self,genotype_1, genotype_2):
            """Reproduction between two parent genotypes."""
```

```
112
        HL_child = round((len(genotype_1) * self.punto_crossover
    [0] + len(genotype_2) * self.punto_crossover[1])/int(7))
114
        length_child = HL_child*int(7)
116
        Total_length = len(genotype_1)+len(genotype_2)
118
        punto_crossover  = [int(round(((len(genotype_1)*1.0)/(
    Total_length*1.0))*length_child)),int(round(((len(genotype_2)
    *1.0)/(Total_length*1.0))*length_child))]
120
        new_genotype = (
122         genotype_1[:punto_crossover[0]] +
            genotype_2[punto_crossover[0]:(punto_crossover[0]+
    punto_crossover[1])])
124     return new_genotype

126
- Mutacion es la funcion que permite aplicar la mutacion de
    acuerdo a la tasa definida, tiene como argumento el individuo
     que va a ser sometido al proceso de mutacion:
128
    def Mutacion(self, child_genotype):
130
        if self.encoding == 'binary':
132
            analisis = self.cadenabit(child_genotype)
134
            try:
136             first_zero = analisis.index('0000000')
                genes_to_consider = analisis[:first_zero + 1]
138         except ValueError:
                # si no hay ceros
140             genes_to_consider = analisis

142 - En este paso consideramos la codificacion de Gray para no
    generar variaciones muy grandes con la mutacion:
            new_child = str()
144         for gene in genes_to_consider:
                gen = GrayCode(7,rank=int(gene,2)).current
146             child = str()
                for c in gen:
148                 if random.random() < self.tasa_mutacion:
                        if c == '0':
150                         c = '1'
                        elif c == '1':
152                         c = '0'
                    child += c
```

```
154                 new_child += gray_to_bin(child)

156             binary_string_length = int(7)
                return new_child.ljust(binary_string_length, '0')
158
        # version entero
160         else:
                try:
162                 first_zero = child_genotype.index(0)
                    genes_to_consider = child_genotype[:first_zero +
     1]
164             except ValueError:
                    genes_to_consider = child_genotype
166
                new_child = []
168             for gene in genes_to_consider:
                    if random.random() < self.options['tasa_mutacion
    ']:
170                     gene += self.options['step_size']
                    new_child.append(gene)
172             return (new_child + [0] * self.max_layers)[:self.
    max_layers]
```

# Bibliografía


[1] Alander, J. T. On optimal population size of genetic algorithms. In *1992 Proceedings Computer Systems and Software Engineering* (1992), IEEE, pp. 65–70.

[2] Alexandridis, A. K., and Zapranis, A. D. *Wavelet neural networks: with applications in financial engineering, chaos, and classification*. John Wiley & Sons, 2014.

[3] Blau, W. *Momentum, direction, and divergence*, vol. 5. John Wiley & Sons, 1995.

[4] Brabazon, A., and O'Neill, M. *Natural computing in computational finance*, vol. 100. Springer Science & Business Media, 2011.

[5] Dunis, C. L., Middleton, P. W., Karathanasopolous, A., and Theofilatos, K. *Artificial Intelligence in Financial Markets: Cutting Edge Applications for Risk Management, Portfolio Optimization and Economics*. Springer, 2016.

[6] Edwards, R. D., Magee, J., and Bassetti, W. C. *Technical analysis of stock trends*. CRC press, 2007.

[7] Goldberg, D. E., and Holland, J. H. Genetic algorithms and machine learning. *Machine learning 3*, 2 (1988), 95–99.

[8] Graupe, D. *Principles of artificial neural networks*, vol. 6. World Scientific, 2007.

[9] Idler, C. *Pattern recognition and machine learning techniques for algorithmic trading*. PhD thesis, MA thesis, FernUniversität, Hagen, Germany, 2014.

[10] JingTao, Y., and Tan, C. L. Guidelines for financial forecasting with neural networks. In *Int. Conf. Neural Information Processing, Shanghai, China* (2001), Citeseer.

[11] Johnson, B. *Algorithmic Trading & DMA: An introduction to direct access trading strategies*, vol. 200. 4Myeloma Press London, 2010.

[12] Kantartzis, I., and Papakostas, A. Las ondas del éxito. *TRADERSÂ´ 04* (2016), 38–41.

[13] Leardi, R. *Nature-inspired methods in chemometrics: genetic algorithms and artificial neural networks*, vol. 23. Elsevier, 2003.

[14] Lo, A. W. Reconciling efficient markets with behavioral finance: the adaptive markets hypothesis.

[15] Mostafa, E. H., Mohammed, E. H., and Abderrahim, E. A. Minimization of value at risk of financial assets portfolio using genetic algorithms and neural networks. *Journal of Applied Finance & Banking 6*, 2 (2016).

[16] Murphy, J. J. *Análisis Técnico De Los Mercados Financieros (Sin colección)*. 2000.

[17] Pring, M. J. *Martin Pring's Introduction to Technical Analysis*. McGraw-Hill Education, 2015.

[18] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning internal representations by error propagation. Tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[19] Thinyane, H., and Millin, J. An investigation into the use of intelligent systems for currency trading. *Computational Economics 37*, 4 (2011), 363–374.

[20] Zavadskaya, A., et al. Artificial intelligence in finance: Forecasting stock market returns using artificial neural networks (available on internet).