ISSN 1870-4069

Optimización de hiperparámetros de una red Long Short Term Memory para pronósticos financieros

Francisco J. Pedroza-Castro, Alfonso Rojas-Domínguez, Martín Carpio, Manuel Ornelas-Rodríguez, Héctor Puga

Tecnológico Nacional de México,
Instituto Tecnológico de León,
México

alfonso.rojas@gmail.com

Resumen. El proceso de optimización de hiperparámetros de una red neuronal artificial, usualmente se lleva cabo de manera manual usando Grid Search o Random Search. En este artículo, estudiamos una metodología para optimizar hiperparámetros de una red Long Short Term Memory (LSTM) usando dos metaheurísticas bio-inspiradas, Particle Swarm Optimization, y Flower Pollination Algorithm y una Estimation of Distribution Algorithm (EDA), Low Number of Function Evaluation, para que la red realice un pronóstico financiero del precio de cierre del siguiente día de las acciones de Google y Nike. Los resultados muestran que las redes LSTM obtenidas mediante el proceso de optimización de hiperparámetros son más simples, resultando en menor tiempo de entrenamiento, y mayor rendimiento, que las redes no optimizadas con metaheurísticas. Los errores de prueba obtenidos de las soluciones son de 10E-4 hasta 10E-6. Adicionalmente realizamos una comparación de las metaheurísticas, concluyendo que la EDA encuentra una solución esperada con menor cantidad de llamadas a función, lo que se traduce en menor costo computacional y tiempo de ejecución, con resultados satisfactorios.

Palabras clave: Long Short Term Memory, estimation of distribution algorithm, bio-inspired metaheuristics, algorithm for a low number of function evaluations, particle swarm optimization, flower pollination algorithm, pronóstico financiero.

Hyperparameter Optimization of a Long Short Term Memory Network for Financial Forecasting

Abstract. The hyperparameter optimization process of an artificial neural network is usually carried out manually using Grid Search or Random Search. In this paper, we study a methodology to optimize hyperparameters of a Long Short Term Memory (LSTM) network using two bio-inspired metaheuristics, Particle Swarm Optimization, and Flower Pollination Algorithm, and an Estimation of Distribution Algorithm (EDA), Low Number of Function Evaluation, for the

network to make a financial forecast of the next day's closing price of Google and Nike stock. The results show that the LSTM networks obtained through the hyperparameter optimization process are simpler, resulting in less training time, and higher performance, than the networks not optimized with metaheuristics. The resulting proof errors of the solutions are 10E-4 up to 10E-6. Additionally, we perform a comparison of the metaheuristics, concluding that the EDA finds an expected solution with fewer function calls, which translates into lower computational cost and execution time, with satisfactory results.

Keywords: Long Short Term Memory, estimation of distribution algorithm, bio-inspired metaheuristics, algorithm for a low number of function evaluations, particle swarm optimization, flower pollination algorithm, financial forecast.

1. Introducción

Inversionistas, academia e investigadores, del campo de pronósticos financieros se han visto interesados por el campo de la inteligencia computacional, ya que las investigaciones muestran ventajas significativas en contraste con los métodos clásicos, como análisis técnico, análisis fundamental, y métodos estadísticos [12, 13, 21].

Las investigaciones en inteligencia computacional aplicadas a pronósticos financieros, muestran que el uso de redes neuronales híbridas con algoritmos evolutivos, obtienen un mayor rendimiento en comparación con las redes no-híbridas. Dichas hibridaciones van desde la optimización de parámetros hasta hiperparámetros [12,13,21]. Este último es un campo del Automated Machine Learning, conocido como Hyperparameter Optimization (HPO). Por lo que en este artículo nos referimos a la hibridación para ajuste de hiperparámetros como HPO [11].

De hecho, el proceso de ajuste de hiperparámetros de una Red Neuronal Artificial, comúnmente se lleva a cabo usando Random Search o Grid Search [15]. HPO busca realizar este proceso de manera automatizada utilizando, comúnmente, diferentes técnicas como Algoritmos Genéticos y Métodos Bayesianos [11, 15].

El presente artículo, muestra una metodología propia para llevar a cabo la optimización de hiperparámetros, probando 3 metaheurísticas: 2 algoritmos bio-inspirados, Particle Swarm Optimization y Flower Pollination Algorithm, y 1 algoritmo perteneciente a las Estimation of Distribution Algorithm, Estimation of Distribution Algorithm Low Number of Function Evaluation (EDALNFE, en este artículo nos referiremos a él como LNFE).

Estos algoritmos se compararon y adicionalmente se contrastaron con una red LSTM que fue ajustada de manera manual por [16]. Los resultados obtenidos muestran que la optimización de hiperparámetros con metaheurísticas diseñan una red más simple con mejores rendimientos en comparación de la hechas manualmente.

El trabajo está organizado como sigue: en la Sección 2, presentamos diferentes trabajos que se han utilizado para realizar pronósticos financieros utilizando redes neuronales artificiales, investigaciones que optimizan hiperparámetros en pronósticos financieros y de otros campos. En la Sección 3, se describe la teoría de la red LSTM y

las metaheurísticas. En la Sección 4, exponemos la metodología propuesta para realizar la optimización de hiperparámetros y comparación de metaheurísticas.

En la sección 5 se muestran los resultados experimentales y discusiones, y finalmente en la Sección 6 las conclusiones de nuestro trabajo.

2. Trabajos relacionados

Los investigadores en pronósticos financieros han encontrado que el uso de inteligencia computacional genera resultados satisfactorios que superan a los métodos clásicos [4, 12–14, 16, 21]. Estos métodos van desde un Algoritmo Genético (AG) [4, 7], Lógica Difusa [4, 6], Sistemas Expertos [4], y Redes Neuronales Artificiales (RNAs) [18], entre otros. Las RNAs se han utilizado mayormente, especialmente la red LSTM, que de acuerdo a la literatura, son las que mayor rendimiento tienen [12–14, 19, 21, 24].

En la literatura para pronósticos financieros usando redes neuronales, del periodo de 2017 a 2019, cerca del 51 % de los artículos revisados utilizan algún tipo de Red Neuronal Recurrente (RNR), de ese porcentaje, el 61 % utilizan redes LSTM, la razón es la misma que se menciona en el párrafo anterior [21], además de ser relativamente fácil de aplicar.

Kumar *et. al* (2021) optimizan hiperparámetros de una red LSTM para pronósticos de tendencias de activos financieros, utiliza PSO y FPA. Su trabajo lo comparan con un modelo sin uso de metaheurísticas, los resultados muestran que utilizar metaheurísticas para la optimización, genera resultados superiores a los casos donde no se usa metaheurísticas [13].

En otro trabajo se realiza un sistema de comercio automatizado usando un AG y un RNA, obteniendo 72.5 % de precisión y 23.3 % de Rendimiento Anual Neto. En su trabajo encontraron que existen tres factores importantes que afectan el rendimiento del pronóstico: Pre-procesamiento de datos adecuado y presentación de los datos, estrategias óptimas de comercio, y la estructura del modelo a pronosticar [7].

