Construcción de un laboratorio interactivo para el modelamiento de Redes Neuronales Recurrentes con aplicaciones en actuaría y finanzas.

Mateo Sebastian Ortiz Higuera

Facultad de Ciencias
Departamento de Matemáticas
Sede Bogotá, Colombia
2025

Construcción de un laboratorio interactivo para el modelamiento de Redes Neuronales Recurrentes con aplicaciones en actuaría y finanzas.

Mateo Sebastian Ortiz Higuera

Tesis presentada como requisito parcial para optar por el título de:
Científico de la Computación

Director(a):
Prof. Arles Ernesto Rodríguez Portela, PhD
Profesor Asistente - Matemáticas
Facultad de Ciencias
Universidad Nacional de Colombia

Universidad Nacional de Colombia
Facultad de Ciencias
Departamento de Matemáticas
2025

Declaración

Me permito afirmar que he realizado esta tesis de manera autónoma y con la única ayuda de los medios permitidos y no diferentes a los mencionados en el presente texto. Todos los pasajes que se han tomado de manera textual o figurativa de textos publicados y no publicados, los he reconocido en el presente trabajo. Ninguna parte del presente trabajo se ha empleado en ningún otro tipo de tesis.

Sede Bogotá., Fecha entrega

Mateo Sebastian Ortiz Higuera

Agradecimientos

Primeramente me gustaría agradecer al profesor Arles Rodríguez, mi director de tesis, por su amabilidad en el acompañamiento semana a semana de este trabajo.

A mi maravillosa familia quien siempre estuvo apoyándome en los momentos más difíciles como lo es en una pandemia. A mis padres por ser la principal motivación para sacar adelante esta carrera y a quienes les debo todo, porque siempre serán los responsables de todo lo que consiga. A mis hermanos quienes siempre estuvieron acompañándome en este duro proceso en una extraña ciudad como lo es Bogotá, sin ustedes no habría sido capaz de terminar la carrera solo.

Y a todas las personas que conocí a lo largo de estos cinco años y medio, porque me ayudaron a formarme como mejor estudiante y persona.

Resumen

Construcción de un laboratorio interactivo para el modelamiento de Redes Neuronales Recurrentes con aplicaciones en actuaría y finanzas

Este trabajo presenta el desarrollo de un laboratorio interactivo orientado al estudio de redes neuronales recurrentes, en particular las arquitecturas LSTM y GRU aplicadas al análisis de series temporales en actuaría y finanzas. Se implementaron ambas arquitecturas desde cero utilizando Numpy y se integraron en un entorno de aprendizaje basado en cuadernos de Jupyter que combina teoría y ejercicios autocalificables. Como caso de estudio, se abordó la predicción de la inflación en Colombia y se compararon los modelos implementados con sus equivalentes en Keras. Los resultados muestran que los modelos implementados desde cero presentan un comportamiento similar a los de Keras. El laboratorio constituye una herramienta didáctica flexible y una base para futuras extensiones y validaciones en contextos académicos.

Palabras clave: Redes Neuronales Recurrentes, Laboratorio Interactivo, Redes de Memoria a Largo Plazo, Unidades Recurrentes Controladas, Series Temporales, Inflación

Abstract

Construction of an Interactive Laboratory for Recurrent Neural Network Modeling with Applications in Actuarial Science and Finance

This work presents the development of an interactive laboratory aimed at studying Recurrent Neural Networks (RNNs), specifically LSTM and GRU architectures, applied to time series analysis in actuarial science and finance. Both architectures were implemented from scratch using NumPy and integrated into a learning environment based on Jupyter Notebooks that combines theory with self-grading exercises. As a case study, Colombian inflation prediction was addressed, and the implemented models were compared with their Keras equivalents. The results show that the models implemented from scratch exhibit behavior similar to those built with Keras. This laboratory constitutes a flexible didactic tool and a foundation for future extensions and validations in academic contexts.

Keywords: Recurrent Neuronal Networks, Interactive Lab, Long short-term Memory networks, Gated Recurrent Units, Time Series, Inflation

Lista de figuras

Lista de tablas

Contenido

Capítulo 1

Introducción

Las redes neuronales recurrentes (RNN) y, en particular, sus variantes con memoria a largo plazo (LSTM, GRU), han demostrado ser herramientas poderosas para el modelado de series temporales en diferentes dominios, incluyendo la actuaría y las finanzas. Su capacidad para capturar dependencias temporales y patrones secuenciales las convierte en candidatas naturales para tareas como la predicción de riesgos, el análisis de tendencias financieras o la estimación de reservas.

Sin embargo, a pesar de su potencial, la comprensión y aplicación práctica de estas redes neuronales puede ser compleja para estudiantes y profesionales que se inician en el área. Existe una brecha entre el conocimiento teórico de los modelos LSTM y GRU y su implementación efectiva en problemas del mundo real. A menudo, el uso de librerías de alto nivel oculta la complejidad matemática interna, limitando el entendimiento profundo de los mecanismos que rigen estos algoritmos. Se necesita un recurso educativo que no solo explique la teoría, sino que desglose la construcción de los algoritmos desde cero y también guíe al usuario en la aplicación computacional y el análisis de resultados en un contexto financiero o actuarial concreto.

El presente trabajo tiene como objetivo el desarrollo de un laboratorio interactivo, basado en cuadernos de Jupyter, que facilite el aprendizaje activo de las redes LSTM y GRU aplicadas a un problema concreto de actuaría o finanzas. Este recurso integrará explicaciones teóricas, ejemplos prácticos y ejercicios autocalificables, con el fin de brindar una comprensión profunda de estas redes desde una perspectiva computacional y aplicada.

Finalmente, el documento se estructura de la siguiente manera: en la primera sección se revisan los fundamentos teóricos de las RNN; la segunda sección detalla la metodología de implementación del laboratorio; la tercera presenta los resultados obtenidos en el caso de estudio de la inflación; y por último, se exponen las conclusiones y trabajos futuros.

Capítulo 2

Objetivos

2.1. Objetivo General

Construir un laboratorio interactivo que permita al estudiante comprender las redes neuronales de memoria larga a corto plazo (LSTM) y Unidades Recurrentes Controladas (GRU) aplicándolo directamente a un problema de actuaría y finanzas.

2.2. Objetivos Específicos

- Estudiar el funcionamiento de las redes neuronales recurrentes, con énfasis en las arquitecturas LSTM y GRU, y su papel en el modelado de datos secuenciales.
- Implementar un modelo LSTM y GRU en Python utilizando librerías como Keras y Numpy aplicándolo al problema seleccionado.
- Desarrollar un laboratorio interactivo en formato Jupyter Notebook que combine teoría, código y actividades autocalificables para facilitar el aprendizaje autónomo.

Capítulo 3

Revisión de literatura

En la presente sección se revisan las aplicaciones más recientes de las arquitecturas LSTM y GRU en dominios financieros y actuariales. El objetivo es explorar el panorama actual, identificar diversas metodologías, puntos débiles comunes y destacar la escasez de recursos educativos especializados que faciliten la adopción de estas tecnologías por parte de estudiantes y profesionales dentro del área de la actuaría y las finanzas.

3.1. Aplicaciones de LSTM y GRU

En los años recientes es notable el aumento del uso de redes neuronales recurrentes para la predicción de diversos factores financieros y actuariales. Algunos estudios como el de *Zhou* [2024] brinda un marco para trabajar utilizando algunos modelos de redes neuronales, justificándose principalmente en el buen manejo de grandes volúmenes de datos y en la alta complejidad asociada a los problemas que los modelos actuales pueden soportar, a diferencia de los métodos estadísticos clásicos. Similarmente, *Lindholm, M. & Palmborg* [2022] estudia la preparación de datos con el uso de LSTM para lograr mejores predicciones en el pronóstico de mortalidad. Estudios como los de *Baillargeon et al.* [2021], *Perla et al.* [2021] y *Cai et al.* [2025] muestran que el uso de algunos modelos, como el HAN[1], el LSTM, el GRU y las GAN[2], producen un mejor rendimiento predictivo en comparación a métodos clásicos estadísticos como el GLM de Poisson, el modelo Lee-Carter, el modelo Chain Ladder y modelos estocásticos tipo Mack, en problemas actuariales como en la estimación de pensiones, seguros de vida, de propiedad y accidentes.

Por otra parte, trabajos como los de *Sako et al.* [2022] y *Fischer & Krauss* [2018] muestran que el uso de

[1] Por las siglas de Hierarchical Attention Network, son modelos cuyas capas son una combinación entre redes recurrentes y redes de autoatención.

[2] Por las siglas de Generative Adversarial Networks, son modelos constituidos por dos redes neuronales: una generadora y otra discriminadora. En donde la generadora intenta engañar a la discriminadora para producir mejores predicciones.

redes recurrentes como LSTM y GRU para predecir el cierre de precios en diversos índices bursátiles pueden a llegar a ser bastante precisos, concluyendo que las GRU suelen tener mejor desempeño en la mayoría de los casos.

Algunos otros estudios como los de *Paranhos* [2021], *Almosova & Andresen* [2018] y *Cárdenas-Cárdenas et al.* [2023] dan cuenta del gran rendimiento de modelos LSTM en la predicción en la inflación en Estados Unidos y en Colombia respectivamente, utilizando algunos factores económicos. Comparándolos con algunos modelos más convencionales como ARIMA, el modelo UC-SV y en modelos autorregresivos como AR/VAR y redes neuronales completamente conectadas.

3.2. Implementaciones educativas y recursos existentes

Pese al crecimiento en aplicaciones, la disponibilidad de recursos educativos especializados es notablemente escasa. Nuestra revisión identificó algunos tutoriales y laboratorios sobre redes recurrentes, proporcionados principalmente en cursos de pago, cuyo enfoque, en la mayoría de casos, vira hacia otras áreas, principalmente el procesamiento del lenguaje natural y la inteligencia artificial generativa (*DeepLearning.AI* [2018], *Imperial College London* [2020], *Packt* [2025]).

Por otro lado, también se encontró algunos otros recursos gratuitos disponibles, como el tutorial presentado por *Meier & Schelldorfer* [2019], el cual es bastante completo al implementar redes LSTM y GRU, sin embargo, al estar escrito solo en lenguaje R, no existe un sistema de evaluación o de retroalimentación para el estudiante que sí permite el uso de lenguajes como Python. Algunos otros tutoriales y recursos gratuitos fueron encontrados en páginas como Kaggle (*Pamukcu* [2023]) y en repositorios como GitHub (*Ömer Berat Sezer* [2018], *campdav* [2017]), sin embargo y a pesar de que ofrecen una explicación relativamente detallada del funcionamiento de las redes recurrentes y de los LSTM, carecen de un sistema de evaluación

Las páginas oficiales de librerías como Keras, TensorFlow y Torch, además de otras como GeeksforGeeks, también ofrecen algunos tutoriales de uso a través de algunos ejemplos básicos. Sin embargo, todas estas páginas están dedicadas a la explicación de la aplicación de las librerías y funciones ya existentes, dirigiendo mayor parte de su esfuerzo en hacer entender el ajuste de parámetros y el uso de clases de forma conjunta, antes que explicar en su totalidad la implementación de estas clases y funciones.

El curso *Ng & deeplearning.ai* [2018] ofrece un par de laboratorios enfocados en redes recurrentes, en el primero de ellos se construyen las celdas de propagación para redes recurrentes genéricas y para LSTM, esto es particularmente útil para nosotros, el laboratorio se construye copiando la estructura del código de los modelos que se hace en el laboratorio de este curso.

