UNIVERSIDAD NACIONAL DE CHIMBORAZO

VICERRECTORADO DE INVESTIGACIÓN, VINCULACIÓN Y POSGRADO

DIRECCIÓN DE POSGRADO

"MODELOS MARKOV SWITCHING Y REDES NEURONALES RECURRENTES PARA PREDECIR SERIES TEMPORALES DE TASAS DE CAMBIO"

TRABAJO DE GRADUACIÓN PARA OPTAR AL TÍTULO DE:

MAGÍSTER EN MATEMÁTICA APLICADA CON MENCIÓN EN MATEMÁTICA COMPUTACIONAL

AUTOR:

Inti Israel Becerra Loaiza

TUTOR:

Saba Rafael Infante Quirpa, PhD.

Riobamba – Ecuador, 2024

Certificación del Tutor

Certifico que el presente trabajo de titulación denominado: **"Modelos markov switching y redes neuronales recurrentes para predecir series temporales de tasas de cambio"**, ha sido elaborado por Inti Israel Becerra Loaiza, el mismo que ha sido orientado y revisado con el asesoramiento permanente de mi persona en calidad de Tutor. Así mismo, refrendo que dicho trabajo de titulación ha sido revisado por la herramienta antiplagio institucional; por lo que certifico que se encuentra apto para su presentación y defensa respectiva.

Es todo cuanto puedo informar en honor a la verdad.

Riobamba, 18 de marzo de 2024

Saba Rafael Infante Quirpa, PhD.
TUTOR

Declaración de Autoría y Cesión de Derechos

Yo, **Inti Israel Becerra Loaiza**, con número único de identificación **110391262-0**, declaro y acepto ser responsable de las ideas, doctrinas, resultados y lineamientos alternativos realizados en el presente trabajo de titulación denominado: "Modelos markov switching y redes neuronales recurrentes para predecir series temporales de tasas de cambio" previo a la obtención del grado de Magíster en Matemática Aplicada con mención en Matemática Computacional.

- Declaro que mi trabajo investigativo pertenece al patrimonio de la Universidad Nacional de Chimborazo de conformidad con lo establecido en el artículo 20 literal j) de la Ley Orgánica de Educación Superior LOES.
- Autorizo a la Universidad Nacional de Chimborazo que pueda hacer uso del referido trabajo de titulación y a difundirlo como estime conveniente por cualquier medio conocido, y para que sea integrado en formato digital al Sistema de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor, dando cumplimiento de esta manera a lo estipulado en el artículo 144 de la Ley Orgánica de Educación Superior LOES.

Riobamba, marzo 2024

Inti Israel Becerra Loaiza
N.U.I. 110391262-0

Actas de superación de observaciones

Dirección de
Posgrado
VICERRECTORADO DE INVESTIGACIÓN,
VINCULACIÓN Y POSGRADO

SGC
SISTEMA DE GESTIÓN DE LA CALIDAD

Riobamba, 18 de marzo de 2024

ACTA DE SUPERACIÓN DE OBSERVACIONES

En calidad de miembro del Tribunal designado por la Comisión de Posgrado, CERTIFICO que una vez revisado el Proyecto de Investigación y/o desarrollo denominado **"MODELOS MARKOV SWITCHING Y REDES NEURONALES RECURRENTES PARA PREDECIR SERIES TEMPORALES DE TASAS DE CAMBIO"**, dentro de la línea de investigación de **Ingeniería Informática, presentado por el maestrante Becerra Loaiza Inti Israel**, portador de la CI. 1103912620, del programa de **Maestría en Matemática Aplicada con mención en Matemática Computacional,** cumple al 100% con los parámetros establecidos por la Dirección de Posgrado de la Universidad Nacional de Chimborazo.

Es todo lo que podemos certificar en honor a la verdad.

Atentamente,

Saba Rafael Infante Quirpa

MIEMBRO DEL TRIBUNAL

Campus La Dolorosa
Av. Eloy Alfaro y 10 de Agosto
Teléfono (593-3) 373-0880, ext. 2002

Dirección de
Posgrado
VICERRECTORADO DE INVESTIGACIÓN,
VINCULACIÓN Y POSGRADO

SGC
SISTEMA DE GESTIÓN DE LA CALIDAD

Riobamba, 18 de marzo de 2024

ACTA DE SUPERACIÓN DE OBSERVACIONES

En calidad de miembro del Tribunal designado por la Comisión de Posgrado, CERTIFICO que una vez revisado el Proyecto de Investigación y/o desarrollo denominado **"MODELOS MARKOV SWITCHING Y REDES NEURONALES RECURRENTES PARA PREDECIR SERIES TEMPORALES DE TASAS DE CAMBIO"**, dentro de la línea de investigación de **Ingeniería Informática, presentado por el maestrante Becerra Loaiza Inti Israel**, portador de la CI. 1103912620, del programa de **Maestría en Matemática Aplicada con mención en Matemática Computacional,** cumple al 100% con los parámetros establecidos por la Dirección de Posgrado de la Universidad Nacional de Chimborazo.

Es todo lo que podemos certificar en honor a la verdad.

Atentamente,

Isidro Rafael Amaro

MIEMBRO DEL TRIBUNAL

Dirección de
Posgrado
VICERRECTORADO DE INVESTIGACIÓN,
VINCULACIÓN Y POSGRADO

SGC
SISTEMA DE GESTIÓN DE LA CALIDAD

Riobamba, 18 de marzo de 2024

ACTA DE SUPERACIÓN DE OBSERVACIONES

En calidad de miembro del Tribunal designado por la Comisión de Posgrado, CERTIFICO que una vez revisado el Proyecto de Investigación y/o desarrollo denominado **"MODELOS MARKOV SWITCHING Y REDES NEURONALES RECURRENTES PARA PREDECIR SERIES TEMPORALES DE TASAS DE CAMBIO"**, dentro de la línea de investigación de **Ingeniería Informática, presentado por el maestrante Becerra Loaiza Inti Israel**, portador de la CI. 1103912620, del programa de **Maestría en Matemática Aplicada con mención en Matemática Computacional,** cumple al 100% con los parámetros establecidos por la Dirección de Posgrado de la Universidad Nacional de Chimborazo.

Es todo lo que podemos certificar en honor a la verdad.

Atentamente,

Firmado electrónicamente por: GUILLERMO EDVIN MACHADO SOTOMAYOR

Guillermo Edvin Machado Sotomayor

MIEMBRO DEL TRIBUNAL

Certificados de similitudes

Dirección de Posgrado
VICERRECTORADO DE INVESTIGACIÓN, VINCULACIÓN Y POSGRADO

en movimiento

Riobamba, **18 de marzo de 2024**

CERTIFICADO

De mi consideración:

Yo Saba Rafael Infante Quirpa, certifico que Inti Israel Becerra Loaiza con cédula de identidad No. 1103912620 estudiante del programa de maestría en Matemática Aplicada con mención en Matemática Computacional , cohorte Primera (2021-2022), presentó su trabajo de titulación bajo la modalidad de Proyecto de titulación con componente de investigación aplicada/desarrollo denominado: MODELOS MARKOV SWITCHING Y REDES NEURONALES RECURRENTES PARA PREDECIR SERIES TEMPORALES DE TASAS DE CAMBIO, el mismo que fue sometido al sistema de verificación de similitud de contenido Turnitin identificando el 6% de similitud en el texto.

Es todo en cuanto puedo certificar en honor a la verdad.

Atentamente,

Saba Rafael Infante Quirpa
CI: 1757134943

Adj.-
- Resultado del análisis de similitud

Campus "La Dolorosa" | Av. Eloy Alfaro y 10 de Agosto | Teléfonos: (593-3) 3730910 - Ext.: 2101-2217-2005

Dirección de Posgrado
VICERRECTORADO DE INVESTIGACIÓN,
VINCULACIÓN Y POSGRADO

en movimiento

Riobamba, 19 de marzo de 2024

CERTIFICADO

De mi consideración:

Yo Kléber Augusto Jaramillo Galarza coordinador académico, certifico que Inti Israel Becerra Loaiza con cédula de identidad No. 1103912620 estudiante del programa de maestría en Matemática Aplicada con mención en Matemática Computacional, cohorte Primera (2021-2022), presentó su trabajo de titulación bajo la modalidad de Proyecto de titulación con componente de investigación aplicada/desarrollo denominado: **MODELOS MARKOV SWITCHING Y REDES NEURONALES RECURRENTES PARA PREDECIR SERIES TEMPORALES DE TASAS DE CAMBIO**, el mismo que fue sometido al sistema de verificación de similitud de contenido TURNITIN identificando el 6% de similitud en el texto, el cual fue realizado por su tutor.

Es todo en cuanto puedo certificar en honor a la verdad.

Atentamente,

Ing. Kléber Augusto Jaramillo Galarza, MSc
CI: 0703748939

Adj.-
- Informe tutor académico
- Resultado del análisis de similitud

Agradecimiento

Quiero agradecer a mis profesores y compañeros de clase que, con sus conocimientos, sus sugerencias a lo largo de este proceso fueron fundamentales para alcanzar los objetivos propuestos.

Un agradecimiento especial al profesor Saba Infante, por su continuo apoyo en esta investigación. Su motivación, entusiasmo e inmenso conocimiento fueron mi motivación para ser perseverante en cada paso de esta tesis.

Por último, quiero agradecer a mis familiares y seres queridos por apoyarme siempre en lo que me propongo.

Gracias a todos.

Dedicatoria

A mis padres Julio y Teresa

A mi esposa Mónica.

A mis hermanos Karina, Stalin y Vanesa.

A mi sobrino Julián.

Índice General

Índice de Figuras

Índice de Algoritmos

Índice de Tablas

Resumen

En la mayoría de los escenarios prácticos de la vida real tales como negocios, finanzas, comercio minorista, energía, economía, etc., los datos que se obtienen de series temporales reflejan cambios temporales no estacionarios. Existen algunos modelos que se utilizan para realizar predicciones de este tipo de información como son los Modelos Markov con Cambio de Estado (switching) que se centran en capturar cambios estructurales en los datos a través de estados discretos y las Redes Neuronales Recurrentes que se centran en modelar dependencias temporales en los datos secuenciales utilizando conexiones recurrentes entre las unidades de la red. En este trabajo, se realiza un modelo hibrido utilizando modelos ocultos de Markov y redes neuronales recurrentes para predecir series de tiempo. Para ello, se realiza una red neuronal recurrente y se agrega el modelo oculto de Markov para que realice la actualización de los cambios de régimen. En la implementación del modelo, se utiliza datos de varias tasas de cambio de divisas (EUR/USD, GBP/USD, USD/CAD, USD/CHF), que van desde 1 de enero de 2020 al 1 de septiembre del 2022. Además, se utilizó el error cuadrático medio (MSE) y la raíz del error cuadrático medio (RMSE) como medidas de bondad de ajuste, obteniendo errores bajos, lo cual indica la calidad de la estimación del modelo.

Palabras claves: *Modelos Markov Switching, Modelos de Markov Ocultos, Redes Neuronales Recurrentes.*

ABSTRACT

In most practical real-life scenarios such as business, finance, retail, energy, economics, etc., time series data reflect non-stationary temporal changes. Some models are used to make predictions of this type of information, such as Markov Switching Models that focus on capturing structural changes in data through discrete states and Recurrent Neural Networks that focus on modeling temporal dependencies in sequential data using recurrent connections between network units. This work uses a hybrid model, Hidden Markov Models, and Recurrent Neural Networks to predict time series. For this purpose, a recurrent neural network is realized, and the hidden Markov model is added to update the regime changes. In the model implementation, data of various currency exchange rates (EUR/USD, GBP/USD, USD/CAD, USD/CHF), ranging from January 1, 2020 to September 1, 2022, is used. In addition, the mean squared error (MSE) and root mean squared error (RMSE) were used as goodness-of-fit measures, obtaining low errors and indicating the quality of the model estimation.

Keywords: Markov Switching Models, Hidden Markov Models, Recurrent Neural Networks.

Reviewed by:
Mgs. Hugo Romero
ENGLISH PROFESSOR
C.C. 0603156258

Introducción

Los Modelos Markov Switching (MSM), también conocidos como modelos Markov con Cambio de Estado y las Redes Neuronales Recurrentes (RNN) son dos enfoques distintos utilizados en el análisis de series temporales y la modelización de datos secuenciales.

Los MSM son modelos estadísticos que se basan en la idea de que las observaciones son generadas por diferentes procesos Markovianos, cada uno con sus propias características estadísticas. Estos modelos son útiles para capturar comportamientos distintos en diferentes períodos de tiempo. Los cambios de estado pueden ocurrir de manera abrupta o gradual, lo que permite capturar la dinámica compleja de los datos(Chang-Jin & Charles, 1999; Nguyen, 2020).

Por otro lado, las Redes Neuronales Recurrentes son un tipo de red neuronal que puede modelar secuencias de datos y capturar dependencias temporales a través de conexiones recurrentes. Las RNN son especialmente útiles para trabajar con datos secuenciales, como series temporales (Hamilton, 1989; Ilhan et al., 2020; Lim et al., 2019), texto (Du et al., 2021; Parthiban et al., 2020; Wang et al., 2017) o señales de audio (Chiu et al., 2021; Makino et al., 2019). Al utilizar conexiones recurrentes, estas redes pueden recordar información de observaciones anteriores y utilizarla para hacer predicciones o clasificaciones en el futuro (Aleksander & Morton, 1990; Haykin, 2009).

En este trabajo, se pretende combinar estos modelos para obtener mejores resultados en la modelización y predicción de series temporales de tasas de cambio entre diferentes monedas que se actualizan diariamente. Estos datos incluyen el valor de una moneda en

términos de otra, lo que permite a los inversores, empresas y analistas monitorear y analizar la evolución de los mercados de divisas.

Algunos trabajos relacionados con el tema son: el realizado por Ilhan et al. (2020), ellos emplean un modelo de Markov oculto (HMM) para las transiciones de régimen, donde cada régimen controla las transiciones de estado ocultas de la celda recurrente de forma independiente. Además, Aydogan-Kilic & Selcuk-Kestel (2023), proponen una estructura híbrida que combina RNN y HMM en el modelado de datos financieros, ellos intentan eliminar la influencia de los parámetros iniciales.

A continuación, se ofrece un resumen de lo que se presenta en cada capítulo: en el capítulo 1 se describe el planteamiento del problema, el objetivo general y los objetivos específicos. El segundo capítulo se presenta el marco teórico donde se resaltan las cadenas de Markov, los Modelos Ocultos de Markov y las Redes Neuronales Recurrentes. En tercer capítulo se muestra la metodología empleada para estimar los parámetros de los modelos y se presenta el algoritmo de aprendizaje del modelo. En el capítulo 4, se describe los resultados obtenidos al aplicar los modelos. En el quinto capítulo se adentra en un marco propositivo en base a los resultados obtenidos. Finalmente, se presentan las conclusiones de este trabajo.

Capítulo 1

Generalidades

1.1 Planteamiento del problema

El comportamiento de la economía es complejo, irregular, de apariencia aleatoria, y de naturaleza no lineal. La mayoría de las series económicas presentan estas características, lo que hace difícil predecir su estados actuales y futuros. Con la ayuda de modelos matemáticos se puede describir el comportamiento de estos sistemas dinámicos que modelan la economía.

Existen muchas evidencias empíricas que demuestran que los comportamientos de las series temporales generadas por sistemas que modelan variables económicas y financieras tienen diferentes patrones a lo largo del tiempo. Para modelar este tipo de series, en lugar de utilizar un modelo clásico para la serie temporal, es natural emplear varios modelos para representar estos patrones.

En este trabajo, se pretende modelar series temporales de tasa de cambio para estimar los parámetros que influyen en los cambios que se producen en la tasa de cambio y predecir la tendencia que estas tendrán. Se propone utilizar un modelo de redes neuronales recurrentes (RNN por sus siglas en inglés) combinado con un Modelo Markov Switching (MSM por sus siglas en inglés). Los modelos RNN se adaptan bien a las series temporales y los modelos MSM en la parte no lineal. Con los resultados que se obtenga del estudio, se podrán realizar análisis que permitan predecir el comportamiento de las tasas de cambio.