En otros trabajos se realiza un sistema de comercio utilizando una red LSTM [22], su red la llevan a un sistema simulado considerando los costos de transacción, y diferentes versiones de la red LSTM, con lo cual llegan a tener rendimientos de hasta 228.94 % en el mejor de los casos.

Sezer *et. al* (2020), además, identifica que en diversos trabajos de pronósticos financieros, con redes neuronales artificiales, el ajuste de hiperparámetros es de vital importancia para un adecuado rendimiento de la red al realizar el pronóstico de activos financieros. Consideran que encontrar los mejores hiperparámetros de las redes neuronales es un problema significativo.

De hecho HPO optimizan los hiperparámetros usando Métodos Bayesianos (MB) como Sequential Model-Based Global Optimization (SMBGO), The Gaussian Process Approach (GPA), Tree-structured Parzen Estimator Approach (TSPEA) [2, 3], entre otros algoritmos. Por ejemplo, se ha realizado optimización de hiperparámetros en campos diferentes a pronósticos financieros, donde se usa GA y MB [11, 15], con resultados satisfactorios. También se ha hecho uso de Aprendizaje por Refuerzo Profundo [5], pero en este caso los resultados superan a GA y MB, incluso, en trabajos relacionados, se ha

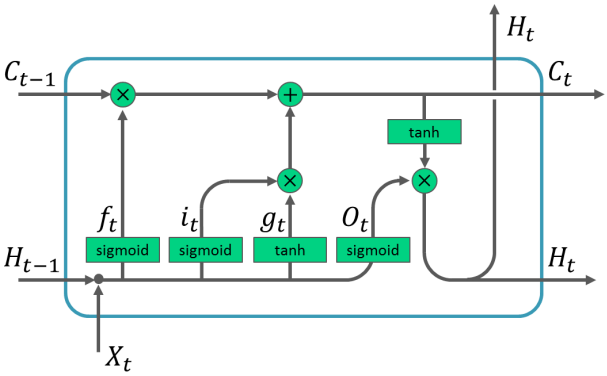

Fig. 1. Estructura de una celda LSTM (Elaboración propia).

hecho uso de una red LSTM para la optimización, teniendo resultados satisfactorios y en un tiempo reducido [13].

Otro trabajo de optimización de hiperparámetros de una red tipo LSTM fue el de [1], quien consideran que es una área de oportunidad para las metaheurísticas con el objeto de buscar los hiperparámetros de la red mencionada. Ellos proponen utilizar Grey Wolf Optimizer (GWO), con lo cual encuentran que una red LSTM de arquitectura simple, la cual puede tener el mismo rendimiento que una red más compleja. Atribuyéndolo a la adecuada combinación de hiperpárametros.

En este trabajo, siguiendo la línea marcada por [12], utilizaremos algoritmos evolutivos para la optimización de hiperparámetros, los cuales son dos bio-inspirados ya utilizados en la literatura, PSO, y FPA, los cuales han tenido resultados satisfactorios al optimizar los hiperparámetros de una red LSTMA; adicionalmente utilizaremos una EDA, EDALNFE que de acuerdo a su diseño hace una menor cantidad de llamadas a función, lo cual puede ser una ventaja, ya que las otras metaheurísticas tiene que hacer llamadas a función a toda la población en cada iteración; en el presente caso, que se usa RNAs, la llamada a función implica entrenar la red en cada iteración, generando un alto costo computacional. El proceso de optimización se seguirá utilizando una metodología propuesta por nosotros. Al final contrastamos cada uno de los algoritmos con una red LSTM que se ajustó sin metaheurísticas por [16].

3. Teoría

3.1. Long Short Term Memory (LSTM)

La red LSTM es un tipo de RNR que puede trabajar con hasta 1000 retrasos en el tiempo, evitando los problemas del *exploding-gradient* y el *vanishing-gradient* [20]. La red trabaja simulando compuertas lógicas que permiten eliminar o agregar información a un *Cell State* (5) como se describe a continuación[1]: primero Forget gate (2) decide que información se olvida o se mantiene; después, la Input gate (1) y Candidate gate

[1] En este trabajo, $\otimes$ representa *element-wise* o el producto Hadamard, sig$(\cdot)$ es la función sigmoide y tanh$(\cdot)$ función tangente hiperbólica.

(4) deciden que información se agrega; finalmente se genera el Hidden state (6) usando Output gate (3) y la Cell state (5) [10]. Las ecuaciones para llevar dicho proceso se muestran a continuación. En la Figura 1 se muestra el funcionamiento de la red LSMT:

$$\textbf{Input gate:} \quad i_t = \text{sig}(x_t U_i + h_{t-1} W_i + b_i), \quad i_t \in \mathbb{R}^{d_H}, \tag{1}$$

$$\textbf{Forget gate:} \quad f_t = \text{sig}(x_t U_f + h_{t-1} W_f + b_f), \quad f_t \in \mathbb{R}^{d_H}, \tag{2}$$

$$\textbf{Output gate:} \quad O_t = \text{sig}(x_t U_o + h_{t-1} W_o + b_o), \quad O_t \in \mathbb{R}^{d_H}, \tag{3}$$

$$\textbf{Candidate gate:} \quad g_t = \tanh(x_t U_g + h_{t-1} W_g + b_g), \quad g_t \in \mathbb{R}^{d_H}, \tag{4}$$

$$\textbf{Cell state:} \quad C_t = f_t \otimes C_{t-1} + i_t \otimes g_t, \quad C_t \in \mathbb{R}^{d_H}, \tag{5}$$

$$\textbf{Hidden state:} \quad H_t = O_t \otimes \tanh(C_t), \tag{6}$$

$$\textbf{State:} \quad s_t = R_{LSTM}(s_{t-1}, x_t) = [c_t; H_t], \quad s_t \in \mathbb{R}^{2d_H}, \tag{7}$$

donde U y $W \in \mathbb{R}^{d_x \times d_H}$ son los pesos, $x_t \in \mathbb{R}$. Si se utiliza tamaños de lote d_b, entonces: $s_t \in \mathbb{R}^{2d_H \times d_b}$, C_t, H_t, i_t, f_t, O_t, $g_t \in \mathbb{R}^{d_H \times d_b}$, y $b \in \mathbb{R}^{d_b \times d_H}$.

