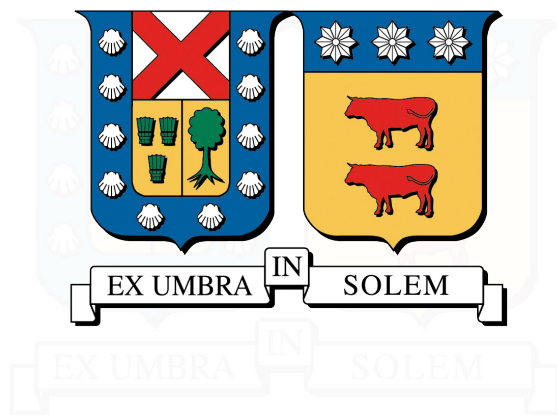


UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INDUSTRIAS
VALPARAÍSO - CHILE



**PREDICCIÓN DE SERIES FINANCIERAS DEL MERCADO
LATINOAMERICANO MEDIANTE REDES NEURONALES
ARTIFICIALES, A TRAVÉS DE UN ALGORITMO DE COLONIAS
DE ABEJAS.**

JULIO IGNACIO VALENZUELA PAVEZ

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL INDUSTRIAL

PROFESOR GUÍA : SR WERNER KRISTJANPOLLER R.
PROFESOR CORREFERENTE : SR. RODOLFO SALAZAR A.

SEPTIEMBRE 2017

Agradecimientos

A lo largo del trabajo y el esfuerzo que significa el paso por la universidad es imposible no sentir el apoyo de las personas que te han acompañado en las buenas, en las malas, en las frustraciones, en las noches de estudio, etc. Gracias a mis padres por todos estos años ayudándome desde levantarme hasta que repasaba la última frase para la prueba del próximo día, los desayunos, los almuerzos, las conversaciones, los consejos. A mi hermana María José, por acogerme durante casi dos años para no tener que viajar desde tan lejos a la universidad. A mi polola por escucharme y contenerme en los momentos difíciles que implica ser un sansano, gracias por hacerme salir del estrés y el agotamiento con todas esas pequeñas grandes cosas. A mis amigos por bancarme siempre, son lo mejor. Al profesor Werner Kristjanpoller por la disposición y la buena onda, así como las sugerencias y el apoyo.

Sin la ayuda de ustedes, así como de las personas que he tenido la fortuna de cruzarme a través de estos casi 6 años, no sería posible nada de esto. Gracias totales.

Resumen ejecutivo

Las redes neuronales han mostrado ser un método capaz de implementar tareas no triviales tales como procedimientos de reconocimiento de imagen, clasificación, ajuste de curvas y pronósticos. En este contexto una tarea crítica es la de la fase de entrenamiento de una red neuronal.

En esta memoria se prueba la eficacia y el desempeño del algoritmo de colonias de abejas (*Artificial Bee Colony Algorithm*) como algoritmo de entrenamiento en una red neuronal para modelar y predecir series financieras correspondientes a los índices del mercado latinoamericano.

Los resultados muestran una coherencia entre el desempeño del algoritmo tanto en la fase de entrenamiento como en el pronóstico, corroborando que aquel método que posee un mejor modelo es más probable que se desempeñe mejor en el pronóstico.

Para finalizar se combina el algoritmo de backpropagation con el algoritmo de colonias de abejas para hacer un híbrido ABC-BP, el cual toma como punto de partida la solución entregada por backpropagation y mejora el modelado mediante el algoritmo ABC. Los experimentos revelan que en tres de las cuatro series en las que fueron testeadas se logró una mejora en el pronóstico respecto al algoritmo clásico de backpropagation.

Índice de Contenidos

1. Problema de investigación	1
1.1. Objetivo General	3
1.2. Objetivos Específicos	3
2. Marco teórico	4
2.1. Métodos de predicción	4
2.1.1. La regresión lineal	5
2.1.2. Econometría de series de tiempo	7
2.1.2.1. Estacionariedad	8
2.1.2.2. Métodos para trabajar series de tiempo	9
2.2. Pronóstico de variables financieras	11
2.2.1. Precios de acción	11
2.2.2. Tipos de cambio	12
2.2.3. Tasas de interés	13
2.3. Redes neuronales	14
3. Metodología	16
3.1. Redes neuronales	16
3.1.1. Introducción	16
3.1.2. Red de neuronas biológicas	16
3.1.3. Neurona artificial	18
3.1.3.1. Funciones de transferencia	20
3.1.4. Perceptrón monocapa	24
3.1.5. Perceptrón multicapa	24
3.1.6. Entrenamiento	26
3.1.6.1. Aprendizaje supervisado	26
3.1.6.2. Aprendizaje no supervisado	27
3.1.6.3. Backpropagation	27
3.2. Algoritmo de colonias de abejas	30
3.2.1. Inteligencia de enjambre	30
3.2.2. Comportamiento de enjambres de abejas	33
3.2.3. Algoritmo ABC para entrenar redes neuronales artificiales	39
4. Datos	41

4.1. Estadística descriptiva y gráficos	41
5. Procedimiento	46
5.1. Ventana móvil	46
5.2. Estructura de la ANN	48
5.2.1. Rentabilidad y normalización	48
5.2.2. Elección de las variables de entrada	49
5.2.3. Elección de la arquitectura	50
5.3. Parámetros algoritmo ABC	51
5.4. Entrenamiento ABC-BP	52
6. Resultados	54
6.1. IPSA	54
6.1.1. Predicción	54
6.1.2. Comparación backpropagation	55
6.1.2.1. Error cuadrático medio	56
6.2. Merval	57
6.2.1. Predicción	57
6.2.2. Comparación backpropagation	59
6.2.2.1. Error cuadrático medio	59
6.3. Colcap	60
6.3.1. Predicción	60
6.3.2. Comparación backpropagation	62
6.3.2.1. Error cuadrático medio	62
6.4. Ibovespa	63
6.4.1. Predicción	63
6.4.2. Comparación backpropagation	64
6.4.2.1. Error cuadrático medio	65
6.5. Inca	65
6.5.1. Predicción	65
6.5.2. Comparación backpropagation	67
6.5.2.1. Error cuadrático medio	67
6.6. IPC	68
6.6.1. Predicción	68
6.6.2. Comparación backpropagation	69
6.6.2.1. Error cuadrático medio	70
6.7. Resultados combinación ABC-BP	71
6.7.1. Inca	71
6.7.2. IPC	72
6.7.3. Colcap	73
6.7.4. Ibovespa	74
7. Conclusiones y recomendaciones	76
Referencias	81

Índice de Tablas

3.1. Tabla resumen de cómo actúa una neurona artificial en ciertos casos. . . .	23
4.1. Estadística descriptiva de las series consideradas.	42
5.1. Tabla resumen de las configuraciones de la ANN.	50
5.2. Parámetros del algoritmo ABC considerados.	51
5.3. Parámetros del algoritmo backpropagation.	51
6.1. Tabla resumen de los errores cuadráticos medios <i>out-sample</i>	70
6.2. Tabla resumen de los errores cuadráticos medios <i>out-sample</i> del algoritmo combinado ABC-BP	75

Índice de Figuras

2.1. Métodos de pronóstico	5
3.1. Esquema básico de una neurona biológica	17
3.2. Neurona de McCulloch y Pitts	19
3.3. Gráfica de la función escalón unitario	20
3.4. Gráfica de la función lineal	21
3.5. Gráfica de la función sigmoidea	21
3.6. Gráfica de la función tangente hiperbólica	22
3.7. Gráfica de la función gaussiana	22
3.8. Esquema de un perceptrón multicapa con una capa oculta y una neurona de salida.	26
3.9. <i>Six-Hump Camel Function</i> . Ejemplo de una función con múltiples óptimos locales.	31
3.10. <i>EggHolder Function</i> . Ejemplo de una función con múltiples óptimos locales.	31
3.11. Diagrama de flujo del funcionamiento del algoritmo ABC.	37
4.1. Gráfica de la serie IPSA de Chile	42
4.2. Gráfica de la serie Merval de Argentina	43
4.3. Gráfica de la serie INCA de Perú	43
4.4. Gráfica de la serie COLCAP de Colombia	44
4.5. Gráfica de la serie IPC de México	44
4.6. Gráfica de la serie IBOVESPA de Brasil	45
5.1. Ejemplo de modelado y pronóstico tradicional.	47
5.2. Ejemplo de ventana móvil.	47
5.3. Diagrama de flujo del funcionamiento del algoritmo ABC inicializado con backpropagation.	53
6.1. Predicción por ventana corrediza de la red neuronal usando el algoritmo ABC para la serie IPSA.	54
6.2. Predicción de la red neuronal con ABC e IPSA real superpuestos.	55
6.3. Predicción de la red neuronal con backpropagation e IPSA real superpuestos.	56
6.4. Error cuadrático medio del algoritmo ABC y backpropagation en la fase de entrenamiento a lo largo de la ventana corrediza.	57

6.5. Predicción por ventana corrediza de la red neuronal usando el algoritmo ABC para la serie Merval.	58
6.6. Predicción de la red neuronal con ABC y Merval real superpuestos. . .	58
6.7. Predicción de la red neuronal con backpropagation y Merval real superpuestos.	59
6.8. Error cuadrático medio del algoritmo ABC y backpropagation en la fase de entrenamiento a lo largo de la ventana corrediza.	60
6.9. Predicción por ventana corrediza de la red neuronal usando el algoritmo ABC para la serie Colcap.	61
6.10. Predicción de la red neuronal con ABC y Colcap real superpuestos. . .	61
6.11. Predicción de la red neuronal con backpropagation y Colcap real superpuestos.	62
6.12. Error cuadrático medio del algoritmo ABC y backpropagation en la fase de entrenamiento a lo largo de la ventana corrediza.	62
6.13. Predicción por ventana corrediza de la red neuronal usando el algoritmo ABC para la serie Ibovespa.	63
6.14. Predicción de la red neuronal con ABC y Ibovespa real superpuestos. .	64
6.15. Predicción de la red neuronal con backpropagation y Ibovespa real superpuestos.	64
6.16. Error cuadrático medio del algoritmo ABC y backpropagation en la fase de entrenamiento a lo largo de la ventana corrediza.	65
6.17. Predicción por ventana corrediza de la red neuronal usando el algoritmo ABC para la serie Inca.	66
6.18. Predicción de la red neuronal con ABC y Inca real superpuestos. . . .	66
6.19. Predicción de la red neuronal con backpropagation e Inca real superpuestos.	67
6.20. Error cuadrático medio del algoritmo ABC y backpropagation en la fase de entrenamiento a lo largo de la ventana corrediza.	68
6.21. Predicción por ventana corrediza de la red neuronal usando el algoritmo ABC para la serie IPC.	68
6.22. Predicción de la red neuronal con ABC y IPC real superpuestos.	69
6.23. Predicción de la red neuronal con backpropagation e IPC real superpuestos. .	69
6.24. Error cuadrático medio del algoritmo ABC y backpropagation en la fase de entrenamiento a lo largo de la ventana corrediza.	70
6.25. Valores de la predicción ABC-BP para la serie Inca	71
6.26. Error cuadrático medio del algoritmo ABC-BP y backpropagation en la fase de entrenamiento a lo largo de la ventana corrediza.	72
6.27. Valores de la predicción ABC-BP para la serie IPC	72
6.28. Error cuadrático medio del algoritmo ABC-BP y backpropagation en la fase de entrenamiento a lo largo de la ventana corrediza.	73
6.29. Valores de la predicción ABC-BP para la serie Colcap	73
6.30. Error cuadrático medio del algoritmo ABC-BP y backpropagation en la fase de entrenamiento a lo largo de la ventana corrediza.	74
6.31. Valores de la predicción ABC-BP para la serie Ibovespa	74

6.32. Error cuadrático medio del algoritmo ABC-BP y backpropagation en la fase de entrenamiento a lo largo de la ventana corrediza.	75
7.1. Nuevo pronóstico INCA con cambio en los parámetros	78
7.2. Nuevo pronóstico IPC con cambio en los parámetros	78



1 | Problema de investigación

A lo largo de la historia el ser humano ha mostrado interés por anteponerse a situaciones futuras, tratar de predecir resultados para así tomar mejores decisiones hoy y estar preparados para el mañana. Bajo este concepto y con la ayuda de los métodos matemáticos nace la econometría para cuantificar las predicciones y la incertidumbre a través de criterios causales multivariados con apoyo en la estocasticidad ([Gujarati y Porter, 2010](#)).

Numerosos han sido los estudios econométricos y las variables a predecir buscando la causalidad de una variable con respecto a otra, por ejemplo, variables de tipo financieras (precio de acción, tasas de interés, tasas de cambio, inflación, etc) cuyo análisis puede favorecer a una decisión más informada.

El estudio de este tipo de variables tiene una implicancia y utilidad que es observable intuitivamente, inversionistas pueden invertir su dinero de una manera más segura y de cierta manera disminuir su riesgo sabiendo si, por ejemplo el precio de acción de una cierta empresa tiene una tendencia al alza, o a la baja. También una empresa podría tomar mejores resoluciones si supiera, con una cierta confianza, entre qué valores es

posible esté la inflación en los próximos periodos, o quizás las tasas de interés, este último ejemplo es extrapolable a cualquier persona natural, por lo que el horizonte es expandible fácilmente.

Nuevas ciencias se han sumado al estudio y es aquí donde conecta la inteligencia artificial, que busca “una teoría computable del conocimiento humano. Es decir, una teoría en la que sus modelos formales puedan ejecutarse en un sistema de cálculo y tener el mismo carácter predictivo que tienen, por ejemplo, las ecuaciones de Maxwell en el electromagnetismo.” (Méndez, J. y Marín, R., 2013). Dentro de la inteligencia artificial es posible usar herramientas que permitan “aprender” el comportamiento de experiencias pasadas y así establecer un modelo que se aproxime a la realidad, como lo son las *Redes Neuronales Artificiales* estudiadas por McCulloch y Pitts (1943), inspirado en el modo de procesamiento de información de la manera que lo hace un cerebro.

El uso de *Redes Neuronales Artificiales* para la predicción de variables de tipo financiera sigue siendo un estudio bastante amplio y existen investigaciones como por ejemplo las de Santana (2006) para la inflación colombiana y de Villada et al. (2012) para predicción de precios de acción que dan cuenta de la eficiencia de estos métodos, pero tiene ciertas complicaciones, por ejemplo, ¿Cómo estas neuronas “aprenden” y ejecutan su resultado? Para resolver esta duda en particular existen numerosos algoritmos, entre los cuales Karaboga (2005) propone un sistema de “colonias de abejas” (*Artificial Bee Colony Algorithm*), el que, al ser bastante nuevo, se hace interesante de estudiar y probar para el pronóstico de este tipo de variables.

Es necesario entonces, tener especial cuidado en el tratamiento de estas redes, por-

que un mal diseño de la red puede llevar a resultados insatisfactorios (Ruiz y Jiménez, 2000).

1.1. Objetivo General

Predecir series financieras correspondientes al mercado latinoamericano mediante redes neuronales artificiales entrenadas con el algoritmo de colonias de abejas para compararlo con algoritmos tradicionales.

1.2. Objetivos Específicos

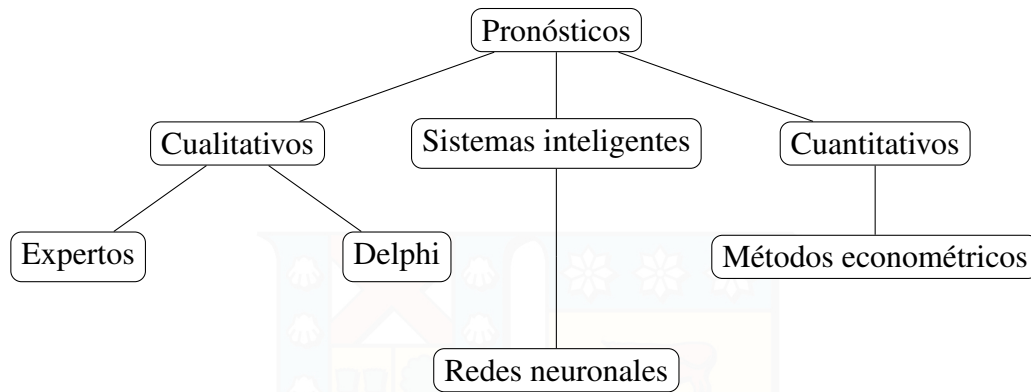
- Estructurar la arquitectura de las redes neuronales consideradas en los experimentos.
- Lograr entrenar una red neuronal mediante el algoritmo de optimización global de colonias de abejas.
- Predecir valores fuera de la muestra de las series consideradas.
- Comparar el desempeño de una red entrenada con el algoritmo de colonias de abejas con respecto a una red entrenada con los algoritmos tradicionales.
- Combinar ambos algoritmos y ver si existen mejoras en el desempeño.

2 | Marco teórico

2.1. Métodos de predicción

Al tomar decisiones en el contexto financiero existen una serie de preocupaciones que pueden complicar o limitar la operación, uno de los más importantes es el comportamiento futuro del mercado. Dentro de este problema, si se tienen datos históricos de cómo se ha comportado anteriormente la variable estudiada, es posible mediante un modelo matemático representar el proceso para generar pronósticos. Sin embargo las características atípicas de algunos fenómenos económicos impiden que la predicción se ajuste al comportamiento real del sistema, por lo que siempre habrán discrepancias.

Para pronosticar existen distintas técnicas y modelos tanto cualitativos como cuantitativos. En el caso de los métodos cualitativos el pronóstico es muchas veces resultado de un análisis realizado por un panel de expertos a través de la intuición, el juicio personal y la experiencia. De igual forma existen estudios de carácter financiero aplicando este tipo de metodologías así como el método Delphi ([Mallo et al., 2003](#)). Lo antes dicho se resume en la Figura 2.1.

**Figura 2.1:** Métodos de pronóstico

(Fuente: Elaboración propia.)

A continuación se describen algunos métodos de pronóstico cuantitativo estudiados principalmente por la probabilidad y estadística y la econometría.

2.1.1. La regresión lineal

En su forma más simple, la regresión lineal ha sido usada en múltiples áreas con el fin de buscar la explicación o relación de una variable con respecto a otra. Por ejemplo tomando como X a la variable *explicativa* y como Y a la variable *explicada* es posible generalizar la relación lineal determinística de la siguiente manera:

$$Y = \beta_0 + \beta_1 X \quad (2.1)$$

Esta expresión determina cada par (X, Y) para los cuales 2.1 describe una línea recta con pendiente β_1 e intercepto en Y de β_0 , cuidando siempre que se está hablando de variables determininísticamente relacionadas.

Para relaciones no determinísticas, entonces el valor de Y pasa a ser *aleatorio* y tiene

el nombre de *variable aleatoria*, por lo que encontrar los parámetros β_0, β_1 *poblacionales* que describan de manera perfecta el comportamiento de Y para un X dado es, con exactitud, imposible (Devore, 2008).

De esta manera la variable *explicada* Y se vuelve una *estimación* de parámetros también *estimados*, de la siguiente forma:

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X + \epsilon \quad (2.2)$$

Donde “^” simboliza que los parámetros son *estimaciones* y ϵ representa un error dado que el modelo se aleja de realizaciones de la variable Y .

Naturalmente la estimación o predicción de una variable Y puede ser extendida para n variables explicativas, de la forma 2.3 y/o 2.4:

$$\hat{Y} = \hat{\beta}_0 + \sum_{i=1}^n \hat{\beta}_i X_i + \epsilon \quad (2.3)$$

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + \hat{\beta}_3 X_3 + \dots + \hat{\beta}_n X_n + \epsilon \quad (2.4)$$

Lo importante eso sí, es que los parámetros *estimados* cumplan con ciertas condiciones que hagan que sean los mejores estimadores lineales insesgados (MELI o BLUE, por su traducción al inglés). Según el Teorema de Gauss Markov $\hat{\beta}$ es el mejor estimador lineal insesgado de β si se cumple (Gujarati y Porter, 2010):

- Es **Lineal** es decir, función lineal de una variable aleatoria, como la variable depen-

diente Y en el modelo de regresión.

- Es **insesgado**, es decir, su valor promedio o esperado, $E(\hat{\beta})$, es igual al valor verdadero, β .
- Tiene varianza mínima dentro de la clase de todos los estimadores lineales insesgados; un estimador insesgado con varianza mínima se conoce como **estimador eficiente**.

2.1.2. Econometría de series de tiempo

Una serie de tiempo $x_1, x_2, \dots, x_t, x_{t+1}, \dots, x_n$ es una secuencia de datos numéricos de una variable que principalmente está indexada por una dimensión temporal, es decir, asociado a un instante específico del tiempo como pueden ser: precios de acción, tasas de interés, precios de commodities, tipos de cambio, inflación, etc.

Para el análisis de datos financieros, el uso de la metodología de series temporales es particularmente útil debido a esta dependencia y los estudios actuales se basan en estos análisis.

Las metodologías existentes actualmente permiten extraer información, interpretar y explicar de manera de obtener información de la serie en momentos no observados a manera de pronóstico futuro, como en el presente estudio. La descomposición clásica de series de tiempo asume que la secuencia de datos incluye un efecto de tendencia (T_t), uno de estacionalidad (S_t), uno de variación cíclica (C_t) y variación errática o aleatoria (E_t).

$$x_t = T_t C_t S_t E_t \quad (2.5)$$

La ecuación 2.5 hace referencia a un método multiplicativo clásico de series de tiempo, sin embargo existe también un método aditivo.