La pregunta que se plantea para la investigación es la siguiente:

¿Cómo pronosticar la tasa de cambio utilizando modelos markovianos de cambio de régimen combinados con los modelos de redes neuronales recurrentes?

1.2 Objetivos

1.2.1 Objetivo general

- Utilizar modelos markovianos switching y redes neuronales recurrentes para predecir la tasa de cambio de diferentes divisas.

1.2.2 Objetivos específicos

- Elaborar una base de datos de tasas de cambio de distintas monedas.
- Elaborar un modelo matemático para ajustar los datos de las tasas de cambio.
- Implementar algoritmos en un software computacional para hacer inferencia.
- Estimar parámetros de los modelos planteados.
- Obtener el pronóstico de las tasas de cambio.
- Implementar algunas medidas de bondad de ajuste para validar los modelos.

Capítulo 2

Marco Teórico

2.1 Cadenas de Markov

2.1.1 Introducción

Para discutir las cadenas de Markov, es necesario mencionar la teoría de los procesos estocásticos. Un proceso estocástico es una familia de variables aleatorias X_n que esta indexado por n, con n ∈ T. El conjunto T es llamado como el conjunto de índices. Existen dos tipos comunes de procesos estocásticos:

- Si T es discreto, X_n es un proceso estocástico de tiempo discreto.
- Si T es continuo, X_n es un proceso estocástico de tiempo continuo.

Cada variable aleatoria X_n puede tener una distribución discreta, continua o mixta. Además, cada X_n toma sus valores en el mismo conjunto, que se denomina espacio de estados y se denota como S, donde S puede ser discreto o continuo. Por lo tanto, $X_n \in S$.

Adicionalmente, la función de distribución conjunta para el modelo de probabilidad de este proceso, donde $\{X_n: n = 0,1,2, \dots\}$ está determinado por:

$$P(X_0 = x_0, X_1 = x_1, \cdots, X_n = x_n)$$

Para todo $n = 0, 1, \dots$ y $x_0, x_1, \dots \in S$.

En general, esta función de distribución conjunta puede ser arbitraria por lo que es muy compleja. Se necesita algún supuesto adicional sobre la función de distribución conjunta para que sea lo suficientemente simple como para poder analizarla. Un ejemplo particular son los supuestos de que las variables sean independientes e idénticamente distribuidas (i.i.d.).

En este caso, la función de distribución conjunta se puede factorizar en productos de funciones marginales. Sin embargo, los supuestos del i.i.d. suelen ser demasiado sólidos para algunos escenarios. Por ejemplo, al modelar la deriva genética, el supuesto iid en cada generación no es un buen modelo.

A continuación, se muestran dos ejemplos en los que los supuestos del i.i.d. no funcionan. Debido a esto, se estudia procesos con una memoria de un solo paso y este proceso estocástico se conoce como cadena de Markov.

2.1.2 Definición y propiedades

Una *Cadena de Markov* (MC por sus siglas en inglés) es un proceso estocástico de tiempo discreto que considera lo siguiente:

Si para cada $x_0, x_1, \cdots, x_{n-2}, i, j \in S$ y $n \geq 0$, entonces

$$P(X_n = i \,|X_{n-1} = j, \cdots, X_0 = x_0) = P(X_n = i \,|X_{n-1} = j)$$

siempre que ambos lados estén bien definidos (Seforzo, 2009). La cadena de Markov tiene memoria de un solo paso.

Por otro lado, a $\{X_n : x \geq 1\}$ se le conoce como una cadena de Markov homogénea (Yen-Chi, 2018) si, la función de distribución $P(X_n = x_n \,|X_{n-1} = x_{n-1}) = p_{ij}$ es independiente de n. De lo contrario, se llama cadena de Markov no homogénea.

Para una cadena de Markov homogénea se tiene que,

$$\sum_{j \in S} p_{ij} = 1, \qquad p_{ij} \geq 0$$

para cada i, j.

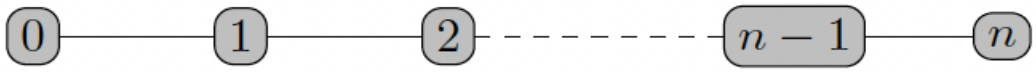
Como S es un conjunto discreto, a menudo lo etiquetamos como $S = \{1,2,3,\cdots,s\}$ y los elementos $\{p_{ij} : i,j = 1,\cdots,s\}$ forman una matriz $\boldsymbol{P} = \{p_{ij}\}$ de dimensión $s \times s$. $\boldsymbol{P}$ se llama matriz de transición (o de probabilidad). La propiedad de cadena de Markov homogénea implica que

$$\boldsymbol{P} \geq 0, \quad \boldsymbol{P}\mathbf{1}_s = \mathbf{1}_s \tag{1}$$

donde $\mathbf{1}_s = (1,1,1,1,\cdots,1)^T$ es el vector de $1's$. Observe que, cualquier matriz que satisface la ecuación (1) es llamada matriz estocástica.

Figura 1

Cadena de Markov utilizando un modelo gráfico simple.



Nota. Yen-Chi (2018)

En las Cadenas de Markov, si la distribución de probabilidad futura para $X = (X_1,\cdots,X_n)$ es una secuencia de variables aleatorias, depende únicamente del valor actual, se consideran procesos "sin memoria", es decir:

$$P(X_{n+1} = x_{n+1}|X_n = x_n,\cdots,X_1 = x_1) = P(X_{n+1} = x_{n+1}|X_n = x_n)$$

Esta identidad es conocida como *propiedad de Márkov* (Kalashnikov, 1994).

A continuación, se muestra un ejemplo llamado La Ruina del Jugador, es el último de los problemas propuesto a los lectores en el Tratado de Huygens, publicado por primera vez en 1657 (Rosenthal et al., 1984), el cual es un problema clásico que plantea un escenario en el que dos participantes inician un juego con un número finito de dinero, en el cual no se

permiten empates. Rocha y Stern (1999), y Katriel (2013), presentan algunas fórmulas para calcular la probabilidad de ruina en diferentes escenarios.

Los jugadores inician partidas entre ellos y el juego consiste en determinar la probabilidad de ruina que tienen los jugadores y la duración del juego bajo esas condiciones. Este problema servirá para explicar de mejor manera cómo funciona una Cadena de Markov.

En este ejemplo presentado en la tesis de Tornero (2017), se gana \$1 con probabilidad $p = 0.5$ y pierdes \$1 con probabilidad $1 - p = 0.5$. Adicionalmente, se supone que, un jugador deja de jugar si llega a tener \$N o \$0. Entonces se tiene lo siguiente:

$$a_{i,i+1} = 0.5, \quad a_{i,i-1} = 0.5, \quad si\ 0 < i < N$$

$$a_{0,0} = 1, \quad a_{N,N} = 1$$

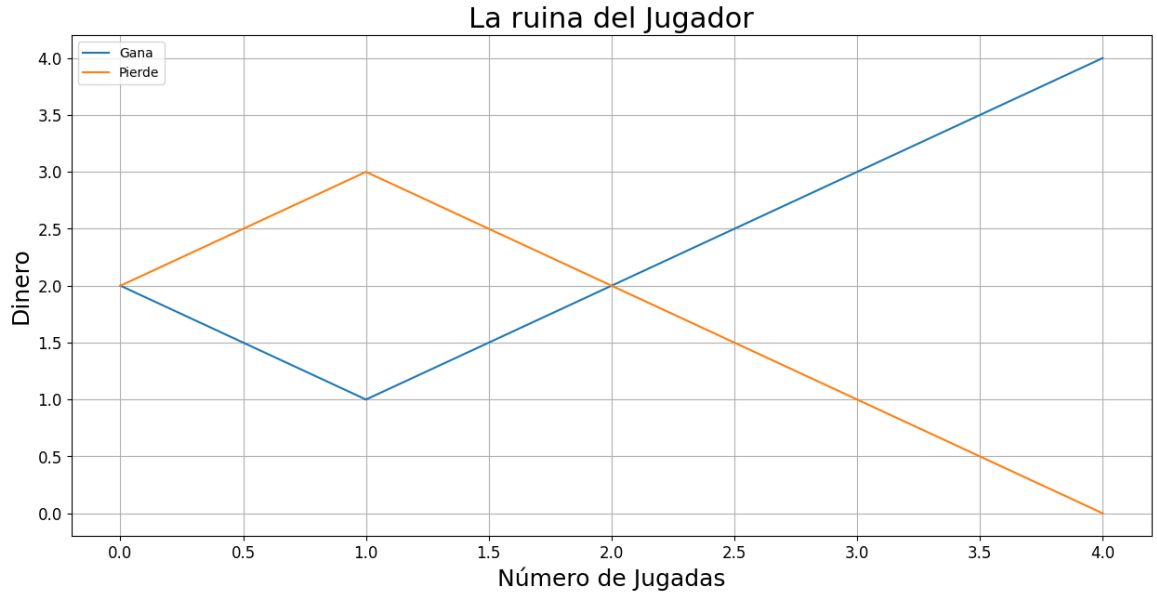
En particular, se presentará un ejemplo para el valor N=4.

Para $N = 4$, se tiene

$$A = \begin{pmatrix} 1.0 & 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0.5 & 0 & 0 \\ 0 & 0.5 & 0 & 0.5 & 0 \\ 0 & 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 1.0 \end{pmatrix}$$

Figura 2

Simulación para la ruina del jugador con N=4.



Los estados fijos en este modelo serán 0 y N, ya que son los únicos estados alcanzables, es decir, que tienen probabilidad mayor que 0, son ellos mismos: $p(0,0) = p(N,N) = 1$. Los estados que tienen ese comportamiento se les denomina absorbentes.

Se ha definido $p(i,j)$ como la probabilidad de ir en un paso del estado i al j. A continuación, se plantea computar la probabilidad de ir de i a j en $m > 1$ pasos

$$p^m(i,j) = P(X_{n+m} = j | X_n = i) \tag{2}$$

Por ejemplo, si se considera la Cadena de Markov del problema *La Ruina del Jugador* para el caso N=4 ¿cuál es la probabilidad de tener \$2 a tener \$0 en dos jugadas?

Entonces se considera los posibles estados y la propiedad

$$P(A|B) = \frac{P(B \cap A)}{P(A)} \tag{3}$$

Entonces se tiene:

$$
\begin{aligned}
P(X_2 = 0|X_0 = 2) &= \sum_{k=0}^{4} P(X_2 = 0, X_1 = k\ |X_0 = 2) \\
&= \sum_{k=0}^{4} \frac{P(X_2 = 0, X_1 = k\ |X_0 = 2)}{P(X_0 = 2)} \\
&= \sum_{k=0}^{4} \frac{P(X_2 = 0, X_1 = k, X_0 = 2)}{P(X_1 = k\ |X_0 = 2)} \frac{P(X_1 = k|X_0 = 2)}{P(X_0 = 2)} \\
&= \sum_{k=0}^{4} P(X_2 = 0, X_1 = k\ |X_0 = 2) P(X_1 = k|X_0 = 2)
\end{aligned}
$$

Aplicando la Propiedad de Markov se observa que:

$$
P(\ X_2 = 0|X_1 = k, X_0 = 2) P(X_1 = k|X_0 = 2) = p(2,k)p(k,0), \qquad \forall\ 0 \leq k \leq 4
$$

Por lo tanto,

$$
\begin{aligned}
P(X_2 = 0\ |X_0 = 2) &= \sum_{k=0}^{4} p(2,k)p(k,0) \\
&= 0 \cdot 0 + 0.5 \cdot 0.5 + 0 \cdot + \cdots = 0.5 \cdot 0.5 = 0.25
\end{aligned}
$$

(Tornero, 2017)

2.1.3 Probabilidad de transición de n-pasos y la ecuación de Chapman-Kolmogorov

Para una cadena de Markov, se define la probabilidad de transición como

$$
p_{ij}^{(n)} = P(X_n = j|X_0 = i)
$$

La probabilidad de transición de n-pasos es invariante en el tiempo.

Adicionalmente, sea $\{X_n\}$ una cadena de Markov homogénea y sea $p_{ij}^{(n)}$ la probabilidad de transición de n-pasos. Entonces, para cualquier $k = 0,1,2,\cdots$,

$$
P(X_{n+k} = j\ |X_k = i) = p_{ij}^{(n)} \tag{4}
$$

Las probabilidades de transición de n pasos están relacionadas entre sí mediante la famosa ecuación de Chapman-Kolmogorov (Kalashnikov, 1994).

Sea $\{X_n: x \geq 1\}$ una cadena de Markov homogénea y sea $p_{ij}^{(n)}$ la probabilidad de transición de n-pasos. Entonces, para cualquier $m, n = 0,1,2,\cdots$,

$$p_{ij}^{(n+m)} = \sum_{k \in S} p_{ik}^{(n)} p_{kj}^{(m)} \tag{5}$$

La ecuación de Chapman-Kolmogorov (ecuación (5)) también implica

$$Ecuación\ Forward:\ p_{ij}^{(n+1)} = \sum_{k \in S} p_{ik}^{(n)} p_{kj}\ \text{, para n} = 1,2,\cdots$$

$$Ecuación\ Backward:\ p_{ij}^{(n+1)} = \sum_{k \in S} p_{ik} p_{kj}^{(n)}\ \text{para n} = 1,2,\cdots$$

La ecuación Forward señala el paso final y tiene el estado inicial i fijo. La ecuación es más útil cuando el interés se centra en los $p_{ij}^{(n)}$ para un *i* particular, pero para todos los valores de *j*. Por el contrario, la ecuación Backward señala el cambio desde el estado inicial *i* y tiene el estado final *j* fijo. Esta ecuación es útil cuando el interés está en los $p_{ij}^{(n)}$ para un *j* particular, pero todos los valores de *i*. La ecuación hacia atrás puede ser cuando existe un estado absorbente *j* del cual no hay escape ($p_{ij} = 1$).

Si se recopila las probabilidades de transición de n-pasos en la matriz $P^{(n)} = \{p_{ij}^{(n)}\}$, entonces las ecuaciones Forward y Backward de Kolmogorov se pueden reescribir en forma matricial como

$$\boldsymbol{P}^{(n+1)} = \boldsymbol{P}^{(n)}\boldsymbol{P} = \boldsymbol{P}\boldsymbol{P}^{(n)}$$

donde, $\boldsymbol{P}^{\mathbf{1}} = \boldsymbol{P}$. Además, se tiene que, $\boldsymbol{P}^{(n)} = \boldsymbol{P}^{n}$ (Guttorp, 1995)

Esta forma matricial también implica una propiedad interesante sobre la distribución marginal de X_n. Supongamos que X_0 tiene una distribución $p_0 = (p_{01}, p_{02}, \cdots, p_{0s})^T$. Sea $p_n = (p_{n1}, p_{n2}, \cdots, p_{ns})^T$ la distribución marginal de X_n, es decir, $p_{nj} = P(X_n = j)$. Entonces

$$\begin{aligned} p_{nj} = P(X_n = j) &= \sum_i P(X_n = j, X_0 = i) \\ &= \sum_i P(X_n = j, X_0 = i)\, P(X_0 = i) \\ &= \sum_i p_{0i} p_{ij}^{(n)} \end{aligned}$$

Utilizando la forma matricial, se obtiene

$$p_n^T = p_0^T \boldsymbol{P}^n$$

2.1.4 Clasificación de Estados

Antes de mencionar la clasificación de los estados, se va a mencionar sobre los tipos de procesos estocásticos que se pueden obtener. Sean T el espacio de parámetros y S el espacio de estados los cuales pueden ser discretos (D) o continuos (C). Además, el conjunto de todos los valores posibles de una variable aleatoria individual X_n es conocido como Espacio estado de un proceso estocástico. Entonces, combinando el espacio de parámetros y el espacio de estados, se puede obtener los diferentes tipos de procesos estocásticos, los cuales se muestran en la Tabla 1:

Tabla 1

Tipos de procesos estocásticos

Espacio de Parámetros	Espacio de estados	Combinación	Proceso
D	D	{D, D}	Cadena de Markov de tiempo discreto
D	C	{D, C}	Cadena de Markov de tiempo continuo
C	D	{C, D}	Proceso de Poisson
C	C	{C, C}	Movimiento Browniano

Este trabajo utiliza procesos estocásticos de tiempo discreto con un espacio de estados discreto.