3.2. Particle Swarm Optimization (PSO)

Esta metaheurística se inspira en el comportamiento social de insectos, animales y humanos. El algoritmo inicia con una población aleatoria, donde cada partícula de la población se considera una solución potencial, las cuales se van desplazando en el espacio de búsqueda con el objeto de mejorar la mejor solución encontrada $\mathbf{g}^*$ en la iteración previa, usando la información de los vecinos $\mathbf{g}^*$ y su mejor posición $\mathbf{p}_i$ (en la que alguna vez estuvo) [9]. Este concepto se presenta en la regla de actualización (8):

$$\mathbf{v}_i^{New} = \chi(w\mathbf{v}_i + \varphi_1 \otimes (\mathbf{p}_i - \mathbf{x}_i) + \varphi_2 \otimes (\mathbf{g}^* - \mathbf{x}_i)), \tag{8}$$

donde $\mathbf{v}_i$ es la velocidad de la partícula i, $\mathbf{x}_i$ es la posición de la partícula i, w es la inercia o peso de inercia, $\mathbf{g}^*$ es la mejor posición global, $\varphi_1 = R_1 c_1$, $\varphi_2 = R_2 c_2$, donde c_1 es el coeficiente cognitivo, c_2 el coeficiente social, $R_1, R_2 \sim \mathcal{U} \in [0, 1]$, y χ el coeficiente de constricción dado por la Ec. (9). La Ec. (10) es la regla actualización de la posición de la partícula. El pseudocódigo del PSO se muestra en Algoritmo 1:

$$\chi = \frac{2}{\left|2 - \varphi' - \sqrt{\varphi'^2 - 4\varphi'}\right|}, \quad \varphi' = c_1 + c_2, \varphi' > 4, \tag{9}$$

$$\mathbf{x}_i^{New} = \mathbf{x}_i + \mathbf{v}_i^{New}. \tag{10}$$

3.3. Flower Pollination Algorithm (FPA)

El FPA es un algoritmo inspirado en el proceso de polinización de las plantas bajo la óptica del proceso de optimización [25]. Tanto FPA como PSO son algoritmos bio-inspirados, los cuales son similares al seguir un proceso de generación de población

Algorithm 1 PSO, asumiendo minimización.

Generación de población inicial de manera aleatoria $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$.
Encontrar la mejor solución $\mathbf{g}^*$ en X.
while *Criterio de paro* no se cumpla, **do**
    for partícula $\mathbf{x}_i$ **do**
        if $f(\mathbf{x}_i) < f(\mathbf{p}_i)$ **then**
            $\mathbf{p}_i \leftarrow \mathbf{x}_i$; actualizar la mejor posición de la partícula i.
        end if
        Identificar la mejor posición de la iteración $\mathbf{g}^*$.
        Actualizar velocidad $\mathbf{v}_i$ vía (8).
        Actualizar posición $\mathbf{x}_i$ vía (10).
    end for
end while
return La mejor solución encontrada $\mathbf{g}^*$.

inicial y mover las soluciones en el espacio de búsqueda, no obstante FPA añade ruido usando una distribución Lévy para explorar, mientras que PSO solo usa números aleatorios uniformes. El FPA usa dos conceptos de manera general, polinización global y polinización local, para realizar la búsqueda. Las Ecs. (11) a (13) representan dichas ideas. El pseudocódigo se muestra en el Algoritmo 2.

Algorithm 2 FPA, asumiendo minimización.

Crear población inicial de manera aleatoria $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$.
Encontrar la mejor solución $\mathbf{g}^*$.
Define el sego de polinizavión $\rho \in [0, 1]$.
while *Criterio de paro* no es encontrado, **do**
    for por cada flor/gameto $\mathbf{x}_i$ ($i = 1$ to n) **do**
        if $rand < \rho$ **then** ▷ Donde $rand \in [0, 1]$.
            Crea números aleatorios por cada variable de la solución usando el vuelo de Lévy.
            Crea solución temporal $\mathbf{x}'$ con la polinización global Ec. (11).
        else
            Crea un número aleatorio ϵ de manera uniforme ($\mathcal{U} \in [0, 1]$) por cada variable.
            Crea solución temporal $\mathbf{x}'$ con la polinización local Ec. (14).
        end if
        if $f(\mathbf{x}') < f(\mathbf{x}_i)$ **then**
            $\mathbf{x}_i \leftarrow \mathbf{x}'$
        end if
    end for
    Identifica la mejor solución $\mathbf{g}^*$ de la iteración.
end while
return La mejor solución encontrada $\mathbf{g}^*$.

La polinización global se define por (11), donde $\mathbf{x}_i^t$ es una flor/gameto, $\mathbf{g}^*$ la mejor solución (flor/gameto) global, γ es el factor de escalamiento del vuelo Lévy, y $L(\lambda)$ es un número aleatorio obtenido mediante la distribución del vuelo de Lévy de acuerdo a la Ec. (12):

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \gamma L(\lambda) \otimes (g^* - \mathbf{x}_i^t), \tag{11}$$

$$L(\lambda) \sim \frac{\lambda \Gamma(\lambda) \sin(\pi\lambda/2)}{\pi} \cdot \frac{1}{s^{1+\lambda}}, \quad s \gg s_0 > 0, \quad s = \frac{U}{|V|^{1/\lambda}}, \tag{12}$$

donde $V \sim \mathcal{N}(0,1)$ y $U \sim \mathcal{N}(0, \sigma_{fpa}^2)$, con desviación estándar σ_{fpa} obtenida con la Ec. (13):

$$\sigma_{fpa}^2 = \left[\frac{\Gamma(1+\lambda)}{\lambda \Gamma((1+\lambda)/2)} \cdot \frac{\sin(\pi\lambda/2)}{2^{(\lambda-1)/2}} \right]^{1/\lambda}. \tag{13}$$

La polinización local se define por (14), donde $\mathbf{x}_k^t$ y $\mathbf{x}_j^t$ son dos flores/gametos tomadas de manera aleatoria de la población. Y ϵ es un número aleatorio en el intervalo $[0, 1]$:

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \epsilon \otimes (\mathbf{x}_j^t - \mathbf{x}_k^t). \tag{14}$$

3.4. Algorithm for a Low Number of Function Evaluations (EDALNFE)

LNFE[2] pertenece a la familia de Estimation of Distribution Algorithms (EDAs). La idea de una EDA es iniciar con una población la cual se usa para crear una media y varianza que es utilizada para generar una nueva muestra poblacional mediante una distribución de probabilidad previamente definida.

La idea central del LNFE es crear un vector direccional con la correlación de Pearson, para crear una nueva población. La ventaja de este algoritmo es que hace una menor cantidad de llamadas a función [17]. El pseudocódigo se muestra en el Algoritmo 3[3].

Este algoritmo inicia con una población, cada solución es evaluada y se le asigna una probabilidad de acuerdo a su fitness. La mejor solución y su fintess, X_{elite}, F_{elite}, se usa para generar una media y varianza (ver [23]). Si el fitness no mejora después de n iteraciones la población se reinicia preservando la mejor solución. La nueva población se genera mediante dos distribuciones (ver línea 8 en Algoritmo 3), una distribución normal multivariada y una distribución sobre el vector dirigido a una dirección prometedora. En cada iteración se verifica que el vector proyección obtenido por la correlación de Pearson mejore al mejor fitness, de lo contrario se reinicia la población. Este proceso se repite hasta cumplir con el criterio de paro (para más detalles ver [17]).