2.1.2.1. Estacionariedad

Un proceso estocástico se dice estacionario si sus propiedades no dependen de la ventana de observación de los datos. Por lo tanto, los parámetros asociados a la media y a la varianza no deberían cambiar a lo largo del tiempo.

La estacionariedad puede presentarse de dos formas: *estrictamente estacionario* y *débilmente estacionario*. La estacionariedad *estricta* o fuerte al ser muy rigurosa es poco comprobable, indica que todos los momentos de un proceso estocástico Y_t son independientes del tiempo. En cambio en la estacionariedad *débil* se dice que los dos primeros momentos de un proceso estocástico Y_t son independientes del tiempo, esto es, su media y su varianza. Por lo tanto:

$$\mu_t = \mu = \text{constante} \quad (2.6)$$

$$\sigma_t^2 = \sigma = \text{constante} \quad (2.7)$$

Si una serie temporal es estacionaria (en media y en varianza) encontrar un modelo que explique su autocorrelación es mucho más fácil. Es por esto que muchas veces se trabajan series de precios mediante su rentabilidad (ver ecuación 5.1).

2.1.2.2. Métodos para trabajar series de tiempo

Suavizamiento exponencial

Uno de los más utilizados sobre todo para el pronóstico de demandas futuras en gestión de operaciones es el de, donde básicamente se pronostica con el período anterior sumado a una penalización por el error del pronóstico de la realización anterior (suavizamiento simple).

$$\hat{x}_t = \hat{x}_{t-1} + \alpha(x_{t-1} - \hat{x}_{t-1}) \quad (2.8)$$

También a este método se han incorporado variantes que permiten incluir tendencia y estacionalidad como es el caso del suavizamiento exponencial doble y Holt-Winters (Kalekar, 2004).

Promedios móviles

El cada punto de una media móvil de una serie temporal es la media aritmética de un número de puntos consecutivos de la serie, donde el número de puntos es elegido de tal manera que los efectos estacionales y / o irregulares sean eliminados.

$$\hat{x}_t = \frac{\sum_{i=1}^n x_{t-i}}{n} \quad (2.9)$$

Familia de modelos GARCH

Dentro de los supuesto del modelo de regresión lineal se encuentra el hecho de que

tiene que ser homocedástico, esto es, varianza constante a lo largo de los datos considerados, sin embargo en una serie de tiempo de característica financiera es altamente probable que esto no sea así, teniendo heterocedasticidad. Engle (1982) introduce los modelos ARCH (autoregressive conditional heteroskedasticity), en los cuales la varianza del error actual es función de los errores de períodos pasados. Posteriormente, Bollerslev (1986), generaliza estos modelos proponiendo el modelo GARCH en el cual la varianza condicional depende no sólo de las varianzas de períodos pasados si no que también depende de las perturbaciones cuadradas de períodos anteriores.

Estos modelos se ocupan para predecir volatilidad en series financieras, a diferencia de los dos anteriormente nombrados que predicen un valor futuro de la serie propiamente tal.

ARIMA

Los modelos autorregresivos integrados de media móvil (ARIMA) es un modelo estadístico que se utiliza para predecir valores futuros de una variable basado en datos del pasado. En un modelo $ARIMA(p,d,q)$ se tienen p términos autorregresivos y q términos de media móvil que en su diferencia d es estacionario.

Además de los antes mencionados, existen más metodologías que permiten el tratamiento de estas series, como lo es

- Procesos estocásticos.
- Modelos VAR.
- Redes neuronales.

2.2. Pronóstico de variables financieras

En el siguiente apartado se revisarán anteriores estudios del último tiempo referido a pronósticos de series financieras tales como precio de acción, tipo de cambio y tasas de interés.

2.2.1. Precios de acción

El pronóstico de precios de acción de un determinado mercado es una interrogante constante para los inversionistas, esto ya que si como tal se pudiera saber con anterioridad qué precio habrá en un futuro, entonces sería posible obtener retornos con toda certeza. Sin embargo la hipótesis de eficiencia débil de los mercados dice que no es posible obtener precios futuros a partir de precios históricos y que por lo tanto, los retornos siguen una caminata aleatoria (Random Walk).

A pesar de lo que dice la teoría, existen estudios como los de [Maya et al. \(2005\)](#) en los cuales se rechaza la hipótesis de caminata aleatoria y por lo tanto sí vale la pena pronosticar precios de acción tanto en mercados emergentes como desarrollados.

Numerosos han sido los trabajos en esta área, por ejemplo un estudio que viene al caso es el de [Mostafa \(2010\)](#) que predice movimientos de la bolsa de valores aplicado en Kuwait mediante redes neuronales para el periodo 2001-2003 y concluye que en mercados emergentes como el de Kuwait las redes neuronales y su modelo de predicción son posibles de aplicar con buenos resultados, además de agregar que los modelos de redes neuronales

se esperan que sobrepasen a las técnicas tradicionales tales como ARIMA.

Mientras [Javedani et al. \(2016\)](#) propone modelos para series de tiempo difusas, [Lahmiri \(2016\)](#) propone un modelo híbrido para predecir precios de un set de 6 acciones “intraday”(en el mismo día) usando métodos de BNN (backpropagation neuronal network) con VMD (variational mode decomposition) probando que se puede aplicar a una serie de tiempo con tanto ruido y no estacionaria como la tratada. También los resultados muestran el uso del modelo híbrido induce menores errores en términos de desviación media absoluta que un método no híbrido.

Otro estudio relacionado a precios de acción, es el de [Kristanpoller y Liberona \(2010\)](#) que hace referencia a los retornos esperados del mercado accionario chileno, comparando mediante modelos CAPM, Fama y French y Reward Beta, concluyendo que el mejor modelo para predecir los retornos del mercado chileno es el de tres factores de Fama y French.

2.2.2. Tipos de cambio

En cuanto al pronóstico de tipos de cambio los estudios de [Carriero et al. \(2009\)](#) proponen un vector auto regresivo (VAR) para el pronóstico de 33 tipos de cambio el cual concluyen que produce mejores resultados que un modelo AR (Random Walk), cabe destacar que entre los tipos de cambio analizados se encuentra la equivalencia [CLP/USD] así como otras monedas latinoamericanas comparadas con la moneda estadounidense.

En esta misma línea [Panda y Narasimhan \(2007\)](#) afirma que la predicción de un tipo

de cambio se da mejor con redes neuronales artificiales. Su estudio se basa en el tipo de cambio indio versus la moneda estadounidense, el cual se analiza desde la perspectiva de las redes de neuronas, modelos autorregresivos lineales y de caminata aleatoria AR, concluyendo que las redes neuronales tienen un mejor desempeño. También es importante mencionar que hace una acotación acerca de que su estudio va en contra de la hipótesis de eficiencia de mercado y que existe **siempre** la posibilidad de extraer información escondida en el tipo de cambio y que permitiría predecirla a futuro.

Otro enfoque tiene que ver con la volatilidad, que es la razón principal de inestabilidad en la economía global afectando las decisiones de apalancamiento corporativo y variables macroeconómicas (Como se cita en [Sanjeev y Sachin, 2016](#)). En este mismo estudio se analizan las monedas británicas e indias midiendo la volatilidad mediante modelos GARCH y luego proyectándolas a futuro mediante redes neuronales.

2.2.3. Tasas de interés

Aquí es interesante mencionar las investigaciones de [Diaz et al. \(2016\)](#) que mencionan una metodología para modelar y pronosticar tasas de interés, teniendo en consideración numerosas variables de tipo económicas, tales como el IMACEC, IPSA, precio del cobre, tipo de cambio, precio del petróleo, tasas de interés cupón cero a 5 años, etc. Sin embargo una de las limitaciones que se presentan dentro de su metodología es que un pequeño error, arrastra resultados sesgados hacia los pasos posteriores.

Un planteamiento distinto es el de [Ju et al. \(1997\)](#) que para datos de series temporales en Korea afirma que las tasas de interés son volátiles y además no lineales, por lo que

exponen un sistema capaz de procesar estos datos a través de lógica difusa y algoritmos genéticos combinados (*Genetic-based fuzzy model*) para el mercado financiero coreano mostrando que es capaz de seguir la volatilidad encontrando buenos pronósticos. Además concluyen que puede ser una buena alternativa a los modelos clásicos y a las ANN.

Como se pudo ver en los apartados anteriores, la utilización de modelos económicos para la predicción de variables financieras se está viendo cada vez más opacada por el uso de redes neuronales con distintas metodologías y algoritmos de tratamiento, debido a que la tendencia de este tipo de trabajos ha ido al alza. Hoy en día se presentan computadores con una capacidad de procesamiento mayor a la que había 10 años atrás y en ese sentido, el tratamiento de la información actualmente es mucho más expedita.

2.3. Redes neuronales

Numerosos son los estudios que abarcan series de tiempo abordadas con redes neuronales, sin embargo esta sección se enfocará en aquellas que usan el algoritmo de colonias de abejas.

El uso del algoritmo de colonias de abejas en redes neuronales es una investigación constante. Variados autores han tratado de mejorar este método, reconociendo sus falencias y tratando de mejorarlas ([Karaboga et al., 2014](#)). En lo que respecta al pronóstico de series de tiempo es importante mencionar el trabajo de [Shah et al. \(2011\)](#) quien para una serie de tiempo de datos acerca de terremotos ajusta una red entrenada mediante el algoritmo de colonias de abejas prediciendo así el valor de testeo de la muestra. Lo importante es que

en este estudio se predice en la fase de testeo que resulta ser *in-sample* es decir dentro de la muestra y no se hace un seguimiento del modelo para ver cómo se comporta con datos nunca antes vistos.

Algo diferente pasa con [Akpinar et al. \(2017\)](#), que para predecir la demanda de gas aplica la técnica de la ventana corrediza, moviendo el modelo y prediciendo un día por delante de la información disponible. En total se prueban 7 días y concluyen que el pronóstico de la red entrenada con el algoritmo ABC es mejor que el pronóstico de la red convencional entrenada por backpropagation. Aunque este estudio es importante por la técnica de la ventana corrediza, la cantidad de predicciones sigue siendo muy baja, sería distinto si pudiéramos testear un año completo de datos fuera de la muestra mediante la técnica de la ventana corrediza.

Por último un estudio más sofisticado de [Hsieh et al. \(2011\)](#) propone una red neuronal recurrente entrenada con el algoritmo de colonias de abejas para distintas series como el DJIA y el Nikkei-225 concluyendo que esta red y el algoritmo de entrenamiento se comportan de mejor forma que los estándares. Sin embargo no se especifica si la fase de testeo es *in-sample* o *out-sample* por lo que si bien representa una cita importante en términos de la arquitectura y modelamiento, en términos del pronóstico no lo es tanto.

En esta memoria se testearán diversas series de índices del mercado latinoamericano para comprobar cuál es la capacidad de pronóstico de una red neuronal entrenada mediante el algoritmo ABC cuando tiene datos que nunca había visto (*out-sample*).

3 | Metodología

3.1. Redes neuronales

3.1.1. Introducción

Una *Red Neuronal Artificial* (RNA o ANN por sus siglas en inglés) es un sistema computacional-matemático basado en el comportamiento biológico de las neuronas del sistema nervioso de nuestro cerebro. Está basado en una unidad fundamental llamada *neurona artificial* que interactúa con otras de su misma especie con el fin de transmitir información dada una entrada para producir un resultado de salida. Al ser células tan importantes para nuestro funcionamiento, están presentes en gran cantidad y ayudan a nuestra toma de decisiones del día a día.

3.1.2. Red de neuronas biológicas

Nuestro cerebro (y en realidad todo nuestro sistema nervioso) se compone de millones de células elementales llamadas neuronas, las cuales están conectadas entre sí mediante

estructuras llamadas *axones*, posee un cuerpo principal llamado *soma* el cual se ramifica formando las llamas *dendritas*.

Son células que poseen alta capacidad de transmitir información a través de la conocida *sinapsis*, que básicamente se origina a través de un impulso eléctrico dado un estímulo. Estas células al estar en gran cantidad en nuestro sistema nervioso forman verdaderas *redes neuronales* las cuales se comunican entre sí para que tomemos las decisiones que tomamos día a día.

Otro concepto importante es la capacidad de aprendizaje, nuestro sistema nervioso está continuamente aprendiendo y entrenándose ante nuestras acciones y reacciones. Por ejemplo: ¿cómo sabemos que no debemos cruzar la calle con luz roja? , en algún momento de nuestras vidas alguien (quizás nuestros padres) nos dijeron que no correspondía a una acción correcta el cruzar la calle con la luz del semáforo en rojo, nuestro cerebro almacenó esa información y hasta el día de hoy aplicamos ese aprendizaje a través de una experiencia pasada para que no resulte en un error (y probablemente en un atropello).

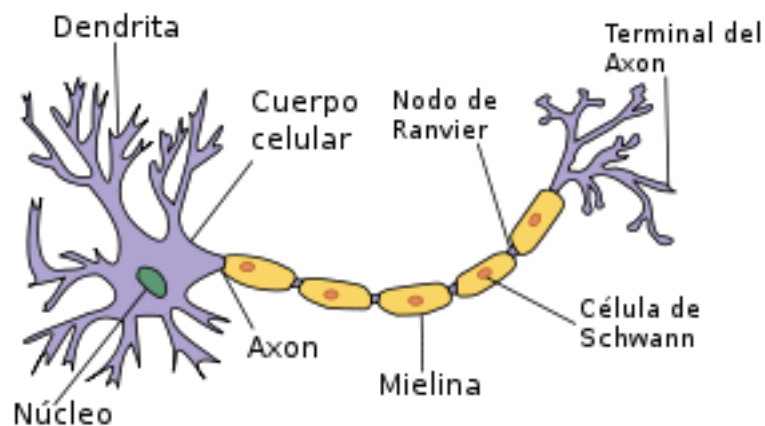


Figura 3.1: Esquema básico de una neurona biológica

(Fuente: <https://www.definicionabc.com>)

3.1.3. Neurona artificial

En el año 1943 Warren S. McCulloch, un neurocientífico en conjunto con Walter Pitts publicaron "Un cálculo lógico de las ideas inmanentes en la actividad nerviosa" (*A logical calculus of the immanent ideas in nervous activity*), una publicación en la cual trataron de entender cómo es que el cerebro humano es capaz de reconocer patrones altamente complejos usando muchas células básicas que están conectadas entre sí. Estas células son las neuronas descritas anteriormente.

En su publicación, [McCulloch y Pitts \(1943\)](#) entregan un modelo muy simplificado de lo que sería una neurona, a la cual llamaron *neurona artificial* también conocida hoy en día como neurona de McCulloch y Pitts.

La neurona propuesta por estos autores tenía como función manipular información de tipo binaria, para lo cual recibía estímulos o entradas y en base a la ponderación de esas entradas la neurona era capaz de responder y llevar a cabo una respuesta. Supongamos que se tiene una neurona k -ésima de una red de neuronas en la figura 3.2. En esta neurona se distinguen m señales de entrada con sus respectivas ponderaciones w_{ki} , $i = 1, 2, \dots, m$ llamadas también *pesos sinápticos*. En la figura la entrada o *input* neto a la neurona está dada por:

$$v_k = \sum_{i=1}^m w_{ki}x_i + b_k \quad (3.1)$$

Donde la expresión b_k representa el sesgo o umbral (*bias* en inglés) el cual cumple la función de compensar la diferencia entre el valor medio de las entradas y el correspondiente

valor medio de las salidas deseadas, una función parecida a la de un intercepto en una regresión lineal (ver ecuaciones 2.1 y 2.2).

Posterior a eso entra en juego la función de activación o transferencia la cual dada la entrada neta v_k va a transformar este input mediante la función $\varphi(v_k)$ en la salida de la neurona.

$$\varphi(v_k) = \varphi\left(\sum_{i=1}^m w_{ki}x_i + b_k\right) = y_k \quad (3.2)$$

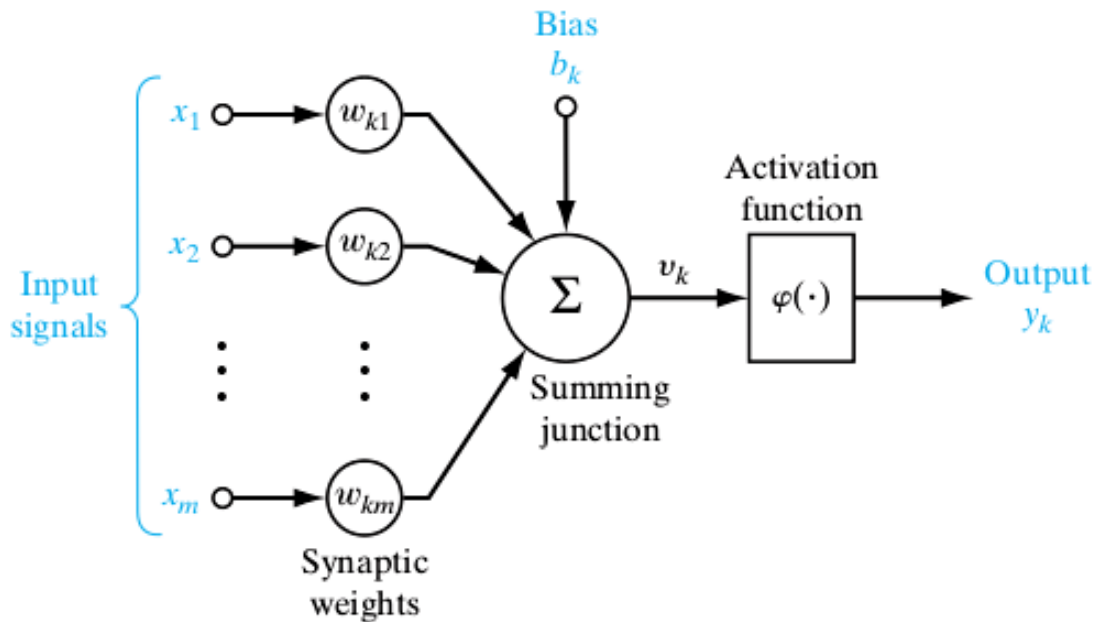


Figura 3.2: Neurona de McCulloch y Pitts

(Fuente: <https://www.opensourceforu.com>)

Esta neurona como ya se dijo antes fue pensada para manejar salidas de manera binaria, por lo que la función de activación tiene ese mismo comportamiento, transforma la entrada v_k en una salida binaria.

3.1.3.1. Funciones de transferencia

La función de transferencia usada por [McCulloch y Pitts \(1943\)](#) fue la función escalón unitario, dando como resulta un 0 o un 1 dependiendo de la entrada. A continuación se muestran las funciones de transferencia de redes neuronales artificiales más comunes.

Función escalón unitario

$$\varphi(a) = \begin{cases} 0 & \text{cuando } a \leq 0 \\ 1 & \text{cuando } a > 0 \end{cases} \quad (3.3)$$

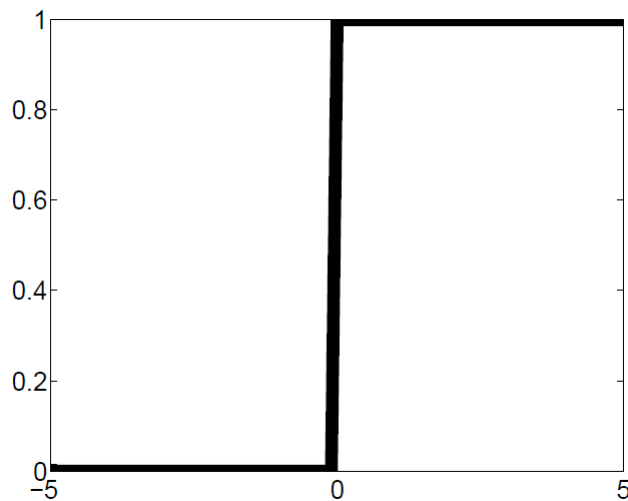


Figura 3.3: Gráfica de la función escalón unitario

(Fuente: [Méndez, J. y Marín, R. \(2013\)](#))

Cabe señalar que esta función al poseer una discontinuidad es sólo usada en el ejemplo de la neurona artificial de McCulloch y Pitts.