Capítulo 4

Marco teórico

En esta sección se presentan las bases teóricas necesarias para comprender el desarrollo del trabajo, tomando como referencias principales los artículos de *Prince* [2023] y *Ghojogh & Ghodsi* [2023]. Primeramente se habla sobre las redes neuronales recurrentes, su formulación matemática y su entrenamiento de parámetros a través del gradiente en descenso con la regla de la cadena (retropropagación de valores). Se da una pequeña noción sobre el mayor problema de esta arquitectura, que tiene que ver con dos problemas numéricos importantes a la hora de entrenar el modelo, esto nos servirá como motivación a introducir las dos arquitecturas trabajadas. Inicialmente se introducen las redes de memoria de larga a corto plazo, su formalización matemática y su entrenamiento. Y finalmente se describe las redes de unidades recurrentes controladas, siguiendo la misma ruta que con las redes de memoria de larga a corto plazo.

4.1. Redes Neuronales Recurrentes

Las *Redes Neuronales Recurrentes* (RNN, por sus siglas en inglés) son un tipo de arquitectura de Redes Neuronales Profundas, especialmente diseñadas para modelar datos secuenciales. Esto incluye tareas como el procesamiento de lenguaje natural, series temporales financieras, entre otras. Su principal característica es la capacidad de mantener información del pasado a través de un estado oculto, lo que les permite generar predicciones contextualizadas basadas en elementos previos de la secuencia. Su arquitectura consiste en una secuencia de bloques recurrentes $h^{\langle t \rangle}$ que tienen dos entradas: el bloque recurrente previo $h^{\langle t-1 \rangle}$ y el elemento $\mathbf{x}^{\langle t \rangle}$, que es el t-ésimo elemento de una muestra $\mathbf{x}$. Dependiendo del problema y del tiempo en la secuencia, cada bloque produce una o dos salidas[1]: una para el bloque siguiente y otra salida correspondiente a una predicción $\hat{y}^{\langle t \rangle}$.

[1] Por practicidad, en este capítulo se supone que cada bloque recurrente produce dos salidas: el estado ocultos $h^{\langle t \rangle}$ y la predicción $\hat{y}^{\langle t \rangle}$.

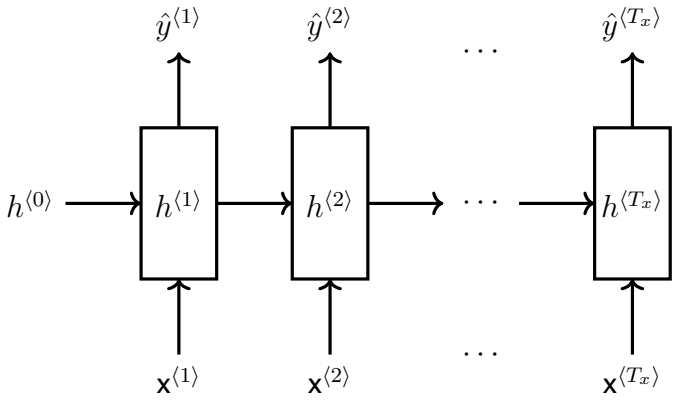


Figura 4-1: Grafo computacional de una red neuronal recurrente con T_x bloques recurrentes.

Otra característica importante de las RNNs es que cada bloque comparte las mismas matrices de pesos y vectores de sesgos por entrada. Es decir, para las entradas de la muestra tenemos una única matriz de pesos W_x para todos los bloques, para las entradas de bloques anteriores tenemos una única matriz de pesos W_h para todos los bloques y para todas las entradas a las salidas tenemos una única matriz de pesos W_y. Esto contribuye a tener un modelo significativamente más pequeño que un modelo de Redes Neuronales Profundas Densas, y a poder manejar datos secuenciales de forma más flexible.

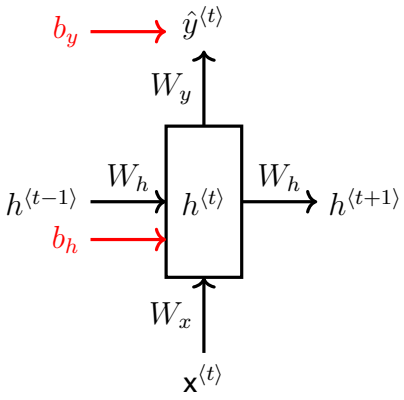


Figura 4-2: Grafo computacional de un solo bloque recurrente con sus matrices de pesos. Debe entenderse que para todo $t = 1, 2, ..., T_x$ se comparten los mismos W_x, W_h, W_y, b_h y b_y.

4.1.1. Formulación matemática

Formalmente, un modelo recurrente se define a través de la siguiente forma recursiva

$$\hat{y}^{\langle t \rangle} = a_y \left(W_y h^{\langle t \rangle} + b_y \right) \tag{4-1}$$

en donde

$$h^{\langle t \rangle} = \begin{cases} 0, & \text{si } t = 0 \\ a_h \left(W_h h^{\langle t-1 \rangle} + W_x \mathsf{x}^{\langle t \rangle} + b_h \right), & \text{en otro caso.} \end{cases} \tag{4-2}$$

y $a_y, a_h : \mathbb{R} \longrightarrow \mathbb{R}$ corresponden a funciones de activación que inducen no linealidad, como $ReLU$, tanh, σ, etc.[2][3]

4.1.2. Retropropagación a través del tiempo

Para el entrenamiento de una RNN se utiliza un método llamado *retropropagación a través del tiempo* (BPTT de sus siglas del inglés), en el cual se calcula la función de pérdida para cada bloque, en orden desde el T_x-ésimo hasta el primero, y al final se suman todas las pérdidas.

Definamos $\phi := \{W_h, W_x, W_y, b_h, b_y\}$ y supongamos que tenemos las siguientes funciones de pérdida

$$\mathcal{L}^{\langle t \rangle}[\phi] = L\left(\hat{y}^{\langle t \rangle}(\phi), y^{\langle t \rangle}\right)$$

para cada instante de tiempo $t = 1, 2, ..., T_x$. Entonces la función de pérdida de la RNN es

$$\mathcal{L}[\phi] = \sum_{t=1}^{T_x} \mathcal{L}^{\langle t \rangle}[\phi] \tag{4-3}$$

El objetivo del entrenamiento es optimizar dicha función de pérdida para ajustar ϕ de tal forma que el modelo aprenda a generalizar y a realizar predicciones de acuerdo a los datos de entrenamiento. Utilizamos el método de gradiente en descenso en conjunto con la regla de la cadena.

Para actualizar los valores de los pesos primero se calcula el gradiente de la función de pérdida con respecto a las salidas. En cada tiempo t se debe calcular la derivada para el término dentro de la función de activación $z_y^{\langle t \rangle} := W_y h^{\langle t \rangle} + b_y$, por la regla de la cadena y por (4-3), tenemos lo siguiente

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial z_y^{\langle t \rangle}} &= \frac{\partial \mathcal{L}}{\partial \mathcal{L}^{\langle t \rangle}} \frac{\partial \mathcal{L}^{\langle t \rangle}}{\partial \hat{y}^{\langle t \rangle}} \frac{\partial \hat{y}^{\langle t \rangle}}{\partial z_y^{\langle t \rangle}} \\ &= \frac{\partial \mathcal{L}^{\langle t \rangle}}{\partial \hat{y}^{\langle t \rangle}} \frac{\partial \hat{y}^{\langle t \rangle}}{\partial z_y^{\langle t \rangle}} \end{aligned} \tag{4-4}$$

Con esto se puede calcular el resto de gradientes. Para calcular el estado o bloque en el tiempo t, por las dos trayectorias tenemos

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial h^{\langle t \rangle}} &= \left(\frac{\partial \mathcal{L}}{\partial \hat{y}^{\langle t \rangle}} \frac{\partial \hat{y}^{\langle t \rangle}}{\partial h^{\langle t \rangle}}\right) + \left(\frac{\partial \mathcal{L}}{\partial h^{\langle t+1 \rangle}} \frac{\partial h^{\langle t+1 \rangle}}{\partial h^{\langle t \rangle}}\right) \\ &= \left(\frac{\partial \mathcal{L}}{\partial \hat{y}^{\langle t \rangle}} W_y\right) + \left(\frac{\partial \mathcal{L}}{\partial h^{\langle t+1 \rangle}} \frac{\partial h^{\langle t+1 \rangle}}{\partial h^{\langle t \rangle}}\right) \end{aligned} \tag{4-5}$$

[2] Véase el apéndice 1.

[3] En esta sección vamos a trabajar con esta notación. Sin embargo, en las siguientes secciones, por la construcción de las arquitecturas, se trabaja con tanh y σ como se puede ver en las ecuaciones (4-17) y (4-45).

En donde, por (4-2) para $t \neq 0$, por regla de la cadena y definiendo $z^{\langle t \rangle} = W_h h^{\langle t-1 \rangle} + W_x \mathbf{x}^{\langle t \rangle} + b_h$, se tiene

$$\begin{aligned}\frac{\partial h^{\langle t+1 \rangle}}{\partial h^{\langle t \rangle}} &= \frac{\partial a_h\left(z^{\langle t+1 \rangle}\right)}{\partial z^{\langle t+1 \rangle}} \frac{\partial z^{\langle t+1 \rangle}}{\partial h^{\langle t \rangle}} \\ &= \frac{\partial a_h\left(z^{\langle t+1 \rangle}\right)}{\partial z^{\langle t+1 \rangle}} \frac{\partial}{\partial h^{\langle t \rangle}} \left(W_h h^{\langle t \rangle} + W_x \mathbf{x}^{\langle t+1 \rangle} + b_h\right) \\ &= \frac{\partial a_h\left(z^{\langle t+1 \rangle}\right)}{\partial z^{\langle t+1 \rangle}} W_h \end{aligned} \tag{4-6}$$

suponiendo que a_h es diferenciable (como ocurre con las funciones $ReLU$ o $tanh$ que son las usadas generalmente en RNNs), entonces esta última expresión existe. Para el último bloque $t = T_x$, el gradiente es simplemente

$$\frac{\partial \mathcal{L}}{\partial h^{\langle T_x \rangle}} = \frac{\partial \mathcal{L}}{\partial \hat{y}^{\langle T_x \rangle}} \frac{\partial \hat{y}^{\langle T_x \rangle}}{\partial h^{\langle T_x \rangle}} = \frac{\partial \mathcal{L}}{\partial \hat{y}^{\langle T_x \rangle}}$$

por 4-4 finalmente se tiene

$$\frac{\partial \mathcal{L}}{\partial h^{\langle T_x \rangle}} = \frac{\partial \mathcal{L}}{\partial \hat{y}^{\langle T_x \rangle}} = \frac{\partial \mathcal{L}^{\langle T_x \rangle}}{\partial \hat{y}^{\langle T_x \rangle}} \tag{4-7}$$

Teniendo estos gradientes ahora sí se puede calcular los gradientes de los pesos. Primero el gradiente de W_y, por la regla de la cadena, por (4-4) y dado que existen T_x salidas $\hat{y}$, se tiene

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial W_y} &= \sum_{t=1}^{T_x} \left(\frac{\partial \mathcal{L}}{\partial \hat{y}^{\langle t \rangle}} \frac{\partial \hat{y}^{\langle t \rangle}}{\partial W_y}\right) \\ &= \sum_{t=1}^{T_x} \left(\frac{\partial \mathcal{L}^{\langle t \rangle}}{\partial \hat{y}^{\langle t \rangle}} \frac{\partial}{\partial W_y} \left(W_y h^{\langle t \rangle} + b_y\right)\right) \\ &= \sum_{t=1}^{T_x} \frac{\partial \mathcal{L}^{\langle t \rangle}}{\partial \hat{y}^{\langle t \rangle}} (h^{\langle t \rangle})^T \end{aligned} \tag{4-8}$$