4. Metodología

Para realizar la optimización de hiperparámetros de la red LSTM y pueda así pronosticar los precios de cierre del siguiente día de las acciones de Google y Nike.

[2] En este articulo nos referimos a EDALNFE como LNFE.

[3] Este algoritmo trabaja en el espacio $[0, 1]$, lo cual separa al problema del algoritmo.

Algorithm 3 Algorithm for a Low Number of Function Evaluations (EDALNFE).

$N_p \leftarrow 8D + 2$; $C_{elite} \leftarrow 0$; $C_{resets} \leftarrow 0$
$X \leftarrow$ initialize(d, N_p) ▷ Inicia población de manera aleatoria X.
$F \leftarrow$ evaluate$(X, f(\cdot))$ ▷ Donde $f(\cdot)$ es la función fitness.
$[ibest, ps] \leftarrow$ selection(F)
$[X_{elite}, F_{elite}] \leftarrow [X_{ibest(1)}, F_{ibest(1)}]$
$f_{reset} \leftarrow F_{elite(1)}$ ▷ Guardar la mejor solución.
while *Criterio de paro* ($var_c > var_{min}$) no se cumple, **do**
    $\left[\hat{X}, U\right] \leftarrow$ generateNewIndividuals$(X, F, X_{elite}, ps, ibest, U \leftarrow rand(D))$
    $\hat{F} \leftarrow$ evaluate$(\hat{X}, f(\cdot))$ ▷ Evaluar la nueva población.
    if $\left|F_{elite} - F_{ibest(1)}\right| > th_{elite}$ **then** ▷ Si no mejora
        $C_{elite} \leftarrow 0$ ▷ Reiniciar población.
        $[X_{elite}, F_{elite}] \leftarrow \left[X_{ibest(1)}, F_{ibest(1)}\right]$ ▷ Actualizar la mejor solución elite.
    else
        $C_{elite} \leftarrow C_{elite} + 1$ ▷ Incrementar el contador de elite.
    end if
    if $C_{elite} = MaxC_{elite}$ **then** ▷ Si la petición de la mejor solución se repite en un limite, reiniciar.
        $C_{resets} \leftarrow C_{resets} + 1$ ▷ Incrementar el contador de reinicio.
        $X_{elite} \leftarrow X_{ibest(1,\ldots,C_{resets})}$
        $F_{elite} \leftarrow F_{ibest(1,\ldots,C_{resets})}$
        $X \leftarrow$ reinitialize$(D, Np - 1, C_{resets}, X_{elite})$ ▷ Reiniciar población.
        $F \leftarrow$ evaluate$(X, f(\cdot))$ ▷ Evaluar la nueva población.
        $[X, F] \leftarrow [[X, X_{elite}], [F, F_{elite}]]$ ▷ Agregar la mejor solución elite a la nueva población.
        $[ibest, F_{elite}] \leftarrow$ selection(F)
        $[X_{ibest}, ps] \leftarrow \left[X_{ibset(1)}, F_{ibest(1)}\right]$
        $C_{elite} \leftarrow 0$
        if $F_{elite(1)} = f_{reset}$ or $C_{resets} = MaxC_{resets}$ **then** ▷ Si no mejora, detener el algoritmo.
            break
        else
            $f_{reset} \leftarrow F_{elite(1)}$ ▷ Si la solución mejora actualizarla.
        end if
    end if
    $[X, F] \leftarrow [X(ibest), F(ibest)]$ ▷ Ordenar las soluciones de acuerdo a su valor fitness.
    $[X, F] \leftarrow [X(1, \ldots, Np/2), F(1, \ldots, Np/2)]$ ▷ Mantener la mitad de la población con mejor fitness.
    $[X, F] \leftarrow \left[\left[X, \hat{X}\right], \left[F, \hat{F}\right]\right]$ ▷ Añadir nuevas soluciones a la población.
end while
return La mejor solución encontrada, $[x_{elite}, f_{elite}]$

Utilizamos 2 algoritmos bio-inspirados (PSO y FPA) y una EDA (LNFE). Para realizar el entrenamiento de las redes LSTM usamos el optimizador Adam y, como función de pérdida el error cuadrático medio. Seleccionamos el 65 % de la serie de tiempo del precio de cierre para el conjunto de entrenamiento y 35 % para el conjunto de prueba.

Tabla 1. Parámetros usados en metaheurísticas.

Parámetro	Valor
PSO	
Coeficiente cognitivo, c_1	2.05
Coeficiente social, c_2	2.05
Peso de inercia, w	0.8
Número de partículas (Soluciones)	25 (Usado en [12])
Número de iteraciones	16
FPA	
Sesgo de polinización, ρ	0.8
Factor de escalamiento de vuelo Lévy, γ	0.1
Número Flores/gametos (Soluciones)	25 (Usado en [12])
Número de iteraciones	16
LNFE	
Número de soluciones	26

Los datos se obtuvieron de la web Yahoo Finance[4]. Los periodos seleccionados para las acciones de Google fuero de 24/08/2004 a 21/01/2022; para el caso de Nike el periodo fue de 02/12/1980 a 24/01/2022. Estas acciones fueron las mismas que usaron [16]; se replicó el trabajo de estos últimos para compararlo con las metaheurísticas. Es menester resaltar que [16] en su trabajo únicamente muestra 4 configuraciones de red LSTM, cambiando únicamente el número de épocas, sin mostrar otras configuraciones con demás hiperparámetros. Al modelo de [16] nos referiremos como Optimización sin Metaheurística (OSM).

En cada experimento se decidió normalizar los datos usando $min - max$ en el intervalo entre $[0, 1]$. Los hiperparámetros que se optimizaron se muestran en la primera columna del Cuadro 2, y su símbolo en la segunda columna. Los hiperparámetros Cl_s, H_s, α, W_s, B_s, y E_s, fueron seleccionados como lo hace [13] (ver Cuadro 2).

Los parámetros de control del PSO fueron seleccionados como sugiere [9] en el caso de FPA se seleccionaron como sugiere [25]. En ambos casos (PSO y FPA) el número de soluciones iniciales se seleccionan como sugiere [13]. En el caso de LNFE la población fue de 26, ya que el diseño del algoritmo sólo así lo permite. Estos parámetros son mostrados en el Cuadro 1.

Tanto PSO como FPA trabajan sobre el espacio de los números continuos. Por lo que decidimos mapear los valores al espacio de los discretos redondeando los números como lo sugiere [8] para solucionar problemas de variables mixtas. De tal modo las soluciones tienen valores de acuerdo al Cuadro 2.