Función lineal

$$\varphi(a) = a \quad (3.4)$$

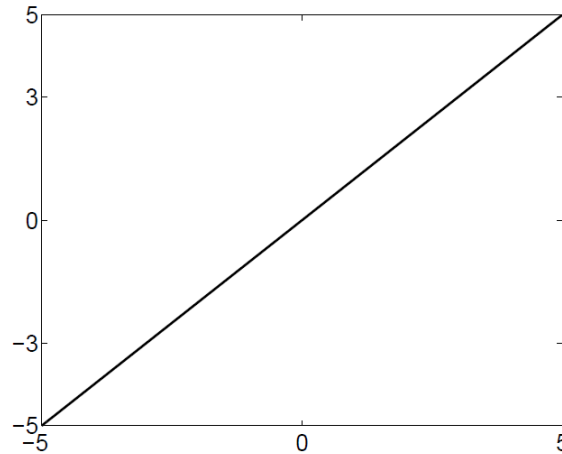


Figura 3.4: Gráfica de la función lineal

(Fuente: Méndez, J. y Marín, R. (2013))

Función sigmoidea

$$\varphi(a) = \frac{1}{1 + e^{-a}} \quad (3.5)$$

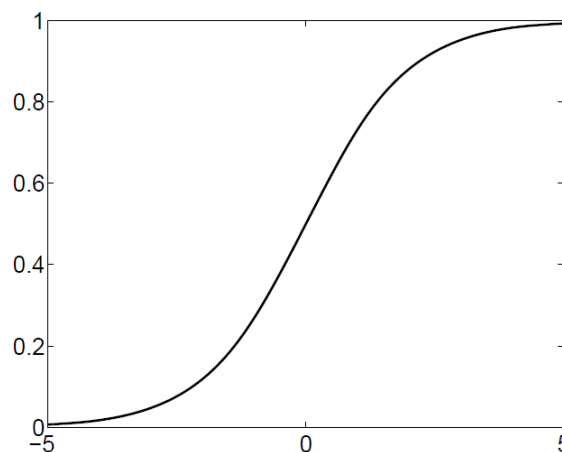


Figura 3.5: Gráfica de la función sigmoidea

(Fuente: Méndez, J. y Marín, R. (2013))

Función tangente hiperbólica

$$\varphi(a) = \tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}} \quad (3.6)$$

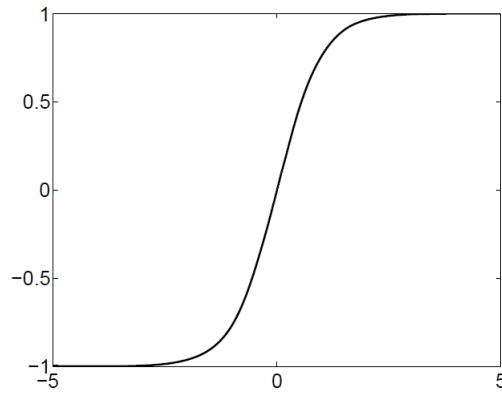


Figura 3.6: Gráfica de la función tangente hiperbólica

(Fuente: Méndez, J. y Marín, R. (2013))

Función gaussiana

$$\varphi(a) = \exp\left(-\frac{(a - \mu)^2}{2\sigma^2}\right) \quad (3.7)$$

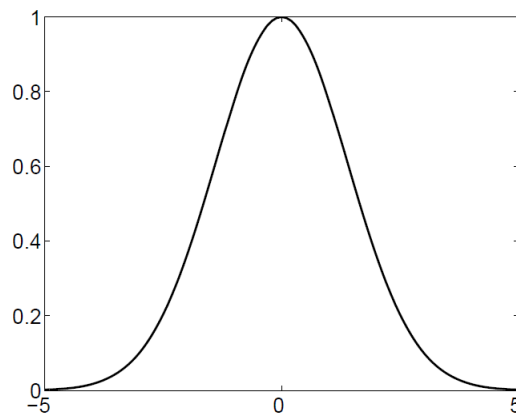


Figura 3.7: Gráfica de la función gaussiana

(Fuente: Méndez, J. y Marín, R. (2013))

Para entender mejor cómo funciona el modelo de la neurona artificial de 1943, a continuación se presenta un ejemplo (Marsalli, 2006).

Supongamos que hay una neurona en el cerebro de un pájaro que tiene dos receptores que, de alguna forma están conectados a los ojos del pájaro. Si el pájaro ve un objeto que es de forma redonda, una señal es enviada al primer receptor, pero si ve otra forma que no sea redonda, entonces no se envía ninguna señal. Así, el primer receptor es un detector de *redondez*. Si el pájaro ve un objeto de color púrpura, una señal es enviada al segundo receptor, si ve otro color, ninguna señal es enviada. Así, el segundo receptor es un detector de color púrpura. Notar que para ambos receptores hay una pregunta que puede ser respondida con un “sí” o un “no” y la señal es solamente es enviada si la respuesta es “sí”. El primer receptor responde a la pregunta *¿es el objeto redondo?* y el segundo receptor responde a la pregunta *¿es el objeto púrpura?*. Por lo tanto, nos gustaría producir un neurona artificial que que le diga al pájaro que coma un arándano, pero que evite comer fresas u otras cosas. En otras palabras queremos hacer que se envíe la señal de “comer” si el objeto es púrpura y redondo y no enviará ninguna señal si el objeto no cumple esas características, por lo tanto nuestro pájaro no comerá ese objeto.

Tabla 3.1: Tabla resumen de cómo actúa una neurona artificial en ciertos casos.

Objeto	¿Púrpura?	¿Redondo?	¿Comer?
Arándano	Sí	Sí	Sí
Bola de golf	No	Sí	No
Piedra	No	Sí	No
Violeta (flor)	Sí	No	No

(Fuente: Marsalli (2006))

Notar que sólo se enviarán señales de sí o no, 1 o 0, esto fue uno de los supuestos hechos por [McCulloch y Pitts \(1943\)](#).

3.1.4. Perceptrón monocapa

[Rosenblatt \(1957\)](#), desarrolló un modelo basado en la neurona artificial de McCulloch y Pitts y en un modelo de aprendizaje en base al error corregido. Este modelo se llamó Perceptrón, sin embargo, lo que llamó la atención en ese entonces fue la capacidad que tenía este modelo de **aprender** en base a la corrección del error y a reconocer patrones. El Perceptrón está constituido por un sensores de entrada y su salida es capaz de clasificar en dos categorías en base a si la salida es 1 (se activa) o 0 (se desactiva). Sin embargo este modelo al ser muy preliminar, tenía ciertas limitaciones que hicieron que se abandonara una vez que hizo su aparición el perceptrón multicapa, el cual hoy en día es altamente utilizado en el campo de la inteligencia artificial tanto para problemas de clasificación, predicción, etc, ya que demostró desempeñarse de manera eficiente en dichos campos.

3.1.5. Perceptrón multicapa

El tipo de arquitectura más conocido es el perceptrón multicapa (*Multi-layer perceptron* o MLP por sus siglas), el cual es una estructura que forma (y de ahí el nombre) una red de neuronas, que se interconectan entre sí. El perceptrón multicapa más básico posee una capa de neuronas de entrada, una capa oculta de neuronas y una capa de salida de neuronas.

El perceptrón multicapa en cada una de sus conexiones maneja pesos sinápticos entre las neuronas presinápticas y postsinápticas, llamemos a este peso w_{ij} que pondera la información que sale de la neurona i y entra a la neurona j . De esta forma al igual que la ecuación 3.1 podríamos decir que la información que le llega a la neurona j queda expresada de la siguiente forma:

$$v_j = \sum_i w_{ij}x_i + b_j \quad (3.8)$$

Es decir la información que maneja la neurona j es igual a la sumatoria ponderada de todas las neuronas i que se conectan con la neurona j . Supongamos ahora que se tiene un perceptrón multicapa con una capa de entrada, una capa oculta y una capa de salida, supongamos también que $f(\theta)$ y $g(\theta)$ son funciones de activación.

$$f\left(\sum_j w_{jk}g\left(\sum_i w_{ij}x_i + b_j\right) + b_k\right) = y_k \quad (3.9)$$

La ecuación 3.9 representa la salida y_k de un perceptrón multicapa. Si un perceptrón multicapa posee sólo una capa de salida, entonces y_k representa al único *output* que tendrá esa red.

En el ejemplo anterior se ha supuesto que la red neuronal es una red que se alimenta hacia adelante, es decir una *Feed-Forward Neural Network* (FFNN) que a pesar de ser una de las más usadas, existen otras estructuras y configuraciones de redes que han probado ser eficientes en cierto tipo de tareas, como por ejemplo las redes neuronales profundas (*deep learning*), redes convolucionales, redes recurrentes, entre otras.

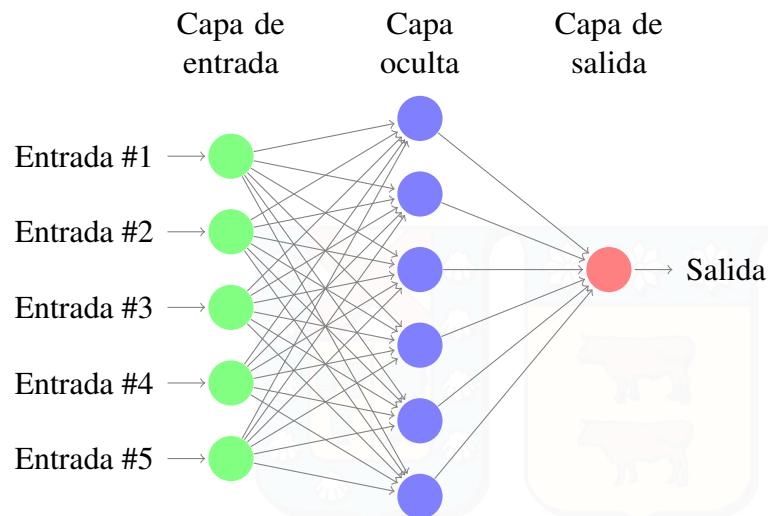


Figura 3.8: Esquema de un perceptrón multicapa con una capa oculta y una neurona de salida.

(Fuente: <http://www.texample.net>)

3.1.6. Entrenamiento

Una pregunta que surge naturalmente luego de las definiciones anteriores es: ¿Cómo encontramos los valores de los pesos w_{ij} y los sesgos b_j ?, y para responder a esta pregunta la fase de entrenamiento es principal, ya que es aquí donde la red aprende a responder, dado una serie de inputs. Existen básicamente dos tipos de aprendizajes, que dependen principalmente del tipo de problema que se desea resolver.

3.1.6.1. Aprendizaje supervisado

El aprendizaje supervisado se refiere al aprendizaje que ocurre al conocer cuáles son las salidas deseadas o *targets*. Se le presenta a la red neuronal un conjunto de patrones en conjunto con la salida deseada y de manera iterativa los pesos y los sesgos de la red se van ajustando hasta llegar a un punto en donde la red haya aprendido a tratar los datos.

3.1.6.2. Aprendizaje no supervisado

Al contrario del aprendizaje supervisado, aquí no se le muestran las salidas deseadas: Un ejemplo clásico de aprendizaje no supervisado es la creación de *clusters* mediante redes neuronales.

3.1.6.3. Backpropagation

El algoritmo de aprendizaje supervisado comúnmente utilizado para entrenar FFNN es el algoritmo de backpropagation o retropropagación de errores. Este algoritmo necesita como entrada una porción del total de los datos entregados a la red (conjunto de entrenamiento), para poder así ver cómo se comportan.

Es natural notar que el entrenamiento de una red naturalmente converge a un problema de optimización, en el cual el objetivo principal es minimizar el error de la señal de salida dado un *target* o salida esperada. El error es representado mediante el Error Cuadrático Medio (MSE):

$$MSE = E = \frac{1}{N} \sum_{n=1}^N e_n \quad (3.10)$$

$$e_n = \frac{1}{2} (s_n - y_n)^2 \quad (3.11)$$

Donde s_n es la salida deseada e y_n es la salida obtenida por la red neuronal (considerando una sola neurona de salida, de no ser este el caso se añadiría una sumatoria adicional).

Notar que y_n puede ser reemplazo por las expresiones obtenidas en la ecuación 3.9. Los pesos de la red se van a ir actualizando mediante un proceso iterativo que en cada iteración modificará los valores antiguos de la red sumándole una magnitud Δ usando el gradiente descendente:

$$w_{t+1} = w_t + \Delta w \quad (3.12)$$

El funcionamiento de Backpropagation se basa en propagar los errores de las señales de manera inversa a cómo se alimenta la red. El primer paso es inicializar la red neuronal con un conjunto de pesos y sesgos (si es que existen), luego alimentar la red hacia adelante según 3.8 y 3.9 para así obtener los valores de salida de cada una de las neuronas hasta llegar al valor y en donde se va a comparar el valor s deseado que dará lugar a el valor δ .

$$\delta = (s - y) \quad (3.13)$$

Este error se va ajustando con los pesos de la red según (k representa la neurona de salida y l una capa de la red):

$$\delta_j^l = w_{jk} \delta \quad (3.14)$$

$$\delta_i^l = \sum_j w_{ij} \delta_j^{l-1} \quad (3.15)$$

Y así sucesivamente. La regla de actualización de pesos Δw^l es:

$$\Delta w^l = -\mu \frac{\partial E}{\partial w^l} \quad (3.16)$$

En la ecuación 3.16 μ representa el factor de aprendizaje o *learning rate*, uno de los parámetros clave en el algoritmo de backpropagation que indica la magnitud del paso en cada iteración.

Denotemos el output de una neurona i cualquiera como v_i^l y como $f(v_i^l) = O_i^l$ su valor en la función de activación (recordar que v_i^l es una suma-producto entre pesos y entradas a la neurona). De esta forma para cualquier capa la actualización de pesos será la multiplicación del factor de aprendizaje con el error propagado según 3.15 y con el valor O_i^{l-1} de la capa anterior.

$$\Delta w_{ij}^l = -\mu \delta_i^l O_j^{l-1} \quad (3.17)$$

Este proceso se repite tantas veces como sea necesario hasta que se cumpla alguna de las condiciones de término del algoritmo.

3.2. Algoritmo de colonias de abejas

3.2.1. Inteligencia de enjambre

Dado que las *Redes Neuronales Artificiales* han sido satisfactoriamente aplicadas en modelar la *no linealidad* de la realidad y tienen características tales como generalización, auto-organización, operaciones a tiempo real y aprendizaje, están involucradas en muchos campos de investigación. Encontrar una estructura/arquitectura de la red que sea apropiada y encontrar los valores óptimos de los pesos es una tarea que ha sido numerosamente estudiada y se traduce en un problema de difícil optimización, por lo que el éxito de una red depende de su arquitectura, el algoritmo de entrenamiento en conjunto con sus parámetros.

Tradicionalmente las *Redes Neuronales Artificiales* han sido llevadas a cabo usando el algoritmo de *Backpropagation* como algoritmo de entrenamiento, sin embargo el uso de este método tiene algunas implicancias como la forma de la superficie de la función de error y los valores iniciales que se le da a los parámetros de la red a ser optimizados mediante el algoritmo, conocidos como semillas (Bányász, 1995; Kunche y Reddy, 2016). Si, por ejemplo, se tiene una función multimodal, el método de *Backpropagation* podría entrar en uno de los muchos mínimos locales y quedarse así sin poder alcanzar el mínimo global, esto puede suceder también en otros algoritmos basados en el gradiente descendente ya que podría darse la situación de que la derivada sea muy pequeña para hacer un cambio significativo en los pesos de la red dando como resultado que la red llegue a un mínimo

incorrecto o llegue a un estado de “parálisis”.

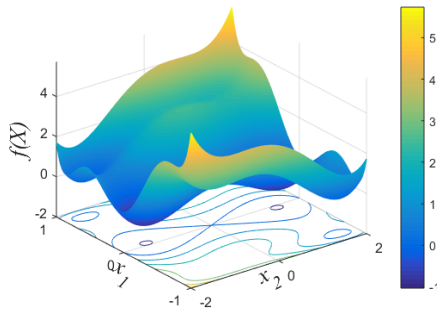


Figura 3.9: *Six-Hump Camel Function*.
Ejemplo de una función con múltiples
óptimos locales.

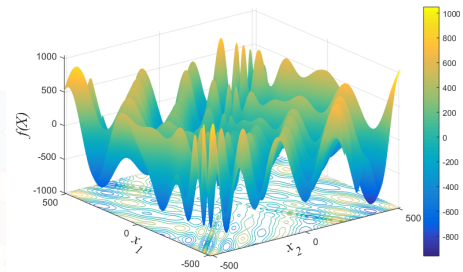


Figura 3.10: *EggHolder Function*.
Ejemplo de una función con múltiples
óptimos locales.

(Fuente: <http://al-roomi.org/>)

Para evitar estas desventajas se han propuesto otros algoritmos con los cuales es posible entrenar una red neuronal, entre los cuales destacan los algoritmos genéticos (GA), optimización por enjambre de partículas (PSO) y la inteligencia de enjambre (Swarm Intelligence) los cuales no sólo son capaces de lidiar con la combinación óptima de pesos de una red, sino que también con su estructura y arquitectura (cantidad de neuronas y capas ocultas).

La inteligencia de enjambre se ha convertido en los últimos años en un tema de interés en la investigación en variados ámbitos, siendo un “intento de diseñar algoritmos basados en el comportamiento colectivo de colonias de insectos u otras sociedades animales” (Bonabeau et al., 1999). En este sentido uno de los clásicos ejemplos de enjambres es el de las abejas alrededor de su colmena y es aquí donde nace el algoritmo de colonias de abejas propuesto por Karaboga (2005).

Existen dos conceptos fundamentales para entender el comportamiento de la inteligencia de enjambre: auto-organización y división del trabajo.

- **Auto-organización:** Esta característica puede ser definida como un conjunto de mecanismos dinámicos que le permiten a un sistema interactuar con sus componentes de menor nivel. Estos mecanismos establecen reglas básicas para la interacción entre los componentes del sistema. (cita)Bonabeau et al. ha caracterizado cuatro propiedades en las que descansa la auto-organización.
 - Retroalimentación positiva, lo cual puede ser interpretado como el reforzamiento de estructuras convenientes, como puede ser una fuente de comida favorable para el enjambre.
 - Retroalimentación negativa, contrario a lo anterior un ejemplo de esto podría ser el agotamiento de una fuente de comida.
 - Fluctuaciones, que hacen referencia principalmente a la aleatoriedad en el descubrimiento de ciertas estructuras.
 - Interacciones entre los individuos y compartir información.
- **División del trabajo:** Dentro de un enjambre cada individuo tiene una tarea distinta basada en su especialidad. Tareas simultáneas realizadas por individuos especializados se creen más eficientes que aquellas tareas que son realizadas de manera secuencial.

Estas dos características son fundamentales para la inteligencia de enjambre en cuanto a la resolución de problemas y adaptación al medio.

3.2.2. Comportamiento de enjambres de abejas

El comportamiento mínimo que se espera en la inteligencia colectiva de los enjambres de abejas consiste en básicamente tres componentes ([Karaboga, 2005](#)):

- **Fuentes de comida:** Una fuente de comida que podría ser interpretado como un lugar de donde las abejas pueden obtener néctar puede ser evaluada y valorada en función de su proximidad con la colmena, la cantidad de néctar que posee o la posibilidad de extraer su energía.
- **Abejas obreras:** Relacionadas con una fuente de comida en particular, las abejas obreras van a **explotar** esa fuente de recursos, evaluando su eficiencia y luego informando a la colmena acerca de su explotación.
- **Abejas no-obreras:** a diferencia de las abejas obreras, las no-obreras no están relacionadas a una fuente de comida encontrada, su trabajo es entonces **explorar** por una nueva fuente de comida, donde se distinguen dos tipos de abejas: las espectadoras, que basadas en la información de las abejas obreras, establecen nuevas fuentes de comida y las exploradoras, que a diferencia de las espectadoras, buscan fuentes de comida completamente diferentes sin basarse en la información de las obreras.

Es posible notar que las propiedades de la auto-organización se cumplen en lo anteriormente descrito:

- **Retroalimentación positiva:** si una fuente de comida es rica en néctar, entonces las abejas obreras informarán de esto a la colmena.

- Retroalimentación negativa: si una fuente de comida no es suficientemente eficiente, entonces se abandonará esa fuente de comida y las abejas no-obreras exploradoras buscarán una nueva fuente de comida.
- Fluctuaciones: en el caso de una abeja exploradora, la manera de buscar una nueva fuente de comida será completamente aleatoria.
- Interacciones: las abejas compartirán información acerca de las fuentes de comida.

En el algoritmo de colonias de abejas (ABC) la posición de una fuente de comida representa una posible solución al problema de optimización y la cantidad de néctar de esta fuente es representada por su calidad o *fitness* a través de 3.19. La cantidad de soluciones en la población (SN) equivale al número de abejas obreras y también coincide con la cantidad de abejas espectadoras. El algoritmo en primera instancia genera de manera aleatoria entre una cota superior y una cota inferior una población de SN soluciones (posiciones de fuentes de comida) según 3.18.

$$x_i = x_{min} + rand(0, 1)(x_{max} - x_{min}) \quad (3.18)$$

$$fit_i = \begin{cases} \frac{1}{1+f(x_i)} & f(x_i) \geq 0 \\ \frac{1}{|f(x_i)|} & f(x_i) < 0 \end{cases} \quad (3.19)$$

Donde $f(x_i)$ representa el valor en la función objetivo de cada solución y $x_i, i = 1, 2, 3, \dots, SN$ es un vector que posee D parámetros a optimizar, por lo que D representa la cantidad de variables de decisión del problema de optimización. Luego de la fase

de inicialización del algoritmo la población de soluciones será sometida a ciclos repetidos $Cycles = 1, 2, \dots, MCN$ (*Max. Cycle Number*) de las fases de abejas obreras, abejas espectadoras y abejas exploradoras.

- **Fase de abejas obreras (Employed):** en esta fase, para cada una de las SN fuentes de comida, cada abeja obrera produce una modificación a partir de esa fuente según 3.20, esta solución modificada necesita ser evaluada, por lo que se calcula su valor en la función objetivo y su *fitness*. Si la solución creada tiene una calidad mayor, entonces se olvida la solución original y se reemplaza por la solución modificada, en otro caso se mantiene la solución original (selección *greedy*). Luego de que todas las abejas obreras han pasado por este proceso, se pasa a la siguiente fase.

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \quad (3.20)$$

Donde $k \in \{1, 2, \dots, SN\}$ y $j \in \{1, 2, \dots, D\}$ son índices elegidos al azar, sin embargo estos índices deben ser distintos entre sí. ϕ_{ij} es un número al azar en el intervalo $[-1, 1]$.