Ahora para el gradiente de W_h, por la regla de la cadena se tiene

$$\frac{\partial \mathcal{L}}{\partial W_h} = \sum_{t=1}^{T_x} \left(\frac{\partial \mathcal{L}}{\partial h^{\langle t \rangle}} \frac{\partial h^{\langle t \rangle}}{\partial W_h}\right) \tag{4-9}$$

en donde la primera expresión corresponde a (4-5) y la segunda se puede expandir

$$\begin{aligned}\frac{\partial h^{\langle t \rangle}}{\partial W_h} &= \frac{\partial h^{\langle t \rangle}}{\partial z^{\langle t \rangle}} \frac{\partial z^{\langle t \rangle}}{\partial W_h} \\ &= \frac{\partial a_h\left(z^{\langle t \rangle}\right)}{\partial z^{\langle t \rangle}} \frac{\partial}{\partial W_h} \left(W_h h^{\langle t-1 \rangle} + W_x \mathbf{x}^{\langle t \rangle} + b_h\right) \\ &= \frac{\partial a_h\left(z^{\langle t \rangle}\right)}{\partial z^{\langle t \rangle}} (h^{\langle t-1 \rangle})^T \end{aligned} \tag{4-10}$$

Para el gradiente de W_x, similarmente a las derivadas parciales anteriores se tiene

$$\frac{\partial \mathcal{L}}{\partial W_x} = \sum_{t=1}^{T_x} \left(\frac{\partial \mathcal{L}}{\partial h^{\langle t \rangle}} \frac{\partial h^{\langle t \rangle}}{\partial W_x} \right) \tag{4-11}$$

en donde la primera expresión del producto corresponde a (4-5) y la segunda se puede expandir así

$$\begin{aligned} \frac{\partial h^{\langle t \rangle}}{\partial W_x} &= \frac{\partial h^{\langle t \rangle}}{\partial z^{\langle t \rangle}} \frac{\partial z^{\langle t \rangle}}{\partial W_x} \\ &= \frac{\partial a_h \left(z^{\langle t \rangle} \right)}{\partial z^{\langle t \rangle}} \frac{\partial}{\partial W_x} \left(W_h h^{\langle t-1 \rangle} + W_x \mathbf{x}^{\langle t \rangle} + b_h \right) \\ &= \frac{\partial a_h \left(z^{\langle t \rangle} \right)}{\partial z^{\langle t \rangle}} (\mathbf{x}^{\langle t \rangle})^T \end{aligned} \tag{4-12}$$

Ahora se calcula la derivada parcial para b_h, se tiene

$$\frac{\partial \mathcal{L}}{\partial b_h} = \sum_{t=1}^{T_x} \left(\frac{\partial \mathcal{L}}{\partial h^{\langle t \rangle}} \frac{\partial h^{\langle t \rangle}}{\partial b_h} \right) \tag{4-13}$$

en donde la primera expresión del producto corresponde a (4-5) y la segunda expresión es

$$\begin{aligned} \frac{\partial h^{\langle t \rangle}}{\partial b_h} &= \frac{\partial h^{\langle t \rangle}}{\partial z^{\langle t \rangle}} \frac{\partial z^{\langle t \rangle}}{\partial b_h} \\ &= \frac{\partial a_h \left(z^{\langle t \rangle} \right)}{\partial z^{\langle t \rangle}} \frac{\partial}{\partial b_h} (W_h h^{\langle t-1 \rangle} + W_x \mathbf{x}^{\langle t \rangle} + b_h) \\ &= \frac{\partial a_h \left(z^{\langle t \rangle} \right)}{\partial z^{\langle t \rangle}} \end{aligned} \tag{4-14}$$

Por último se puede calcular la derivada parcial de b_y, se tiene

$$\frac{\partial \mathcal{L}}{\partial b_y} = \sum_{t=1}^{T_x} \left(\frac{\partial \mathcal{L}}{\partial \hat{y}^{\langle t \rangle}} \frac{\partial \hat{y}^{\langle t \rangle}}{\partial b_y} \right) \tag{4-15}$$

Finalmente, se utiliza el método del gradiente en descenso para actualizar cada parámetro. Dado un learning rate $\eta > 0$ y por (4-5), (4-9), (4-11), (4-8), (4-13) y (4-15), se puede tener:

$$\begin{aligned} W_x &\longleftarrow W_x - \eta \frac{\partial \mathcal{L}}{\partial W_x} \quad b_h \longleftarrow b_h - \eta \frac{\partial \mathcal{L}}{\partial b_h} \\ W_y &\longleftarrow W_y - \eta \frac{\partial \mathcal{L}}{\partial W_y} \quad b_y \longleftarrow b_y - \eta \frac{\partial \mathcal{L}}{\partial b_y} \\ W_h &\longleftarrow W_h - \eta \frac{\partial \mathcal{L}}{\partial W_h} \end{aligned} \tag{4-16}$$

para todo $t = 1, 2, 3, \cdots, T_x$.

Sin embargo, al final la actualización de cada parámetro puede depender de un *optimizador*. En este trabajo, cada parámetro se actualiza como se ve en la ecuación (5-2).

4.1.3. Problema de gradientes

Como en cualquier red neuronal profunda, esta arquitectura de red neuronal no es más que una larga composición de transformaciones no lineales, como se puede ver al expandir la fórmula definida en (4-1).

$$\hat{y}^{\langle t \rangle} = a_y \left(W_y \left(a_h \left(W_h \left(\cdots a_h \left(W_h h^{\langle 1 \rangle} + W_{\mathbf{x}} \mathbf{x}^{\langle 1 \rangle} + b_h \right) \cdots \right) + W_{\mathbf{x}} \mathbf{x}^{\langle t \rangle} + b_h \right) \right) + b_y \right)$$

Como resultado de esto, esta arquitectura es también susceptible a la propagación de errores numéricos durante el proceso de entrenamiento. En particular, puede presentarse la propagación de valores extremadamente pequeños, que tienden a anularse por aproximarse al cero numérico, o, en el extremo opuesto, la propagación de valores que crecen exponencialmente a lo largo de las iteraciones.

Estos comportamientos, conocidos respectivamente como desvanecimiento y explosión del gradiente, constituyen una de las principales dificultades en el entrenamiento de redes neuronales recurrentes, especialmente cuando se trabaja con secuencias extensas de celdas, donde los efectos acumulativos del gradiente se vuelven más pronunciados.

En la literatura, así como para las redes neuronales profundas en general, existen diversos métodos y técnicas para prevenir y mitigar estos problemas de gradientes. No obstante, en este trabajo, estos problemas funcionan como una motivación más para presentar las dos arquitecturas propuestas: las LSTM y las GRU.

4.2. Redes de Memoria larga a corto plazo (LSTM)

Una *Long Short-Term Memory* (*LSTM*) es una arquitectura de red neuronal recurrente introducida por *Hochreiter & Schmidhuber* [1997], cuyo propósito fundamental consiste en mitigar el problema de desvanecimiento del gradiente presente en arquitecturas *Recurrent Neural Network* (*RNN*) más simples, como la analizada previamente. En esta arquitectura, se incorpora una *célula de memoria* $c^{\langle t \rangle}$ que permite el flujo estable de los gradientes durante el proceso de retropropagación, reduciendo significativamente la pérdida de información a lo largo del tiempo.

Desde una perspectiva funcional, una LSTM posee la capacidad de retener o descartar información dependiendo de su relevancia en la secuencia de entrada. Para esto incorpora una *compuerta de olvido* Γ_f, una *compuerta de actualización* Γ_u y una *compuerta de salida* Γ_o.

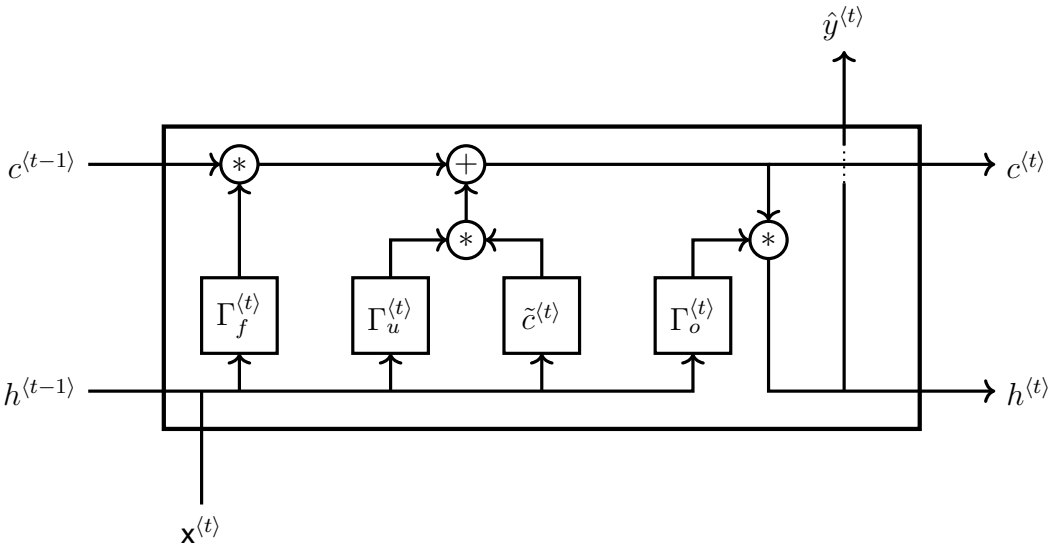


Figura 4-3: Arquitectura de la celda LSTM en el tiempo t.

El funcionamiento de cada sección o compuerta de la arquitectura está dado como sigue

- *Compuerta de olvido*: La compuerta de olvido está constituida de una matriz $\Gamma_f^{\langle t \rangle}$ que contiene valores en el rango $(0, 1) \subset \mathbb{R}$, estos valores dependen del estado oculto anterior $h^{\langle t-1 \rangle}$ y la entrada actual $\mathsf{x}^{\langle t \rangle}$. Dicha matriz realiza un producto elemento a elemento con la célula de memoria del estado anterior $c^{\langle t-1 \rangle}$, de esta forma 'olvida' valores (cuando los valores de Γ_f son cercanos a 0) o 'retiene' (cuando los valores de Γ_f son cercanos a 1) valores propagados en la célula de memoria.
- *Compuerta de actualización o entrada:* La compuerta de reinicio está constituida de una matriz $\Gamma_u^{\langle t \rangle}$ y una matriz candidata $\tilde{c}^{\langle t \rangle}$, el primer contiene valores en el rango $(0, 1) \subset \mathbb{R}$ y el segundo contiene valores en el rango $(-1, 1) \subset \mathbb{R}$, estos valores dependen del estado oculto anterior $h^{\langle t-1 \rangle}$

y de la entrada actual $\mathsf{x}^{\langle t\rangle}$. Entre ambas matrices se realiza un producto elemento a elemento, similarmente como con la compuerta anterior, Γ_f retiene o elimina valores del candidato $\tilde{c}$ y estos valores restantes se suman a la célula de memoria $c^{\langle t\rangle}$ actualizándola.

- *Compuerta de salida:* La compuerta de salida está constituida por una matriz tensor $\Gamma_o^{\langle t\rangle}$, que nuevamente contiene valores en el rango $(0,1) \subset \mathbb{R}$, estos valores también dependen del estado oculto anterior $h^{\langle t-1\rangle}$ y la entrada actual $\mathsf{x}^{\langle t\rangle}$. Este tensor realiza un producto elemento a elemento con la célula de memoria actual $c^{\langle t\rangle}$, que primero pasa por una normalización con la función tanh. Similarmente a las compuertas anteriores, Γ_o retiene o elimina valores de la célula de memoria para la salida del estado oculto y de la predicción del tiempo actual.