El espacio de tamaño de lote se seleccionó como se usa típicamente en RNAs. Las épocas se seleccionaron de acuerdo a experimentos previos y el trabajo de [16]. El tamaño de la ventana se usa de acuerdo a las investigaciones realizadas por [21]. El número de estados ocultos se seleccionó para reducir el espacio de búsqueda, no obstante, se incluye la solución de [16]. El número de celdas LSTM se selecciona como sugiere [13] (ver Cuadro 2). Para realizar una comparación justa de las metaheurísticas

[4] https://finance.yahoo.com/

Tabla 2. Hiperparámetros y espacio de búsqueda.

hiperparámetro	Simbolo	Espacio de búsqueda	Limite
Celda LSTM	Cl_S	$\{1, 2, 3, 4, 5, 6, 7, 8\}$	$[1, 8]$
Estado oculto	H_S	$\{1, 20, 50, 80, 96, 110, 150\}$	$[1, 7]$
Tamaño de la ventana	W_S	$\{20, 40, 60, 80, 90, 100\}$	$[1, 6]$
Factor de aprendizaje	α	$\left\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\right\}$	$[1, 5]$
Tamaño de lote	B_S	$\{8, 16, 32, 64\}$	$[1, 4]$
Número de épocas	E_S	$\{12, 25, 50, 100, 150\}$	$[1, 5]$

decidimos utilizar un mínimo de 400 llamadas a función, dicho número se fijó por experimentos previos.

Para entrenar la red se consideró añadir condiciones de paro para disminuir el costo computacional, consistiendo detener el entrenamiento de no mejora la pérdida de la red después de 5 épocas. La metodología de optimización de hiperparámetros se puede ver en la Figura 2.

Por otro lado, también se decidió utilizar la misma semilla de generación de número aleatorios de las soluciones iniciales para comparar a los algoritmos bajo condiciones similares.

Una vez obtenidas las soluciones por cada metaheurística, se realizaron 31 experimentos de entrenamiento de cada solución para obtener evidencia estadística de cada solución.

Por último, para revisar el fitness de convergencia, realizamos gráficas de convergencia, así como comparar sus desviaciones estándar de la primera generación a la última iteración de cada metaheurística.

5. Resultados experimentales y discusiones

Los experimentos se realizaron en una computadora con procesador RYZEN 5 5600x, una tarjeta gráfica GPU-GTX-1050TI, y RAM de 16GB. Se utilizó el lenguaje Python, usando principalmente NumPy, matplotlib y Keras de TensorFlow. El tiempo de ejecución fue de entre 4 y 7 horas tanto para PSO, FPA y LNFE, para cada uno de los conjuntos de datos (Google y Nike).

Los hiperparámetros encontrados por cada una de las metaheuristicas se encuentra en el Cuadro 3. Se analizó la convergencia de cada uno de los algoritmos. En la Figura 3 se muestran algunas gráficas de convergencia ya que de agregar todas las imágenes el número de páginas sería mayor a lo requerido.

Adicionalmente en el Cuadro 4 se muestran las diferencias de las dispersión de la primera población y la última población de cada una de las variables, de cada algoritmos y de cada acción[5]. En el Cuadro 4 se aprecia que en PSO la convergencia en cada variable disminuyó a excepción de FPA, que incluso en algunos casos aumenta la dispersión. En apariencia PSO muestra mejor convergencia que LNFE, no obstante como se aprecia en la Figura 4 y 3 en 400 llamadas a función PSO llegó a la solución

[5] Las soluciones fueron escaladas en el intervalo $[0, 1]$ para poder hacer una comparación relativa.

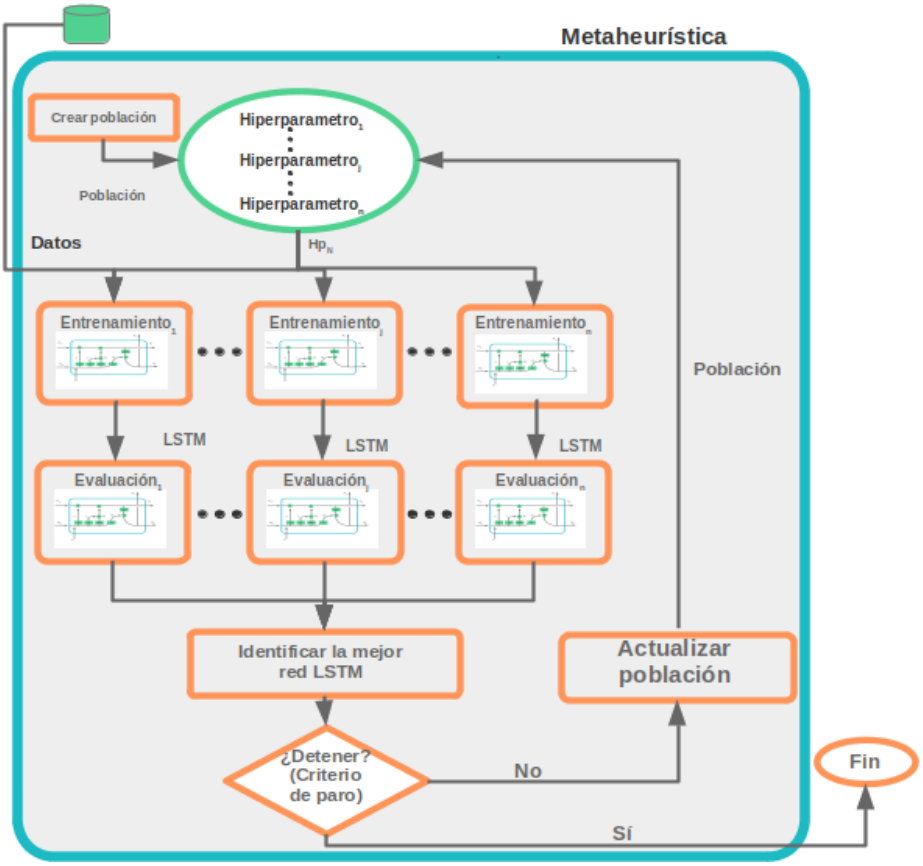

Fig. 2. Metodología de optimización. Hp_N representa el N hiperparámetro (Elaboración propia).

mostrada en el Cuadro 3, no obstante LNFE, llegó a la solución en una menor cantidad de iteraciones y con menor cantidad de llamadas a función.

En la última iteración LNFE tiene mayor dispersión a comparación de PSO, empero el algoritmo esta diseñado para renovar su población si se repite el fitness después de n iteraciones, por lo que explicaría una mayor dispersión que PSO en su población final (ver Cuadro 4).

Por otro lado, FPA no disminuyó en gran medida su dispersión en la población final, la cual se mantuvo en cada iteración como se muestra en la Figura 3 y el Cuadro 4, incluso en algunos casos aumentó su dispersión. En el Cuadro 3 se muestran las soluciones encontradas, así como los resultados de 31 experimentos de entrenamiento de cada una de las redes.