Ahora, se presenta una clasificación de los estados de una cadena de Markov que es crucial para comprender el comportamiento de las cadenas de Markov. Antes se menciona la definición de una relación de equivalencia "$\sim$" como lo menciona Yen-Chi Chen (2018). Una relación de equivalencia es una relación binaria entre elementos de un conjunto que satisface lo siguiente:

1. Reflexividad: $i \sim i$ para todo i

2. Simetría: $i \sim j \Rightarrow j \sim i$

3. Transitividad: $i \sim j, j \sim k \Rightarrow i \sim k$.

Para un conjunto S y $a \in S$, $\{s \in S: s \sim a\}$ se llama clase de equivalencia. Las relaciones de equivalencia nos permitirán dividir los espacios de estados de la cadena de Markov en clases de equivalencia.

El estado j es alcanzable desde el estado i $(i \to j)$ sí existe m ≥ 0 tal que $p_{ij}^{(m)} > 0$. Por otro lado, se dice que i se comunica con j $(i \leftrightarrow j)$ si j es alcanzable desde i y i es accesible desde j. A dos estados que se comunican se dice que están en la misma clase.

Ahora, se tiene que $f_{ii} = P_i(T_i < \infty)$ es la probabilidad de regresar a i dado que comenzamos en i y $\mathbb{E}_i(T_i)$ es el tiempo de regreso esperado. Entonces, según sea el caso, el estado i toma los siguientes nombres:

1. Recurrente si $f_{ii} = 1$.
2. Transitorio si $f_{ii} < 1$.
3. Recurrente positivo si $f_{ii} = 1$ y $\mathbb{E}_i(T_i) < \infty$.
4. Recurrente nulo si $f_{ii} = 1$ y $\mathbb{E}_i(T_i) = \infty$.
5. Periódico con periodo d_i si $p_{ii}^{(n)} = 0$ para todo n no divisible para d_i(>1) es el mayor entero de ese tipo.
6. Aperiódico si $d_i = 1$.
7. Ergódico si se cumplen (3) y (6) .
8. Absorbente si $p_{ii} = 1$. (Yen-Chi, 2018)

Las cadenas de Markov convergen a una distribución estacionaria si la cadena es ergódica, es decir, es aperiódica, irreductible y recurrente positiva.

Si se retoma el ejemplo de *La Ruina del Jugador* (Tornero, 2017), se observa que la cadena se queda atascada en $0 o $N. Para el caso N = 4 es fácil comprobar que estos dos

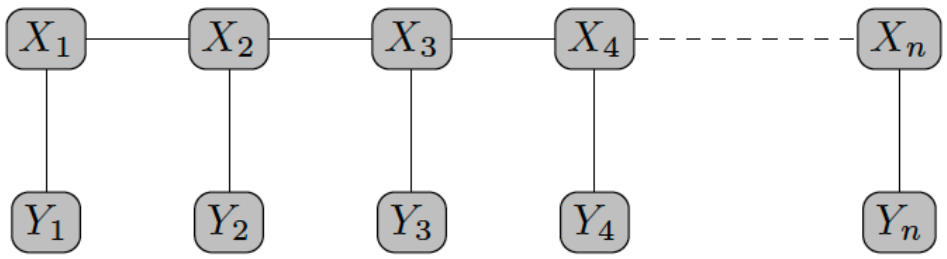
estados son recurrentes, ya que $p_{00} = 1$. De forma similar, se puede ver que *N* es recurrente. En forma general, si *i* es un estado absorbente, entonces *i* es también recurrente. Lo que confirma que la cadena eventualmente se queda atascada bien en \$0 o en \$4.

2.2 Modelos Ocultos de Markov

El Modelo Oculto de Markov (HMM por sus siglas en inglés) consta de dos variables, la variable observada Y_t y una variable de estado oculta X_t. Las variables $\{Y_1, \cdots, Y_n\}$ son las que observamos mientras que las variables ocultas $\{X_1, \cdots, X_n\}$ son estados no observados en cada instante de tiempo (Liisberg et al., 2016; Yen-Chi, 2018).

Figura 3

Modelo Oculto de Markov.



Nota. Yen-Chi (2018)

Los estados ocultos se los representa con $k_t \in \{1, \dots, K\}$, donde K es el número de estados. La evolución del sistema en el tiempo está definida por las matrices de transición y emisión, junto con la distribución inicial. La distribución conjunta tiene la siguiente forma:

$$p(k_{1:T}, \boldsymbol{y}_{1:T}) = \prod_{t=1}^{T} p(k_t \mid k_{t-1}; \Psi) p(\boldsymbol{y}_t \mid k_t; \boldsymbol{\theta}) \tag{6}$$

donde,

- $p(k_1 \mid k_0) = \pi(k_1)$ es la distribución del estado inicial.

- $p(k_t \mid k_{t-1}; \Psi)$ es el modelo de transmisión definido por una matriz de transmisión $\Psi \in \mathbb{R}^{K \times K}$ tal que $\Psi_{ij} \triangleq p(k_t = j \mid k_{t-1} = i)$.

En algunas ocasiones, el modelo de observación (modelo de emisión) se modela como gaussiano, de modo que $p(\boldsymbol{y}_t \mid k_t = k; \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{y}_t \mid \boldsymbol{\mu}_k, \Sigma_k)$.

El estado posterior $p(k_t \mid \boldsymbol{y}_{1:T})$ también se denomina estado de creencia filtrado y puede estimarse recursivamente mediante lo siguiente:

$$p(k_t \mid \boldsymbol{y}_t) = \frac{p(\boldsymbol{y}_t \mid k_t) p(k_t \mid \boldsymbol{y}_{1:t-1})}{p(\boldsymbol{y}_t \mid \boldsymbol{y}_{1:t-1})} \propto p(\boldsymbol{y}_t \mid k_t) \sum_{k_t=1}^{K} p(k_t \mid k_{t-1}) p(k_{t-1} \mid \boldsymbol{y}_{t-1}) \tag{7}$$

Sea $\alpha_{t,k} \triangleq p(k_t = k \mid \boldsymbol{y}_{1:T})$ la creencia para el k-ésimo estado, se define $\boldsymbol{\alpha}_t = [\ldots, \alpha_{t,k}, \ldots]^T$ como el vector de estado de creencia K-dimensional, y, $\boldsymbol{\phi}_t = [\ldots, p(\boldsymbol{y}_t \mid k_t = k), \ldots]^T$ como el K-dimensional vector de probabilidad, respectivamente. Entonces la ecuación (7) se puede expresar como

$$\boldsymbol{\alpha}_t \propto \boldsymbol{\phi}_t \odot (\Psi^T \boldsymbol{\alpha}_{t-1}) \tag{8}$$

Ahora, para obtener el vector de estado de creencia filtrado, se normaliza la expresión (8) y se la divide por la suma de valores (Ilhan et al., 2020).

Adicionalmente, para evitar ambigüedades entre los estados del modelo de Markov y los estados ocultos de la red neuronal, se introduce el término “regímenes” para referirse a los estados del modelo oculto de Markov. Esta distinción en términos facilitará la comprensión precisa de los elementos de cada modelo utilizado, evitando confusiones en el análisis e interpretación de resultados.

2.2.1 Componentes de un HMM

Para definir completamente un HMM, se necesitan cinco elementos (Liu et al., 2015):

1. Los *K* estados ocultos del modelo $S = \{S_1, \dots, S_K\}$.
2. Los M diferentes símbolos $V = \{V_1, \dots, V_M\}$ que pueden observarse. Si las observaciones son continuas, M es infinito.
3. La matriz de transición $\Psi = \{a_{ij}\}$, donde $a_{ij} = P(q_{t+1} = j | q_t = i)$, siendo q_t el estado actual. Además, si uno de los a_{ij} es definido cero, permanecerá cero durante todo el proceso de entrenamiento.
4. La probabilidad de distribución de las variables en cada estado, $B = \{b_j(k)\}$ donde $b_j(k)$ es la probabilidad de observar la variable v_j en el estado S_j, viene dada por

$$b_j(k) = P(o_t = v_k \,|q_t = j),\ \forall j \in \{1, \dots, N\}, \forall k \in \{1, \dots, M\}$$

donde v_k denota el *k*-ésima variable del alfabeto, y o_t el actual vector de observaciones. Se deben satisfacer ciertas restricciones:

$b_j(k) \geq 0,\ \forall j \in \{1, \dots, N\}, \forall k \in \{1, \dots, M\}$ y además $\sum_{k=1}^{M} b_j(k) = 1,\ \forall j \in \{1, \dots, N\}$

En el caso continuo, se tiene una función de densidad continua, en lugar de un conjunto de probabilidades discretas. En este caso, se establece el conjunto de parámetros de la función de densidad, aproximada como una suma ponderada de M distribuciones Gaussianas (GHMM),

$$b_j(o_t) = \sum_{m=1}^{M} c_{jm} N(\mu_{jm}, \Sigma_{jm}, o_t)$$

donde c_{jm} son los pesos, μ_{jm} los vectores de medias, y Σ_{jm} las matrices de covarianzas.

Observar que c_{jm} debe satisfacer las condiciones estocásticas $c_{jm} \geq 0, \forall m \in \{1, \dots, M\}$ y

$$\sum_{m=1}^{M} c_{jm} = 1, \forall j \in \{1, \dots, N\}$$

5. La distribución inicial de estados $\pi = \{\pi_i\}$, donde π_i es la probabilidad de que el modelo está en el estado S_i en el tiempo inicial $t = 0$, con

$$\pi_i = P(q_1 = i), \forall_i \in \{1, \dots, N\}$$

Para denotar los parámetros de un HMM a menudo se utiliza

$$\lambda = (A, B, \pi)$$

para denotar distribuciones discretas, o bien

$$\lambda = \left(A, c_{jm}, \mu_{jm}, \Sigma_{jm}, \pi\right)$$

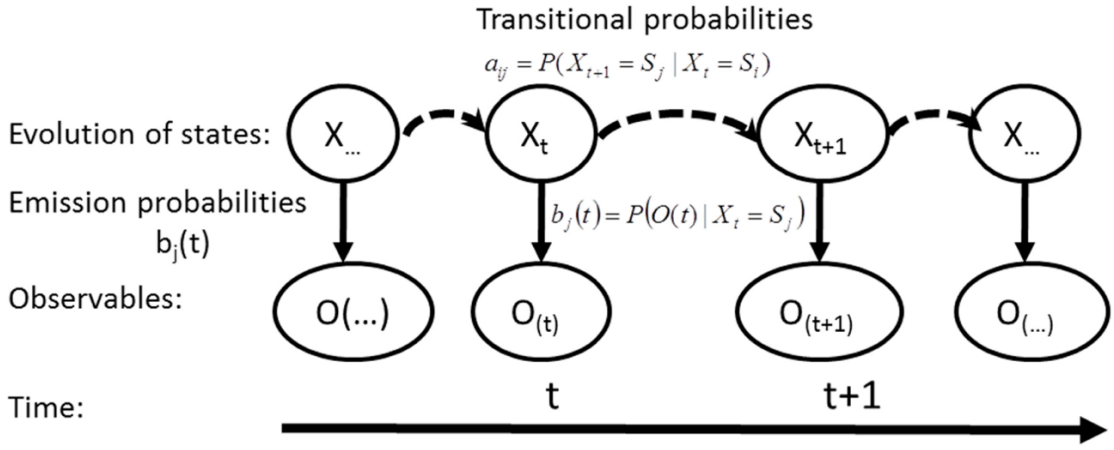
cuando se trata de distribuciones continúas asociadas a funciones de densidad.

2.2.2 Arquitectura de los HMM

En la Figura 4 se muestra un esquema del funcionamiento general de un Modelo Oculto de Markov.

Figura 4

Esquema de funcionamiento de un HMM.



Nota. Liu et al. (2015)

A cada variable aleatoria $S(t)$ le corresponde su respectivo tiempo *t.* Esta representa el estado oculto. La variable *O* es la de las observaciones.

En el primer proceso estocástico, resulta que, en el tiempo t el valor de la variable oculta $S(t)$, solo depende de $S(t-1)$, cuando los valores de S son conocidos. Por lo tanto, la propiedad de Markov es satisfecha.

En cambio, en el segundo proceso estocástico, el valor de $O(t)$ viene de la variable oculta asociada a $S(t)$.

2.2.3 Algoritmos utilizados en los Modelos de Markov Ocultos

Los Modelos Ocultos de Markov tienen como objetivo aprender a clasificar los datos proporcionados. Tienen que ser capaces de descifrar las diferencias de comportamiento que posean los diferentes estados a partir de las observaciones con el que se lo alimenta (Tornero, 2017). Históricamente, los HMM se han destacado en el estudio de tres problemas que se mencionan a continuación:

2.2.3.1 Problema de Evaluación: Forward Algorithm

Dada una secuencia de observaciones $O = \{o_1, \dots, o_m\}$ y un modelo $\lambda = (A, B, \pi)$, se trata de averiguar $P(O \mid \lambda)$. Lo primero que se plantea es el cálculo mediante la fórmula de Bayes:

$$P(O \mid \lambda) = \frac{P(\lambda \mid O)P(O)}{P(\lambda)}$$

No obstante, las operaciones requeridas están en orden de N^T, lo que computacionalmente no es efectivo. Sin embargo, existe un método de menos complejidad en el cual se utiliza una variable auxiliar *forward* (α) de la forma

$$\alpha_t(i) = P(o_1, o_2, \dots, o_t, q_t = i \mid \lambda)$$

Para el cálculo de esta variable se puede definir una relación recursiva de la forma

$$\begin{cases} \alpha_1(j) = \pi_j b_j(o_1), \forall j \in \{1, \dots, N\} \\ \alpha_{t+1}(j) = b_j(o_{t+1}) \sum_{i=1}^{N} \alpha_t(i)\alpha_{ij}\,; \;\; \forall j \in \{1, \dots, N\}, \forall t \in \{1, \dots, T-1\} \end{cases}$$

Utilizando recursión, se puede calcular las variables α_T. Así que, la probabilidad requerida es dada por

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

Este método es conocido comúnmente como *forward algorithm* (Tornero, 2017; Yen-Chi, 2018).

De igual manera se puede definir la variable *backward* $\beta_t(i)$

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, \dots, o_T \mid q_t = i, \lambda)$$

Donde, $\beta_t(i)$ es la probabilidad de tener en *t* el estado *i,* conociendo la historia futura parcial $o_{t+1}, o_{t+2}, \dots, o_T$.

Al igual que para la variable *forward* se plantea la fórmula recursiva para la variable *backward*

$$\begin{cases} \beta_t(i) = \sum_{j=1}^{N} \beta_{t+1}(j) a_{ij} b_j(o_{t+1});\ \forall i \in \{1, \dots, N\}, \forall t \in \{1, \dots, T-1\} \\ \beta_T(i) = 1, \forall i \in \{1, \dots, N\} \end{cases}$$

Combinando las ecuaciones anteriores se obtiene

$$\alpha_t(i)\beta_{t(i)} = P(O, q_t = i \mid \lambda) \qquad \forall i \in \{1, \dots, N\}, \forall t \in \{1, \dots, T\}$$

Finalmente, se obtiene que

$$P(O \mid \lambda) = \sum_{i=1}^{N} P(O, q_t = i \mid \lambda) = \sum_{i=1}^{N} \alpha_t(i)\beta_t(i)$$

(Tornero, 2017; Yen-Chi, 2018)

2.2.3.2 Algoritmo de Viterbi

Dada una secuencia de observaciones $O = o_1, o_2, \dots, o_T$ y un modelo $\lambda = (A, B, \pi)$ se desea conocer cuál es la secuencia de estados más probable, este proceso se lo conoce como problema de decodificación.