- **Fase de abejas espectadoras (Onlookers):** con la información proporcionada por las abejas obreras, las abejas espectadoras calculan una probabilidad basada en la calidad según 3.21 para cada fuente de alimento y acorde a esa probabilidad se elige una solución y se sigue el mismo procedimiento anterior: se forma una solución

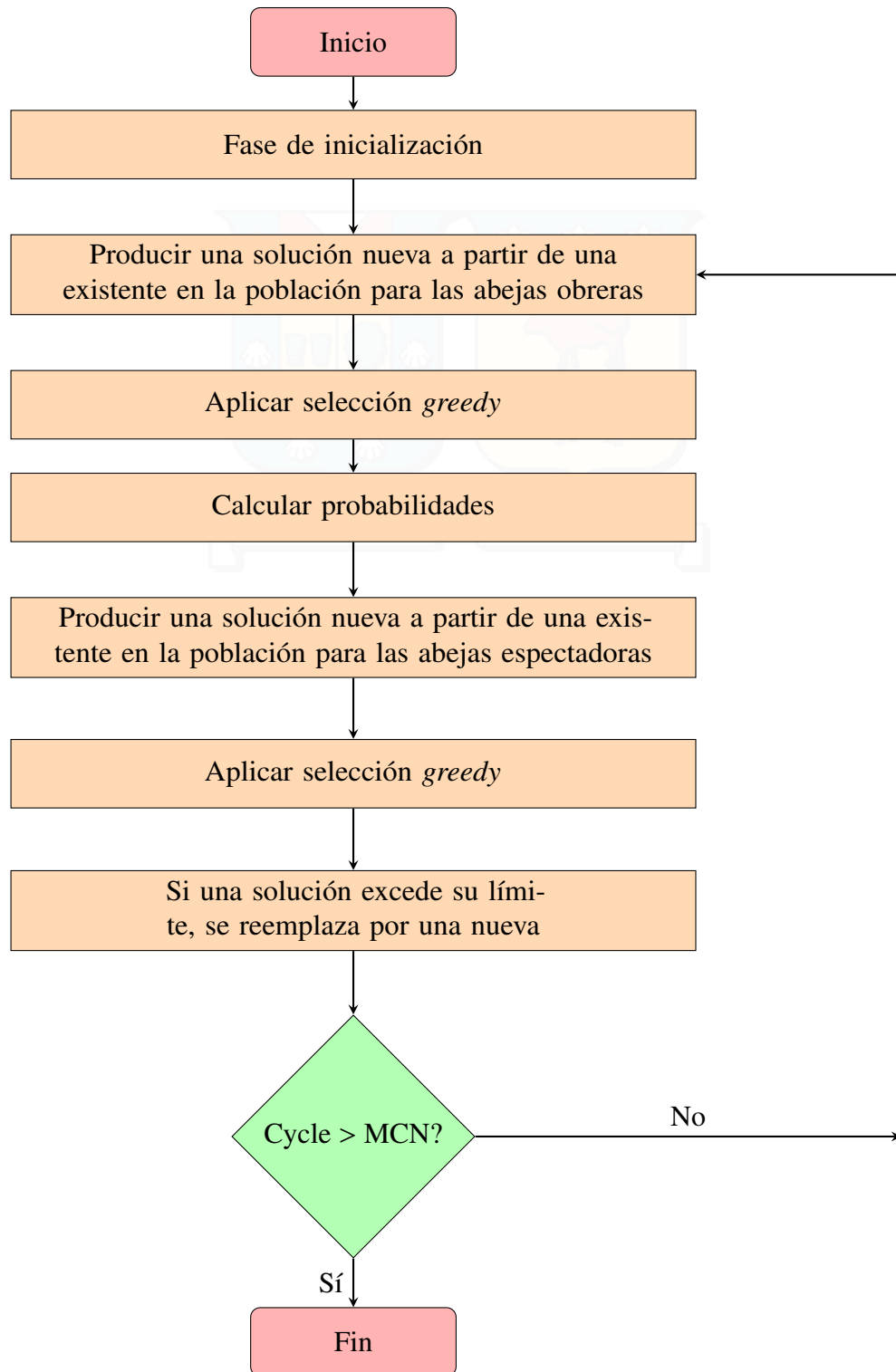
nueva con 3.20 y se aplica una selección *greedy*.

$$p_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n} \quad (3.21)$$

Una vez que todas las abejas han hecho este trabajo se sigue con la última fase.

- **Fase de abejas exploradoras (Scouts):** cada fuente de alimento tiene un contador, el cual aumenta cada vez que una solución **no** puede ser mejorada, en caso contrario este contador vuelve a su valor inicial cero. Cuando el contador llega a un valor dado (límite) se abandona esta solución y la abeja que estaba asociada a esta solución se convierte en una abeja exploradora, la cual produce una nueva solución aleatoriamente según 3.18, reemplazando aquella que fue abandonada y que no pudo mejorar.

A continuación se presenta un diagrama de flujo y un pseudo-código para el algoritmo de colonias de abejas (*Artificial Bee Colony Algorithm*).

**Figura 3.11:** Diagrama de flujo del funcionamiento del algoritmo ABC.

(Fuente: Elaboración propia.)

Algorithm 1 Pseudo-código ABC

```
1: Inicializar la población de soluciones  $x_i, i = 1, 2, \dots, SN$ .
2: Evaluar la población.
3:  $Cycle = 1$ 
4: while  $Cycle < MCN$  do
5:   Producir nuevas soluciones  $v_i$  para las abejas obreras y evaluarlas.
6:   Aplicar selección Greedy.
7:   Calcular probabilidades  $p_i$  para las soluciones  $x_i$ .
8:   Producir nuevas soluciones  $v_i$  para la población de abejas espectadoras a partir de
   las soluciones  $x_i$  seleccionadas por su probabilidad  $p_i$  y evaluarlas.
9:   Aplicar selección Greedy.
10:  if Existe una solución abandonada then Reemplazarla por una solución generada
   aleatoriamente por las abejas exploradoras.
11:  Memorizar la mejor solución encontrada.
12:   $Cycle = Cycle + 1$ 
```

En resumen, el algoritmo ABC:

- Está inspirado en el comportamiento forrajero de las abejas melíferas.
- Es un algoritmo de optimización global.
- Fue inicialmente propuesto para optimización numérica por [Karaboga \(2005\)](#).
- También puede ser utilizado para problemas de optimización combinatorial ([Pan et al., 2011](#)).
- Ha mostrado flexibilidad al ser usado en problemas irrestrictos y con restricciones ([Domínguez \(2009\)](#), [Karaboga y Basturk \(2007\)](#), [Karaboga y Akay \(2009\)](#)).
- Emplea tres parámetros de control principales: Tamaño de población, *Max Cycle Number* y *limit* que deben ser predeterminados por el usuario al igual que las cotas de las soluciones x_{min} y x_{max} .
- Es bastante sencillo, flexible y robusto ([Rao et al., 2008](#)).

3.2.3. Algoritmo ABC para entrenar redes neuronales artificiales

A la hora de modificar el algoritmo ABC de colonias de abejas para trabajar sobre una RNA surgen dudas que no son triviales de responder:

- ¿Cuáles son los parámetros de entrada al algoritmo?
- ¿Cómo se relacionan con la red neuronal?
- ¿Se necesitan otros parámetros de control?
- ¿De qué manera el algoritmo ajustará los pesos y sesgos de la red?

Estas preguntas pueden ser respondidas una vez que entendemos qué es lo que se está haciendo y por qué. Si se vuelve a la base del algoritmo de backpropagation es posible darse cuenta que su objetivo es optimizar los pesos y sesgos (que serían las variables) de una red neuronal dada una función objetivo que en este caso sería el MSE o error cuadrático medio.

En el algoritmo ABC entonces, necesitamos aplicar la información anteriormente obtenida: los pesos w_{ij} y sesgos b_j de la red serán las *food positions* o variables del problema y el MSE de la red será la función objetivo que hasta el momento no sabemos cuál es su forma, sin embargo sabemos que a menor MSE se ajusta un modelo mejor.

De esta forma debemos encontrar la mejor configuración de pesos y sesgos comparando cada vez que nuestro MSE sea menor en cada iteración.

- **Fase de inicialización:** en esta fase se debe crear la población de abejas en conjunto con las soluciones según la ecuación 3.18. Estas soluciones a su vez deben ser

evaluadas en la función objetivo 3.10, para lo cual cada vector de pesos y sesgos debe ser introducido a la red neuronal para luego calcular su MSE respectivo. También en esta fase quedan definidos otros parámetros del algoritmo ABC como el parámetro *limit*, los x_{min} y x_{max} , la cantidad SN y el *Max Cycle Number (MCN)* que van a definir la capacidad de **explotación** y **exploración** del algoritmo.

- **Fase de abejas obreras (employed):** aquí para cada una de las SN configuraciones de pesos y sesgos se va a producir una solución nueva y se aplicará una selección *greedy* basada en su *fitness* con la ecuación 3.19 reemplazando el valor del MSE calculado.
- **Fase de abejas espectadoras (Onlookers):** se calculan probabilidades y se repite el paso anterior.
- **Fase de abejas exploradoras (Scouts):** una vez que no se puede mejorar la solución y su contador de *trials* ha superado al valor *limit* se reemplaza esa solución por una nueva, evaluando esa solución en la red neuronal y rescatando su MSE respectivo.
- **Término:** el algoritmo termina cuando se alcanza el *Max Cycle Number* y la red neuronal queda configurada con aquella combinación de pesos y sesgos que imputan el menor MSE alcanzado.

4 | Datos

4.1. Estadística descriptiva y gráficos

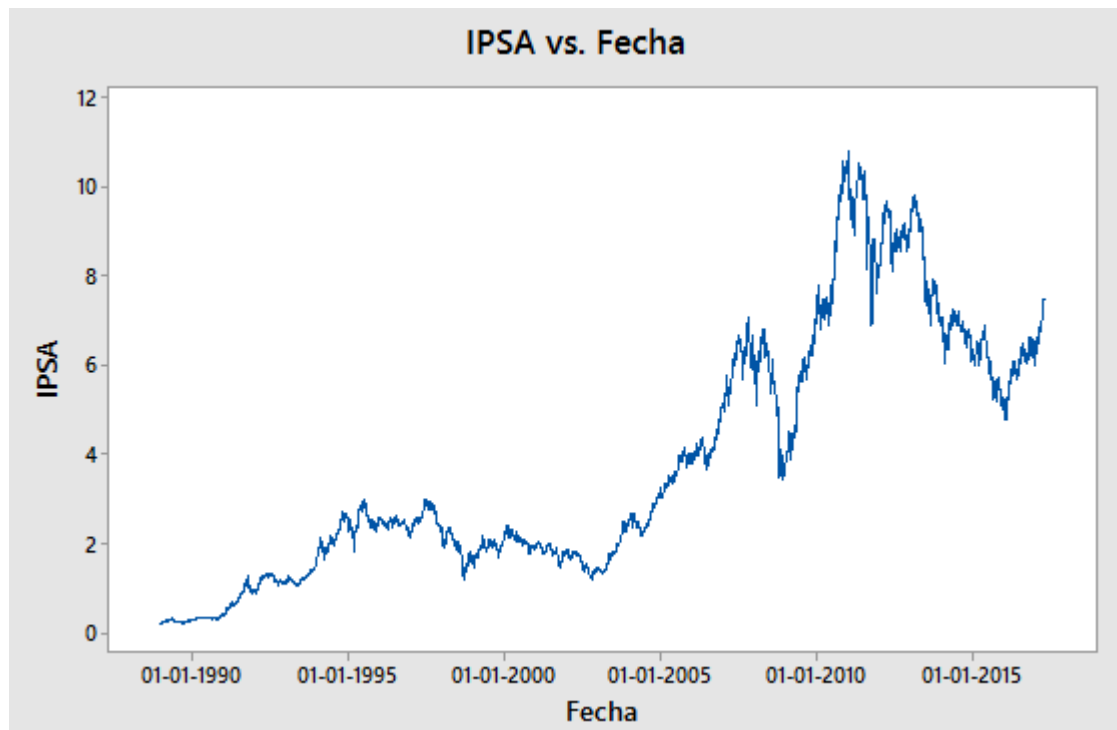
Los datos analizados en este estudio corresponden a los principales índices de acciones de Perú, Argentina, Colombia, México, Brasil y Chile. Todas las series están expresadas en dólares [*USD*] y los rangos de fechas de las series son las siguientes:

- Argentina: Merval 01/1990 - 04/2017
- Brasil: IBOVESPA 01/1968 - 04/2017
- Chile: IPSA 01/1989 - 04/2017
- Colombia: COLCAP 08/2007 - 04/2017
- México: IPC 01/1982 - 04/2017
- Perú: INCA 06/2007 - 04/2015

A continuación se presenta una tabla con los indicadores de la estadística descriptiva de las series consideradas.

Tabla 4.1: Estadística descriptiva de las series consideradas.

Estadísticas	IPSA	MERVAL	INCA	COLCAP	IPC	IBOVESPA
Número de datos	7051	6725	1969	2357	8824	12145
Promedio	3,7823	568,8	32,455	0,68887	1140	8093
Desviación estándar	2,7537	259,76	7,507	0,22341	1115,3	11051
Mínimo	0,193	75,21	10,795	0,28772	7,7	98
Máximo	10,7814	1486,85	46,837	1,06726	3679,5	44616
Rango	10,5884	1411,64	36,042	0,77954	3671,9	44518
Curtosis	-0,68	0,77	0,16	-1,53	-0,98	1,45
Coefficiente de variación	72,81	45,67	23,13	32,43	97,83	136,55