Como se aprecia en las soluciones encontradas por las metaheurísticas, la arquitectura de la red LSTM es similar en cada uno de los casos a excepción de LSTM-PSO-Nike[6]. Empero, cada solución es más simple que las redes LSTM-OSM-Google y

[6] Cuando nos referimos a LSTM-PSO-Nike significa, red LSTM optimizada con PSO para pronosticar las acciones de Nike.Del mismo modo para la combinación de las diferentes siglas.

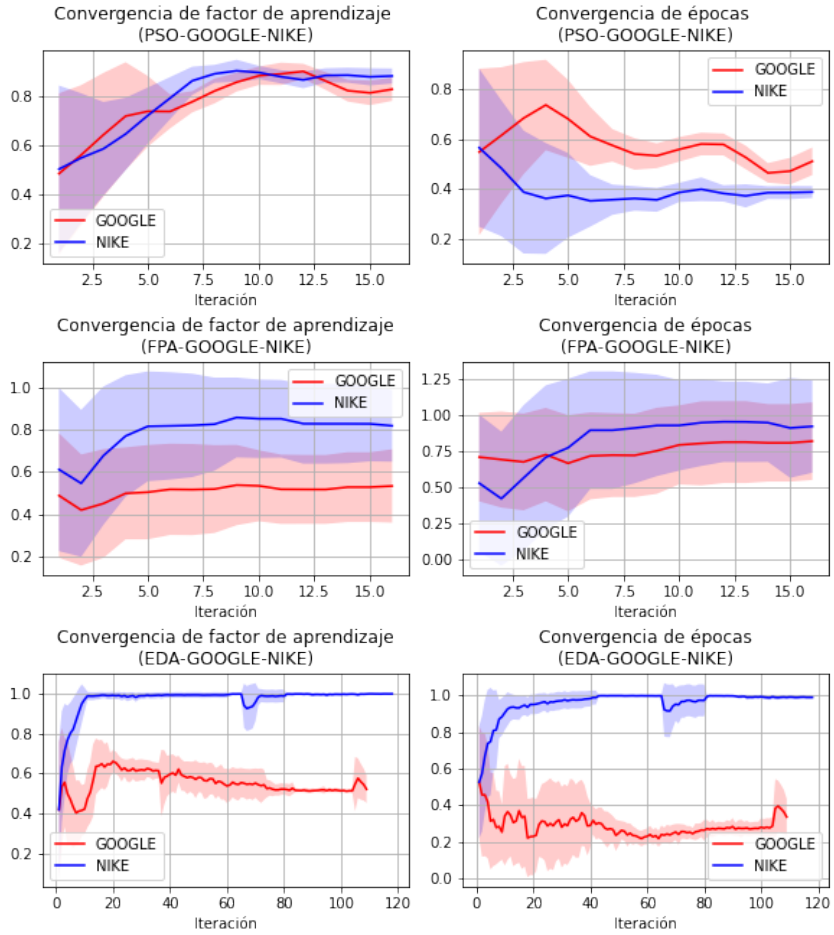

Fig. 3. Convergencia del factor de aprendizaje y número de épocas. Las gráficas están escaladas en el intervalo $[0, 1]$.

LSTM-OSM-Nike, inclusive la media de la pérdida es menor en todos los casos.

En el caso de LSTM-PSO-Google, se muestra que es el mejor fitness entre LSTM-FPA-Google, LSTM-LNFE-Google y LSTM-OSM-Google, no obstante la media de los 31 experimentos muestra que LSTM-LNFE-Google tiene mejor rendimiento, mientras que en el caso de LSTM-FPA-Nike y LSTM-LNFE-Nike muestra un mejor rendimiento que LSTM-PSO-Nike, con fitness similar como se aprecia en la Figura 6.

Empero LNFE, encontró la solución con una menor cantidad de llamadas a función como se ven Figura 3. Los resultados promedio de pronósticos de 31 experimentos de LNFE y OSM se muestra en la Figura 5. En dichas imágenes se aprecia la mejora sustancial de usar LNFE, lo cual sucede de igual manera con los demás pronósticos y se puede apreciar en el Cuadro 3.

No se presentan los demás experimentos para evitar tener un número excesivo de imágenes. Dichos pronósticos se logran con una red LSTM más simple que las redes

Tabla 3. LSTM optimizada para realizar el pronóstico de las acciones.

	Google				Nike			
hiperparámetro	**PSO**	**FPA**	**LNFE**	**OSM**	**PSO**	**FPA**	**LNFE**	**OSM**
Cl_s	1	1	1	4	1	1	1	4
H_s	150	150	150	96	110	150	150	96
W_s	60	80	40	50	90	100	40	50
α	10^{-4}	10^{-3}	10^{-4}	10^{-2}	10^{-4}	10^{-4}	10^{-4}	10^{-2}
B_s	16	32	8	16	16	16	8	16
E_s	50	150	100	100	50	150	150	100
Datos de solución	**PSO**	**FPA**	**LNFE**	**OSM**	**PSO**	**FPA**	**LNFE**	**OSM**
Llamadas a función:	400	400	401	-	400	400	402	-
Fitness:	3.18E-5	3.41E-5	3.32E-5	-	5.25E-6	4.23E-6	1.24E-7	-
Media de pérdida:	3.53E-5	2.91E-5	2.91E-5	1.86E-3	5.35E-6	3.96E-6	3.97E-6	1.02E-3
Media de prueba:	2.60E-4	3.28E-4	1.49E-4	2.22E-3	1.78E-4	1.04E-4	1.08E-4	2.97E-3
Desviación estándar de pérdida:	2.10E-6	1.58E-6	1.04E-6	7.80E-5	1.93E-7	9.13E-8	1.51E-7	5.61E-5
Desviación estándar de prueba:	1.29E-4	1.88E-4	6.47E-5	2.01E-4	6.01E-5	4.05E-5	2.38E-5	5.48E-4

Tabla 4. Diferencia de desviación estándar entre la primera y última generación de cada variable.

	PSO		FPA		LNFE	
hiperparámetro	**Google**	**Nike**	**Google**	**Nike**	**Google**	**Nike**
Cl_s	0.23	0.26	0.19	0.13	0.26	0.23
H_s	0.27	0.25	0.32	0.34	0.29	0.28
W_s	0.29	0.26	0.10	0.09	0.25	0.23
α	0.31	0.28	0.12	0.20	0.27	0.33
B_s	0.23	0.20	-0.03	-0.03	0.17	0.18
E_s	0.29	0.27	0.04	0.12	0.21	0.30

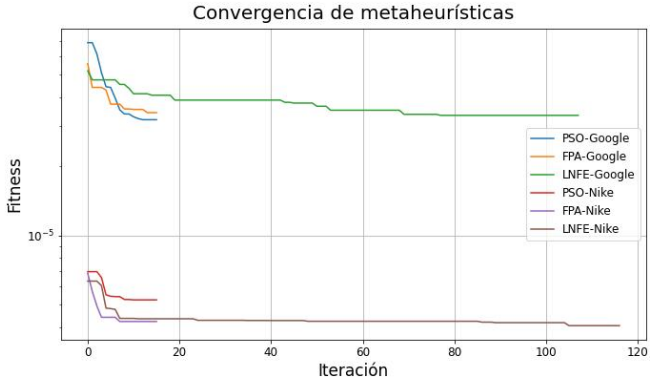

Fig. 4. Convergencia (fitness vs. Iteración) de la optimización de hiperparámetros de la red LSTM.