Formalmente, una LSTM se define a través de la siguiente forma recursiva

$$\hat{y}^{\langle t\rangle} = \mathsf{tanh}\left(W_y h^{\langle t\rangle} + b_y\right) \tag{4-17}$$

en donde

$$h^{\langle t\rangle} = \begin{cases} 0, & \text{si } t = 0 \\ \Gamma_o^{\langle t\rangle} * \mathsf{tanh}\left(c^{\langle t\rangle}\right), & \text{en otro caso.} \end{cases} \tag{4-18}$$

$$c^{\langle t\rangle} = \left(c^{\langle t-1\rangle} * \Gamma_f^{\langle t\rangle}\right) + \left(\Gamma_u^{\langle t\rangle} * \tilde{c}^{\langle t\rangle}\right) \tag{4-19}$$

$$\tilde{c}^{\langle t\rangle} = \mathsf{tanh}\left(W_c\left[h^{\langle t-1\rangle}, \mathsf{x}^{\langle t\rangle}\right] + b_c\right) \tag{4-20}$$

$$\Gamma_o^{\langle t\rangle} = \sigma\left(W_o\left[h^{\langle t-1\rangle}, \mathsf{x}^{\langle t\rangle}\right] + b_o\right) \tag{4-21}$$

$$\Gamma_u^{\langle t\rangle} = \sigma\left(W_u\left[h^{\langle t-1\rangle}, \mathsf{x}^{\langle t\rangle}\right] + b_u\right) \tag{4-22}$$

$$\Gamma_f^{\langle t\rangle} = \sigma\left(W_f\left[h^{\langle t-1\rangle}, \mathsf{x}^{\langle t\rangle}\right] + b_f\right) \tag{4-23}$$

en donde $*$ denota el producto elemento a elemento entre matrices y, por simplicidad

$$W_\ell\left[h^{\langle t-1\rangle}, \mathsf{x}^{\langle t\rangle}\right] = W_{\ell,1} h^{\langle t-1\rangle} + W_{\mathsf{x}} \mathsf{x}^{\langle t\rangle} \tag{4-24}$$

4.2.1. Retropropagación a través del tiempo

En la sección anterior se mostró cómo se pueden calcular las derivadas de una RNN. Aquí la idea es similar, sin embargo, al existir la célula de memoria y las compuertas, se deben calcular las derivadas de los parámetros asociados a cada uno de estos elementos.

Dada la *retropropagación a a través del tiempo*, se considera la misma función de pérdida $\mathcal{L}[\phi]$ definida en (4-3) para un conjunto de parámetros $\phi := \{W_c, W_o, W_u, W_f, W_y, b_c, b_o, b_u, b_f, b_y\}$.

Para la salida, por la regla de la cadena, por (4-3) y por (4-4)

$$\frac{\partial \mathcal{L}}{\partial z_y^{\langle t\rangle}} = \frac{\partial \mathcal{L}}{\partial \hat{y}^{\langle t\rangle}} = \frac{\partial \mathcal{L}^{\langle t\rangle}}{\partial \hat{y}^{\langle t\rangle}} \frac{\partial \hat{y}^{\langle t\rangle}}{\partial z_y^{\langle t\rangle}} \tag{4-25}$$

Por (A-2), entonces

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial z_y^{\langle t \rangle}} &= \frac{\partial \mathcal{L}^{\langle t \rangle}}{\partial \hat{y}^{\langle t \rangle}} \frac{\partial}{\partial z_y^{\langle t \rangle}} \left(\tanh \left(z_y^{\langle t \rangle} \right) \right) \\ &= \frac{\partial \mathcal{L}^{\langle t \rangle}}{\partial \hat{y}^{\langle t \rangle}} \left(1 - \tanh^2 \left(z_y^{\langle t \rangle} \right) \right)\end{aligned} \tag{4-26}$$

Con esto se pueden calcular las derivadas para W_y, gracias a (4-8) y por (4-26) se tiene

$$\frac{\partial \mathcal{L}}{\partial W_y} = \sum_{t=1}^{T_x} \left(\frac{\partial \mathcal{L}}{\partial z_y^{\langle t \rangle}} \left(h^{\langle t \rangle} \right)^T \right) \tag{4-27}$$

y para b_y, por (4-15) y por (4-26), entonces

$$\frac{\partial \mathcal{L}}{\partial b_y} = \sum_{t=1}^{T_x} \frac{\partial \mathcal{L}}{\partial z_y^{\langle t \rangle}} \tag{4-28}$$

Para el bloque en el tiempo t se tiene

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial h^{\langle t \rangle}} &= \left(\frac{\partial \mathcal{L}}{\partial \hat{y}^{\langle t \rangle}} \frac{\partial \hat{y}^{\langle t \rangle}}{\partial h^{\langle t \rangle}} \right) + \left(\frac{\partial \mathcal{L}}{\partial h^{\langle t+1 \rangle}} \frac{\partial h^{\langle t+1 \rangle}}{\partial h^{\langle t \rangle}} \right) \\ &= \left(\frac{\partial \mathcal{L}}{\partial \hat{y}^{\langle t \rangle}} \frac{\partial}{\partial h^{\langle t \rangle}} \left(W_y h^{\langle t \rangle} + b_y \right) \right) + \left(\frac{\partial \mathcal{L}}{\partial h^{\langle t+1 \rangle}} \frac{\partial h^{\langle t+1 \rangle}}{\partial h^{\langle t \rangle}} \right) \\ &= \left(\frac{\partial \mathcal{L}}{\partial \hat{y}^{\langle t \rangle}} W_y^T \right) + \left(\frac{\partial \mathcal{L}}{\partial h^{\langle t+1 \rangle}} \frac{\partial h^{\langle t+1 \rangle}}{\partial h^{\langle t \rangle}} \right)\end{aligned} \tag{4-29}$$

Nótese que el segundo término depende completamente del bloque en el tiempo $t+1$. Esto sugiere un enfoque recursivo: es eficiente iniciar la retropropagación calculando el gradiente en el último instante de tiempo $\partial \mathcal{L} / \partial h^{\langle T_x \rangle}$ y luego retropropagar hacia atrás iterativamente. Para obtener $\partial \mathcal{L} / \partial h^{\langle t \rangle}$ en instantes intermedios, es necesario primero calcular las derivadas respecto a las compuertas y la célula de memoria en el tiempo t, ya que estas definen directamente $h^{\langle t \rangle}$ y también influyen en la célula de memoria que se propaga hacia atrás.

Para el último bloque $t = T_x$, el gradiente es simplemente

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial h^{\langle T_x \rangle}} &= \frac{\partial \mathcal{L}}{\partial \hat{y}^{\langle T_x \rangle}} \frac{\partial \hat{y}^{\langle T_x \rangle}}{\partial h^{\langle T_x \rangle}} \\ &= \frac{\partial \mathcal{L}}{\partial \hat{y}^{\langle t \rangle}} \frac{\partial}{\partial h^{\langle t \rangle}} \left(W_y h^{\langle t \rangle} + b_y \right) \\ &= \frac{\partial \mathcal{L}}{\partial \hat{y}^{\langle T_x \rangle}} W_y^T\end{aligned} \tag{4-30}$$

Con la célula de memoria, por la regla de la cadena y por (4-19), se tiene

$$\frac{\partial \mathcal{L}}{\partial c^{\langle t \rangle}} = \left(\frac{\partial \mathcal{L}}{\partial h^{\langle t \rangle}} * \frac{\partial h^{\langle t \rangle}}{\partial c^{\langle t \rangle}} \right) + \left(\frac{\partial \mathcal{L}}{\partial c^{\langle t+1 \rangle}} * \frac{\partial c^{\langle t+1 \rangle}}{\partial c^{\langle t \rangle}} \right) \tag{4-31}$$

En donde, por (A-2), se tiene que

$$\begin{aligned}\frac{\partial h^{\langle t\rangle}}{\partial c^{\langle t\rangle}} &= \frac{\partial}{\partial c^{\langle t\rangle}}\left(\Gamma_o^{\langle t\rangle} * \tanh\left(c^{\langle t\rangle}\right)\right)\\ &= \Gamma_o^{\langle t\rangle} * \left(1 - \tanh^2\left(c^{\langle t\rangle}\right)\right)\end{aligned} \tag{4-32}$$

y

$$\begin{aligned}\frac{\partial c^{\langle t+1\rangle}}{\partial c^{\langle t\rangle}} &= \frac{\partial}{\partial c^{\langle t\rangle}}\left(c^{\langle t\rangle} * \Gamma_f^{\langle t+1\rangle} + \Gamma_u^{\langle t+1\rangle} * \tilde{c}^{\langle t+1\rangle}\right)\\ &= \Gamma_f^{\langle t+1\rangle}\end{aligned} \tag{4-33}$$

Entonces, por (4-32) y (4-33), la ecuación (4-31) está dada como sigue

$$\frac{\partial \mathcal{L}}{\partial c^{\langle t\rangle}} = \frac{\partial \mathcal{L}}{\partial h^{\langle t\rangle}} * \left(\Gamma_o^{\langle t\rangle} * \left(1 - \tanh^2\left(c^{\langle t\rangle}\right)\right)\right) + \frac{\partial \mathcal{L}}{\partial c^{\langle t+1\rangle}} * \Gamma_f^{\langle t+1\rangle} \tag{4-34}$$

Para el último instante de tiempo T_x es simplemente

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial c^{\langle T_x\rangle}} &= \frac{\partial \mathcal{L}}{\partial h^{\langle T_x\rangle}} * \frac{\partial h^{\langle T_x\rangle}}{\partial c^{\langle t\rangle}}\\ &= \frac{\partial \mathcal{L}}{\partial h^{\langle T_x\rangle}} * \left(\Gamma_o^{\langle T_x\rangle} * \left(1 - \tanh^2\left(c^{\langle T_x\rangle}\right)\right)\right)\end{aligned}$$

Ahora bien, para la primer compuerta en la retropropagación $\Gamma_o^{\langle t\rangle}$ se tiene que calcular la derivada para el término dentro de la función de activación $z_o^{\langle t\rangle} := W_o[h^{\langle t-1\rangle}, \mathrm{x}^{\langle t\rangle}] + b_o$. Por (4-18), (4-21) y por regla de la cadena, se tiene

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial z_o^{\langle t\rangle}} &= \frac{\partial \mathcal{L}}{\partial h^{\langle t\rangle}} * \frac{\partial h^{\langle t\rangle}}{\partial \Gamma_o^{\langle t\rangle}} * \frac{\partial \Gamma_o^{\langle t\rangle}}{\partial z_o^{\langle t\rangle}}\\ &= \frac{\partial \mathcal{L}}{\partial h^{\langle t\rangle}} * \frac{\partial}{\partial \Gamma_o^{\langle t\rangle}}\left(\Gamma_o^{\langle t\rangle} * \tanh\left(c^{\langle t\rangle}\right)\right) * \frac{\partial \Gamma_o^{\langle t\rangle}}{\partial z_o^{\langle t\rangle}}\\ &= \frac{\partial \mathcal{L}}{\partial h^{\langle t\rangle}} * \tanh\left(c^{\langle t\rangle}\right) * \frac{\partial \Gamma_o^{\langle t\rangle}}{\partial z_o^{\langle t\rangle}}\end{aligned} \tag{4-35}$$

Por (A-3), el último término se ve como sigue

$$\begin{aligned}\frac{\partial \Gamma_o^{\langle t\rangle}}{\partial z_o^{\langle t\rangle}} &= \frac{\partial}{\partial z_o^{\langle t\rangle}}\left(\sigma\left(z_o^{\langle t\rangle}\right)\right)\\ &= \Gamma_o^{\langle t\rangle} * \left(1 - \Gamma_o^{\langle t\rangle}\right)\end{aligned} \tag{4-36}$$