En este caso, para encontrar el estado de secuencias más probable, se utiliza el *Algoritmo de Viterbi*.

Para ello se utiliza una variable auxiliar

$$\delta_t(i) = \max_{q_1, q_2, \dots, q_{t-1}} P(q_1, q_2, \dots, q_{t-1}, q_t = 1, o_1, o_2, \dots., q_t \mid \lambda)$$

Para el caso $t = 1$ se tiene $\delta_1(j) = \pi_j b_j(o_1), \forall j \in \{1, \dots, N\}$, para los siguientes términos se usa la siguiente fórmula

$$\delta_{t+1}(j) = b_j(o_{t+1}) \left[\max_{1 \le i \le N} \delta_t(i) a_{ij}\right] \quad \forall i \in \{1, \dots, N\}, \forall t \in \{1, \dots, T-1\}$$

Por lo tanto, se inicia el cálculo desde $\delta_T(j), \forall j \in \{1, \dots, N\}$, se mantiene un puntero al estado que será el "ganador", y $j* = \max_{1 \le i \le N} \delta_T(j)$, y a partir de esto se realiza un paso hacia atrás en la secuencia, con lo cual se logra redefinir $j*$ en cada paso, se obtiene de esta manera el conjunto de estados que se busca (Nguyen, 2020).

2.2.3.3 Algoritmo Baum-Welch

El algoritmo Baum-Welch se centra en encontrar la manera de ajustar los parámetros del Modelo de Markov Oculto para que se ajusten de mejor manera a las observaciones.

A este algoritmo también se lo conoce como *Forward-Backward*, debido a que utiliza variables que se definen previamente. Además, utiliza una cantidad auxiliar $Q(\lambda, \lambda')$ para comparar dos modelos λ y λ'.

$$Q(\lambda, \lambda') = \sum_q P(q \mid O, \lambda) \log [P(O, q, \lambda')]$$

A partir de las *backward* y *forward* se definen nuevas variables.

Ahora, dadas las observaciones *O* y el modelo λ, se puede obtener la probabilidad de estar en el estado *i* en el instante *t*. Esta probabilidad es conocida como *smoothed*, Se obtiene combinando α y β de acuerdo al Teorema de Bayes.

$$\gamma_t(i) = P(q_t(i) | O, \gamma) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \gamma_t(i)}$$

También se puede definir la probabilidad de estar en el estado *i* en el instante *t*, y en el estado *j* en el instante $t+1$; dadas las observaciones *O* y el modelo γ.

$$\xi_t(i,j) = \frac{P((q_t = i, q_{t+1} = j, O\ |\lambda)}{P(O\ |\lambda)} = \frac{\alpha_i(t) a_{i,j}\, \beta_j(t+1) b_j(y_{t+1})}{\sum_{k=1}^{N}\sum_{l=1}^{N} \alpha_k(t) a_{kl} \beta_l(t+1) b_l(y_{t+1})}$$

Por lo tanto, la relación entre $\gamma_t(i)$ y $\xi_t(i)$ viene dada por

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j) \quad \forall i \in \{1, \dots, N\}, \forall t \in \{1, \dots, M\}$$

Luego de calcular las nuevas variables, se definen los nuevos parámetros $\lambda' = (A', B',\ \pi')$ del HMM utilizando las *smoothed* de acuerdo a lo siguiente

Probabilidades iniciales: probabilidades de estar en el estado *i* en *t* = 1:

$$\pi' = \gamma_1(i) \quad \forall i \in \{1, \dots, N\}$$

Matriz de transición: cada a_{ij} viene representado por el cociente entre el número esperado de transiciones entre *i* y *j*, entre el número total de transiciones desde *i*:

$$a'_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \quad \forall i \in \{1, \dots, N\}, \forall j \in \{1, \dots, N\}$$

Probabilidades de emisión: cociente entre el número de veces que se tiene el estado *j* y se observa o_k, y el número esperado de veces que se pasa por el estado *j*:

$$b'_j(o_k) \frac{\sum_{t=1;o_t=o_k}^{T} \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(j)} \quad \forall i \in \{1, \dots, N\}, \forall j \in \{1, \dots, N\}$$

(Nguyen, 2020; Tornero, 2017; Yen-Chi, 2018)

2.3 Redes Neuronales Artificiales

Las redes neuronales artificiales, comúnmente denominadas “redes neuronales”, se pueden definir de la siguiente manera: “Las redes neuronales artificiales consisten en un grupo densamente interconectado de unidades de conmutación de umbral simples similares a neuronas. Cada unidad toma una cantidad de insumos de valor real y produce una única salida de valor real. Basándose en la conectividad entre las unidades de umbral y los parámetros de los elementos, estas redes pueden modelar un comportamiento global complejo” (Staudemeyer & Morris, 2019).

El estudio de las redes neuronales artificiales está motivado, desde sus inicios, por conocer cómo el cerebro humano computa de manera diferente a la forma que lo hace una computadora digital convencional. Por ejemplo, el cerebro realiza rutinariamente tareas de reconocimiento perceptivo (reconocer una cara familiar incrustada en una escena desconocida) en aproximadamente 100 a 200 ms, mientras que tareas de mucha menor complejidad toman mucho más tiempo en una computadora potente.

En forma general, una red neuronal es una máquina diseñada para modelar la forma en que el cerebro realiza una tarea o función de interés particular; la red generalmente se implementa mediante el uso de componentes electrónicos o se simula en software en una computadora digital. Para lograr un buen rendimiento, las redes neuronales emplean una interconexión masiva de células informáticas simples denominadas "neuronas" o "unidades de procesamiento" (Haykin, 2009).

En contraste, se puede tomar la definición de Aleksander & Morton (1990) vista como una máquina adaptativa:

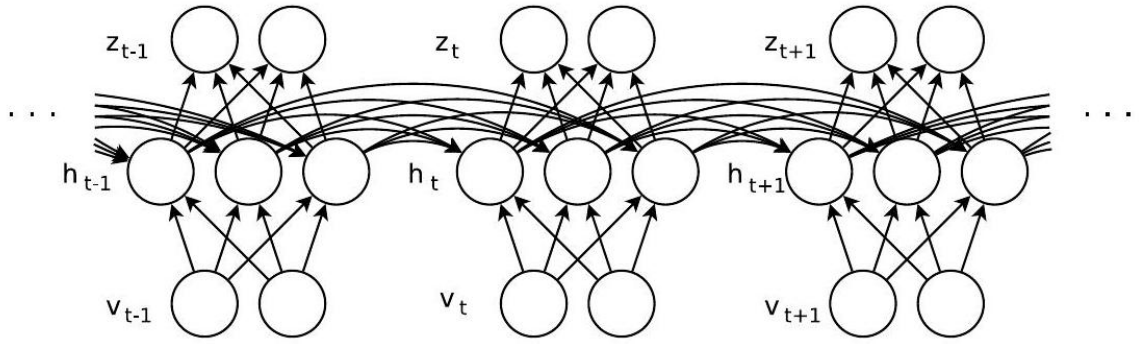
> "Una red neuronal es un procesador distribuido masivamente en paralelo formado por unidades de procesamiento simples que tiene una propensión natural a almacenar conocimiento experiencial y ponerlo a disposición para su uso. Se parece al cerebro en dos aspectos:
>
> 1. El conocimiento lo adquiere la red de su entorno mediante un proceso de aprendizaje.
>
> 2. Las fuerzas de conexión entre neuronas, conocidas como pesos sinápticos, se utilizan para almacenar el conocimiento adquirido."

2.4 Redes Neuronales Recurrentes

Las Redes Neuronales Recurrentes (RNN por sus siglas en inglés) es un sistema dinámico no lineal que asigna secuencias a secuencias. Está parametrizado con tres matrices de peso y tres vectores de sesgo $[W_{hy}; W_{hh}; W_{oh}; b_h; b_o; h_o]$ cuya concatenación θ describe completamente la RNN Figura 5.

Figura 5

Una Red Neuronal Recurrente.



Nota. Sutskever (2013)

Dada una secuencia de entrada $(v_1, \dots, v_T)$ (que denotamos por v_1^T), la RNN calcula una secuencia de estados ocultos h_1^T y una secuencia de salidas z_1^T mediante el siguiente algoritmo:

Algoritmo 1. Algoritmo para encontrar las secuencias de estados ocultos y salida

1. Para t desde 1 hasta T hacer
2. $u_t \leftarrow W_{hv} v_t + W_{hh} h_{t-1} + b_h$
3. $h_t \leftarrow e(u_t)$
4. $o_t \leftarrow W_{oh} h_t + b_o$
5. $z_t \leftarrow g(o_t)$
6. Fin Para

donde $e(\cdot)$ y $g(\cdot)$ son las no linealidades ocultas y de salida del RNN, y h_0 es un vector de parámetros que almacena el primer estado oculto. La pérdida del RNN suele ser una suma de las pérdidas por paso de tiempo:

$$L(z, y) = \sum_{t=1}^{T} L(z_t; y_t)$$

Las derivadas de las RNN se calculan fácilmente con el algoritmo de retropropagación en el tiempo (Werbos, 1990):

Algoritmo 2. Algoritmo backpropagation through time (BPTT)

1. $do_t \leftarrow g'(o_t) \cdot \frac{dL(z_t; y_t)}{dz_t}$
2. $db_o \leftarrow db_o + do_t$
3. $dW_{oh} \leftarrow dW_{oh} + do_t h_t^\top$
4. $dh_t \leftarrow dh_t + W_{oh}^\top do_t$

5. $dz_t \leftarrow e'(z_t) \cdot dh_t$
6. $dW_{hv} \leftarrow dW_{hv} + dz_t v_t^\top$
7. $db_h \leftarrow db_h + dz_t$
8. $dW_{hh} \leftarrow dW_{hh} + dz_t h_{t-1}^\top$
9. $dh_{t-1} \leftarrow W_{hh}^\top dz_t$
10. Fin Para
11. Retorna $d\theta = [dW_{hv}, dW_{hh}, dW_{oh}, db_h, db_o, dh_0]$.

En las RNN, los gradientes de los RNN son fáciles de calcular, pero son difíciles de entrenar, especialmente en problemas con dependencias temporales de largo alcance (Bengio et al., 1994; S. Hochreiter & Schmidhuber, 1997; Martens & Sutskever, 2011), debido a su naturaleza iterativa no lineal. Un pequeño cambio en un proceso iterativo puede generar efectos muy grandes, puede agravar el proceso muchas iteraciones después; esto se conoce coloquialmente como "el efecto mariposa". La implicación es que, en un RNN, la derivada de la función de pérdida en un momento dado puede ser exponencialmente grande con respecto a las activaciones ocultas en un momento mucho anterior. Por tanto, la función de pérdida es muy sensible a pequeños cambios, por lo que se vuelve efectivamente discontinua.

Las RNN también sufren el problema de desaparición de gradiente, Hochreiter (1991) y Bengio et al. (1994) fueron los que plantearon esto por primera vez.

Se considera el término $\partial \frac{\partial L(z_T; y_T)}{\partial W_{hh}}$:

$$\frac{\partial L(z_T; y_T)}{\partial W_{hh}} = \sum_{t=1}^{T} dz_t h_{t-1}^\top$$

donde

$$dz_t = \left(\prod_{\tau=t+1}^{T} W_{hh}^{\top} e'(z_t) \right) \left(W_{oh}^{\top} g'(o_t) \frac{\partial L(z_T; y_T)}{\partial p_T} \right)$$

Si todos los valores propios de W_{hh} son considerablemente menores que 1, entonces las contribuciones $dz_t h_{t-1}^{\top}$ a dW_{hh} disminuirán rápidamente debido a que dz_t tiende a cero exponencialmente a medida que $T - t$ aumenta.

Este último fenómeno se conoce como gradiente de fuga. Se garantiza que ocurrirá en cualquier RNN que pueda almacenar un bit de información indefinidamente y al mismo tiempo sea resistente al menos cierto nivel de ruido (Bengio et al., 1994), una condición que debería ser satisfecho por la mayoría de los RNN.

Un dz_t que desaparece no es deseable, porque convierte a BPTT en BPTT truncado, que probablemente sea incapaz de entrenar a las RNN para explotar la estructura temporal a largo plazo.

Los problemas de gradiente que desaparecen y explotan dificultan la optimización de las RNN en secuencias con dependencias temporales de largo alcance.

En el caso de estudio planteado, se estudia la predicción de series de tiempo mediante la utilización de una RNN. Con este propósito, se emplea lo siguiente:

$$\begin{aligned} \boldsymbol{h}_t &= f_h(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t; \boldsymbol{\theta}_h) \\ &= f_h(\mathbf{W}_{hh}\boldsymbol{h}_{t-1} + \mathbf{W}_{xh}\boldsymbol{x}_t) \end{aligned} \tag{9}$$

$$\begin{aligned} \hat{\boldsymbol{y}}_t &= f_y(\boldsymbol{h}_t; \boldsymbol{\theta}_y) \\ &= f_y(\mathbf{W}_{hy}\boldsymbol{h}_t) \end{aligned} \tag{10}$$

donde,

- $f_h(x) = \sigma_{\tanh(x)}$

- $f_y(x) = x$
- $\boldsymbol{h}_t \in \mathbb{R}^{n_h}$
- $\mathbf{W}_{hh} \in \mathbb{R}^{n_h \times n_h}$
- $\mathbf{W}_{xh} \in \mathbb{R}^{n_x \times n_h}$
- $\mathbf{W}_{hy} \in \mathbb{R}^{n_h \times n_y}$
- $\boldsymbol{\theta}_h = \{\mathbf{W}_{hh}, \mathbf{W}_{xh}\}$
- $\boldsymbol{\theta}_y = \{\mathbf{W}_{hy}\}$

La función tanh se la puede expresar con exponenciales de la siguiente manera $\sigma_{\tanh(x)} = \left(\frac{e^x - e^{-x}}{e^x + e^{-x}}\right)$. Adicionalmente, $\boldsymbol{h}_t$ es el vector de estados ocultos en un tiempo t. $\mathbf{W}_{hh} \in \mathbb{R}^{n_h \times n_h}, \mathbf{W}_{xh} \in \mathbb{R}^{n_x \times n_h}$ y $\mathbf{W}_{hy} \in \mathbb{R}^{n_h \times n_y}$ son las matrices de pesos. Se utiliza $\boldsymbol{\theta}_h = \{\mathbf{W}_{hh}, \mathbf{W}_{xh}\}$ y $\boldsymbol{\theta}_y = \{\mathbf{W}_{hy}\}$ para denotar la transición de los estados y la de los parametros de estados a observaciones respectivamente (Ilhan et al., 2020).

Capítulo 3

Diseño Metodológico

En este capítulo, se presenta la estructura de las Redes Neuronales Recurrentes con el modelo de Markov Switching. Luego, se describe la red para múltiples regímenes internos basados en un HMM. Además, se presenta el algoritmo de aprendizaje secuencial.