**Figura 4.1:** Gráfica de la serie IPSA de Chile

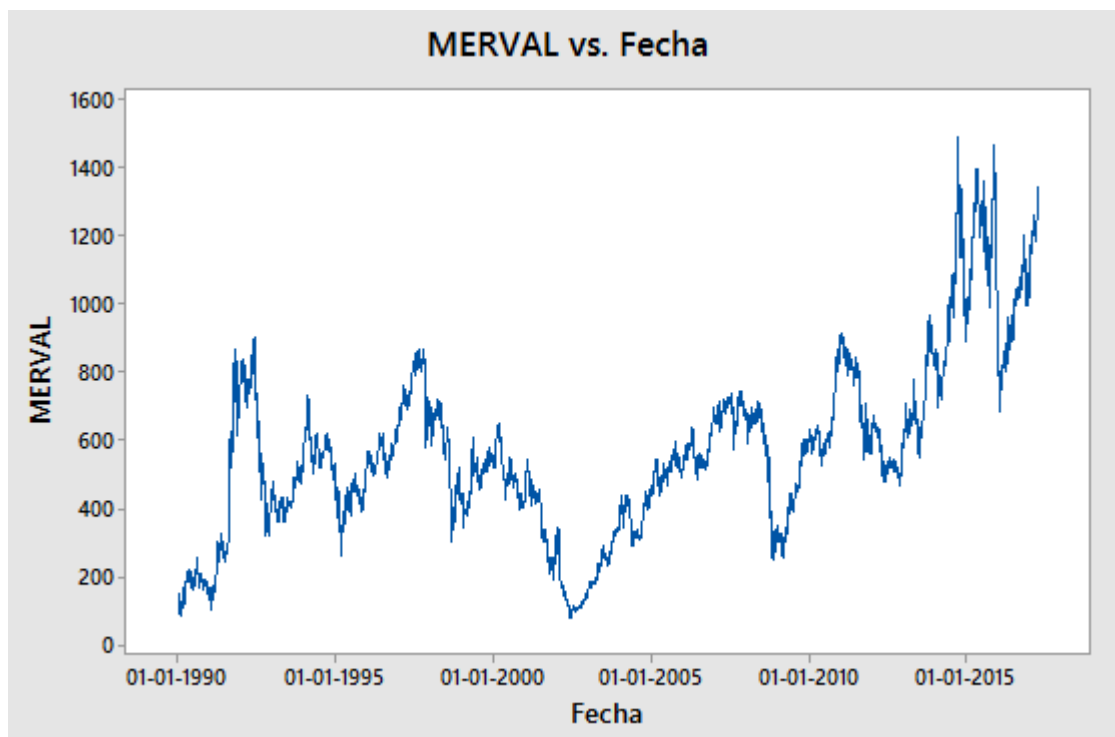


Figura 4.2: Gráfica de la serie MERVAL de Argentina

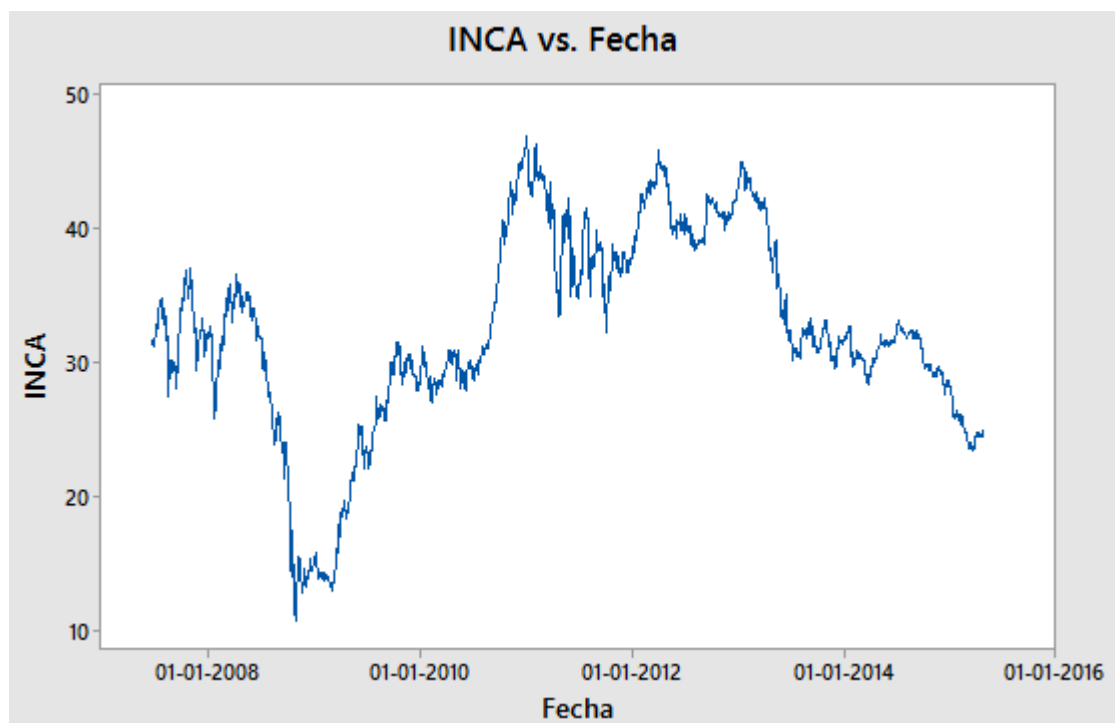


Figura 4.3: Gráfica de la serie INCA de Perú

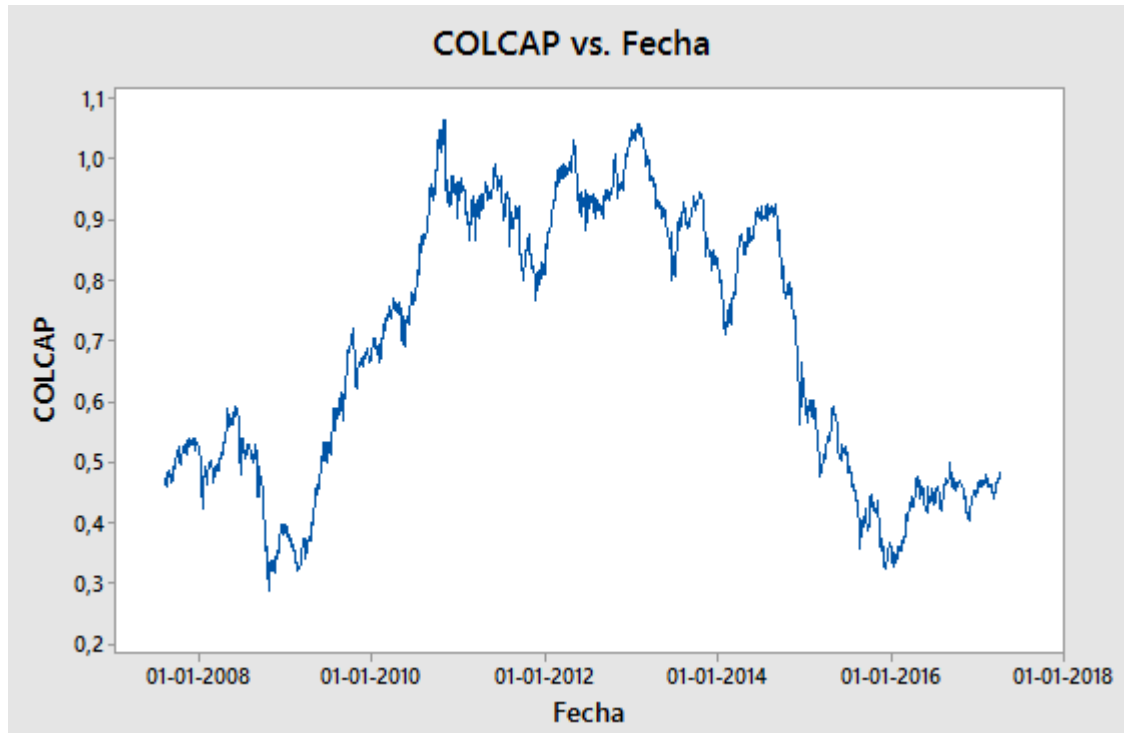


Figura 4.4: Gráfica de la serie COLCAP de Colombia

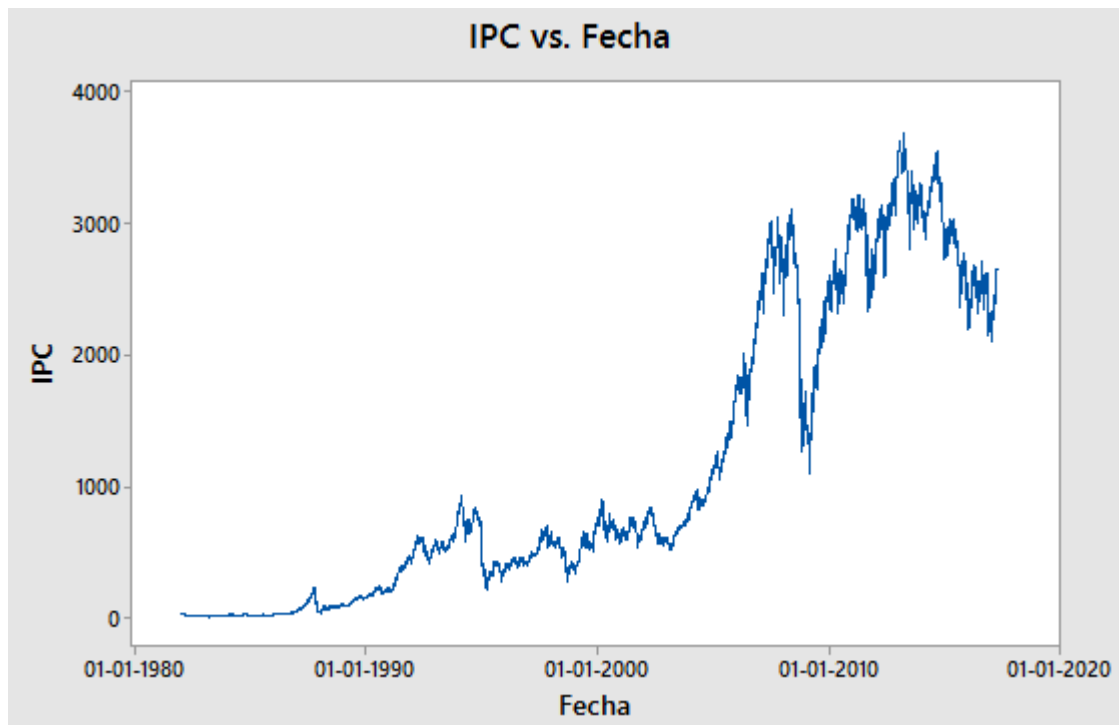


Figura 4.5: Gráfica de la serie IPC de México

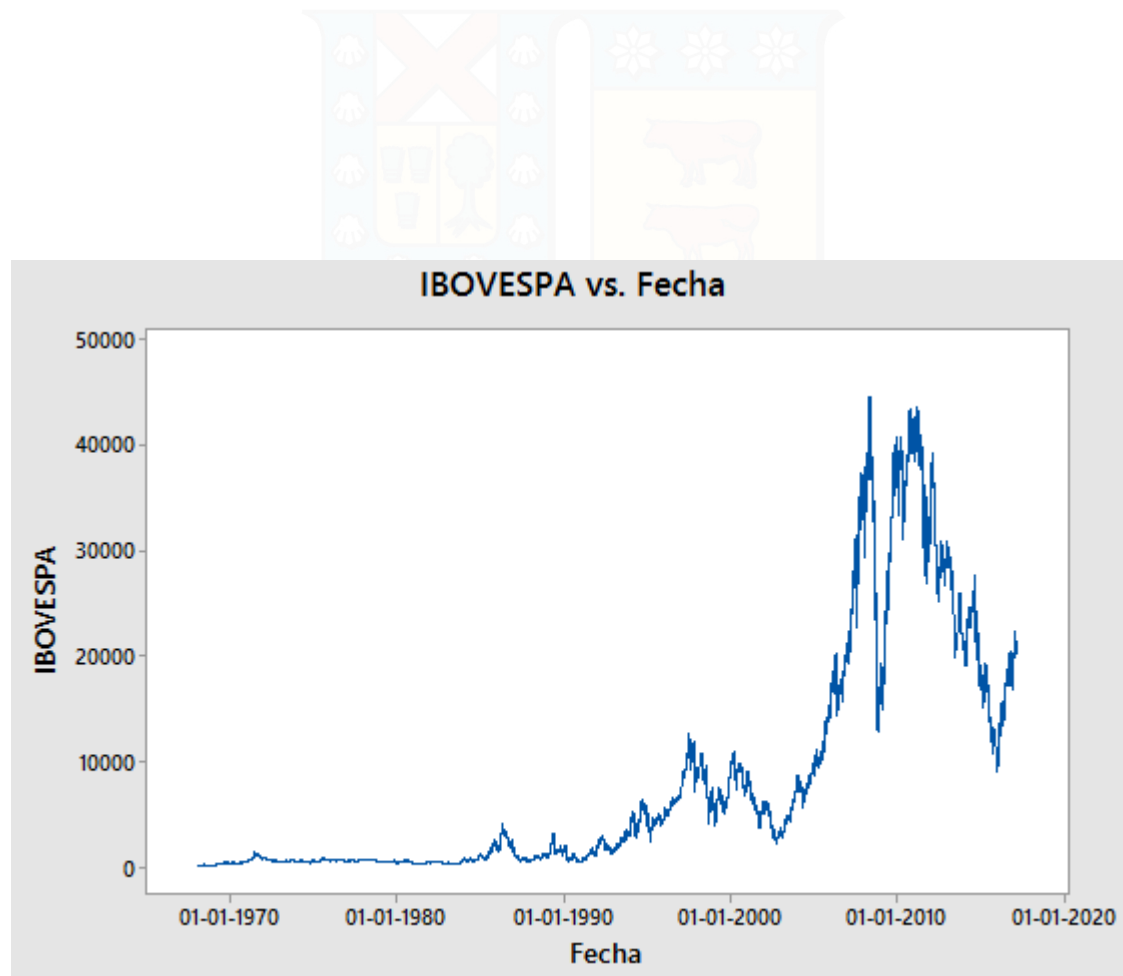


Figura 4.6: Gráfica de la serie IBOVESPA de Brasil

5 | Procedimiento

A continuación se presentan los parámetros considerados para probar la capacidad de predicción del algoritmo ABC en series financieras del mercado latinoamericano, en conjunto con su posterior comparación con el algoritmo convencional de backpropagation.

5.1. Ventana móvil

La ventana móvil o ventana corrediza es una técnica que se utiliza en predicción de series de tiempo para testear un modelo con datos fuera de la muestra (*out-sample*). Supongamos que se tiene una serie de datos ordenados temporalmente de tamaño N y elegimos una ventana de datos de tamaño $s < N$, esto es $1, 2, \dots, s$. Con una red neuronal es posible modelar con esta ventana de s datos y pronosticar el valor $s + 1$. En la siguiente iteración tomando el mismo tamaño de ventana se puede mover para modelar con los datos $2, 3, \dots, s + 1$ y pronosticar el dato $s + 2$, posteriormente se modelará con los datos $3, 4, \dots, s + 2$ y pronosticar el dato $s + 3$. Este procedimiento se repite $N - s$ veces hasta que seamos capaces de pronosticar el dato N .

Como resultado tendremos un vector de $N - s$ datos pronosticados mediante la red neuronal de los cuales conocemos cuál es su valor real gracias a que ya teníamos nuestro vector de N datos original. Con estos datos entonces podemos sacar medidas tales como el MSE 3.10 y corroborar cómo nuestra red pronostica datos.

Tradicionalmente se usa el set de datos entero para pronosticar un valor en el futuro, sin embargo esto no nos da luces de cómo pronostica nuestro modelo. En la siguiente figura las bolas azules representan datos para el modelamiento y la bola roja representa la predicción.

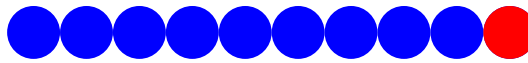


Figura 5.1: Ejemplo de modelado y pronóstico tradicional.

(Fuente: Elaboración propia.)

Con el método de ventana móvil podemos ver qué tan eficiente es nuestro modelo a la hora de pronosticar datos nunca antes vistos.

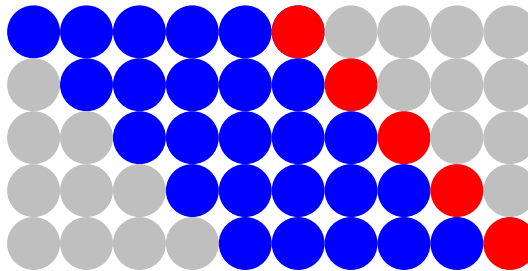


Figura 5.2: Ejemplo de ventana móvil.

(Fuente: Elaboración propia.)

En este trabajo se considerará un tamaño $N = 500$ y las ventanas serán de tamaño $s = 250$, lo cual representa aproximadamente un año hábil.

5.2. Estructura de la ANN

Una red neuronal posee una arquitectura, esto es, número de capas ocultas, número de neuronas por capa, número de neuronas en la capa de entrada, número de neuronas en la capa de salida, etc. De esta arquitectura depende el modelamiento de la red y por ende su rendimiento.

Los parámetros propiamente tales de la red neuronal a considerar en este apartado serán: número de capas ocultas, número de neuronas en cada capa y los inputs de la red.

Para los experimentos realizados se consideró que cada capa oculta tiene el mismo número de neuronas, en cambio para la capa de salida esta posee una sola neurona, la cual representa el precio de salida.

5.2.1. Rentabilidad y normalización

En general las series de tiempo financieras se analizan desde el punto de vista de la rentabilidad más que el precio en sí. [Campbell y Andrew \(1997\)](#) entregan dos razones para ello:

- La rentabilidad es el indicador clave a medir para el rendimiento de inversiones.
- Las series de rentabilidad son más fáciles de manejar para su análisis y se ajustan mejor a propiedades estadísticas de series.

Para todas las series se calculará la rentabilidad de forma logarítmica. Suponer x_{t-1} y x_t dos valores consecutivos de una serie de tiempo:

$$R_t = \ln\left(\frac{x_t}{x_{t-1}}\right) \quad (5.1)$$

Con tal de evitar que la red neuronal esté mal condicionada, luego de rentabilizar los datos se va a aplicar una transformación para que los datos estén en el intervalo $[0, 1]$, esto permite que que todas las entradas a la red estén en un rango comparable y favorece el entrenamiento de manera más eficiente. La normalización queda expresada de la siguiente forma:

$$X'_{normalizado} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (5.2)$$

5.2.2. Elección de las variables de entrada

La red neuronal necesita *inputs* o variables de entrada, las cuales van a explicar nuestra variable de salida que en este caso será el precio de la serie financiera. Si bien según la econometría existen muchas variables que se podrían considerar para explicar el precio de una serie del mercado latinoamericano, se considerará en este caso los rezagos de la misma variable. Es decir, para pronosticar el dato x_t se considerará como variable explicativa los valores $x_{t-1}, x_{t-2}, \dots, x_{t-k}$. Se probará $k = 5, k = 10, k = 15$ y $k = 20$ y se escogerá aquella cantidad de rezagos que logre un MSE menor.

5.2.3. Elección de la arquitectura

La elección del número de capas y neuronas será en base a un proceso iterativo de probar numerosas configuraciones de capas y número de neuronas por capa, eligiendo aquella que impute el menor error cuadrático medio.

El proceso será el siguiente: se comenzará con 2 capas y 5 neuronas por capa, luego la cantidad de neuronas irá aumentando a 10 y luego a 15, una vez que se llegue a 15 neuronas se aumenta el número de capas en 1 y se vuelve a 5 neuronas, repitiendo el proceso hasta llegar a 10 capas ocultas y 15 neuronas.

Para cada configuración se entrenará solamente la primera ventana de la serie y se probarán $k = 5$, $k = 10$, $k = 15$ y $k = 20$ rezagos, guardando su valor del error cuadrático medio en cada caso. A continuación se detalla la cantidad de capas, la cantidad de neuronas de cada capa y la cantidad de rezagos que se considerarán como variables de entrada a la red, en conjunto con su error cuadrático medio obtenido.

Tabla 5.1: Tabla resumen de las configuraciones de la ANN.

	# Capas	# Neuronas	# Rezagos	MSE
IPSA (CHI)	3	10	5	0,0200
MERVAL (ARG)	5	10	5	0,057
INCA (PER)	2	15	20	0,0106
COLCAP (COL)	3	15	20	0,0133
IPC (MEX)	4	15	20	0,0131
IBOVESPA (BRA)	2	15	20	0,0151

(Fuente: Elaboración propia.)

5.3. Parámetros algoritmo ABC

Como se mencionó anteriormente, los parámetros del algoritmo ABC van a definir la capacidad de exploración y explotación del algoritmo. También a medida que aumentamos estos parámetros y al hacer la búsqueda más intensa, aumentan los esfuerzos computacionales.

Los valores de x_{min} y x_{max} utilizados serán -10 y $+10$ respectivamente, como es sugerido en los trabajos de Karaboga et al. (2007), Akpinar et al. (2017) y Shah et al. (2011). El parámetro *limit* utilizado tendrá un valor de 365 y la población de abejas será de 100.

Tabla 5.2: Parámetros del algoritmo ABC considerados.

Parámetro	Valor
limit	365
MCN	300
x_{min}	-10
x_{max}	10
Población de abejas	100

(Fuente: Elaboración propia.)

Posteriormente cuando se compare con el algoritmo de backpropagation se usará un *learning rate* de $\mu = 0,01$ y la función de activación será la función 3.5.

Tabla 5.3: Parámetros del algoritmo backpropagation.

Parámetro	Valor
μ	0.01
Función de activación	Sigmoidea

(Fuente: Elaboración propia.)

5.4. Entrenamiento ABC-BP

Para finalizar los experimentos computacionales se tratará de combinar ambos algoritmos de entrenamiento y ver si existen sinergias entre ambos. Para lograr esto la idea es tomar la solución entregada por backpropagation (combinación de pesos y sesgos) e inicializar con esta solución el algoritmo ABC (recordar que el algoritmo ABC normalmente se inicia de manera aleatoria).

De esta forma al inicializar la población de soluciones, se repetirá la solución a través de las SN filas de la matriz de soluciones y se empezará a correr el algoritmo

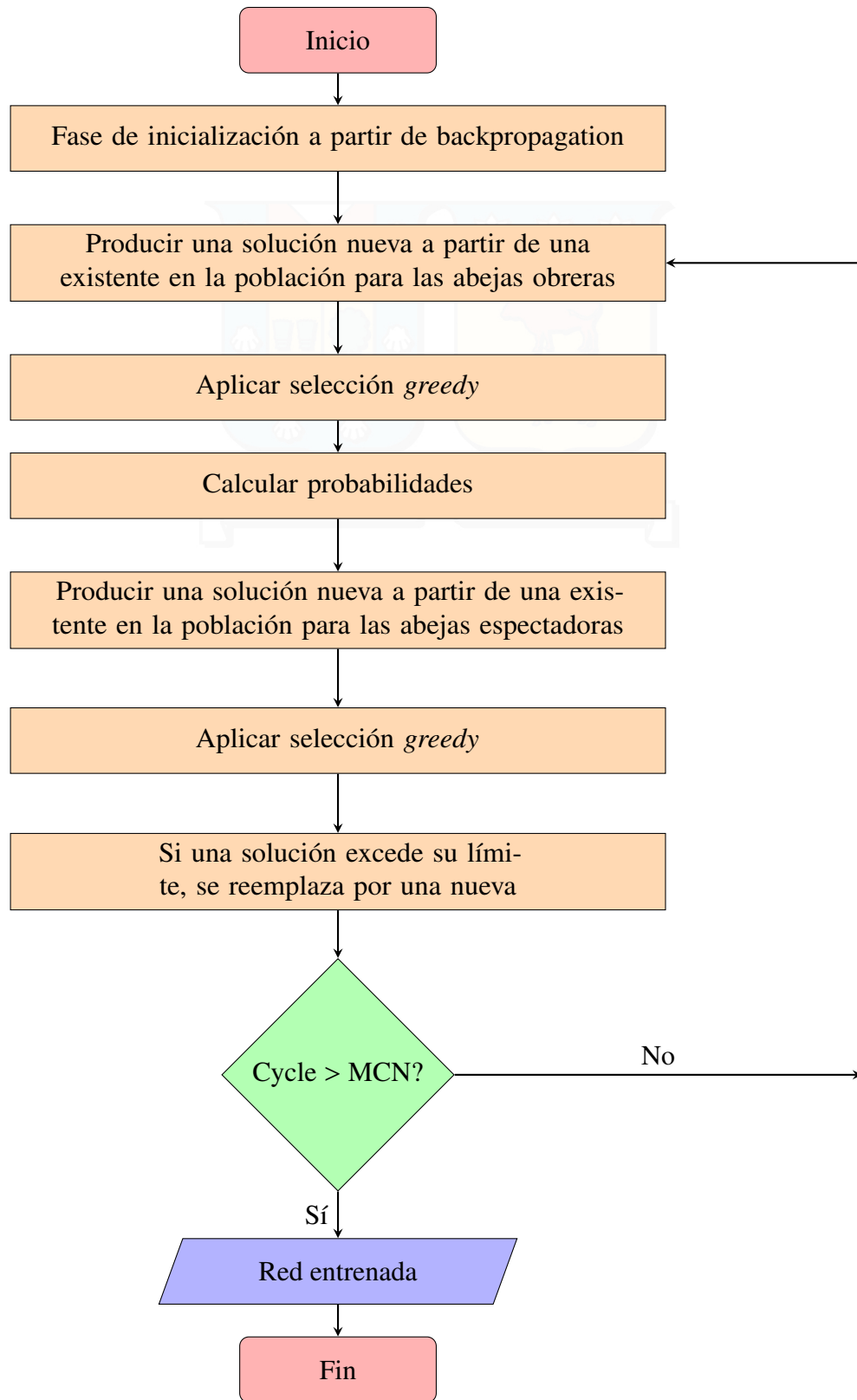


Figura 5.3: Diagrama de flujo del funcionamiento del algoritmo ABC inicializado con backpropagation.

(Fuente: Elaboración propia.)

6 | Resultados

6.1. IPSA

6.1.1. Predicción

En la figura 6.1 se puede observar la predicción de la red neuronal entrenada mediante el algoritmo ABC para la serie IPSA a lo largo de la ventana corrediza.

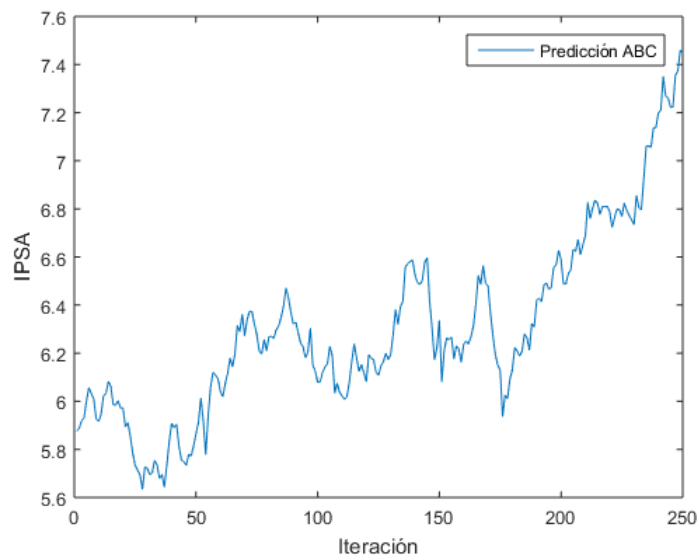


Figura 6.1: Predicción por ventana corrediza de la red neuronal usando el algoritmo ABC para la serie IPSA.

En la figura 6.2 se han superpuesto la predicción del algoritmo ABC en conjunto con la serie real IPSA, se puede observar que en su mayoría la red neuronal está leyendo de buena forma la tendencia y cambios en la serie.

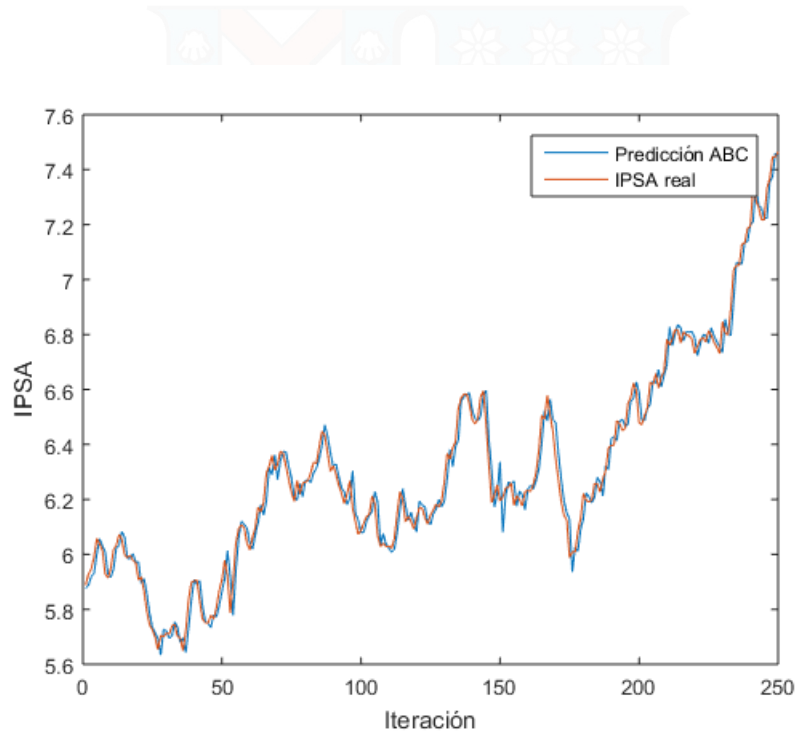


Figura 6.2: Predicción de la red neuronal con ABC e IPSA real superpuestos.