Así entonces, por (4-36), la ecuación (4-35) se calcula como sigue

$$\frac{\partial \mathcal{L}}{\partial z_o^{\langle t\rangle}} = \frac{\partial \mathcal{L}}{\partial h^{\langle t\rangle}} * \tanh\left(c^{\langle t\rangle}\right) * \Gamma_o^{\langle t\rangle} * \left(1 - \Gamma_o^{\langle t\rangle}\right) \tag{4-37}$$

Para la compuerta $\Gamma_u^{\langle t \rangle}$, también se calcula la derivada para $z_u^{\langle t \rangle}$. Por (4-18), (4-19), (4-22), por regla de la cadena y por (A-3), se tiene

$$\frac{\partial \mathcal{L}}{\partial z_u^{\langle t \rangle}} = \left(\frac{\partial \mathcal{L}}{\partial c^{\langle t \rangle}} * \frac{\partial c^{\langle t \rangle}}{\partial \Gamma_u^{\langle t \rangle}} * \frac{\partial \Gamma_u^{\langle t \rangle}}{\partial z_u^{\langle t \rangle}} \right) + \left(\frac{\partial \mathcal{L}}{\partial h^{\langle t \rangle}} * \frac{\partial h^{\langle t \rangle}}{\partial c^{\langle t \rangle}} * \frac{\partial c^{\langle t \rangle}}{\partial \Gamma_u^{\langle t \rangle}} * \frac{\partial \Gamma_u^{\langle t \rangle}}{\partial z_u^{\langle t \rangle}} \right)$$

cuya primera parte se puede desarrollar así

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial c^{\langle t \rangle}} * \frac{\partial c^{\langle t \rangle}}{\partial \Gamma_u^{\langle t \rangle}} * \frac{\partial \Gamma_u^{\langle t \rangle}}{\partial z_u^{\langle t \rangle}} &= \frac{\partial \mathcal{L}}{\partial c^{\langle t \rangle}} * \frac{\partial c^{\langle t \rangle}}{\partial \Gamma_u^{\langle t \rangle}} * \frac{\partial \Gamma_u^{\langle t \rangle}}{\partial z_u^{\langle t \rangle}} \\
&= \frac{\partial \mathcal{L}}{\partial c^{\langle t \rangle}} * \frac{\partial}{\partial \Gamma_u^{\langle t \rangle}} \left(c^{\langle t-1 \rangle} * \Gamma_f^{\langle t \rangle} + \tilde{c}^{\langle t \rangle} * \Gamma_u^{\langle t \rangle} \right) * \frac{\partial \Gamma_u^{\langle t \rangle}}{\partial z_u^{\langle t \rangle}} \\
&= \frac{\partial \mathcal{L}}{\partial c^{\langle t \rangle}} * \tilde{c}^{\langle t \rangle} * \frac{\partial}{\partial z_u^{\langle t \rangle}} \left(\sigma \left(z_u^{\langle t \rangle} \right) \right) \\
&= \frac{\partial \mathcal{L}}{\partial c^{\langle t \rangle}} * \tilde{c}^{\langle t \rangle} * \Gamma_u^{\langle t \rangle} * \left(1 - \Gamma_u^{\langle t \rangle} \right)
\end{aligned}$$

y la segunda expresión como sigue

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial h^{\langle t \rangle}} * \cdots * \frac{\partial \Gamma_u^{\langle t \rangle}}{\partial z_u^{\langle t \rangle}} &= \frac{\partial \mathcal{L}}{\partial h^{\langle t \rangle}} * \frac{\partial}{\partial c^{\langle t \rangle}} \left(\Gamma_o^{\langle t \rangle} * \tanh \left(c^{\langle t \rangle} \right) \right) * \frac{\partial c^{\langle t \rangle}}{\partial \Gamma_u^{\langle t \rangle}} * \frac{\partial \Gamma_u^{\langle t \rangle}}{\partial z_u^{\langle t \rangle}} \\
&= \frac{\partial \mathcal{L}}{\partial h^{\langle t \rangle}} * \Gamma_o^{\langle t \rangle} * \left(1 - \tanh^2 \left(c^{\langle t \rangle} \right) \right) * \tilde{c}^{\langle t \rangle} * \Gamma_u^{\langle t \rangle} * \left(1 - \Gamma_u^{\langle t \rangle} \right)
\end{aligned}$$

así se tiene finalmente para la compuerta $\Gamma_u^{\langle t \rangle}$

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial z_u^{\langle t \rangle}} &= \left(\frac{\partial \mathcal{L}}{\partial c^{\langle t \rangle}} * \tilde{c}^{\langle t \rangle} * \Gamma_u^{\langle t \rangle} * \left(1 - \Gamma_u^{\langle t \rangle} \right) \right) \\
&\quad + \left(\frac{\partial \mathcal{L}}{\partial h^{\langle t \rangle}} * \Gamma_o^{\langle t \rangle} * \left(1 - \tanh^2 \left(c^{\langle t \rangle} \right) \right) * \tilde{c}^{\langle t \rangle} * \Gamma_u^{\langle t \rangle} * \left(1 - \Gamma_u^{\langle t \rangle} \right) \right) \\
&= \left(\frac{\partial \mathcal{L}}{\partial c^{\langle t \rangle}} + \frac{\partial \mathcal{L}}{\partial h^{\langle t \rangle}} * \Gamma_o^{\langle t \rangle} * \left(1 - \tanh^2 \left(c^{\langle t \rangle} \right) \right) \right) * \tilde{c}^{\langle t \rangle} * \Gamma_u^{\langle t \rangle} * \left(1 - \Gamma_u^{\langle t \rangle} \right)
\end{aligned} \tag{4-38}$$

Para la compuerta de la célula candidata $\tilde{c}^{\langle t \rangle}$, por (4-18), (4-19), (4-20), por regla de la cadena y por (A-2), se tiene

$$\frac{\partial \mathcal{L}}{\partial \tilde{c}^{\langle t \rangle}} = \left(\frac{\partial \mathcal{L}}{\partial c^{\langle t \rangle}} * \frac{\partial c^{\langle t \rangle}}{\partial \tilde{c}^{\langle t \rangle}} * \frac{\partial \tilde{c}^{\langle t \rangle}}{\partial z_c^{\langle t \rangle}} \right) + \left(\frac{\partial \mathcal{L}}{\partial h^{\langle t \rangle}} * \frac{\partial h^{\langle t \rangle}}{\partial c^{\langle t \rangle}} * \frac{\partial c^{\langle t \rangle}}{\partial \tilde{c}^{\langle t \rangle}} * \frac{\partial \tilde{c}^{\langle t \rangle}}{\partial z_c^{\langle t \rangle}} \right)$$

en cuya primera expresión se tiene

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial c^{\langle t \rangle}} * \frac{\partial c^{\langle t \rangle}}{\partial \tilde{c}^{\langle t \rangle}} * \frac{\partial \tilde{c}^{\langle t \rangle}}{\partial z_c^{\langle t \rangle}} &= \frac{\partial \mathcal{L}}{\partial c^{\langle t \rangle}} * \frac{\partial}{\partial \tilde{c}^{\langle t \rangle}} \left(c^{\langle t-1 \rangle} * \Gamma_f^{\langle t \rangle} + \Gamma_u^{\langle t \rangle} * \tilde{c}^{\langle t \rangle} \right) * \frac{\partial \tilde{c}^{\langle t \rangle}}{\partial z_c^{\langle t \rangle}} \\
&= \frac{\partial \mathcal{L}}{\partial c^{\langle t \rangle}} * \Gamma_u^{\langle t \rangle} * \frac{\partial}{\partial z_c^{\langle t \rangle}} \left(\tanh \left(z_c^{\langle t \rangle} \right) \right) \\
&= \frac{\partial \mathcal{L}}{\partial c^{\langle t \rangle}} * \Gamma_u^{\langle t \rangle} * \left(1 - \tilde{c}^2 \right)
\end{aligned}$$

y del segundo término

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial h^{\langle t\rangle}} * \cdots * \frac{\partial \tilde{c}^{\langle t\rangle}}{\partial z_c^{\langle t\rangle}} &= \frac{\partial \mathcal{L}}{\partial h^{\langle t\rangle}} * \frac{\partial}{\partial c^{\langle t\rangle}} \left(\Gamma_o^{\langle t\rangle} * \tanh\left(c^{\langle t\rangle}\right)\right) * \frac{\partial c^{\langle t\rangle}}{\partial \tilde{c}^{\langle t\rangle}} * \frac{\partial \tilde{c}}{\partial z_c^{\langle t\rangle}} \\ &= \frac{\partial \mathcal{L}}{\partial h^{\langle t\rangle}} * \Gamma_o^{\langle t\rangle} * \left(1 - \tanh^2\left(c^{\langle t\rangle}\right)\right) * \Gamma_u^{\langle t\rangle} * \left(1 - \tilde{c}^2\right)\end{aligned}$$

de esta forma, la derivada de la célula candidata es

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \tilde{c}^{\langle t\rangle}} &= \left(\frac{\partial \mathcal{L}}{\partial c^{\langle t\rangle}} * \Gamma_u^{\langle t\rangle} * \left(1 - \tilde{c}^2\right)\right) \\ &\quad + \left(\frac{\partial \mathcal{L}}{\partial h^{\langle t\rangle}} * \Gamma_o^{\langle t\rangle} * \left(1 - \tanh^2\left(c^{\langle t\rangle}\right)\right) * \Gamma_u^{\langle t\rangle} * \left(1 - \tilde{c}^2\right)\right) \\ &= \left(\frac{\partial \mathcal{L}}{\partial c^{\langle t\rangle}} + \Gamma_o^{\langle t\rangle} * \left(1 - \tanh^2\left(c^{\langle t\rangle}\right)\right)\right) * \Gamma_u^{\langle t\rangle} * \left(1 - \tilde{c}^2\right)\end{aligned} \tag{4-39}$$

Para la última compuerta $\Gamma_f^{\langle t\rangle}$, por (4-18), (4-19), (4-23), por la regla de la cadena y por (A-3), se tiene

$$\frac{\partial \mathcal{L}}{\partial z_f^{\langle t\rangle}} = \left(\frac{\partial \mathcal{L}}{\partial c^{\langle t\rangle}} * \frac{\partial c^{\langle t\rangle}}{\partial \Gamma_f^{\langle t\rangle}} * \frac{\partial \Gamma_f^{\langle t\rangle}}{\partial z_f^{\langle t\rangle}}\right) + \left(\frac{\partial \mathcal{L}}{\partial h^{\langle t\rangle}} * \frac{\partial h^{\langle t\rangle}}{\partial c^{\langle t\rangle}} * \frac{\partial c^{\langle t\rangle}}{\partial \Gamma_f^{\langle t\rangle}} * \frac{\partial \Gamma_f^{\langle t\rangle}}{\partial z_f^{\langle t\rangle}}\right)$$

en donde el primer término se calcula así

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial c^{\langle t\rangle}} * \frac{\partial c^{\langle t\rangle}}{\partial \Gamma_f^{\langle t\rangle}} * \frac{\partial \Gamma_f^{\langle t\rangle}}{\partial z_f^{\langle t\rangle}} &= \frac{\partial \mathcal{L}}{\partial c^{\langle t\rangle}} * \frac{\partial}{\partial \Gamma_f^{\langle t\rangle}} \left(c^{\langle t-1\rangle} * \Gamma_f^{\langle t\rangle} + \Gamma_u^{\langle t\rangle} * \tilde{c}^{\langle t\rangle}\right) * \frac{\partial \Gamma_f^{\langle t\rangle}}{\partial z_f^{\langle t\rangle}} \\ &= \frac{\partial \mathcal{L}}{\partial c^{\langle t\rangle}} * c^{\langle t-1\rangle} * \frac{\partial}{\partial z_f^{\langle t\rangle}} \left(\sigma\left(z_f^{\langle t\rangle}\right)\right) \\ &= \frac{\partial \mathcal{L}}{\partial c^{\langle t\rangle}} * c^{\langle t-1\rangle} * \Gamma_f^{\langle t\rangle} * \left(1 - \Gamma_f^{\langle t\rangle}\right)\end{aligned}$$

y la segunda expresión como sigue

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial h^{\langle t\rangle}} * \cdots * \frac{\partial \Gamma_f^{\langle t\rangle}}{\partial z_f^{\langle t\rangle}} &= \frac{\partial \mathcal{L}}{\partial h^{\langle t\rangle}} * \frac{\partial}{\partial c^{\langle t\rangle}} \left(\Gamma_o^{\langle t\rangle} * \tanh\left(c^{\langle t\rangle}\right)\right) * \frac{\partial c^{\langle t\rangle}}{\partial \Gamma_f^{\langle t\rangle}} * \frac{\partial \Gamma_f^{\langle t\rangle}}{\partial z_f^{\langle t\rangle}} \\ &= \frac{\partial \mathcal{L}}{\partial h^{\langle t\rangle}} * \Gamma_o^{\langle t\rangle} * \left(1 - \tanh^2\left(c^{\langle t\rangle}\right)\right) * c^{\langle t-1\rangle} * \Gamma_f^{\langle t\rangle} * \left(1 - \Gamma_f^{\langle t\rangle}\right)\end{aligned}$$