3.1 Red Neuronal Recurrente con múltiples regímenes internos

Se describe la estructura RNN y HMM con múltiples regímenes. Los regímenes son los que se encargan de controlar la transición de estado oculta de forma independiente. Para este fin, las ecuaciones (9) y (10), que están de la forma tradicional, se modifican para adaptarlas al modelo de la siguiente manera

$$\begin{aligned} \boldsymbol{h}_t^{(k)} &= f_h\left(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t; \boldsymbol{\theta}_h^{(k)}\right) \\ &= f_h\left(\mathbf{W}_{hh}^{(k)} \boldsymbol{h}_{t-1} + \mathbf{W}_{xh}^{(k)} \boldsymbol{x}_t\right) \end{aligned} \tag{11}$$

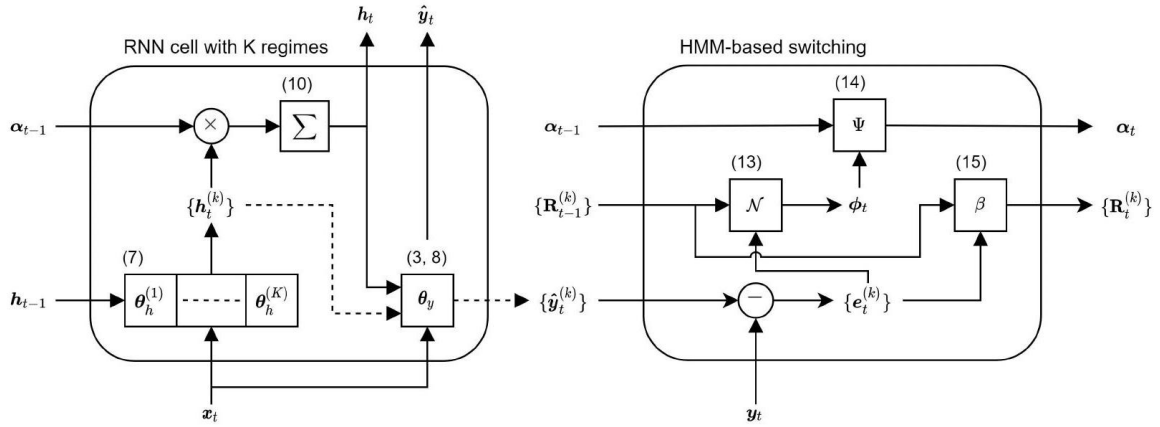
$$\begin{aligned} \hat{\boldsymbol{y}}_t^{(k)} &= f_y\left(\boldsymbol{h}_t^{(k)}; \boldsymbol{\theta}_y\right) \\ &= f_y\left(\mathbf{W}_{hy} \boldsymbol{h}_t^{(k)}\right) \end{aligned} \tag{12}$$

donde $k \in \{1, \dots, K\}$ es el índice de régimen y K es el número de regímenes.

En el lado izquierdo de la Figura 6se puede observar la celda modificada de la red neuronal recurrente con múltiples regímenes. En esta celda, el vector de estado oculto se desplaza al siguiente paso de tiempo, con pesos distintos $\boldsymbol{\theta}_h^{(k)}$ en cada nodo. Este paso lo realiza de forma independiente.

Figura 6

Esquema detallado de la celda RNN con HMM



Nota. Ilhan et al. (2020)

La estimación final del estado oculto en el paso de tiempo t se obtiene con la ecuación (13), la cual realiza el promedio ponderado de los estados ocultos de cada régimen, conociendo que $w_{t,k}$ es el peso para el k-ésimo régimen

$$\boldsymbol{h}_t = \sum_{k=1}^{K} w_{t,k} \boldsymbol{h}_t^{(k)} \tag{13}$$

Finalmente, se estima la predicción usando la ecuación (10). En la siguiente sección se muestra cómo se calcula los pesos $w_{t,k}$.

El número de regímenes, K, se considera un hiperparámetro y se puede seleccionar mediante validación cruzada.

3.2 Mecanismo de conmutación (switching) basado en HMM

Para realizar la conmutación entre los regímenes internos se utiliza un Modelo Oculto de Markov. Los pesos que se muestran en la ecuación (13) se representan mediante los valores de creencia del modelo de la siguiente manera:

$$\boldsymbol{h}_t = \sum_{k=1}^{K} \alpha_{t-1,k} \boldsymbol{h}_t^{(k)} \tag{14}$$

donde $\alpha_{t-1,k} \triangleq w_{t,k}$ denota la creencia para el k-ésimo régimen.

Ahora, para llevar a cabo la actualización de creencias como se indica en la ecuación (8), se necesita calcular los valores de probabilidad de $\boldsymbol{\phi}_t$ para el t-ésimo paso de tiempo después de observar $\boldsymbol{y}_t$. Adicionalmente, para la pérdida MSE, se considera el modelo de error con distribución gaussiana tal que

$$\boldsymbol{y}_t = \hat{\boldsymbol{y}}_t^{(k)} + \boldsymbol{e}_t^{(k)} \tag{15}$$

$$\boldsymbol{e}_t^{(k)} \sim \mathcal{N}\left(0, \mathbf{R}_{t-1}^{(k)}\right) \tag{16}$$

donde $\boldsymbol{e}_t^{(k)}$ es el vector de error y $\mathbf{R}_{t-1}^{(k)}$ es la matriz de error de covarianza para el k-ésimo régimen, la cual almacena los errores del régimen correspondiente hasta el t-ésimo paso de tiempo, excluyendo el último paso.

Luego se calcula la probabilidad mediante

$$p(\boldsymbol{y}_t \mid k_t = k) = \frac{1}{\sqrt{(2\pi)^{n_y}\left|\mathbf{R}_{t-1}^{(k)}\right|}} \exp\left(-\frac{1}{2}\boldsymbol{e}_t^{(k)^T}\mathbf{R}_{t-1}^{(k)^{-1}}\boldsymbol{e}_t^{(k)}\right) \tag{17}$$

Una vez que se obtiene las probabilidades, se actualiza el vector de creencias del régimen usando (8) como

$$\begin{aligned} \tilde{\boldsymbol{\alpha}}_t &= \boldsymbol{\phi}_t \odot (\Psi^T \boldsymbol{\alpha}_{t-1}) \\ \boldsymbol{\alpha}_t &= \frac{\tilde{\boldsymbol{\alpha}}_t}{\text{sum}(\tilde{\boldsymbol{\alpha}}_t)} \end{aligned} \tag{18}$$

donde se calcula $\boldsymbol{\phi}_t = [\ldots, p(\boldsymbol{y}_t \mid k_t = k), \ldots]^T$ con (17). Finalmente, se actualiza la matriz de error de covarianza usando suavizamiento exponencial mediante lo siguiente:

$$\mathbf{R}_t^{(k)} = (1-\beta)\mathbf{R}_{t-1}^{(k)} + \beta \boldsymbol{e}_t^{(k)} \boldsymbol{e}_t^{(k)^T} \quad (19)$$

donde $\beta \in [0,1)$ controla el efecto de suavizado, que se puede seleccionar mediante validación cruzada (Aydogan-Kilic & Selcuk-Kestel, 2023; Ilhan et al., 2020).

3.3 Algoritmo de aprendizaje

El siguiente algoritmo, presentado por Ilhan et al. (2020), muestra cómo se realiza la secuencia de aprendizaje en el modelo.

Algoritmo 3. Algoritmo de la secuencia de aprendizaje

Entrada: Series temporales de entrada y salida: $\{x_t\}_{t=1}^{T}$ y $\{y_t\}_{t=1}^{T}$.

Parámetros: parámetro de actualización de error de covarianza $\beta \in [0,1)$, Taza de aprendizaje $\eta \in \mathbb{R}^+$, número de épocas $n \in \mathbb{N}^+$, Tolerancia de parada $n_{"tolerance}$" $\in \mathbb{N}$, Duraciones de conjuntos de entrenamiento/validación T_{train} and T_{val} .

Salida: θ_{best}

1. Inicializar: θ (pesos).
2. Inicializar: $\boldsymbol{\theta}_{\text{best}} = \boldsymbol{\theta}$ (mejores pesos)
3. Inicializar: $v = \infty$ (menor perdida de validación)
4. Inicializar: $j = 0$ (contador para parada)
5. Para épocas $e = 1$ hasta n hacer

Fase de entrenamiento:

6. Para time step $t = 1$ hasta T_{train} hacer
7. Inicializar: $\boldsymbol{h}_1, \boldsymbol{\alpha}_1$ y $\left\{\mathbf{R}_t^{(k)}\right\}_{k=1}^{K}$.

Celda RNN Forward Pass:

8. Recibir $\boldsymbol{x}_t$
9. Para regímen $k = 1$ hasta K hacer
10. $\boldsymbol{h}_t^{(k)} = f_h\left(\mathbf{W}_{hh}^{(k)}\boldsymbol{h}_{t-1} + \mathbf{W}_{xh}^{(k)}\boldsymbol{x}_t\right)$
11. $\hat{\boldsymbol{y}}_t^{(k)} = f_y\left(\mathbf{W}_{hy}\boldsymbol{h}_t^{(k)}\right)$
12. Fin Para
13. $\boldsymbol{h}_t = \sum_{k=1}^{K} \alpha_{t-1,k}\boldsymbol{h}_t^{(k)}$
14. $\hat{\boldsymbol{y}}_t = f_y\left(\mathbf{W}_{hy}\boldsymbol{h}_t\right)$

Calcular pérdidas:

15. Recibir $\boldsymbol{y}_t$
16. $\boldsymbol{e}_t = \boldsymbol{y}_t - \hat{\boldsymbol{y}}_t$
17. $\ell_{\text{MSE}}(\boldsymbol{y}_t, \hat{\boldsymbol{y}}_t) = \boldsymbol{e}_t^T\boldsymbol{e}_t$

Backward Pass:

18. Actualice los pesos del modelo a través de la propagación hacia atrás usando

$\frac{\partial \ell}{\partial \theta}\,\Psi_{ij} \leftarrow \exp\left(\Psi_{ij}\right)/\sum_{j'=1}^{K}\exp\left(\Psi_{ij'}\right)$

Base HMM para conmutación (Switching):

19. $\boldsymbol{\phi}_t = [\ldots, p(\boldsymbol{y}_t \mid k_t = k), \ldots,]^T$
20. $\tilde{\boldsymbol{\alpha}}_t = \boldsymbol{\phi}_t \odot (\Psi^T\boldsymbol{\alpha}_{t-1})$
21. $\boldsymbol{\alpha}_t = \frac{\tilde{\alpha}_t}{\text{sum}(\tilde{\alpha}_t)}$
22. $\boldsymbol{e}_t^{(k)} = \boldsymbol{y}_t - \hat{\boldsymbol{y}}_t^{(k)}$
23. $\mathbf{R}_t^{(k)} = (1-\beta)\mathbf{R}_{t-1}^{(k)} + \beta\boldsymbol{e}^{(k)}\boldsymbol{e}^{(k)^T}$

24. Fin Para

Fase de Validación

25. $L_{val} = 0$ (validación de las pérdidas)

26. for time step $t = T_{\text{train}}$ hasta $T_{\text{train}} + T_{\text{val}}$, hacer

27. Hacer predicciones $\hat{\boldsymbol{y}}_t$

28. $L_{val} = L_{val} + \ell_{\text{MSE}}(\boldsymbol{y}_t, \hat{\boldsymbol{y}}_t)$

29. Fin para

30. $\bar{L}_{val} = \frac{L_{val}}{T_{val}}$

31. si $\bar{L}_{val} < v$, entonces

32. $v = \bar{L}_{val}$

33. $\boldsymbol{\theta}_{\text{best}} = \boldsymbol{\theta}$

34. $j = 0$

35. Si no

36. $j = j + 1$

37. Fin si

38. Si $j > n_{\text{tolerance}}$, entonces

39. Retorna $\boldsymbol{\theta}_{\text{best}}$

40. Fin si

41. Fin para.

42. Retorna θ_{best}

El algoritmo sigue la siguiente secuencia, en primer lugar, se inicializa los pesos del modelo, los estados ocultos, el vector del régimen de creencia y las matrices de error de covarianzas.

Para una secuencia dada con longitud temporal T, después de recibir la entrada x_t en cada paso de tiempo t, se calcula los estados ocultos para cada régimen interno usando la ecuación (11). Luego, se utiliza la ecuación (12) y los estados ocultos de cada régimen para realizar la predicción de la salida. Después del forward-pass de cada régimen interno, se generan h_t y la predicción $\hat{\boldsymbol{y}}$, usando (10) y (3). Después de recibir la salida objetivo y_t, calculamos la pérdida usando

$$L_{MSE} = \frac{1}{T}\sum_{t=1}^{T} \ell_{MSE}(\boldsymbol{y_t}, \hat{\boldsymbol{y_t}})$$

donde

$$\ell_{MSE}(\boldsymbol{y_t}, \hat{\boldsymbol{y_t}}) = e_t^T e_t$$

Adicionalmente, se actualizan los pesos mediante backpropagation de las derivadas. Las derivadas básicas utilizadas se muestran a continuación:

$$\begin{aligned}
\frac{\partial \ell_t}{\partial \hat{\boldsymbol{y}}_t} &= -2\boldsymbol{e}_t^T \\
\frac{\partial \hat{\boldsymbol{y}}_t}{\partial \boldsymbol{h}_t} &= \mathbf{W}_{hy} \\
\frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_t^{(k)}} &= \alpha_{t-1,k} \\
\frac{\partial \boldsymbol{h}_t^{(k)}}{\partial \boldsymbol{h}_{t-1}} &= \mathbf{W}_{hh}^{(k)} \odot \operatorname{diag}\left(f_h'\left(z_t^{(k)}\right)\right)
\end{aligned}$$

$$\frac{\partial \boldsymbol{h}_t}{\partial \alpha_{t-1,k}} = \boldsymbol{h}_t^{(k)}$$
$$\frac{\partial \alpha_{t,k}}{\partial \tilde{\alpha}_{t,k'}} = \frac{\delta_{kk'}\text{sum}\,(\tilde{\boldsymbol{\alpha}}_t) - \tilde{\alpha}_{t,k}}{\text{sum}\,(\tilde{\boldsymbol{\alpha}}_t)^2}$$

donde $z_t^{(k)} = \mathbf{W}_{hh}^{(k)}\boldsymbol{h}_{t-1} + \mathbf{W}_{xh}^{(k)}\boldsymbol{x}_t$. (Aydogan-Kilic & Selcuk-Kestel, 2023; Ilhan et al., 2020)

Capítulo 4

Análisis y Discusión de los Resultados

En este capítulo se proporciona detalles sobre los procedimientos y tratamientos que se realizaron a los datos. Se realiza una exploración inicial, luego, se revisa si existen valores ausentes y valores atípicos, se realiza normalización de estos y finalmente la validación de los datos. Adicionalmente, se realiza el análisis de los resultados obtenidos. Para todo este proceso se utiliza el lenguaje de programación Python.

4.1 Preparación, edición y limpieza de los datos.

Para este trabajo, se utiliza información sobre las tasas de cambio entre diferentes monedas que se actualizan diariamente. Los valores que se utiliza en base al dólar estadounidense (USD) y se compara con la libra esterlina (GBP), euro (EUR), franco suizo (CHF) y dólar canadiense (CAD). Los valores que se emplean abarcan el lapso de tiempo desde el 1 de enero de 2019 hasta el 1 de septiembre de 2022. El valor tomado para el estudio que se plantea de cada tasa de cambio es el de cierre del día, teniendo 1340 observaciones (ver Tabla 2)

Tabla 2

Algunos valores de las variables

Tiempo	EUR/USD	GBP/USD	USD/CAD	USD/CHF
1/1/19	1.14612	1.26060	1.36347	0.99
2/1/19	1.13121	1.26278	1.35763	0.98669
3/1/19	1.13899	1.27170	1.34879	0.98616
⋮	⋮	⋮	⋮	⋮

30/8/22	1.00234	1.16202	1.31263	0.97728
31/8/22	1.0038	1.15427	1.31538	0.98126
1/9/22	0.99514	1.15005	1.31239	0.9811

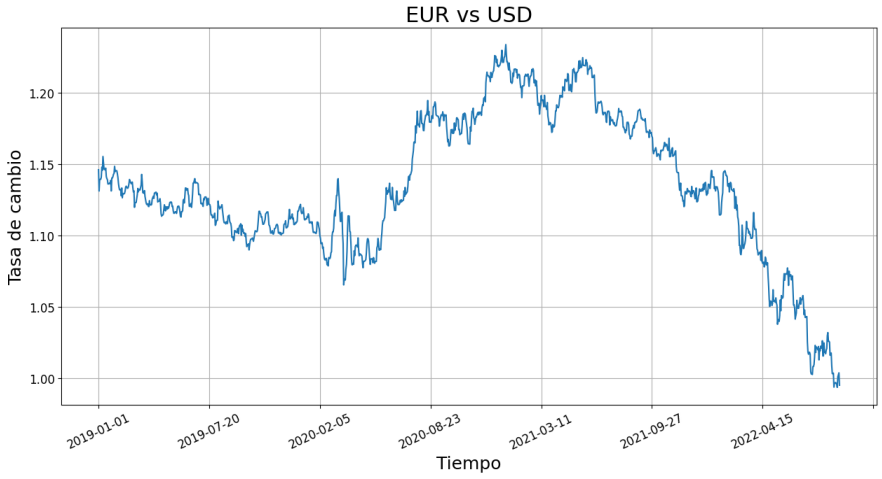
Los datos fueron obtenidos de la página web www.dukascopy.com, en la cual se proporciona información y noticias sobre Forex (Foreing Currencies Exchange en inglés) o también conocido como mercado de divisas.