6.1.2. Comparación backpropagation

Al mismo tiempo que fue entrenada la red con el algoritmo ABC se entrenó con backpropagation, obteniendo así los distintos outputs a lo largo de la ventana. En la figura 6.3 se observa el output de backpropagation y el valor de IPSA real. A priori no existe manera de ver qué método predice mejor la serie, es por esto que se calcula el error cuadrático medio de de la serie *out-sample* o fuera de la muestra.

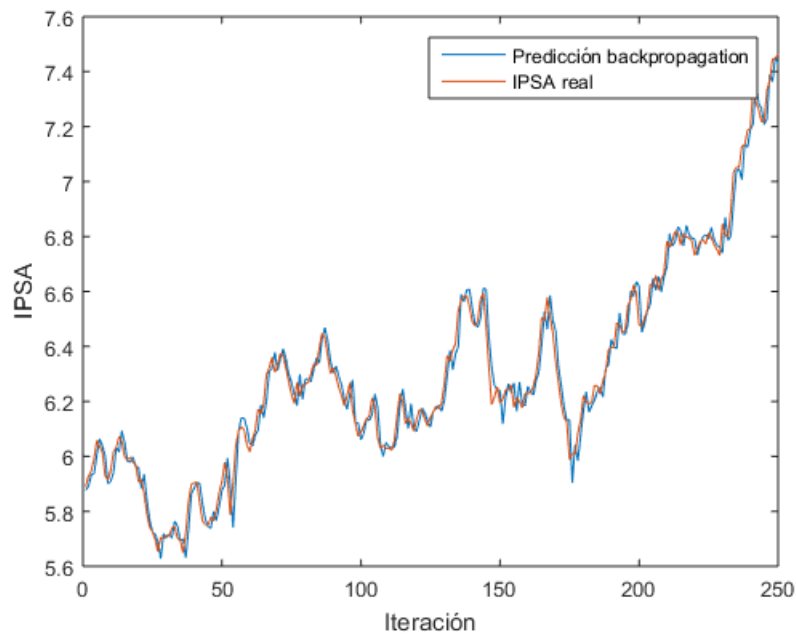


Figura 6.3: Predicción de la red neuronal con backpropagation e IPSA real superpuestos.

6.1.2.1. Error cuadrático medio

En términos del error cuadrático medio, este se calculó con los valores de los precios, a diferencia del MSE de la fase de entrenamiento el cual está en función de la rentabilidad y de la posterior normalización. El error cuadrático medio del algoritmo ABC *out-sample* fue de 0,002901 y el del algoritmo de backpropagation fue de 0,002957, siendo el algoritmo ABC el que predijo de mejor forma la serie. También en la figura 6.4 se puede ver el mse de la fase de entrenamiento a lo largo de las iteraciones de la ventana corregida. De esta figura se puede inferir que el algoritmo ABC tuvo un error cuadrático medio algo más controlado y mayoritariamente menor que el de backpropagation durante las fases de entrenamiento.

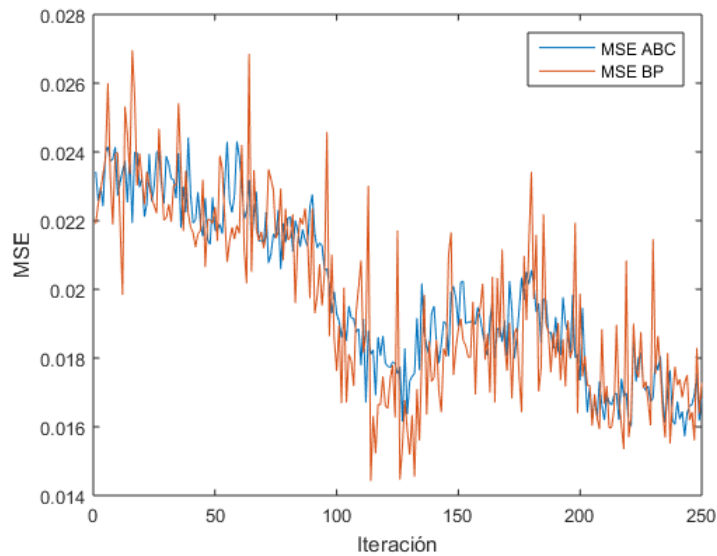


Figura 6.4: Error cuadrático medio del algoritmo ABC y backpropagation en la fase de entrenamiento a lo largo de la ventana corrediza.

6.2. Merval

6.2.1. Predicción

Los resultados para la predicción del algoritmo ABC sobre la serie Merval puede verse en la figura 6.5.

La superposición entre la predicción y los valores reales de la serie puede observarse en la figura 6.6, se evidencia un seguimiento correcto de las propiedades de la serie. Nuevamente para saber cómo se adaptó la red neuronal es necesario calcular el error cuadrático medio respectivo.

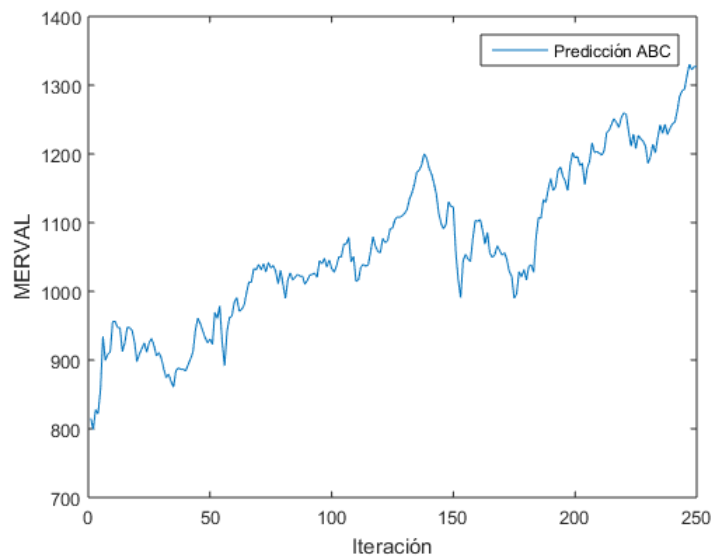


Figura 6.5: Predicción por ventana corrediza de la red neuronal usando el algoritmo ABC para la serie MERVAL.

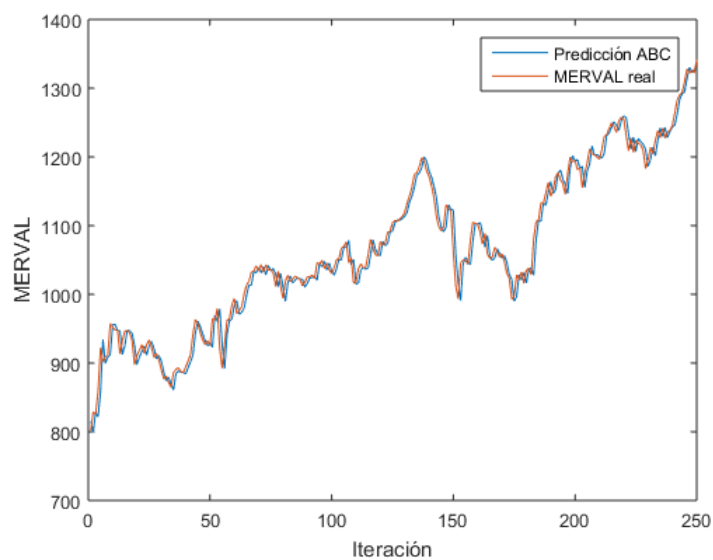


Figura 6.6: Predicción de la red neuronal con ABC y MERVAL real superpuestos.

6.2.2. Comparación backpropagation

En la figura 6.7 se puede ver el ajuste de la red con backpropagation. Nuevamente se hace necesario calcular los errores cuadráticos medios.

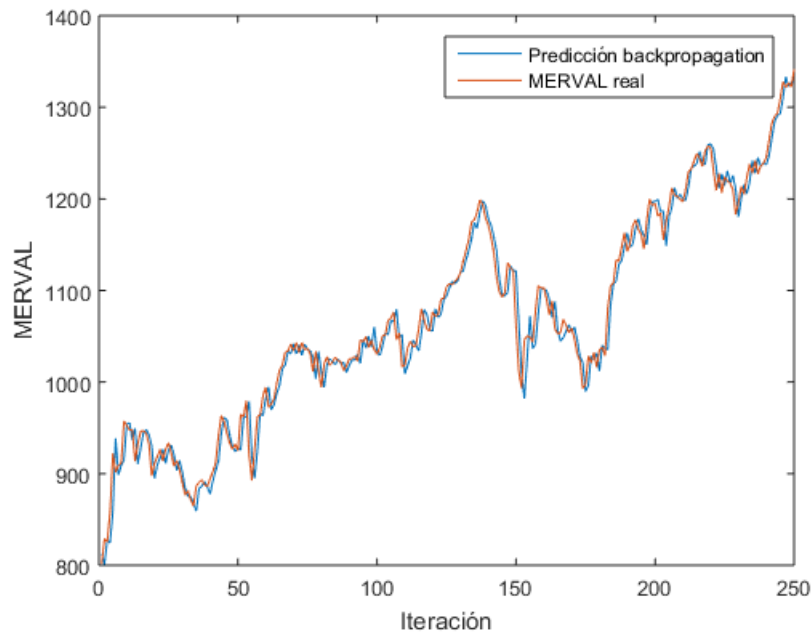


Figura 6.7: Predicción de la red neuronal con backpropagation y Merval real superpuestos.

6.2.2.1. Error cuadrático medio

Para el caso de la red entrenada con el algoritmo de colonias de abejas se tiene un MSE de 303,9617 y con el algoritmo de backpropagation tiene un MSE de 317,3521 siendo mayor que el del algoritmo ABC, también los errores cuadráticos medios *in-sample* tienen este comportamiento

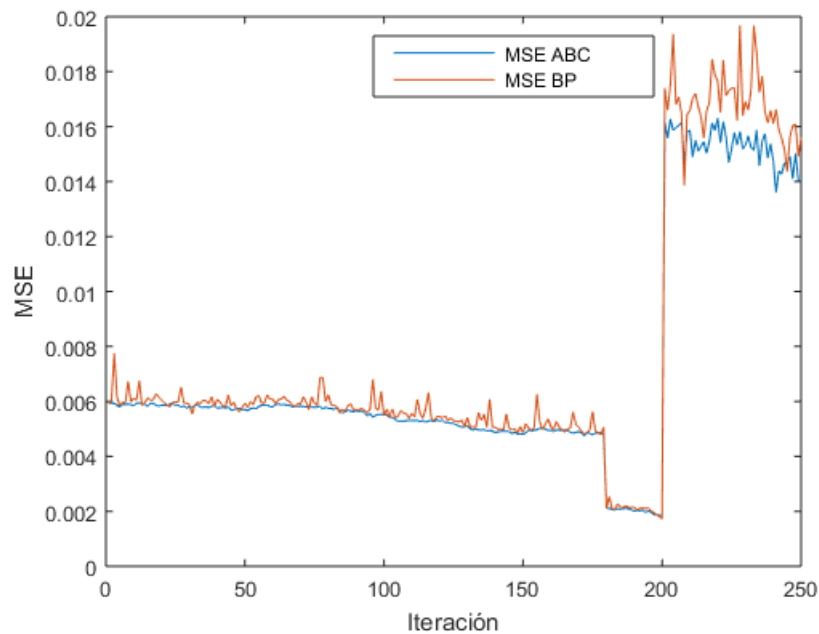


Figura 6.8: Error cuadrático medio del algoritmo ABC y backpropagation en la fase de entrenamiento a lo largo de la ventana corrediza.

6.3. COLCAP

6.3.1. Predicción

Para el mercado colombiano se muestran los resultados de la predicción en la figura 6.9. La superposición entre la serie real y la predicción se puede ver en la figura 6.10 dando cuenta que para este caso la red neuronal también se adapta de buena forma a la forma de la serie manteniendo la forma, a pesar de que puede observarse que la serie presenta varios altos y bajos.

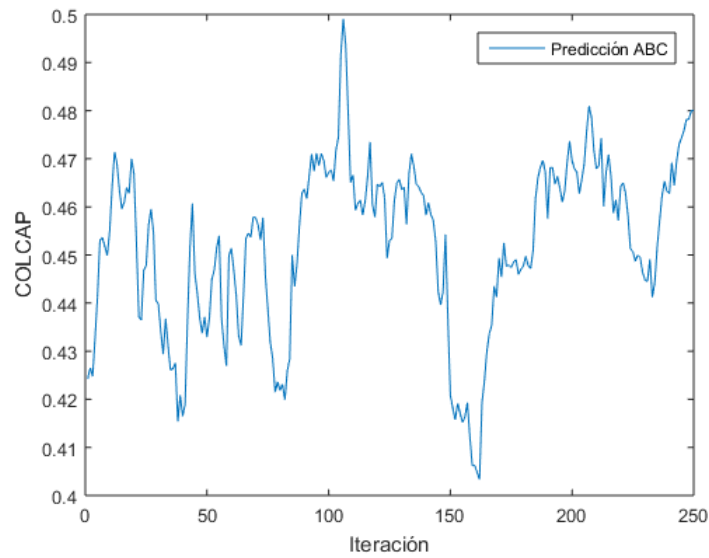


Figura 6.9: Predicción por ventana corrediza de la red neuronal usando el algoritmo ABC para la serie COLCAP.

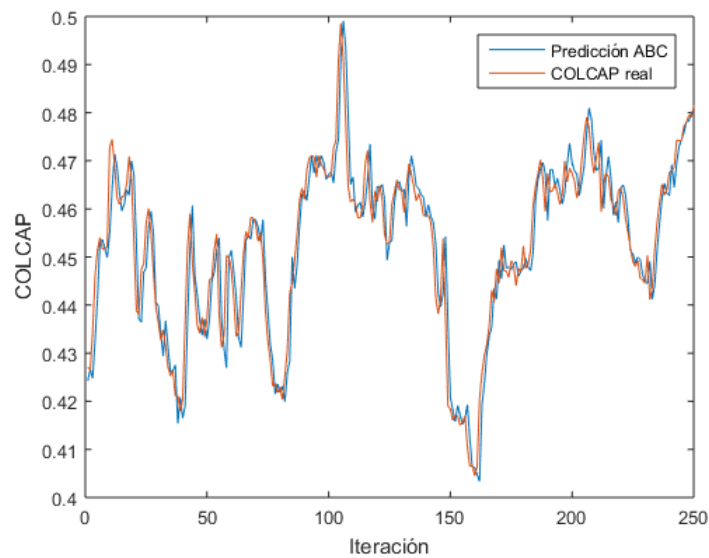


Figura 6.10: Predicción de la red neuronal con ABC y COLCAP real superpuestos.

6.3.2. Comparación backpropagation

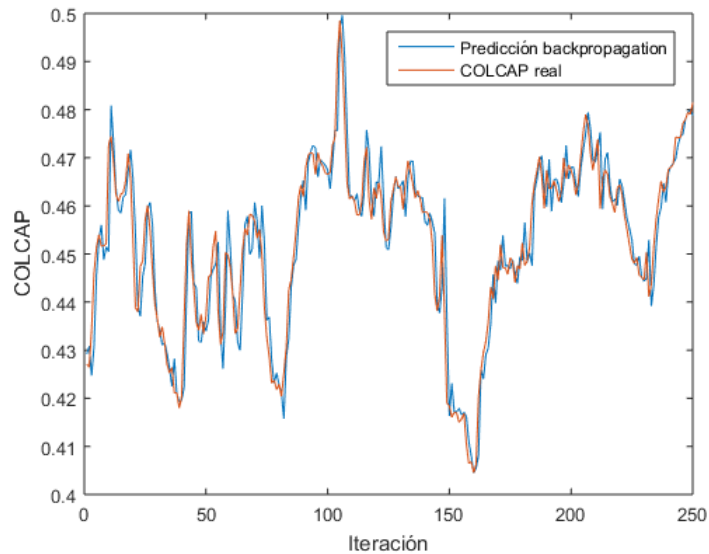


Figura 6.11: Predicción de la red neuronal con backpropagation y COLCAP real superpuestos.

6.3.2.1. Error cuadrático medio

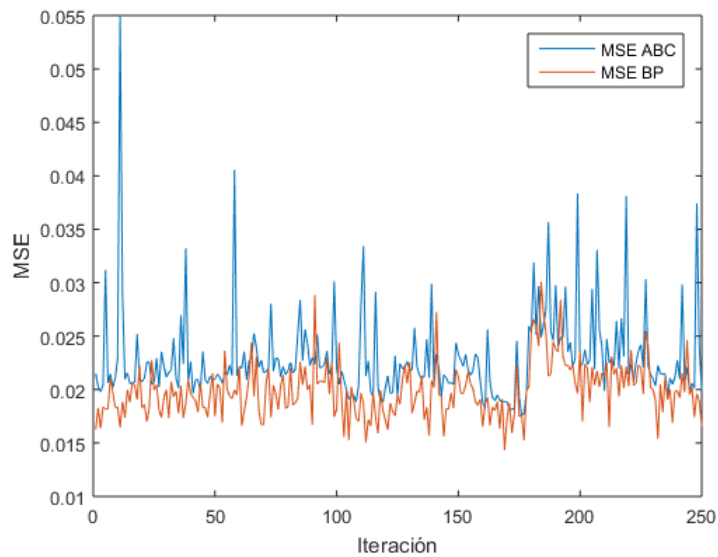


Figura 6.12: Error cuadrático medio del algoritmo ABC y backpropagation en la fase de entrenamiento a lo largo de la ventana corrediza.

La serie entrenada por backpropagation y sus valores reales están graficados en la figura 6.11. El valor para el MSE en el algoritmo ABC fue de $3,7136E - 05$ mientras que para el algoritmo de backpropagation fue de $3,4878E - 05$. Se puede notar que esta ventaja en términos del error cuadrático medio *out-sample* también se evidencia en términos *in-sample* (figura 6.12).

6.4. IBOVESPA

6.4.1. Predicción

Para la serie IBOVESPA la predicción se vuelve algo más ruidosa, a simple vista puede verse que la predicción se escapa de los valores reales .

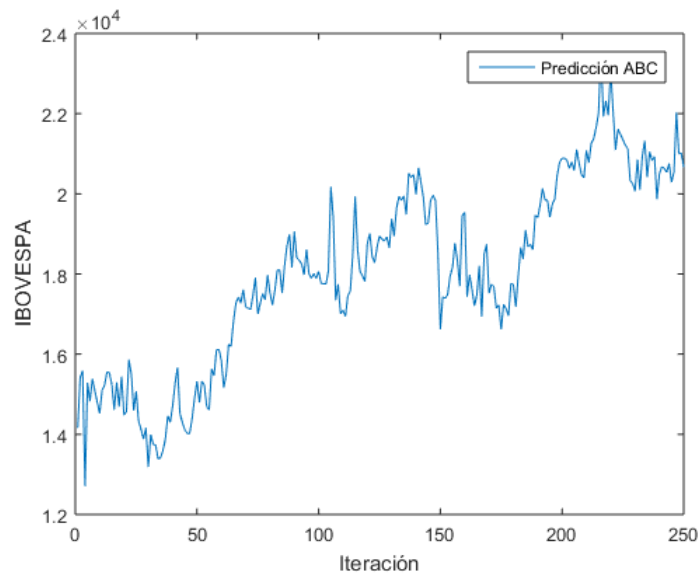


Figura 6.13: Predicción por ventana corrediza de la red neuronal usando el algoritmo ABC para la serie IBOVESPA.

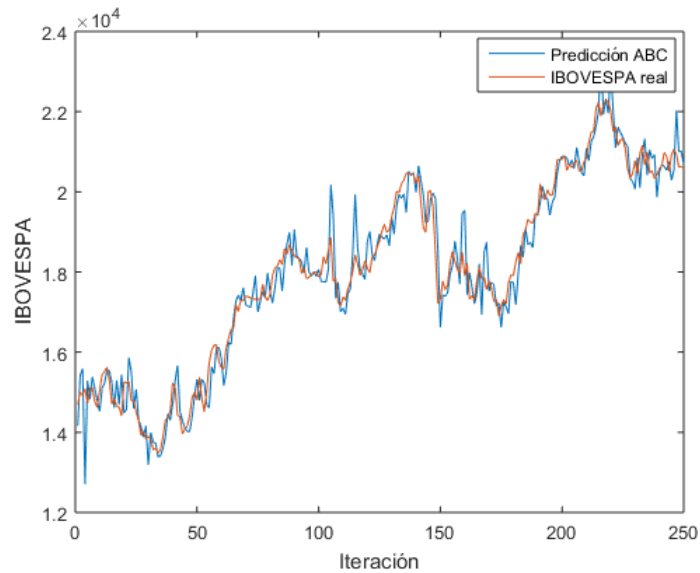


Figura 6.14: Predicción de la red neuronal con ABC y IBOVESPA real superpuestos.

6.4.2. Comparación backpropagation

La predicción con backpropagation se observa más apegada a la serie real. Este comportamiento es coincidente con los valores del MSE.