LSTM-OSM, similar a lo que refiere [1] quién asegura que encontrando las hiperparámetros adecuados se puede obtener la misma precisión a una red más compleja, e incluso se supera el rendimiento, como se aprecia en en este caso en el diagrama de caja y bigote de la Figura 6 y en la gráfica de entrenamiento de la Figura 7.

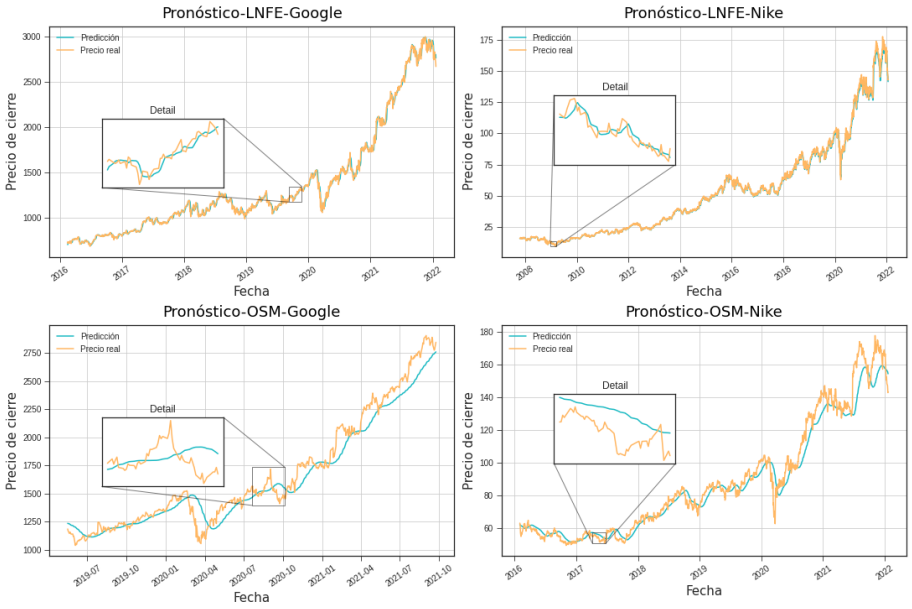

Fig. 5. Pronóstico promedio de 31 experimentos de LNFE y OSM.

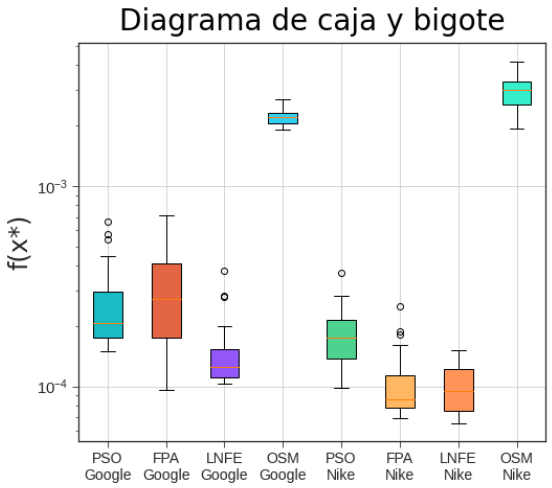

Fig. 6. Diagrama de caja y bigote de prueba de 31 experimentos.

6. Conclusiones

Las topologías que retornaron cada una de las metaheurísticas fueron similares, a excepción de PSO-Nike, sin embargo esto da indicios de que se puede usar una misma arquitectura tanto para Nike como Google, y usar con menor cantidad de Cl_s, usando más estados ocultos, y diferentes configuraciones de hiperparámetros; de la misma manera en el caso de α que se repite en todos a excepción de FPA-Google.

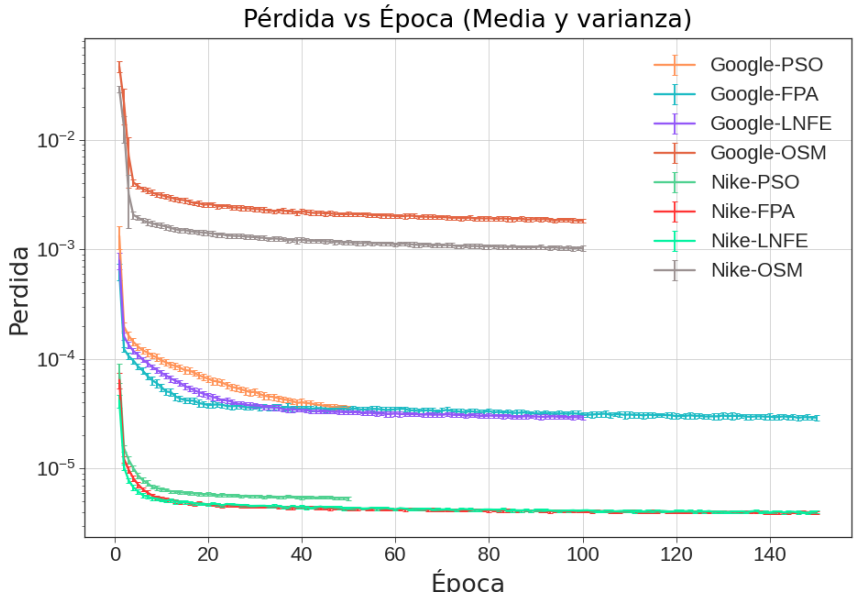

Fig. 7. Media y varianza de 31 experimentos de entrenamiento de la red optimizada con PSO, FPA, LNFE y OSM.

Mientras que LNFE retorna soluciones iguales tanto para Google como Nike, y convergencia claramente definida en una menor cantidad de llamadas a función logrando rendimientos similares a PSO-Google, lo cual sugiere continuar probando con más acciones y diferentes EDAs, por ejemplo, CMA-ES.

Por otro lado la convergencia de LNFE puede aumentar su dispersión en la población final, empero con soluciones consistentes, pues el mismo algoritmo guarda la mejor solución aunque se reinicie su población, por lo cual se puede correr el algoritmo con una menor cantidad de llamadas a función e iteraciones obteniendo resultados satisfactorios.

En referente a la metodología para optimizar la metaheurística, muestra resultados satisfactorios. Se sugiere probar la metodología fijando el número de E_s, y reduciendo el conjunto de entrenamiento, siguiendo un proceso similar de optimización Random Search, pero automatizado; de tal modo que se puede reducir el tiempo de búsqueda, y al final el resultado solo entrenarlo con un mayor número de E_s y con un conjunto de entrenamiento mayor.