4.2 Análisis exploratorio

En la gráfica de euro vs dólar estadounidense (EUR/USD) que se muestran en la Figura 7, se puede observar que los datos tienen una tendencia decreciente desde enero de 2019 hasta aproximadamente marzo de 2020. Luego, cambia la tendencia y se vuelve creciente hasta abril del 2021. Finalmente, se observa una tendencia decreciente hasta agosto de 2022, donde se aprecia que llegan a contar con valores similares el dólar y el euro.

Figura 7

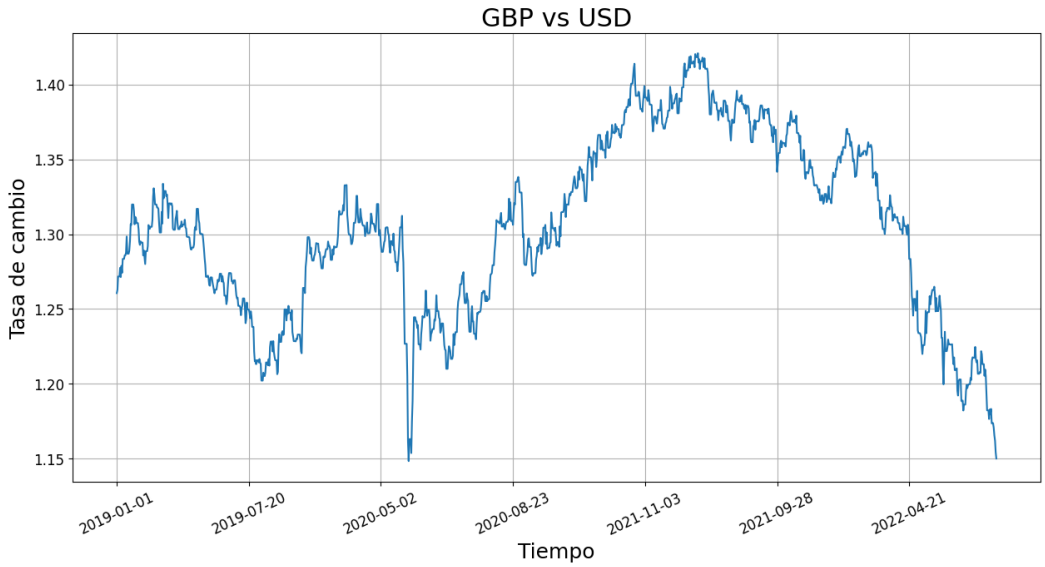
Serie de tiempo de los valores de tasa de cambio entre euros y dólares estadounidenses entre el 1 de enero de 2019 a 1 de agosto de 2022.



Para la serie de tiempo de libra esterlina vs dólar estadounidense (GBP/USD) se presenta una estacionalidad hasta aproximadamente mayo de 2020. También, se observa un decrecimiento grande alrededor del mes de mayo del 2020. Luego, la serie crece hasta junio del 2021 y finalmente tiene un decrecimiento de su valor (Figura 8).

Figura 8

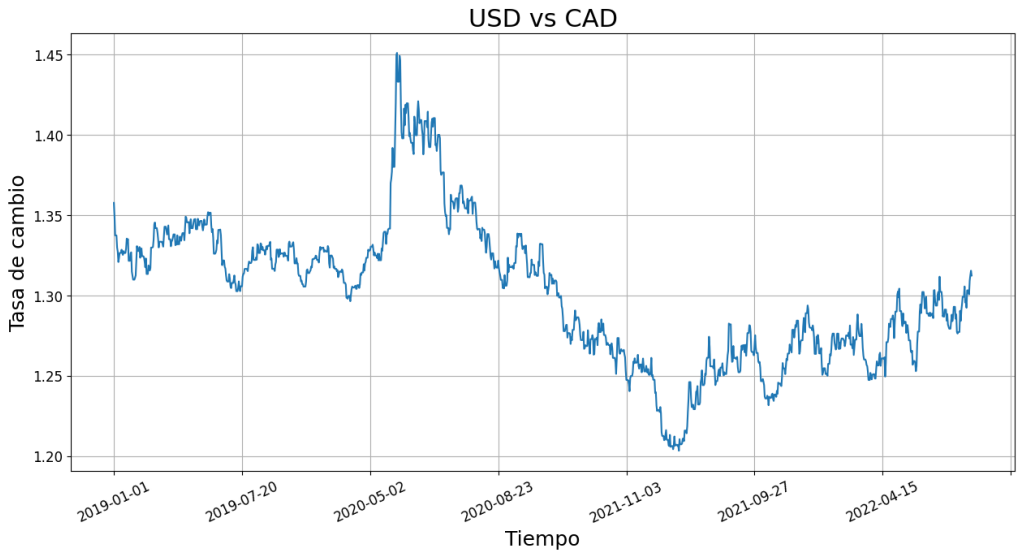
Serie de tiempo de los valores de tasa de cambio entre libra esterlina y dólares estadounidenses entre el 1 de enero de 2019 a 1 de agosto de 2022.



En cambio, en la Figura 9 se muestran los valores de dólar estadounidense vs dólar canadiense (USD/CAD) tienen una tendencia lineal hasta mayo de 2020, sigue con un crecimiento pronunciado para seguir con un decrecimiento hasta mayo del 2021. Finalmente, se tiene un crecimiento leve hasta el final de la serie.

Figura 9

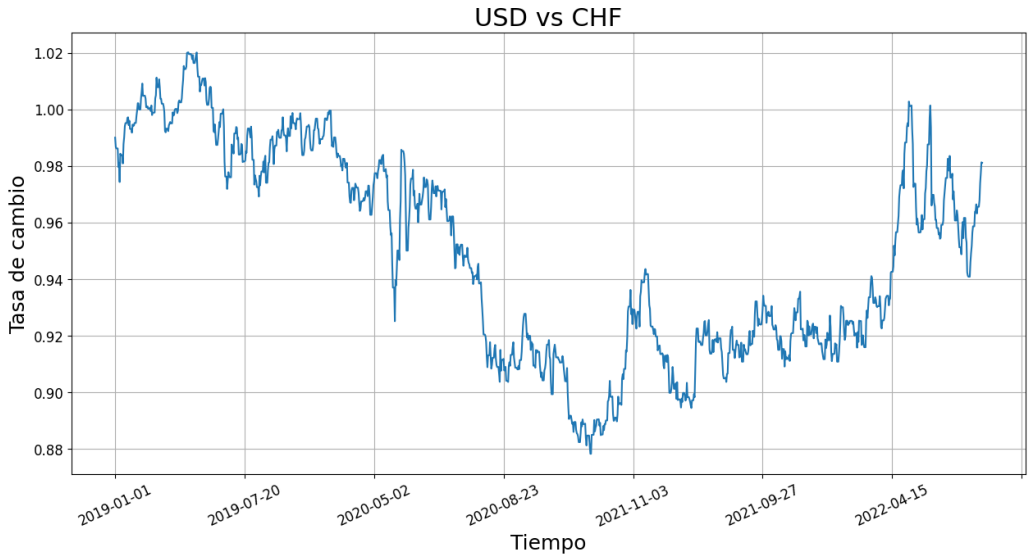
Serie de tiempo de los valores de tasa de cambio entre dólar estadounidenses y dólar canadiense entre el 1 de enero de 2019 a 1 de agosto de 2022.



En la serie de valores de dólar estadounidense vs franco suizo (USD/CHF) inicia con un decrecimiento de la serie, para luego crecer hasta el final. (ver Figura 10)

Figura 10

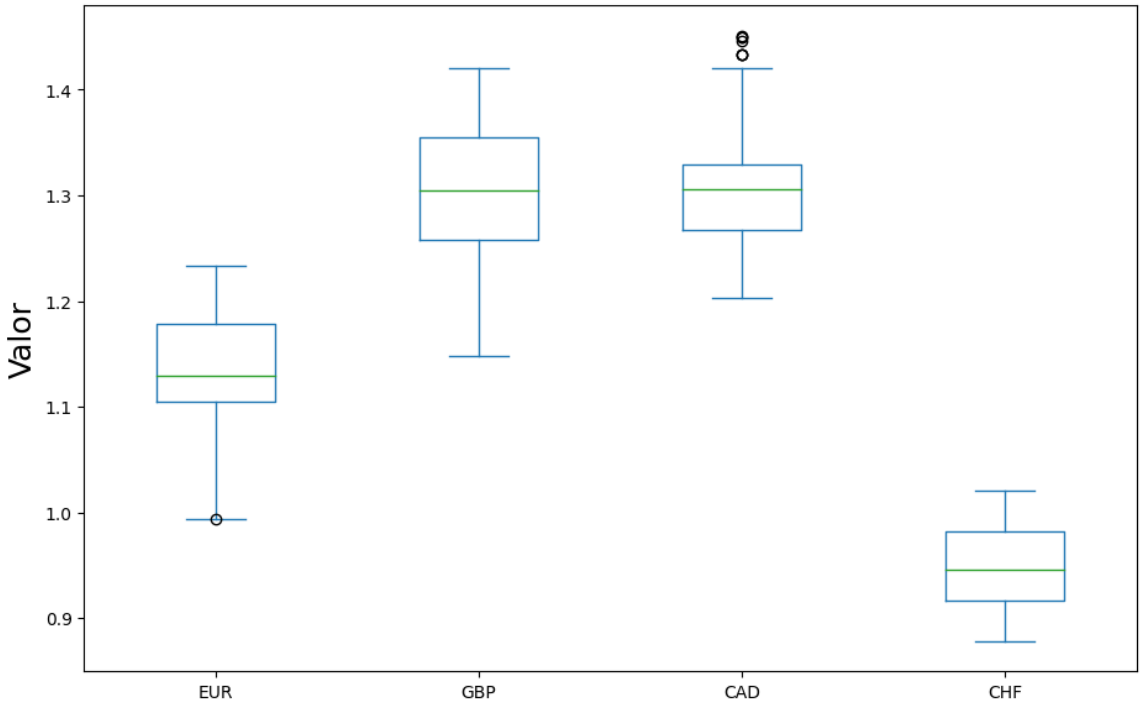
Serie de tiempo de los valores de tasa de cambio entre dólar estadounidenses y franco suizo entre el 1 de enero de 2019 a 1 de agosto de 2022.



Adicional a la gráfica de las series de tiempo, se realiza un diagrama de cajas para analizar algunas características de los datos (ver Figura 11). Se observa que los datos de EUR/USD están entre 1.0 a 1.25 aproximadamente, los de GBP/USD entre 1.20 y 1.40, los de USD/CAD entre 1.40 y 1.20 y USD/CHF entre 0.90 y 1.00. Además, el 50% de los datos son mayores a 1.10 y menores a 1.20. Finalmente, existe un valor atípico (menor a 1.00) en los EUR que está muy cercano al límite inferior y varios valores atípico en la gráfica de CAD.

Figura 11

Diagrama de cajas de las variables.

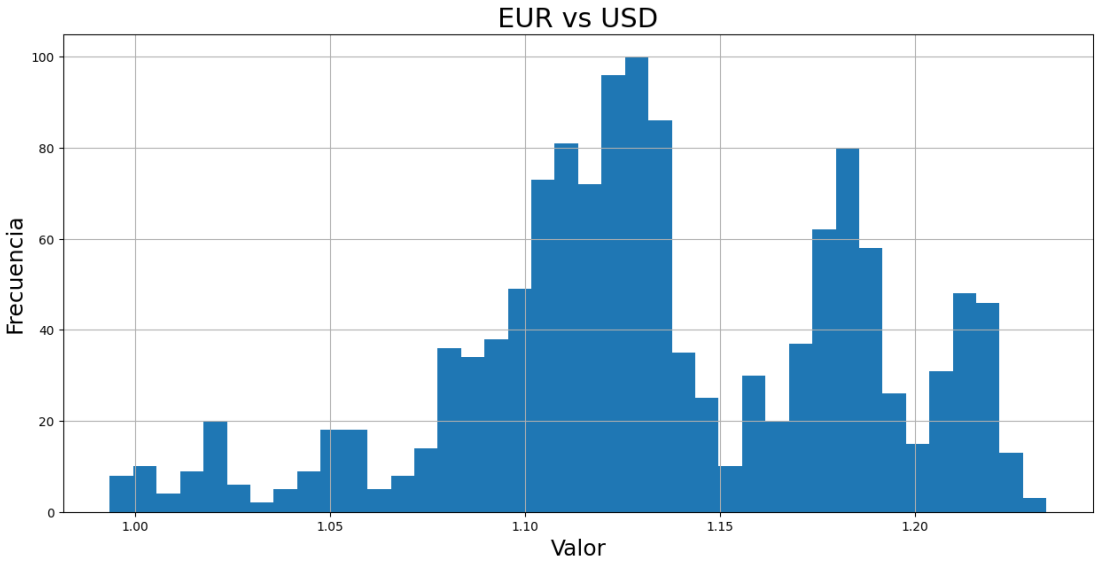


Del mismo modo, se realiza histogramas de las diferentes tasas de cambio. Para EUR/USD (ver Figura 12) en el cual se puede observar la distribución de los valores utilizados. En La mayoría de los valores están entre 1.10 y 1.15, esto sugiere que en la mayor parte del tiempo el euro tiene mayor valor que el dólar. También, se puede observar valores con poca frecuencia a la izquierda. Estos valores son menores a 1.0, esto nos indica que en

algún momento el dólar tuvo mayor valor que el euro. Finalmente, se puede concluir que, en este periodo de tiempo, el euro no tuvo valores mayores a 1.25 con respecto al dólar americano.

Figura 12

Histograma de euros vs dólares.

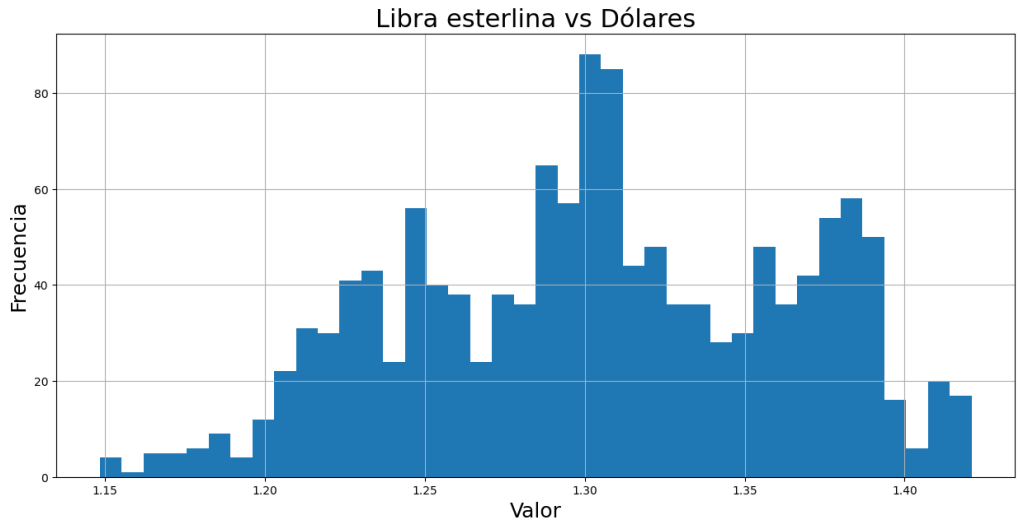


Para GBP/USD (ver Figura 13) en la mayor parte de los valores están entre 1.20 y 1.40, de lo que se concluye que la mayor parte del tiempo la libra esterlina tiene mayor valor que el dólar. Además, se puede observar valores con poca frecuencia a la izquierda del histograma, estos valores son menores a 1.20. Finalmente, en este periodo de tiempo, la libra esterlina no superó valores mayores a 1.45 con respecto al dólar estadounidense.