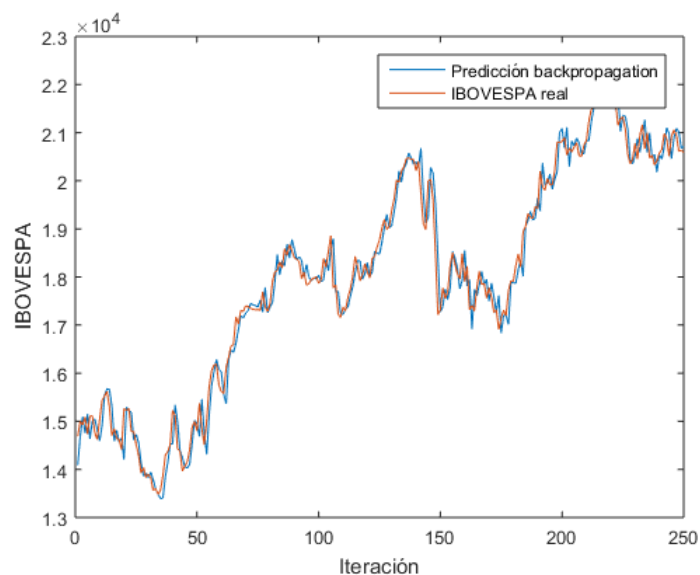


Figura 6.15: Predicción de la red neuronal con backpropagation y IBOVESPA real superpuestos.

6.4.2.1. Error cuadrático medio

El MSE para la red con el algoritmo ABC fue de $2,7151E + 05$ en cambio para el algoritmo de backpropagation fue de $1,1897E + 05$ y durante la fase de entrenamiento se muestran los resultados de la figura 6.16.

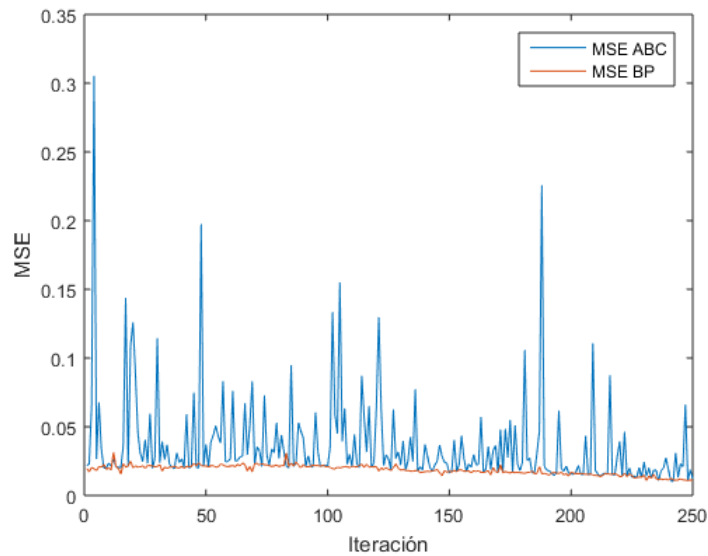


Figura 6.16: Error cuadrático medio del algoritmo ABC y backpropagation en la fase de entrenamiento a lo largo de la ventana corrediza.

6.5. INCA

6.5.1. Predicción

Para la serie INCA la predicción por las primeras iteraciones sigue la forma de la serie real, sin embargo entre la iteración 150 y 200 la red pronostica unos valores que distan de la tendencia de la serie.

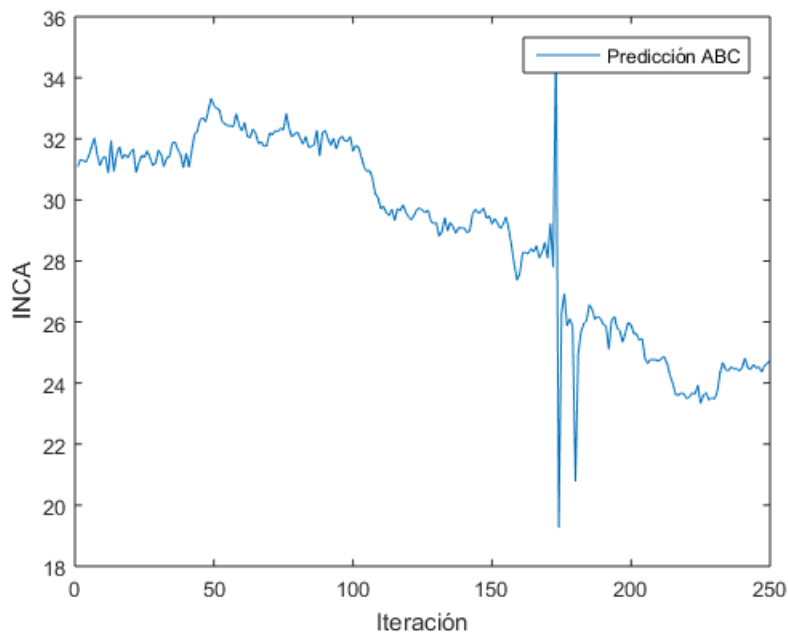


Figura 6.17: Predicción por ventana corrediza de la red neuronal usando el algoritmo ABC para la serie INCA.

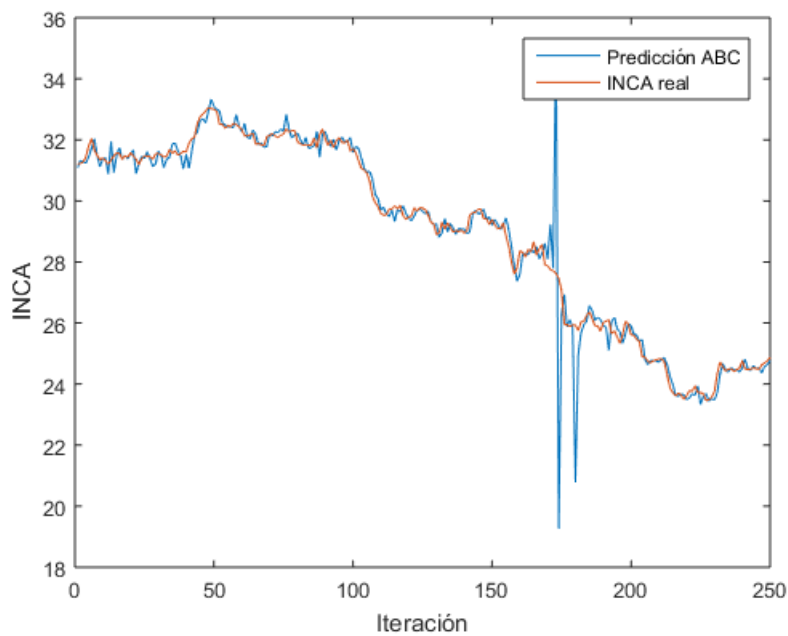


Figura 6.18: Predicción de la red neuronal con ABC y INCA real superpuestos.

6.5.2. Comparación backpropagation

La predicción con backpropagation fue más acertada, a simple vista se puede ver que sigue de mejor forma a la serie y además no tuvo los problemas que tuvo el algoritmo ABC.

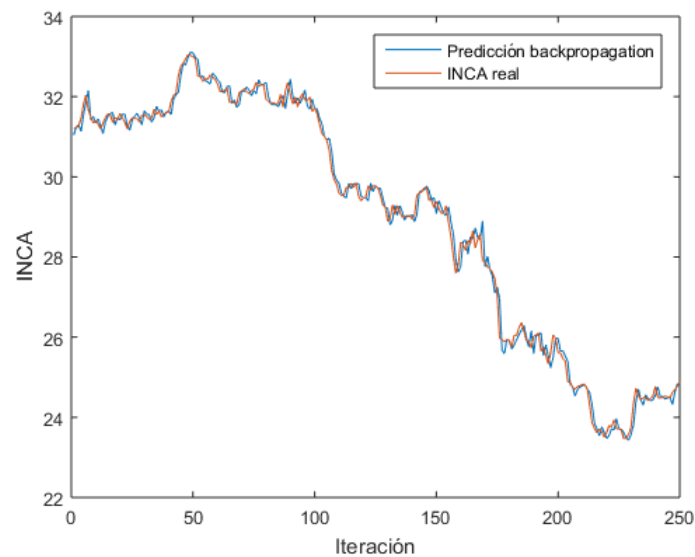


Figura 6.19: Predicción de la red neuronal con backpropagation e INCA real superpuestos.

6.5.2.1. Error cuadrático medio

El error para la red con el algoritmo ABC fue de 0,6944 y para el algoritmo de backpropagation fue de 0,05114. En la figura 6.24 se puede ver que el error de ABC es mayor durante la etapa de entrenamiento.

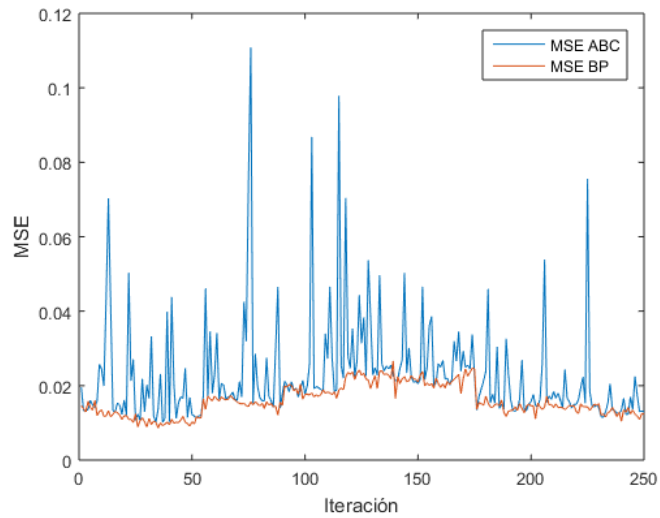


Figura 6.20: Error cuadrático medio del algoritmo ABC y backpropagation en la fase de entrenamiento a lo largo de la ventana corrediza.

6.6. IPC

6.6.1. Predicción

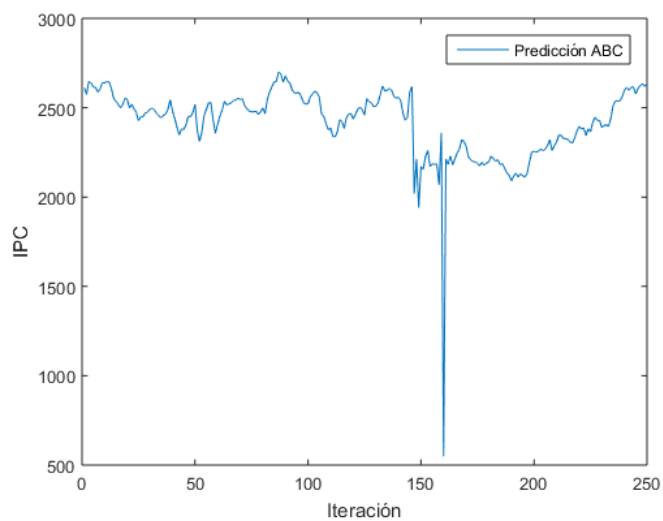


Figura 6.21: Predicción por ventana corrediza de la red neuronal usando el algoritmo ABC para la serie IPC.

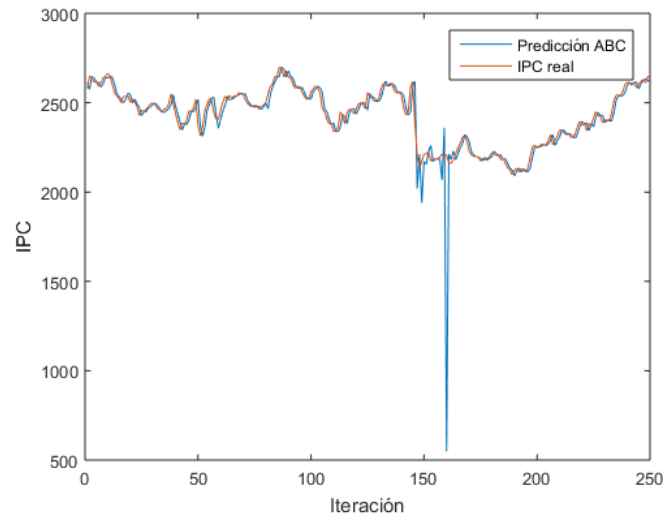


Figura 6.22: Predicción de la red neuronal con ABC y IPC real superpuestos.

Al igual que la serie anterior existen datos que escapan bastante de lo que se espera de la serie.

6.6.2. Comparación backpropagation

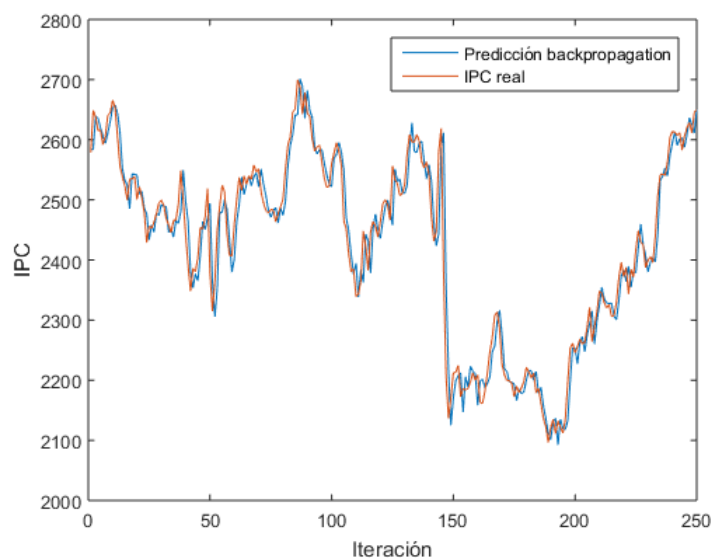


Figura 6.23: Predicción de la red neuronal con backpropagation e IPC real superpuestos.

6.6.2.1. Error cuadrático medio

El valor del MSE de la predicción con el algoritmo ABC fue de $1,2924E+04$ mientras que para el caso de backpropagation fue de $1,3661E+03$ lo cual no se condice con los valores MSE de la etapa de entrenamiento, esto podría ser explicado por el buen rendimiento que presentaba el algoritmo ABC hasta la iteración 147 aproximadamente.

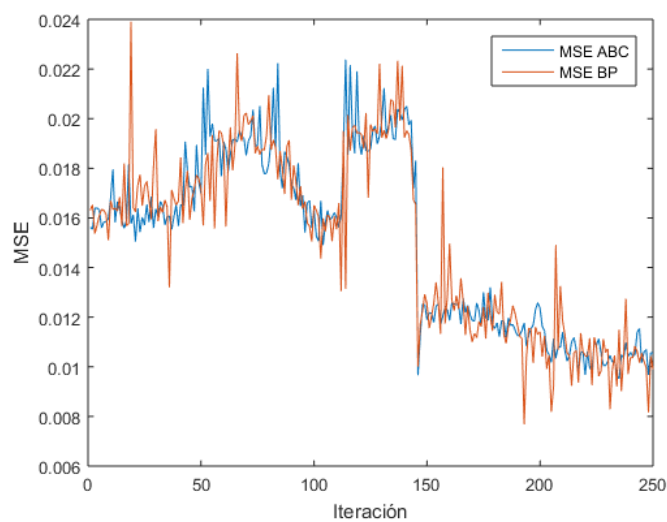


Figura 6.24: Error cuadrático medio del algoritmo ABC y backpropagation en la fase de entrenamiento a lo largo de la ventana corrediza.

A continuación se presenta una tabla resumen con los valores de los errores cuadráticos sobre las distintas series.

Tabla 6.1: Tabla resumen de los errores cuadráticos medios *out-sample*

	MSE BP	MSE ABC
IPSA	0,002957	0,002901
MERVAL	317,3521	303,9617
INCA	0.05114	0.6944
COLCAP	3,4878E-05	3,7136E-05
IPC	1.3661E+03	1.2924E+04
IBOVESPA	1.1897E+05	2.7151E+05

6.7. Resultados combinación ABC-BP

A continuación se presentan los resultados obtenidos al combinar el algoritmo de backpropagation con el algoritmo ABC, inicializando este último con la solución obtenida con backpropagation. Las series probadas para este análisis serán aquellas series que fueron desfavorables en el apartado anterior, es decir **INCA**, **IPC**, **IBOVESPA** y **COLCAP**, las cuales pasarán por una nueva fase de entrenamiento y predicción por backpropagation para luego tomar esa solución y empezar las iteraciones respectivas del algoritmo de colonias de abejas.

6.7.1. INCA

En el caso de la serie INCA pudo verse una mejora con respecto a la predicción de backpropagation, pasando de un valor de MSE 0,04936 a un valor de 0,04822.

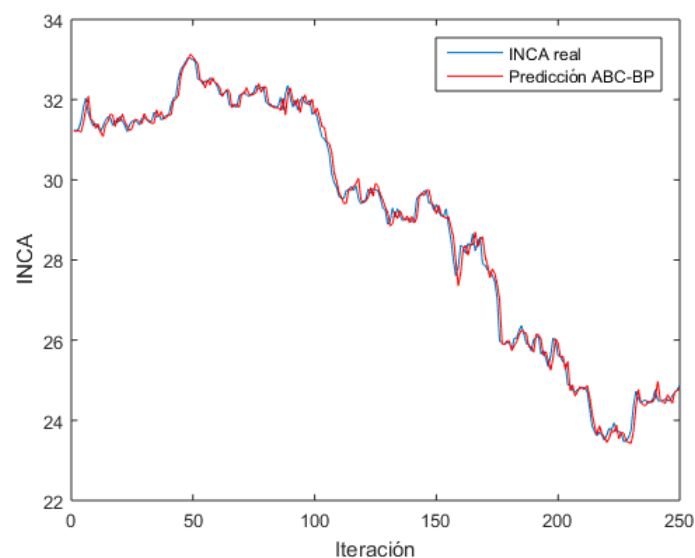


Figura 6.25: Valores de la predicción ABC-BP para la serie INCA

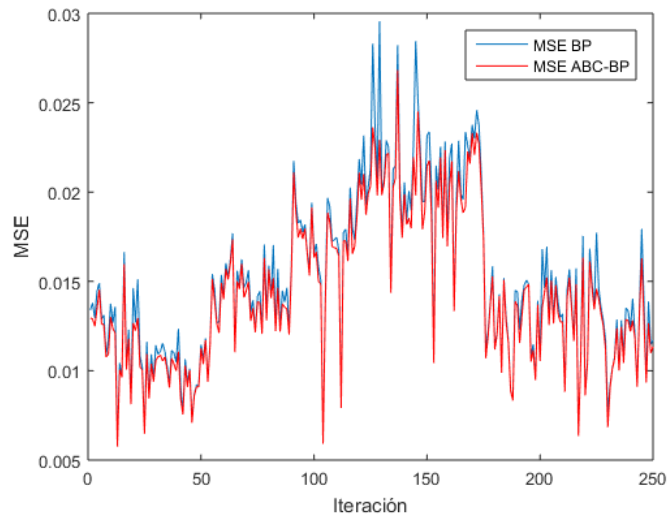


Figura 6.26: Error cuadrático medio del algoritmo ABC-BP y backpropagation en la fase de entrenamiento a lo largo de la ventana corrediza.

6.7.2. IPC

Para la serie IPC el valor del error cuadrático medio descendió de $1,5762E + 03$ para la fase de backpropagation a $1,5675E + 03$ para el entrenamiento con ambos algoritmos combinados.

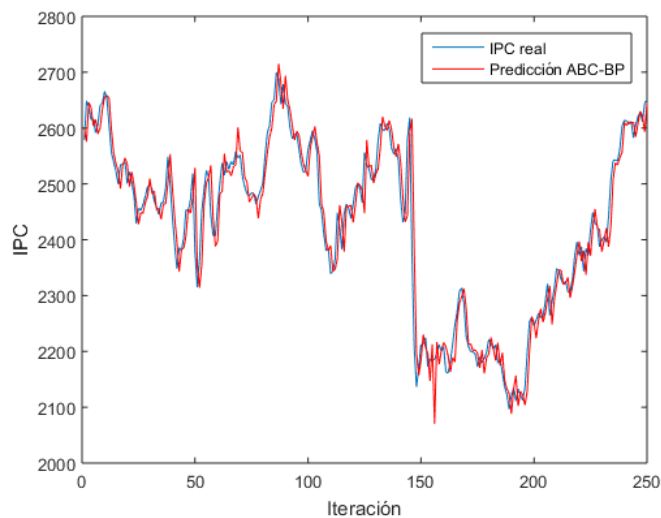


Figura 6.27: Valores de la predicción ABC-BP para la serie IPC

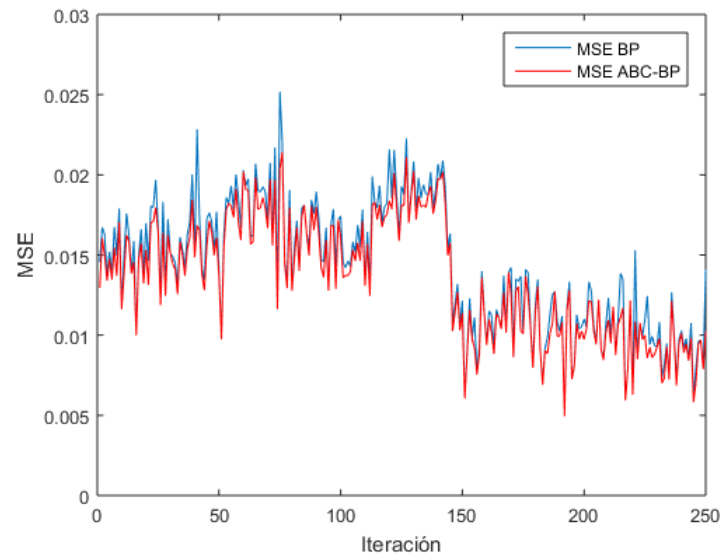


Figura 6.28: Error cuadrático medio del algoritmo ABC-BP y backpropagation en la fase de entrenamiento a lo largo de la ventana corrediza.