Es menester puntualizar que mantener a los algoritmos trabajando en el espacio de los continuos y luego mapear cada solución a números enteros da resultados satisfactorios. Se sugiere experimentar con un número mayor de acciones, para saber si la arquitectura se repite, quizás en tales casos solo será necesario usar una LNFE, ya que implica menor costo computacional.

Por último, usar gráficas de convergencia permite un análisis más detallado del comportamiento de los algoritmos para saber que algoritmo usar para determinado caso, en el presente, pronósticos.

Agradecimientos. Este trabajo es parcialmente financiado por CONACYT de México a través: Beca de Posgrado No. 774627 (F. Pedroza) y Proyecto CÁTEDRAS-2598 (A. Rojas).

Referencias

1. Aufa, B. Z., Suyanto, S., Arifianto, A.: Hyperparameter setting of LSTM-based language model using grey wolf optimizer. In: International conference on data science and its Applications (ICoDSA). pp. 1–5. IEEE (2020) doi: 10.1109/ICoDSA50139.2020.9213031
2. Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. Advances in neural information processing systems, vol. 24 (2011)
3. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of machine learning research, vol. 13, no. 2 (2012)
4. Cavalcante, R. C., Brasileiro, R. C., Souza, V. L., Nobrega, J. P., Oliveira, A. L.: Computational intelligence and financial markets: A survey and future directions. Expert systems with applications, vol. 55, pp. 194–211 (2016) doi: https://doi.org/10.1016/j.eswa.2016.02.006
5. Chen, S., Wu, J., Chen, X.: Deep reinforcement learning with model-based acceleration for hyperparameter optimization. In: IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI). pp. 170–177. IEEE (2019) doi: 10.1109/ICTAI.2019.00032
6. Chourmouziadis, K., Chatzoglou, P. D.: An intelligent short term stock trading fuzzy system for assisting investors in portfolio management. Expert Systems with Applications, vol. 43, pp. 298–311 (2016) doi: 10.1016/j.eswa.2015.07.063
7. Evans, C., Pappas, K., Xhafa, F.: Utilizing artificial neural networks and genetic algorithms to build an algo-trading model for intra-day foreign exchange speculation. Mathematical and Computer Modelling, vol. 58, no. 5–6, pp. 1249–1266 (2013) doi: 10.1016/j.mcm.2013.02.002
8. Filomeno-Coelho, R., Xiao, M., Guglielmetti, A., Herrera, M., Zhang, W.: Investigation of three genotypes for mixed variable evolutionary optimization. In: Advances in evolutionary and deterministic methods for design, optimization and control in engineering and sciences, pp. 309–319. Springer, Cham (2015)
9. Gendreau, M., Potvin, J.-Y.: Metaheuristics in combinatorial optimization. Annals of Operations Research, vol. 140, no. 1 (2005) doi: 10.1007/s10479-005-3971-7
10. Goldberg, Y.: Neural network methods for natural language processing. Synthesis lectures on human language technologies, vol. 10, no. 1, pp. 1–309 (2017) doi: 10.2200/S00762ED1V01Y201703HLT037
11. Gorgolis, N., Hatzilygeroudis, I., Istenes, Z., Gyenne, L. G.: Hyperparameter optimization of LSTM network models through genetic algorithm. In: 10th International Conference on Information, Intelligence, Systems and Applications (IISA). pp. 1–4. IEEE (2019) doi: 10.1109/IISA.2019.8900675
12. Kumar, G., Jain, S., Singh, U. P.: Stock market forecasting using computational intelligence: A survey. Archives of Computational Methods in Engineering, vol. 28, no. 3, pp. 1069–1101 (2021) doi: doi.org/10.1007/s11831-020-09413-5

13. Kumar, K., Haider, M., Uddin, T.: Enhanced prediction of intra-day stock market using metaheuristic optimization on RNN–LSTM network. New Generation Computing, vol. 39, no. 1, pp. 231–272 (2021) doi: 10.1007/s00354-020-00104-0
14. Li, A. W., Bastos, G. S.: Stock market forecasting using deep learning and technical analysis: a systematic review. IEEE, vol. 8, pp. 185232–185242 (2020) doi: 10.1109/ACCESS.2020.3030226
15. Li, W., Ng, W. W., Wang, T., Pelillo, M., Kwong, S.: Help: An LSTM-based approach to hyperparameter exploration in neural network learning. Neurocomputing, vol. 442, pp. 161–172 (2021) doi: 10.1016/j.neucom.2020.12.133
16. Moghar, A., Hamiche, M.: Stock market prediction using LSTM recurrent neural network. Procedia Computer Science, vol. 170, pp. 1168–1173 (2020) doi: 10.1016/j.procs.2020.03.049
17. Mora, K. M. F., Marín, J. A., Cerda, J., Carrasco-Ochoa, J. A., Martínez-Trinidad, J. F., Olvera-López, J. A.: Pattern Recognition: 12th Mexican Conference, MCPR'20, vol. 12088. Springer Nature (2020)
18. Ozbayoglu, A. M., Gudelek, M. U., Sezer, O. B.: Deep learning for financial applications: A survey. Applied Soft Computing, vol. 93, pp. 106384 (2020) doi: 10.1016/j.asoc.2020.106384
19. Qu, Y., Zhao, X.: Application of LSTM neural network in forecasting foreign exchange price, vol. 1237, no. 4, pp. 042036 (2019) doi: 10.1088/1742-6596/1237/4/042036
20. Schmidhuber, J., Hochreiter, S.: Long short-term memory. Neural Comput, vol. 9, no. 8, pp. 1735–1780 (1997) doi: 10.1162/neco.1997.9.8.1735
21. Sezer, O. B., Gudelek, M. U., Ozbayoglu, A. M.: Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. Applied soft computing, vol. 90, pp. 106181 (2020) doi: 10.1016/j.asoc.2020.106181
22. Silva, T. R., Li, A. W., Pamplona, E. O.: Automated trading system for stock index using LSTM neural networks and risk management. In: International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2020) doi: 10.1109/IJCNN48605.2020.9207278
23. Valdez-Peña, S. I., Hernández-Aguirre, A., Botello-Rionda, S.: Approximating the search distribution to the selection distribution in EDAs. In: Proceedings of the 11th Annual conference on Genetic and evolutionary computation. pp. 461–468 (2009) doi: 10.1145/1569901.1569965
24. Wen, Y., Lin, P., Nie, X.: Research of stock price prediction based on PCA-LSTM model, vol. 790, no. 1, pp. 012109 (2020) doi: 10.1088/1757-899X/790/1/012109
25. Yang, X. S. (ed): Nature-inspired computation and swarm intelligence: Algorithms, theory and applications. Academic Press (2020)