En cambio, en la Figura 14 se observa la serie USD/CAD la cual tiene valores con poca frecuencia a la derecha de 1.35. La mayoría de los valores están alrededor del valor 1.30.

Figura 13

Histograma de libra esterlina vs dólares.



En la Figura 15, se muestra la serie para USD/CHF se observa dos agrupaciones, la primera se encuentra alrededor de 0.91 y la segunda se encuentra alrededor de 0.98. Además, se puede observar pocos valores con poca frecuencia a la izquierda del histograma, estos valores son menores a 0.90. Finalmente, el franco suizo pocas veces superó al dólar estadounidense.

Figura 14

Histograma de dólar estadounidense vs dólar canadiense.

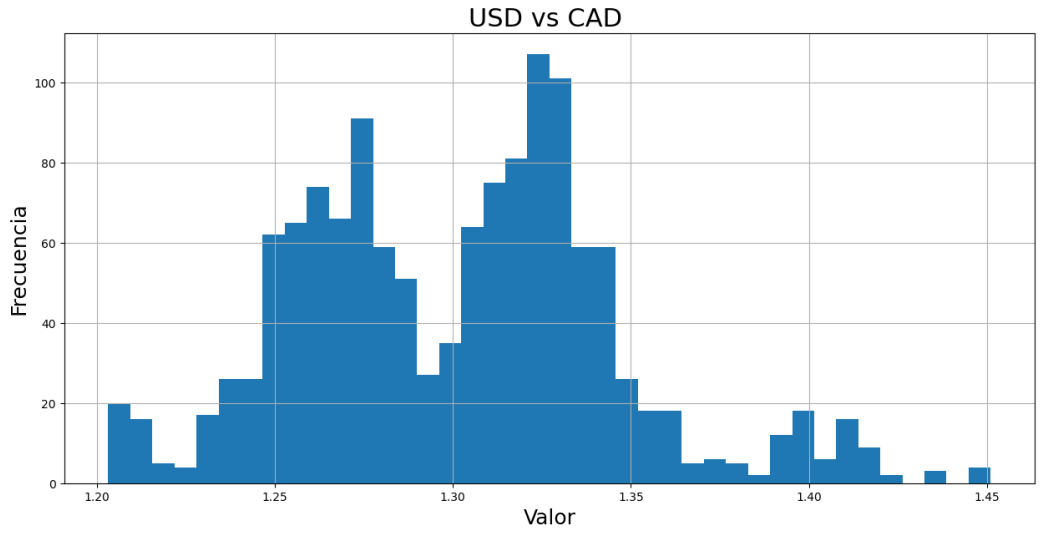
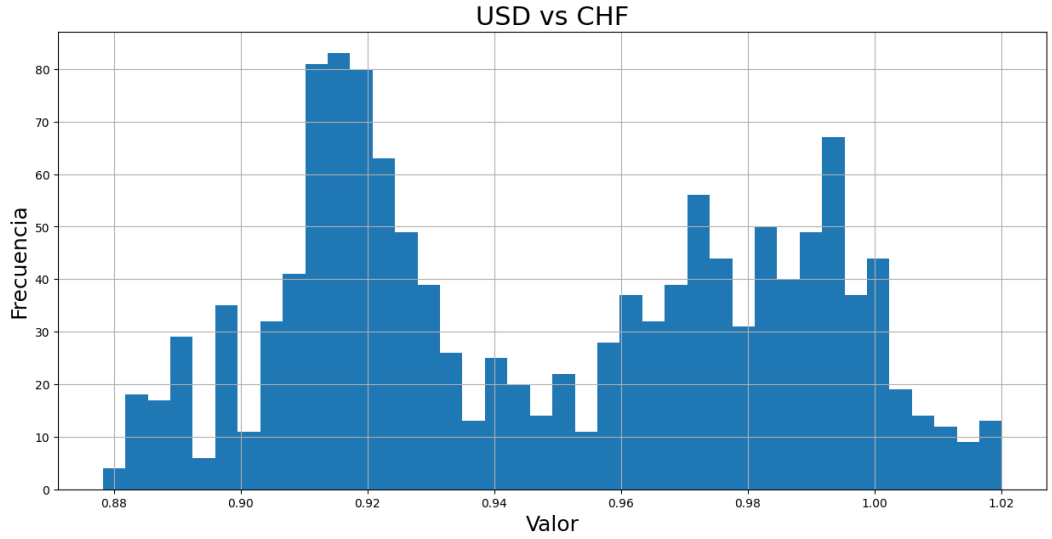


Figura 15

Histograma de dólar estadounidense vs franco suizo.



Además, en esta sección se realiza un resumen estadístico de los datos. Se puede observar que, en una parte del periodo de tiempo en estudio, el dólar tuvo mejor valor que el euro, el valor que se tuvo fue 0.99 dólares por cada euro; del mismo modo el franco suizo con respecto al dólar americano. Los valores máximos alcanzados fueron los siguientes: 1.23 dólares por euro, 1.42 dólares estadounidenses por libra esterlina, 1.45 dólares canadienses por dólar estadounidense y 1.02 franco suizo por dólar. Adicionalmente, cada euro estuvo la mayor parte del tiempo en un valor mayor que 1.18 dólares, la libra esterlina mayor a 1.35 dólares, los dólares canadienses a 1,45 por dólar y el franco suizo estuvo a 0.98 respecto al dólar americano.

Tabla 3

Resumen estadístico de las diferentes variables

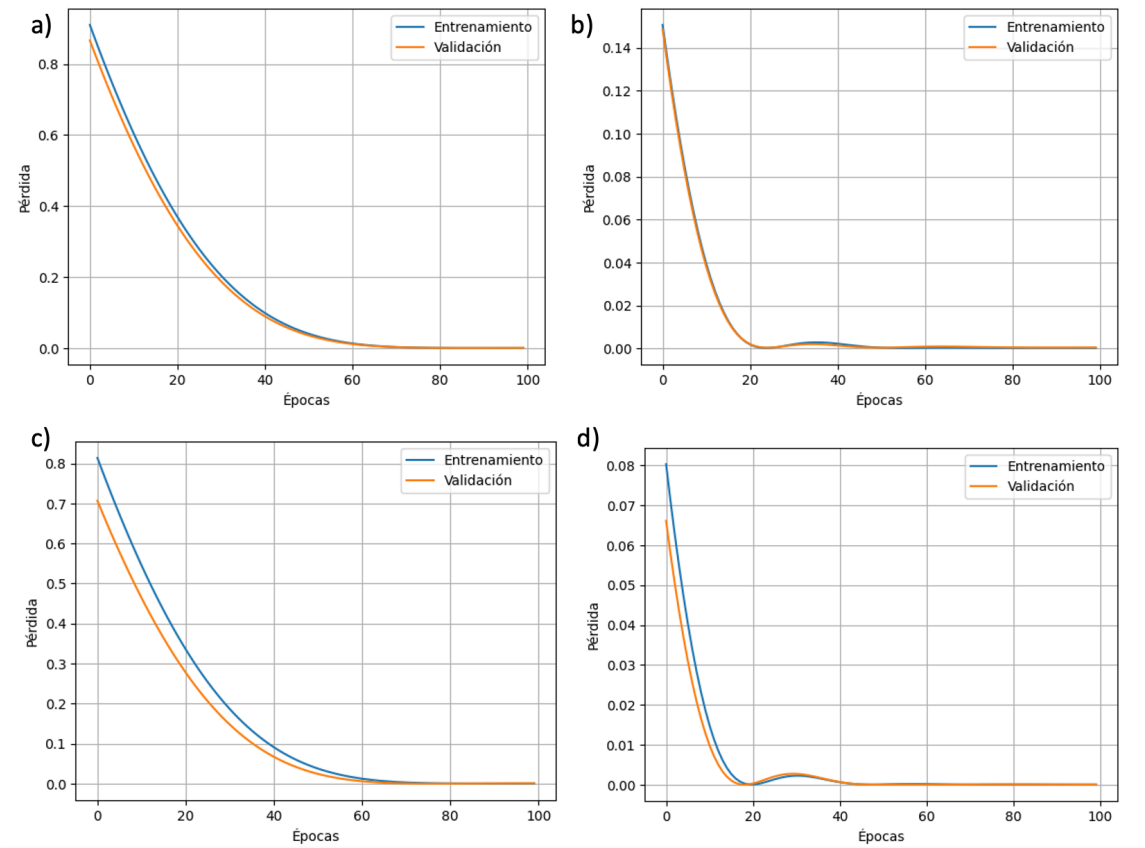
	EUR	GBP	CAD	CHF
Mínimo	0.99	1.15	1.20	0.88
1er cuartil	1.10	1.26	1.27	0.92
Mediana	1.12	1.30	1.31	0.95
Media	1.13	1.30	1.30	0.99
3er cuantil	1.18	1.36	1.33	0.98
Máximo	1.23	1.42	1.45	1.02
Desviación estándar	0.05	0.06	0.04	0.04

4.3 Resultados

El modelo está alimentado con datos históricos, los datos se los divide en 60% para el entrenamiento y 40% para la validación. Luego del entrenamiento y validación del modelo, se realiza un gráfico de las pérdidas de la fase de entrenamiento y validación para todas las variables en estudio. En la Figura 16, el gráfico de pérdida muestra un inicio turbulento, con valores altos que reflejan la incertidumbre y la falta de dirección en las primeras etapas del entrenamiento. A medida que avanza el entrenamiento, se observa que la pérdida disminuye hasta estabilizarse aproximadamente en el valor cero para todas las variables en estudio.

Figura 16

Gráfico de pérdidas: a) EUR/USD, b) GBP/USD, c) USD/CAD, d) USD/CHF



El entrenamiento se lo realizó con 300 épocas, pero al ver los resultados se puede concluir que no se necesitaba más de 100 épocas para que la pérdida decrezca hasta los niveles necesarios, indicando que la red está aprendiendo de sus errores y ajustando sus pesos y conexiones para mejorar su rendimiento.

Luego de que se ajustó la pérdida del modelo, se realizan las predicciones (ver figuras 17, 18, 19 y 20). Las series de tiempo originales y las series de tiempo de predicción tiene similitud. Al parecer, el modelo está ajustando bien los parámetros necesarios para realizar las predicciones.

Figura 17

Gráfico de EUR/USD original y la predicción.

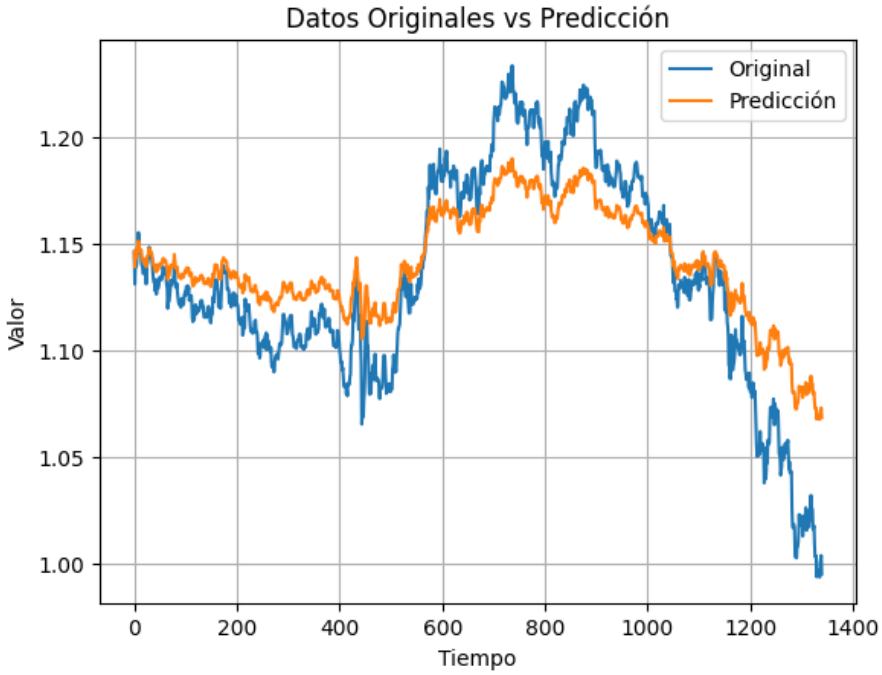


Figura 18

Gráfico de GBP/USD original y la predicción.

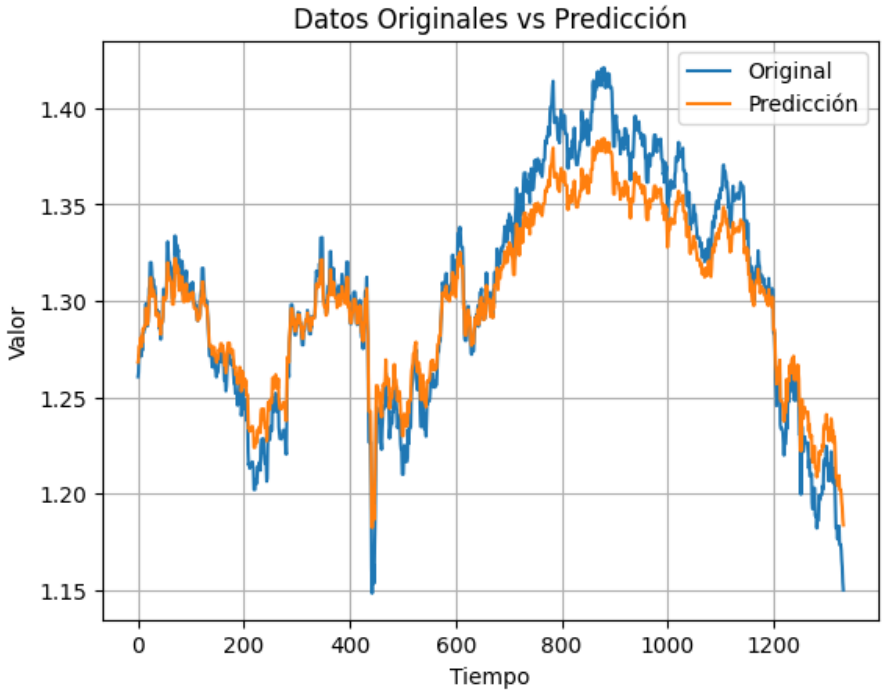


Figura 19

Gráfico de USD/CAD original y la predicción.