De esta forma la derivada de la compuerta $\Gamma_f^{\langle t\rangle}$ es

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial z_f^{\langle t\rangle}} &= \left(\frac{\partial \mathcal{L}}{\partial c^{\langle t\rangle}} * c^{\langle t-1\rangle} * \Gamma_f^{\langle t\rangle} * \left(1 - \Gamma_f^{\langle t\rangle}\right)\right) \\ &\quad + \left(\frac{\partial \mathcal{L}}{\partial h^{\langle t\rangle}} * \Gamma_o^{\langle t\rangle} * \left(1 - \tanh^2\left(c^{\langle t\rangle}\right)\right) * c^{\langle t-1\rangle} * \Gamma_f^{\langle t\rangle} * \left(1 - \Gamma_f^{\langle t\rangle}\right)\right) \\ &= \left(\frac{\partial \mathcal{L}}{\partial c^{\langle t\rangle}} + \frac{\partial \mathcal{L}}{\partial h^{\langle t\rangle}} * \Gamma_o^{\langle t\rangle} * \left(1 - \tanh^2\left(c^{\langle t\rangle}\right)\right)\right) * c^{\langle t-1\rangle} * \Gamma_f^{\langle t\rangle} * \left(1 - \Gamma_f^{\langle t\rangle}\right)\end{aligned} \tag{4-40}$$

Una vez obtenido el valor de las derivadas parciales de cada compuerta, podemos calcular las derivadas de cada uno de los parámetros, del bloque en el tiempo anterior y de la entrada actual.

Primero veamos el cálculo del bloque en el tiempo anterior. Por la regla de la cadena, por (4-20), (4-21), (4-22) y por (4-23)

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial h^{\langle t-1\rangle}} &= \frac{\partial \mathcal{L}}{\partial z_f^{\langle t\rangle}}\frac{\partial z_f^{\langle t\rangle}}{\partial h^{\langle t-1\rangle}} + \frac{\partial \mathcal{L}}{\partial z_u^{\langle t\rangle}}\frac{\partial z_u^{\langle t\rangle}}{\partial h^{\langle t-1\rangle}} + \frac{\partial \mathcal{L}}{\partial z_c^{\langle t\rangle}}\frac{\partial z_c^{\langle t\rangle}}{\partial h^{\langle t-1\rangle}} \\ &\quad + \frac{\partial \mathcal{L}}{\partial z_o^{\langle t\rangle}}\frac{\partial z_o^{\langle t\rangle}}{\partial h^{\langle t-1\rangle}}\end{aligned} \tag{4-41}$$

En donde, para cada $z_\ell^{\langle t\rangle}$, por (4-24), se tiene:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial z_\ell^{\langle t\rangle}}\frac{\partial z_\ell^{\langle t\rangle}}{\partial h^{\langle t-1\rangle}} &= \frac{\partial \mathcal{L}}{\partial z_\ell^{\langle t\rangle}}\frac{\partial}{\partial h^{\langle t-1\rangle}}\left(W_\ell\left[h^{\langle t-1\rangle}, \mathsf{x}^{\langle t\rangle}\right] + b_\ell\right) \\ &= \frac{\partial \mathcal{L}}{\partial z_\ell^{\langle t\rangle}}\frac{\partial}{\partial h^{\langle t-1\rangle}}\left(W_{\ell,1}h^{\langle t-1\rangle} + W_{\mathsf{x}}\mathsf{x}^{\langle t\rangle} + b_\ell\right) \\ &= W_{\ell,1}^T\frac{\partial \mathcal{L}}{\partial z_\ell^{\langle t\rangle}}\end{aligned} \tag{4-42}$$

de esta forma

$$\frac{\partial \mathcal{L}}{\partial h^{\langle t-1\rangle}} = W_{f,1}^T\frac{\partial \mathcal{L}}{\partial z_f^{\langle t\rangle}} + W_{u,1}^T\frac{\partial \mathcal{L}}{\partial z_u^{\langle t\rangle}} + W_{c,1}^T\frac{\partial \mathcal{L}}{\partial z_c^{\langle t\rangle}} + W_{o,1}^T\frac{\partial \mathcal{L}}{\partial z_o^{\langle t\rangle}} \tag{4-43}$$

De forma similar, por la regla de la cadena, por (4-20), (4-21), (4-22) y por (4-23), para el dato $\mathsf{x}^{\langle t\rangle}$ se tiene

$$\frac{\partial \mathcal{L}}{\partial \mathsf{x}^{\langle t\rangle}} = W_{\mathsf{x}}^T\frac{\partial \mathcal{L}}{\partial z_f^{\langle t\rangle}} + W_{\mathsf{x}}^T\frac{\partial \mathcal{L}}{\partial z_u^{\langle t\rangle}} + W_{\mathsf{x}}^T\frac{\partial \mathcal{L}}{\partial z_c^{\langle t\rangle}} + W_{\mathsf{x}}^T\frac{\partial \mathcal{L}}{\partial z_o^{\langle t\rangle}} \tag{4-44}$$

Finalmente, se puede calcular la derivada de cada uno de los parámetros. Por (4-10) y por (4-13) se tiene

$$\frac{\partial \mathcal{L}}{\partial W_o} = \sum_{t=1}^{T_x}\left(\frac{\partial \mathcal{L}}{\partial z_o^{\langle t\rangle}}\left[h^{\langle t-1\rangle}, \mathsf{x}^{\langle t\rangle}\right]^T\right) \qquad \frac{\partial \mathcal{L}}{\partial b_o} = \sum_{t=1}^{T_x}\frac{\partial \mathcal{L}}{\partial z_o^{\langle t\rangle}}$$

$$\frac{\partial \mathcal{L}}{\partial W_u} = \sum_{t=1}^{T_x}\left(\frac{\partial \mathcal{L}}{\partial z_u^{\langle t\rangle}}\left[h^{\langle t-1\rangle}, \mathsf{x}^{\langle t\rangle}\right]^T\right) \qquad \frac{\partial \mathcal{L}}{\partial b_u} = \sum_{t=1}^{T_x}\frac{\partial \mathcal{L}}{\partial z_u^{\langle t\rangle}}$$

$$\frac{\partial \mathcal{L}}{\partial W_c} = \sum_{t=1}^{T_x}\left(\frac{\partial \mathcal{L}}{\partial z_c^{\langle t\rangle}}\left[h^{\langle t-1\rangle}, \mathsf{x}^{\langle t\rangle}\right]^T\right) \qquad \frac{\partial \mathcal{L}}{\partial b_c} = \sum_{t=1}^{T_x}\frac{\partial \mathcal{L}}{\partial z_c^{\langle t\rangle}}$$

$$\frac{\partial \mathcal{L}}{\partial W_f} = \sum_{t=1}^{T_x}\left(\frac{\partial \mathcal{L}}{\partial z_f^{\langle t\rangle}}\left[h^{\langle t-1\rangle}, \mathsf{x}^{\langle t\rangle}\right]^T\right) \qquad \frac{\partial \mathcal{L}}{\partial b_f} = \sum_{t=1}^{T_x}\frac{\partial \mathcal{L}}{\partial z_f^{\langle t\rangle}}$$

Para terminar la retropropagación se realiza el método del gradiente en descenso para actualizar cada parámetro como se hace en (4-16).

4.3. Unidades Recurrentes Controladas (GRU)

Una Gated Recurrent Unit (GRU) es una arquitectura de red neuronal introducida por *Cho et al.* [2014], cuyo objetivo es el de mitigar el problema de desvanecimiento de gradientes, pero con menos parámetros en comparación con la arquitectura LSTM. Aquí ya no está presente la célula de memoria, en cambio de eso, esa misma información se va propagando por el mismo estado oculto $h^{\langle t \rangle}$.

En este sentido, una GRU, similarmente a una LSTM, posee la capacidad de retener o descartar información dependiendo de su relevancia en la secuencia de entrada. Sin embargo, tan solo incorpora dos compuertas: una *compuerta de reinicio* Γ_r y una *compuerta de actualización* Γ_u.

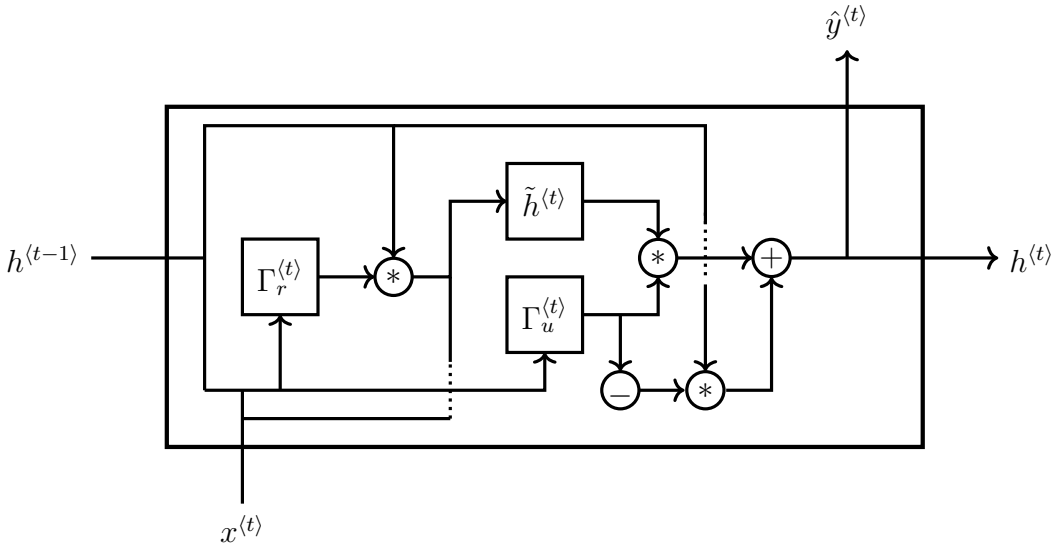


Figura 4-4: Arquitectura de la celda GRU en el tiempo t.

El funcionamiento de cada sección o compuerta de la arquitectura está dado como sigue

- *Compuerta de reinicio*: La compuerta de reinicio está constituida de una matriz $\Gamma_r^{\langle t \rangle}$ que contiene valores en el rango $(0, 1) \subset \mathbb{R}$, estos valores dependen del estado oculto anterior $h^{\langle t-1 \rangle}$ y de la entrada actual $x^{\langle t \rangle}$. Dicha matriz realiza un producto elemento a elemento con el estado oculto anterior, lo que da como resultado la eliminación o retención de valores que considera irrelevantes o no.