6.7.3. COLCAP

Para el mercado colombiano la serie pasó de tener un error de predicción de $3,7579E-05$ a un error de $3,6899E-05$ con los algoritmos combinados.

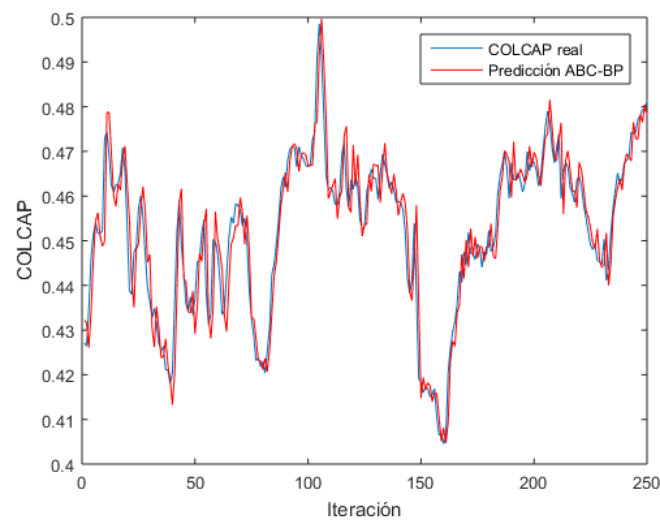


Figura 6.29: Valores de la predicción ABC-BP para la serie COLCAP

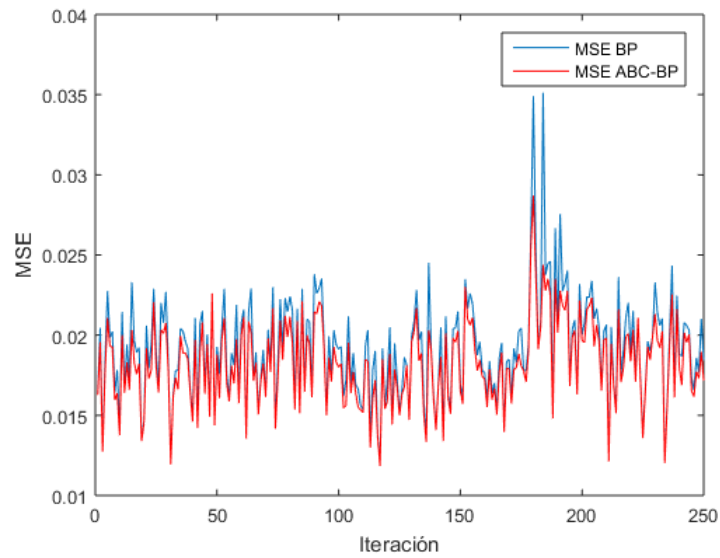


Figura 6.30: Error cuadrático medio del algoritmo ABC-BP y backpropagation en la fase de entrenamiento a lo largo de la ventana corrediza.

6.7.4. IBOVESPA

Distinto a lo anterior, la serie IBOVESPA fue la única que no mejoró su predicción, pasando de un error de $1,1496E + 05$ a un error de $1,2428E + 05$.

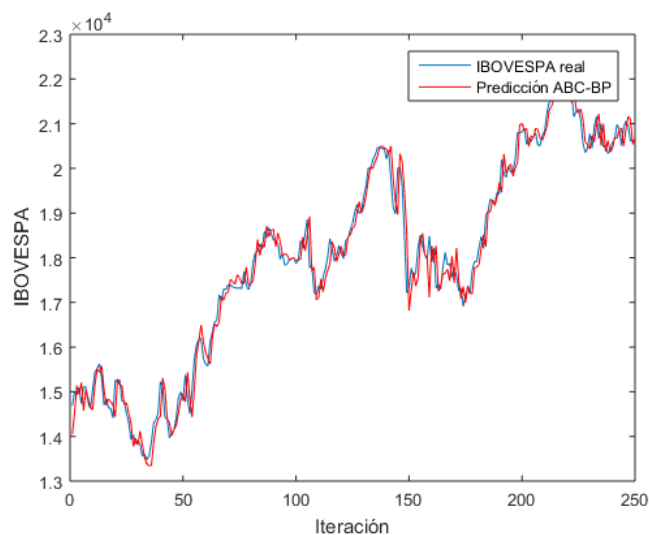


Figura 6.31: Valores de la predicción ABC-BP para la serie IBOVESPA

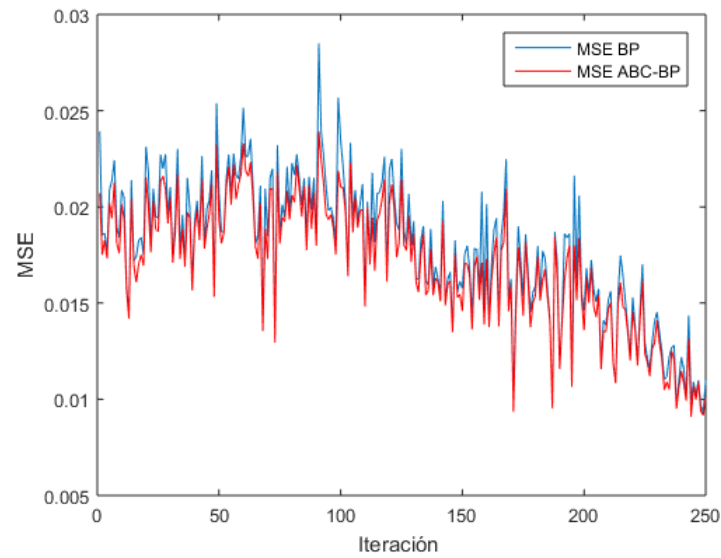


Figura 6.32: Error cuadrático medio del algoritmo ABC-BP y backpropagation en la fase de entrenamiento a lo largo de la ventana corrediza.

En la tabla 6.2 se muestran los valores antes mencionados para los pronósticos

Tabla 6.2: Tabla resumen de los errores cuadráticos medios *out-sample* del algoritmo combinado ABC-BP

	MSE BP	MSE ABC-BP
INCA	0,04936	0.04822
COLCAP	3,7579E-05	3,6899E-05
IPC	15762	15675
IBOVESPA	1.1496E+05	1.2428E+05

7 | Conclusiones y recomendaciones

En este trabajo se testeó la capacidad predictiva de una red neuronal artificial entrenada con el algoritmo de colonias de abejas presentado por [Karaboga \(2005\)](#) para distintas series del mercado latinoamericano durante aproximadamente un año hábil de datos.

Luego de finalizados los experimentos computacionales es posible rescatar que el entrenamiento de una red neuronal por el algoritmo ABC es relativamente sencillo y tiene capacidad tanto para explotar como explorar soluciones dentro de la superficie del error. Sin embargo también hay que decir que requiere de un trabajo computacional importante para ser aplicado a redes neuronales, ya que al tener tantos ciclos anidados hace que se el tiempo se alargue bastante más de lo que demora el algoritmo tradicional de entrenamiento, cosa que no sucede cuando el algoritmo se usa para optimización global de funciones, esto porque en una red neuronal cada vez tendremos que ir actualizando la red en pos de evaluar la función objetivo: el MSE o error cuadrático medio.

En términos del MSE se van a distinguir dos medidas: el MSE *in-sample* o dentro de la muestra y el MSE *out-sample* o fuera de la muestra, el primero hará referencia al error alcanzado durante la etapa de entrenamiento y el segundo se referirá al error alcanzado

una vez que se hicieron las predicciones, es decir, cuando la red neuronal predice un valor nunca antes visto.

Gracias al método de la ventana corrediza fue posible simular un año hábil de pronósticos y los resultados arrojaron que para los casos de la serie Merval e IPSA el algoritmo ABC se desempeñaba de mejor forma que el algoritmo backpropagation para predecir. Además también se puede notar que en las figuras 6.4 y 6.8 la superioridad del algoritmo ABC también se da en términos del error de entrenamiento, por lo que el método utilizado está modelando de mejor manera; ergo, prediciendo mejor.

Para los casos de las series IBOVESPA y COLCAP se puede contrastar que el error *out-sample* de la predicción es menor en el caso de backpropagation y existe una coherencia con los valores *in-sample* en donde el desempeño de ABC es menor quedándose a veces muy por encima que su competencia, para los mismos parámetros considerados (figuras 6.12 y 6.16).

El punto distinto fue el resultado que se obtuvo al probar las series INCA e IPC. Para ambas series entre la iteración 140 y 200 los valores se disparaban, lo cual era bastante dudoso ya que antes de eso la red estuvo prediciendo bien o al menos no tan alejado y después de eso siguió prediciendo bien, por lo que aparentemente algo pasaba en esas iteraciones que hacía que la red fallara. Por lo tanto, descartando que fuera un problema de la red en sí, o del script (ya que antes de eso se desempeñaba bien) se llegó a la conclusión de que podría ser: problema de datos o problema del algoritmo ABC que no estaba convergiendo suficiente. Si el problema era del algoritmo ABC bastaba con agregar una mayor población de abejas y/o más iteraciones. Si el problema era de los datos podría

tratarse de la forma en la cual los datos se introducen a la red (scaling 5.2).

Dado que en el mismo rango de iteraciones continuaba sucediendo lo mismo, se decidió aumentar los parámetros de ABC y cambiar el tipo de normalización por:

$$x'_{normalizado} = \frac{x - \bar{x}}{s_x} \quad (7.1)$$

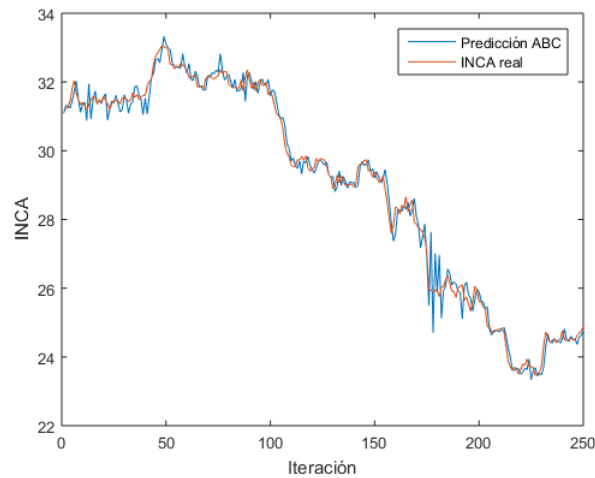


Figura 7.1: Nuevo pronóstico INCA con cambio en los parámetros

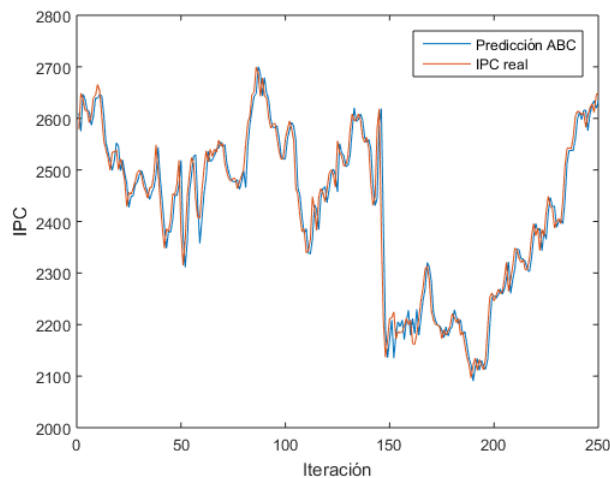


Figura 7.2: Nuevo pronóstico IPC con cambio en los parámetros

Si bien la predicción no mejoró de manera relevante, desaparecieron aquellos datos que se habían disparado. En su artículo, [Akpinar et al. \(2017\)](#) no utiliza la normalización 5.2 (scaling) si no que lleva los datos a valores entre [0,1;0,9] para evitar que la red tenga que lidiar con datos que tengan valor cero.

Por último, aprovechando las características del algoritmo de colonias de abejas se decide inicializar este algoritmo con la solución que entrega backpropagation, con la idea de mejorar el pronóstico de las cuatro series en las que el método tradicional se ejecuto mejor que el método propuesto, es decir: IBOVESPA, COLCAP, INCA e IPC. Una vez inicializado el algoritmo, se **asegura** que el **modelado** será igual o mejor (en términos de error cuadrático medio) que lo obtenido por el algoritmo de backpropagation. Los resultados arrojaron que INCA, COLCAP e IPC mejoraron sus pronósticos con este método a lo largo de un año hábil lo cual reafirma lo anteriormente dicho: un mejor modelo debería predecir mejor. Sin embargo para el caso de IBOVESPA el pronóstico no mejoró, a pesar de tener un modelo a lo largo de las iteraciones igual o mejor a lo entregado por backpropagation. Esto podría explicarse por el *overfitting* o el sobreajuste que pueda haber experimentado el modelo.

El sobreajuste de un modelo es similar a lo que sucede cuando ajustamos un modelo a una nube de puntos de una serie de datos y obtenemos una función que pasa por todos los puntos con un $R^2 = 0,9999$: el modelo **particulariza** y pierde la capacidad de **generalizar**, en el contexto de las redes neuroales, la red está **memorizando** y no está **aprendiendo**, algo que justamente el algoritmo de backpropagation trata de evitar en su fase de validación y que podría explicar el por qué a pesar de tener errores cuadráticos medios menores a lo

largo de las distintas iteraciones se tiene un pronóstico con un error mayor.

Así, el algoritmo de colonias de abejas ha probado un buen desempeño en labores de modelado y predicción de redes neuronales e incluso mostró sinergia al ser combinado con el algoritmo de backpropagation. A pesar de esto hay que tener especial cuidado con los parámetros del algoritmo que pueden depender tanto de la serie como de la arquitectura de la red (cantidad de parámetros a optimizar) y también con el tratamiento de los datos para evitar situaciones como las de las series INCA e IPC en este estudio.

Referencias

- Akpinar, Mustafa; Adak, M Fatih; y Yumusak, Nejat (2017). Day-ahead natural gas demand forecasting using optimized abc-based neural network with sliding window technique: The case study of regional basis in turkey. *Energies*, 10(6), 781. 2.3, 5.3, 7
- Bányász, Csilla (1995). *Adaptive systems in control and signal processing 1995: a postscript volume from the Fifth IFAC Symposium, Budapest, Hungary, 14-16 June 1995*. Pergamon. 3.2.1
- Bollerslev, Tim (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3), 307 – 327. 2.1.2.2
- Bonabeau, Eric; Dorigo, Marco; y Theraulaz, Guy (1999). *Swarm intelligence: from natural to artificial systems*. Número 1. Oxford university press. 3.2.1
- Campbell, John Y y Andrew, W (1997). Lo, and a. craig mackinlay, 1997. *The econometrics of financial markets*, 2. 5.2.1
- Carriero, A; Kapetanios, G; y Marcellino, M (2009). Forecasting exchange rates with a large Bayesian VAR. *International Journal of Forecasting*, 25(2), 400–417. 2.2.2
- Devore, Jay L. (California Polytechnic State University) (2008). *Probability and Statistics for Engineering and the Sciences*. Cengage, 7 edición. 2.1.1
- Díaz, David; Theodoulidis, Babis; y Dupouy, Carlos (2016). Modelling and forecasting interest rates during stages of the economic cycle : A knowledge-discovery approach R. *Expert Systems With Applications*, 44, 245–264. 2.2.3
- Domínguez, Omar Cetina (2009). An adaptation of the scout bee behavior in the artificial bee colony algorithm to solve constrained optimization problems. *Master of Computer Science*. 3.2.2
- Engle, Robert F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica*, 50(4), 987–1007. 2.1.2.2
- Gujarati, D. y Porter, D. (2010). *Econometría*. New York: Mc Graw Hill, quinta edición. 1, 2.1.1

- Hsieh, Tsung-Jung; Hsiao, Hsiao-Fen; y Yeh, Wei-Chang (2011). Forecasting stock markets using wavelet transforms and recurrent neural networks: An integrated system based on artificial bee colony algorithm. *Applied soft computing*, 11(2), 2510–2525. 2.3
- Javedani, Hossein; Enayatifar, Rasul; Hisyam, Muhammad; y Mahmud, Maqsood (2016). A hybrid model based on differential fuzzy logic relationships and imperialist competitive algorithm for stock market forecasting. *Applied Soft Computing Journal*, 40, 132–149. 2.2.1
- Ju, Y. J.; Kim, C. E.; y Shim, J. C. (1997). Computers ind. Engng. *Computers ind. Engng*, 33(1), 561–564. 2.2.3
- Kalekar, Prajakta S (2004). Time series forecasting using holt-winters exponential smoothing. *Kanwal Rekhi School of Information Technology*, 4329008, 1–13. 2.1.2.2
- Karaboga, D. (2005). An idea based on Honey Bee Swarm for Numerical Optimization. *Technical Report TR06, Erciyes University*, (TR06), 10. 1, 3.2.1, 3.2.2, 3.2.2, 7
- Karaboga, Dervis y Akay, Bahriye (2009). A comparative study of artificial bee colony algorithm. *Applied mathematics and computation*, 214(1), 108–132. 3.2.2
- Karaboga, Dervis; Akay, Bahriye; y Ozturk, Celal (2007). Artificial bee colony (abc) optimization algorithm for training feed-forward neural networks. *MDAI*, 7, 318–319. 5.3
- Karaboga, Dervis y Basturk, Bahriye (2007). Artificial bee colony (abc) optimization algorithm for solving constrained optimization. In *Problems, LNCS: Advances in Soft Computing: Foundations of Fuzzy Logic and Soft Computing*, Springer-Verlag, IFSA (2007: Citeseer. 3.2.2
- Karaboga, Dervis; Gorkemli, Beyza; Ozturk, Celal; y Karaboga, Nurhan (2014). A comprehensive survey: artificial bee colony (abc) algorithm and applications. *Artificial Intelligence Review*, 42(1), 21–57. 2.3
- Kristanpoller, Werner y Liberona, Carolina (2010). Comparación de modelos de predicción de retornos accionarios en el Mercado Accionario Chileno:. 2.2.1
- Kunche, Prajna y Reddy, KVVS (2016). *Metaheuristic Applications to Speech Enhancement*. Springer. 3.2.1
- Lahmiri, Salim (2016). Intraday stock price forecasting based on variational mode decomposition. *Journal of Computational Science*, 12, 23–27. 2.2.1
- Mallo, Paulino E; Artola, María Antonia; Galante, Marcelo Javier; Martínez, Diego; Pascual, Mariano Enrique; y Morettini, Mariano (2003). Aplicación del método delphi a las decisiones financieras en situaciones de incertidumbre. 2.1
- Marsalli, Michael (2006). Mcculloch pitts neurons. 3.1.3.1, 3.1

- Maya, C.; Torres, G.; et al. (2005). Las caminatas aleatorias no son de este mundo. teoría y revisión bibliográfica sobre evidencia empírica. 2.2.1
- McCulloch, W. y Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4), 115–133. 1, 3.1.3, 3.1.3.1, 3.1.3.1
- Méndez, J. y Marín, R. (2013). *Inteligencia artificial: Métodos, técnicas y aplicaciones*. Murcia: Mc Graw Hill. 1, 3.3, 3.4, 3.5, 3.6, 3.7
- Mostafa, Mohamed M (2010). Expert Systems with Applications Forecasting stock exchange movements using neural networks : Empirical evidence from Kuwait. *Expert Systems With Applications*, 37(9), 6302–6309. 2.2.1
- Pan, Quan-Ke; Tasgetiren, M Fatih; Suganthan, Ponnuthurai N; y Chua, Tay Jin (2011). A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem. *Information sciences*, 181(12), 2455–2468. 3.2.2
- Panda, Chakradhara y Narasimhan, V (2007). Forecasting exchange rate better with artificial neural network. 29, 227–236. 2.2.2
- Rao, R Srinivasa; Narasimham, SVL; y Ramalingaraju, M (2008). Optimization of distribution network configuration for loss reduction using artificial bee colony algorithm. *International Journal of Electrical Power and Energy Systems Engineering*, 1(2), 116–122. 3.2.2
- Rosenblatt, Frank (1957). *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory. 3.1.4
- Ruiz, R. y Jiménez, J. (2000). Las redes neuronales en su aplicación a las finanzas. *Banca y Finanzas*, (54), 19–26. 1
- Sanjeev, Gupta y Sachin, Kashyap (2016). Modelling Volatility and Forecasting of Exchange Rate of British Pound Sterling and Indian Rupee. *Journal of Modelling in Management*, 11(2). 2.2.2
- Santana, J. (2006). Predicción de series temporales con redes neuronales: Una aplicación a la inflación Colombiana. *Revista Colombiana de Estadística*, 29(1), 77–92. 1
- Shah, Habib; Ghazali, Rozaida; y Nawi, Nazri Mohd (2011). Using artificial bee colony algorithm for mlp training on earthquake time series data prediction. *arXiv preprint arXiv:1112.4628*. 2.3, 5.3
- Villada, F.; Muñoz, N.; y García, E. (2012). Aplicación de las Redes Neuronales al Pronóstico de Precios en el Mercado de Valores. *Informacion Tecnologica*, 23(4), 11–20. 1