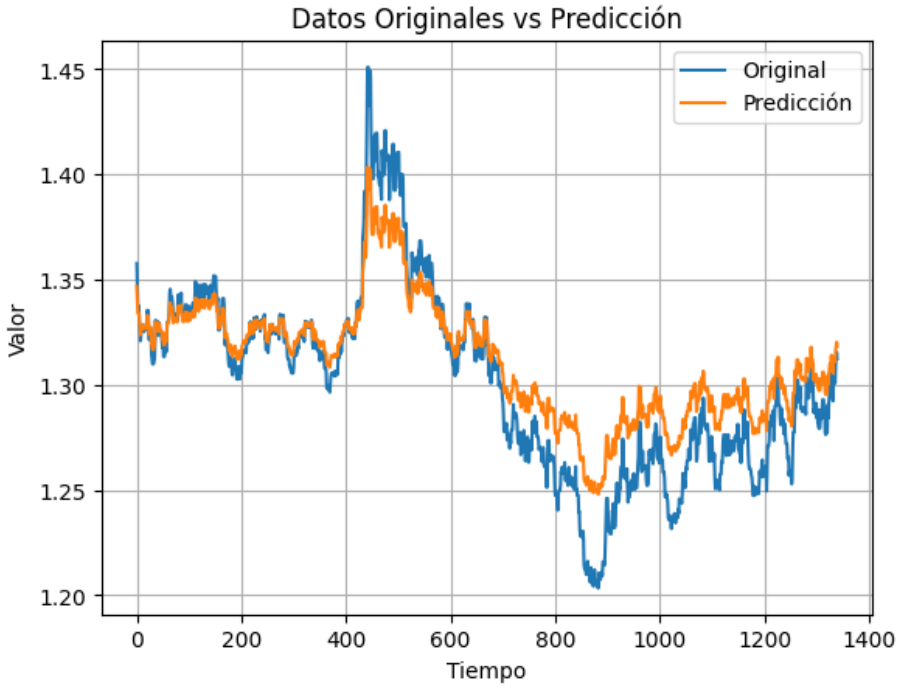
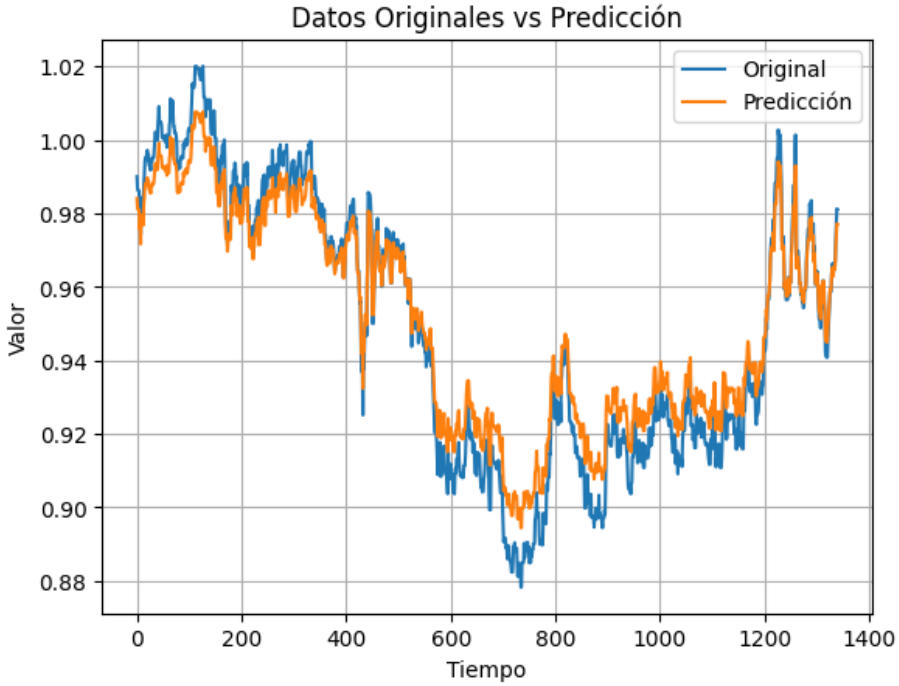


Figura 20

Gráfico de USD/CHF original y la predicción.



La siguiente tabla (Tabla 4) presenta las predicciones generadas por el modelo. Estas predicciones representan las estimaciones del modelo sobre las diferentes variables en estudio, las cuales proporcionan una visión útil para la toma de decisiones en el futuro.

Tabla 4

Predicciones de las variables.

Tiempo	EUR/USD	GBP/USD	USD/CAD	USD/CHF
1/1/19	1.14319	1.27164	1.36347	0.98234
2/1/19	1.13448	1.27300	1.35763	0.97991
3/1/19	1.13904	1.27855	1.34879	0.97952
⋮	⋮	⋮	⋮	⋮
30/8/22	1.05610	1.20888	1.31263	0.97300
31/8/22	1.05702	1.20383	1.31538	0.97592
1/9/22	1.05156	1.20108	1.31239	0.97580

Para evaluar la precisión de las predicciones realizadas por el modelo, ya sea para el entrenamiento y la validación, se utiliza el error cuadrático medio (MSE) y la raíz cuadrada del error cuadrático medio (RMSE) como se muestra en las siguientes ecuaciones:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y_i})^2 \qquad RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y_i})^2}$$

Los valores obtenidos se los muestra en la Tabla 5.

Tabla 5

MSE y RMSE de los datos de entrenamiento y validación de cada variable.

	Entrenamiento		Validación	
	MSE	RMSE	MSE	RMSE
EUR/USD	0.0004114	0.0202864	0.0009749	0.0252855
GBP/USD	0.0004491	0.0211912	0.0014362	0.0290730
USD/CAD	0.0002111	0.0145306	0.0008382	0.0215594
USD/CHF	0.0004177	0.0204375	0.0003739	0.0199999

Los valores de MSE y RMSE en todas las variables son bajos, esto quiere decir que las predicciones del modelo se ajustan de manera adecuada a los datos reales.

Capítulo 5

Marco Propositivo

5.1 Planificación de la Actividad Preventiva

En la actualidad, el crecimiento acelerado de información, entre ellos datos secuenciales, y la necesidad de obtener el mayor beneficio de estos, han dado a la creación de nuevas técnicas de tratamiento de datos. Un caso particular es, el desarrollo de modelos híbridos. En nuestro caso se fusionaron los Modelos de Markov con Cambio de Estado y las Redes Neuronales Recurrentes para capturar tanto las dependencias temporales a largo plazo como los cambios estructurales en los datos secuenciales.

Este modelo proporcionar una herramienta versátil y eficaz que se lo puede utilizar en varias áreas como las finanzas, predicciones de clima, biología, entre otros. Además, se desea desarrollar un modelo que integre la capacidad de adaptación de las RNN con la capacidad de capturar cambios estructurales de los Modelos Markov con Cambio de Estado. Así, se puede mejorar la capacidad predictiva y la robustez del modelo cuando tengan cambios significativos en su estructura.

Para la implementación, se debe tener algun software o entorno de programación como Python para poder realizar el modelo y las simulaciones. Adicionalmente, se puede crear una interfaz de usuario simple de utilizar, que permita a los usuarios entrenar el modelo con sus propios datos y realizar predicciones de manera intuitiva.

Lo que se espera de estos modelos es tener una mejora significativa en la capacidad predictiva en comparación con modelos convencionales de RNN o Modelos Markov con Cambio de Estado por separado. Asimismo, aplicar el modelo a diferentes conjuntos de datos de series temporales, con eso poder demostrar su utilidad en diferentes campos.

Finalmente, se espera contar con el modelo en código abierto, para que toda la comunidad en general, pueda hacer uso de esta herramienta.

Conclusiones

Se presenta la combinación de Modelos Markov con Cambio de Estado y Redes Neuronales Recurrentes, la cual permite al modelo adaptarse a cambios en la estructura de los datos secuenciales, lo que lo hace robusto frente a fluctuaciones y evoluciones en el comportamiento de los datos. Además, el modelo mejora las predicciones en casos donde se requiere capturar tanto dependencias temporales a largo plazo como cambios estructurales en los datos.

Al realizar un análisis exploratorio de los datos se puede observar que las divisas en el último año tienen una tendencia a decrecer con respecto al dólar estadounidense. Adicionalmente, se tienen pocos datos atípicos en las variables, la mayor parte de datos atípicos se concentran en la serie USD/CAD. Además, se puede ver que las variables no tienen una distribución normal.

Las medidas de bondad de ajuste utilizadas MSE y RMSE, nos muestran errores bajos en cada una de las variables en estudio. Esto quiere decir que el modelo está funcionando de forma adecuada. Esta afirmación se puede contrastar con las figuras en las que se compara la serie original y las predicciones del modelo.

Dependiendo de la implementación y el tamaño de los datos, el modelo puede presentar una complejidad computacional más alta en comparación con otros enfoques, lo que podría requerir recursos computacionales adicionales para su entrenamiento y validación.

Referencias Bibliográficas

Aleksander, I., & Morton, H. (1990). *Introduction to Neural Computing*.

Aydogan-Kilic, D., & Selcuk-Kestel, A. S. (2023). Modification of hybrid RNN-HMM model in asset pricing: univariate and multivariate cases. *Applied Intelligence*, *53*(20), 23812–23833. https://doi.org/10.1007/s10489-023-04762-7

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning Long-Term Dependencies with Gradient Descent is Difficult. *IEEE TRANSACTIONS ON NEURAL NETWORKS*, *5*(2).

Chang-Jin, K., & Charles, N. (1999). State-Space Models with Regime Switching: Classical and Gibbs-Sampling Approaches with Applications. *MIT Press Books*, *1*.

Chiu, C. C., Narayanan, A., Han, W., Prabhavalkar, R., Zhang, Y., Jaitly, N., Pang, R., Sainath, T. N., Nguyen, P., Cao, L., & Wu, Y. (2021). RNN-T Models Fail to Generalize to Out-of-Domain Audio: Causes and Solutions. *2021 IEEE Spoken Language Technology Workshop, SLT 2021 - Proceedings*, 873–880. https://doi.org/10.1109/SLT48900.2021.9383518

Du, J., Vong, C. M., & Philip Chen, C. L. (2021). Novel Efficient RNN and LSTM-Like Architectures: Recurrent and Gated Broad Learning Systems and Their Applications for Text Classification. *IEEE Transactions on Cybernetics*, *51*(3), 1586–1597. https://doi.org/10.1109/TCYB.2020.2969705

Guttorp, P. (1995). *Stochastic Modeling of Scientific Data*. https://doi.org/https://doi.org/10.1201/9780203738252

Hamilton, J. D. (1989). A New Approach to the Economic Analysis of Nonstationary Time Series and the Business Cycle. *Econometrica*, *57*(2), 357–384. http://www.jstor.org/stable/1912559

Haykin, S. (2009). *Neural networks and learning machines*. Prentice Hall/Pearson.

Hochreiter, J. (1991). *Untersuchungen zu dynamischen neuronalen Netzen* [Diploma thesis]. technische universitat munchen.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neuronal Computation*, 1735–1780.

Ilhan, F., Karaahmetoglu, O., Balaban, I., & Kozat, S. S. (2020). Markovian RNN: An adaptive time series prediction network with HMM-based switching for nonstationary environments. *IEEE Transactions on Neural Networks and Learning Systems*.

Kalashnikov, V. V. (1994). Mathematical Methods in Queuing Theory. En *Mathematical Methods in Queuing Theory*. Springer Netherlands. https://doi.org/10.1007/978-94-017-2197-4

Katriel, G. (2013). Gambler's ruin probability-A general formula. *Statistics and Probability Letters*, *83*(10), 2205–2210. https://doi.org/10.1016/j.spl.2013.06.005

Liisberg, J., Møller, J., Bloem, H., Cipriano, J., Mor, G., & Madsen, H. (2016). Hidden Markov Models for indirect classification of occupant behaviour. *Sustainable Cities and Society*, *27*, 83–98. https://doi.org/10.1016/j.scs.2016.07.001

Lim, B., Zohren, S., & Roberts, S. (2019). *Recurrent Neural Filters: Learning Independent Bayesian Filtering Steps for Time Series Prediction*. http://arxiv.org/abs/1901.08096

Liu, J., Zhu, L., Wang, Y., Liang, X., Hyyppä, J., Chu, T., Liu, K., & Chen, R. (2015). Reciprocal estimation of pedestrian location and motion state toward a smartphone geo-context computing solution. *Micromachines*, *6*(6), 699–717. https://doi.org/10.3390/mi6060699

Makino, T., Liao, H., Assael, Y., Shillingford, B., Garcia, B., Braga, O., & Siohan, O. (2019). Recurrent Neural Network Transducer for Audio-Visual Speech Recognition. *2019 IEEE Automatic Speech Recognition and Understanding Workshop, ASRU 2019 - Proceedings*, 905–912. https://doi.org/10.1109/ASRU46091.2019.9004036

Martens, J., & Sutskever, I. (2011). Learning Recurrent Neural Networks with Hessian-Free Optimization Ilya Sutskever. *In ICML-28*, 1033–1040.

Nguyen, B. A. (2020). *Markov Regime-switching in Forecasting Models*. Carleton University.

Parthiban, R., Ezhilarasi, R., & Saravanan, D. (2020, julio 3). Optical Character Recognition for English Handwritten Text Using Recurrent Neural Network. *2020 International Conference on System, Computation, Automation and Networking, ICSCAN 2020*. https://doi.org/10.1109/ICSCAN49426.2020.9262379

Rocha, A. L., & Stern, F. (1999). The gambler's ruin problem with n players and asymmetric play. En *Statistics & Probability Letters* (Vol. 44).

Rosenthal, R. W., Murray, H., & Rubinstein, A. (1984). Repeated Two-Player Games with Ruin 1. *International Journal of Game Theory*, *13*, 155–177.

Seforzo, R. (2009). *Basics of applied stochastic processes*. Springer Science & Business Media. https://doi.org/10.1007/978-3-540-89332-5

Staudemeyer, R. C., & Morris, E. R. (2019). *Understanding LSTM -- a tutorial into Long Short-Term Memory Recurrent Neural Networks*. http://arxiv.org/abs/1909.09586

Sutskever, I. (2013). *Training Recurrent Neural Networks*. University of Toronto.

Tornero, J. (2017). *Machine Learning: Modelos Ocultos de Markov (HMM) y Redes Neuronales Artificiales (ANN)*.

Wang, C., Jiang, F., & Yang, H. (2017). A hybrid framework for text modeling with convolutional RNN. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *Part F129685*, 2061–2070. https://doi.org/10.1145/3097983.3098140

Werbos, P. J. (1990). Backpropagation Through Time: What It Does and How to Do It. *Proceedings of the IEEE*, *78*(10), 1550–1560.

Yen-Chi, C. (2018). *Stochastic Modeling of Scientific Data*.

Apéndice

Apéndice A. Código en Python del modelo

En el modelo se utilizó las librerías hmmlearn y Pytorch

```
# Entrenar el modelo oculto de Markov (HMM)
model_hmm = hmm.GaussianHMM(n_components=2, covariance_type="tied",
algorithm="map", n_iter=100, implementation="scaling")

model_hmm.fit(data)
```

```
# Definir la arquitectura de la RNN
input_size = 1
hidden_size = 16
output_size = 1
rnn = torch.nn.RNN(input_size, hidden_size, batch_first=True)
criterion = torch.nn.MSELoss()
optimizer = torch.optim.Adam(rnn.parameters(), lr=0.001)
```

```
# Entrenamiento de la RNN
epochs = 300
train_losses = []
val_losses = []
for epoch in range(epochs):
    rnn.train()
    optimizer.zero_grad()
    output, _ = rnn(torch.tensor(train_data).view(1, -1, 1).float())
    loss = criterion(output.view(-1),
torch.tensor(train_data).view(1,-1,1).float())
    loss.backward()
    optimizer.step()
    train_losses.append(loss.item())

    rnn.eval()
    val_outputs, _ = rnn(torch.tensor(val_data).view(1, -1,
1).float())
    val_loss = criterion(val_outputs.view(-1),
torch.tensor(val_data).view(1, -1, 1).float())
    val_losses.append(val_loss.item())

    if epoch % 10 == 0:
```

```
        print(f'Epoch [{epoch+1}/{epochs}], Train Loss:
{loss.item():.4f}, Val Loss: {val_loss.item():.4f}')
```

```
# Predicción con la RNN
rnn.eval()
with torch.no_grad():
    prediction, _ = rnn(torch.tensor(data).view(1, -1, 1).float())
prediction = prediction.view(-1).numpy()
```