- *Compuerta de actualización*: La compuerta de actualización está constituida de una matriz $\Gamma_u^{\langle t \rangle}$ y una matriz candidata $\tilde{h}^{\langle t \rangle}$. La matriz candidata contiene valores en el intervalo $(-1, 1) \subset \mathbb{R}$ y depende de la entrada actual $x^{\langle t \rangle}$ y de la compuerta de reinicio. La matriz Γ_u tiene una función muy similar a la vista en la LSTM, decide qué valores de la matriz candidata son buenos y nos retiene, el resto nos borra. De forma inversa (en el espacio de probabilidad), hace lo mismo para la matriz del estado oculto anterior.

Al final, la actualización se completa al realizar la ponderación entre las matrices resultantes de operar con Γ_u.

Formalmente, una GRU se define a través de la siguiente forma recursiva

$$\hat{y}^{\langle t\rangle} = \tanh\left(W_y h^{\langle t\rangle} + b_y\right) \tag{4-45}$$

en donde

$$h^{\langle t\rangle} = \begin{cases} 0, & \text{si } t = 0 \\ \Gamma_u^{\langle t\rangle} * \tilde{h}^{\langle t\rangle} + \left(1 - \Gamma_u^{\langle t\rangle}\right) * h^{\langle t-1\rangle}, & \text{en otro caso.} \end{cases} \tag{4-46}$$

$$\tilde{h}^{\langle t\rangle} = \tanh\left(W_h\left[\Gamma_r^{\langle t\rangle} * h^{\langle t-1\rangle}, \mathsf{x}^{\langle t\rangle}\right] + b_h\right) \tag{4-47}$$

$$\Gamma_u^{\langle t\rangle} = \sigma\left(W_u\left[h^{\langle t-1\rangle}, \mathsf{x}^{\langle t\rangle}\right] + b_u\right) \tag{4-48}$$

$$\Gamma_r^{\langle t\rangle} = \sigma\left(W_r\left[h^{\langle t-1\rangle}, \mathsf{x}^{\langle t\rangle}\right] + b_r\right) \tag{4-49}$$

y, por simplicidad, se tiene que

$$W_\ell\left[h^{\langle t-1\rangle}, \mathsf{x}^{\langle t\rangle}\right] = W_{\ell,1} h^{\langle t-1\rangle} + W_{\mathsf{x}} \mathsf{x}^{\langle t\rangle} \tag{4-50}$$

4.3.1. Retropropagación a través del tiempo

Para calcular las derivadas de los parámetros de una GRU tomamos la misma idea con la que se calcula las derivadas en LSTM.

Tomando el método de la *retropropagación a través del tiempo*, considerando la función de pérdida $\mathcal{L}[\phi]$ definida en (4-3) para el conjunto de parámetros $\phi := \{W_h, W_u, W_r, W_y, b_h, b_u, b_r, b_y\}$, podemos calcular las derivadas.

Para la salida, por la regla de la cadena, por (4-3), por (4-4) y por (A-2), se tiene

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial z_y^{\langle t\rangle}} &= \frac{\partial \mathcal{L}^{\langle t\rangle}}{\partial \hat{y}^{\langle t\rangle}} \frac{\partial \hat{y}^{\langle t\rangle}}{\partial z_y^{\langle t\rangle}} \\ &= \frac{\partial \mathcal{L}^{\langle t\rangle}}{\partial \hat{y}^{\langle t\rangle}} \left(1 - \tanh^2\left(z_y^{\langle t\rangle}\right)\right) \end{aligned} \tag{4-51}$$

Con esto se calculan las derivadas para W_y y para b_y, por (4-8) y por 4-51 se tiene

$$\frac{\partial \mathcal{L}}{\partial W_y} = \sum_{t=1}^{T_x} \left(\frac{\partial \mathcal{L}}{\partial z_y^{\langle t\rangle}} \left(h^{\langle t\rangle}\right)^T\right) \tag{4-52}$$

y, por (4-15) y por 4-51

$$\frac{\partial \mathcal{L}}{\partial b_y} = \sum_{t=1}^{T_x} \frac{\partial \mathcal{L}}{\partial z_y^{\langle t\rangle}} \tag{4-53}$$

Similarmente a como ocurría en (4-30), la derivada parcial de $h^{\langle t\rangle}$ respecto a $\mathcal{L}$ depende del bloque en el tiempo siguiente $t+1$. De esta forma se considera lo mismo, se calcula el gradiente para el último instante de tiempo $\partial\mathcal{L}/\partial h^{\langle T_x\rangle}$ y luego se retropropaga iterativamente hacia atrás, es decir, se calcula el gradiente del estado $h^{\langle t-1\rangle}$.

Para el último bloque $t = T_x$, el gradiente es el mismo visto en (4-31)

$$\frac{\partial\mathcal{L}}{\partial h^{\langle T_x\rangle}} = \frac{\partial\mathcal{L}}{\partial\hat{y}^{\langle T_x\rangle}} W_y^T$$

Ahora se puede calcular el resto de derivadas, primeramente para el estado candidato $\tilde{h}^{\langle t\rangle}$ se calcula la derivada para $z_h^{\langle t\rangle} := W_h[h^{\langle t-1\rangle}, \mathsf{x}^{\langle t\rangle}] + b_h$. Por (4-46), por regla de la cadena y por (A-2), se tiene

$$\begin{aligned}
\frac{\partial\mathcal{L}}{\partial z_h^{\langle t\rangle}} &= \frac{\partial\mathcal{L}}{\partial h^{\langle t\rangle}} * \frac{\partial h^{\langle t\rangle}}{\partial\tilde{h}^{\langle t\rangle}} * \frac{\partial\tilde{h}^{\langle t\rangle}}{\partial z_h^{\langle t\rangle}} \\
&= \frac{\partial\mathcal{L}}{\partial h^{\langle t\rangle}} * \frac{\partial}{\partial\tilde{h}^{\langle t\rangle}} \left(\Gamma_u^{\langle t\rangle} * \tilde{h}^{\langle t\rangle} + \left(1-\Gamma_u^{\langle t\rangle}\right) * h^{\langle t-1\rangle}\right) * \frac{\partial\tilde{h}^{\langle t\rangle}}{\partial z_h^{\langle t\rangle}} \\
&= \frac{\partial\mathcal{L}}{\partial h^{\langle t\rangle}} * \Gamma_u^{\langle t\rangle} * \left(1 - \tanh^2\left(z_h^{\langle t\rangle}\right)\right)
\end{aligned} \tag{4-54}$$

Para la compuerta de actualización $\Gamma_u^{\langle t\rangle}$, por (4-46), por (4-48), por la regla de la cadena y por (A-3)

$$\begin{aligned}
\frac{\partial\mathcal{L}}{\partial z_u^{\langle t\rangle}} &= \frac{\partial\mathcal{L}}{\partial h^{\langle t\rangle}} * \frac{\partial h^{\langle t\rangle}}{\partial\Gamma_u^{\langle t\rangle}} * \frac{\partial\Gamma_u^{\langle t\rangle}}{\partial z_u^{\langle t\rangle}} \\
&= \frac{\partial\mathcal{L}}{\partial h^{\langle t\rangle}} * \frac{\partial}{\partial\Gamma_u^{\langle t\rangle}} \left(\Gamma_u^{\langle t\rangle} * \tilde{h}^{\langle t\rangle} + \left(1-\Gamma_u^{\langle t\rangle}\right) * h^{\langle t-1\rangle}\right) * \frac{\partial\Gamma_u^{\langle t\rangle}}{\partial z_u^{\langle t\rangle}} \\
&= \frac{\partial\mathcal{L}}{\partial h^{\langle t\rangle}} * \left(\tilde{h}^{\langle t\rangle} - h^{\langle t-1\rangle}\right) * \frac{\partial}{\partial z_u^{\langle t\rangle}} \left(\sigma(z_u^{\langle t\rangle})\right) \\
&= \frac{\partial\mathcal{L}}{\partial h^{\langle t\rangle}} * \left(\tilde{h}^{\langle t\rangle} - h^{\langle t-1\rangle}\right) * \Gamma_u^{\langle t\rangle} * \left(1-\Gamma_u^{\langle t\rangle}\right)
\end{aligned} \tag{4-55}$$

Para la compuerta de reinicio $\Gamma_r^{\langle t\rangle}$, por (4-47), por (4-49), por regla de la cadena y por (A-3), se tiene

$$\begin{aligned}
\frac{\partial\mathcal{L}}{\partial z_r^{\langle t\rangle}} &= \frac{\partial\mathcal{L}}{\partial\tilde{h}^{\langle t\rangle}} * \frac{\partial\tilde{h}^{\langle t\rangle}}{\partial\Gamma_r^{\langle t\rangle}} * \frac{\partial\Gamma_r^{\langle t\rangle}}{\partial z_r^{\langle t\rangle}} \\
&= \frac{\partial\mathcal{L}}{\partial\tilde{h}^{\langle t\rangle}} * \frac{\partial\tilde{h}^{\langle t\rangle}}{\partial\Gamma_r^{\langle t\rangle}} * \Gamma_r^{\langle t\rangle} * \left(1-\Gamma_r^{\langle t\rangle}\right)
\end{aligned} \tag{4-56}$$

Ahora bien, en la primera expresión, por regla de la cadena generalizada a matrices (dado que en $\Gamma_r^{\langle t\rangle}$

se multiplica elemento a elemento con $h^{\langle t-1\rangle}$ en $\tilde{h}^{\langle t\rangle}$), por (4-54) y por (4-50), se tiene

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \tilde{h}^{\langle t\rangle}} * \frac{\partial \tilde{h}^{\langle t\rangle}}{\partial \Gamma_r^{\langle t\rangle}} &= \frac{\partial \mathcal{L}}{\partial \tilde{h}^{\langle t\rangle}} * \frac{\partial}{\partial \Gamma_r^{\langle t\rangle}} \left(\tanh\left(z_h^{\langle t\rangle}\right)\right) \\ &= \frac{\partial \mathcal{L}}{\partial z_h^{\langle t\rangle}} * \frac{\partial z_h^{\langle t\rangle}}{\partial \Gamma_r^{\langle t\rangle}} \\ &= \frac{\partial \mathcal{L}}{\partial z_h^{\langle t\rangle}} * \frac{\partial}{\partial \Gamma_r^{\langle t\rangle}} \left(W_h \left[\Gamma_r^{\langle t\rangle} * h^{\langle t-1\rangle}, \mathbf{x}^{\langle t\rangle}\right] + b_h\right) \\ &= \underbrace{\left(W_{h,1}^T \frac{\partial \mathcal{L}}{\partial z_h^{\langle t\rangle}}\right)}_{\text{Backprop a través de } W} * \frac{\partial}{\partial \Gamma_r^{\langle t\rangle}} \left(\Gamma_r^{\langle t\rangle} * h^{\langle t-1\rangle}\right) \\ &= \left(W_{h,1}^T \frac{\partial \mathcal{L}}{\partial z_h^{\langle t\rangle}}\right) * h^{\langle t-1\rangle} \end{aligned} \tag{4-57}$$

De esta forma la ecuación (4-56), por (4-57) se puede escribir como sigue

$$\frac{\partial \mathcal{L}}{\partial z_r^{\langle t\rangle}} = \left(W_{h,1}^T \frac{\partial \mathcal{L}}{\partial z_h^{\langle t\rangle}}\right) * h^{\langle t-1\rangle} * \Gamma_r^{\langle t\rangle} * \left(1 - \Gamma_r^{\langle t\rangle}\right) \tag{4-58}$$

Teniendo estas derivadas se puede calcular la derivada del bloque anterior $t-1$. Por la regla de la cadena, por (4-46), (4-54), (4-55) y (4-58), se tiene

$$\frac{\partial \mathcal{L}}{\partial h^{\langle t-1\rangle}} = \frac{\partial \mathcal{L}}{\partial h^{\langle t\rangle}} * \frac{\partial h^{\langle t\rangle}}{\partial h^{\langle t-1\rangle}} + \frac{\partial \mathcal{L}}{\partial \Gamma_u^{\langle t\rangle}} \frac{\partial \Gamma_u^{\langle t\rangle}}{\partial h^{\langle t-1\rangle}} + \frac{\partial \mathcal{L}}{\partial \Gamma_r^{\langle t\rangle}} \frac{\partial \Gamma_r^{\langle t\rangle}}{\partial h^{\langle t-1\rangle}} + \frac{\partial \mathcal{L}}{\partial \tilde{h}^{\langle t\rangle}} \frac{\partial \tilde{h}^{\langle t\rangle}}{\partial h^{\langle t-1\rangle}} \tag{4-59}$$

en donde

$$\begin{aligned}\frac{\partial h^{\langle t\rangle}}{\partial h^{\langle t-1\rangle}} &= \frac{\partial}{\partial h^{\langle t-1\rangle}} \left(\Gamma_u^{\langle t\rangle} * \tilde{h}^{\langle t\rangle} + \left(1 - \Gamma_u^{\langle t\rangle}\right) * h^{\langle t-1\rangle}\right) \\ &= \left(1 - \Gamma_u^{\langle t\rangle}\right) \end{aligned} \tag{4-60}$$

y, por regla de la cadena y por (4-50), se tiene

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \tilde{h}^{\langle t\rangle}} \frac{\partial \tilde{h}^{\langle t\rangle}}{\partial h^{\langle t-1\rangle}} &= \frac{\partial \mathcal{L}}{\partial \tilde{h}^{\langle t\rangle}} \frac{\partial}{\partial h^{\langle t-1\rangle}} \left(\tanh\left(z_h^{\langle t\rangle}\right)\right) \\ &= \frac{\partial \mathcal{L}}{\partial z_h^{\langle t\rangle}} \frac{\partial z_h^{\langle t\rangle}}{\partial h^{\langle t-1\rangle}} \\ &= \frac{\partial \mathcal{L}}{\partial z_h^{\langle t\rangle}} \frac{\partial}{\partial h^{\langle t-1\rangle}} \left(W_h \left[\Gamma_r^{\langle t\rangle}, h^{\langle t-1\rangle}\right] + b_h\right) \\ &= \left(W_{h,1}^T \frac{\partial \mathcal{L}}{\partial z_h^{\langle t\rangle}}\right) * \Gamma_r^{\langle t\rangle} \end{aligned} \tag{4-61}$$

similarmente, para cada Γ_ℓ se te tiene

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \Gamma_\ell^{\langle t\rangle}}\frac{\partial \Gamma_\ell^{\langle t\rangle}}{\partial h^{\langle t-1\rangle}} &= \frac{\partial \mathcal{L}}{\partial z_\ell^{\langle t\rangle}}\frac{\partial z_\ell^{\langle t\rangle}}{\partial h^{\langle t-1\rangle}} \\ &= W_{\ell,1}^T \frac{\partial \mathcal{L}}{\partial z_\ell^T}\end{aligned} \tag{4-62}$$

de esta forma la ecuación (4-59) finalmente tiene la siguiente forma

$$\frac{\partial \mathcal{L}}{\partial h^{\langle t-1\rangle}} = \frac{\partial \mathcal{L}}{\partial h^{\langle t\rangle}} * \left(1-\Gamma_u^{\langle t\rangle}\right) + W_{u,1}^T \frac{\partial \mathcal{L}}{\partial z_u^{\langle t\rangle}} + W_{r,1}^T \frac{\partial \mathcal{L}}{\partial z_r^{\langle t\rangle}} + \left(W_{h,1}^T \frac{\partial \mathcal{L}}{\partial z_h^{\langle t\rangle}}\right) * \Gamma_r^{\langle t\rangle} \tag{4-63}$$

De forma similar, por la regla de la cadena, por (4-47), (4-48), (4-49) y (4-50), para la salida $\mathsf{x}^{\langle t\rangle}$ se tiene

$$\frac{\partial \mathcal{L}}{\partial \mathsf{x}^{\langle t\rangle}} = W_{\mathsf{x}}^T \frac{\partial \mathcal{L}}{\partial z_u^{\langle t\rangle}} + W_{\mathsf{x}}^T \frac{\partial \mathcal{L}}{\partial z_r^{\langle t\rangle}} + W_{\mathsf{x}}^T \frac{\partial \mathcal{L}}{\partial z_h^{\langle t\rangle}} \tag{4-64}$$

Con esto, se puede calcular la derivada de cada parámetro, por (4-9) y por (4-13) se tiene

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial W_u} &= \sum_{t=1}^{T_x}\left(\frac{\partial \mathcal{L}}{\partial z_u^{\langle t\rangle}}\left[h^{\langle t-1\rangle}, \mathsf{x}^{\langle t\rangle}\right]^T\right) & \frac{\partial \mathcal{L}}{\partial b_u} &= \sum_{t=1}^{T_x}\frac{\partial \mathcal{L}}{\partial z_u^{\langle t\rangle}} \\ \frac{\partial \mathcal{L}}{\partial W_h} &= \sum_{t=1}^{T_x}\left(\frac{\partial \mathcal{L}}{\partial z_h^{\langle t\rangle}}\left[h^{\langle t-1\rangle}, \mathsf{x}^{\langle t\rangle}\right]^T\right) & \frac{\partial \mathcal{L}}{\partial b_h} &= \sum_{t=1}^{T_x}\frac{\partial \mathcal{L}}{\partial z_h^{\langle t\rangle}} \\ \frac{\partial \mathcal{L}}{\partial W_r} &= \sum_{t=1}^{T_x}\left(\frac{\partial \mathcal{L}}{\partial z_r^{\langle t\rangle}}\left[h^{\langle t-1\rangle}, \mathsf{x}^{\langle t\rangle}\right]^T\right) & \frac{\partial \mathcal{L}}{\partial b_r} &= \sum_{t=1}^{T_x}\frac{\partial \mathcal{L}}{\partial z_r^{\langle t\rangle}}\end{aligned}$$

Para terminar la retropropagación se realiza el método del gradiente en descenso para actualizar cada parámetro como se hace en (4-16).

Capítulo 5

Desarrollo de los modelos

En el presente capítulo se expone el desarrollo de los modelos planteados para abordar el problema de la predicción de la inflación. En primer lugar, se describe el *dataset* utilizado y el proceso de preparación de los datos. Posteriormente, se presenta el primer modelo propuesto, una red LSTM implementada desde cero, junto con su procedimiento de entrenamiento y los resultados obtenidos. Finalmente, se realiza el mismo análisis para el segundo modelo, basado en una arquitectura GRU.

El laboratorio desarrollado puede consultarse en el repositorio del proyecto.[1]

5.1. Conjunto de Datos

El problema a tratar en este laboratorio corresponde a la predicción de la inflación en Colombia. Para ello, se emplea como *dataset* el registro histórico del porcentaje de inflación mensual desde el año 2000, proporcionado por el Banco de la República.[2]

Al graficar la serie temporal —como se observa en la Figura **5-1**— se identifican patrones de variación notablemente extremos entre los años 1955 y 2000. A partir del año 2000, el comportamiento de la serie se torna más estable, con excepción de los años 2021, 2022 y 2023 por razón de la pandemia. Por esta razón, se decidió entrenar los modelos únicamente con los datos a partir del año 2000, para evitar sesgo en los datos de entrenamiento.

Además, se propone desarrollar los modelos sobre ventanas de 12 meses. Así pues, se crean secuencias con los datos que sirven como entrada de los modelos que predicen el décimo tercer mes de la secuencia de meses; con este tamaño de ventanas se espera llegar a predecir la inflación de hasta un año entero.

[1] `https://github.com/MateoOrtiz001/Laboratory-LSTM-GRU-Inflation`
[2] `https://suameca.banrep.gov.co/buscador-de-series/#/`

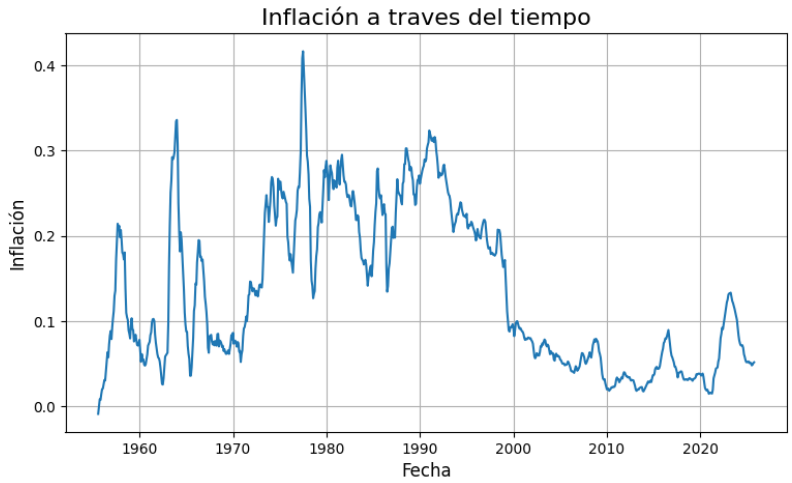


Figura 5-1: Serie temporal de la inflación desde 1955 hasta 2025

De esta forma se tiene un total de 297 muestras, de las cuales se ha decidido tomar el 80 % para entrenamiento y el 20 % restante para test. En la Figura **5-2** se puede observar el corte entre los datos de entrenamiento y los datos de prueba, que se da en noviembre del año 2020.

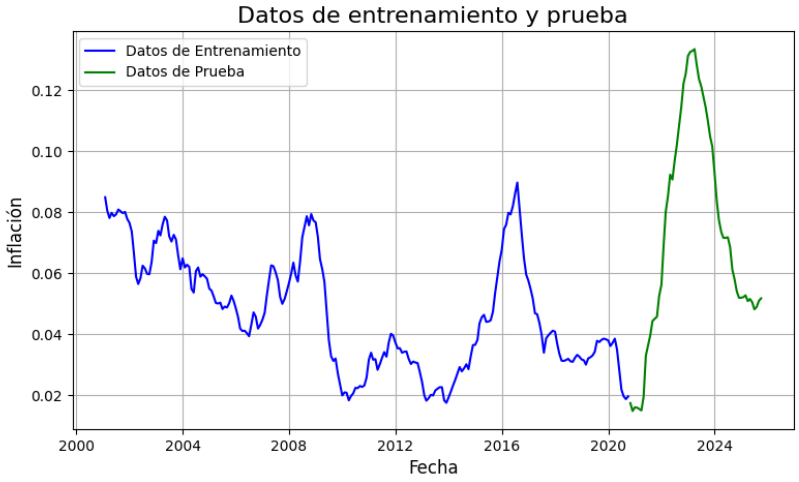


Figura 5-2: Serie temporal de los datos de entrenamiento y de los datos de prueba.

5.1.1. Normalización de los datos

Previo al entrenamiento, los datos se someten a un proceso de normalización con el fin de mejorar la estabilidad numérica del modelo y favorecer la convergencia durante la retropropagación. Este procedimiento, ampliamente utilizado en el entrenamiento de redes neuronales, consiste en escalar cada variable de entrada mediante la denominada *escala estándar*.

Sea $X \in \mathbb{R}^d$ una observación del conjunto de entrenamiento, con componentes $X_1, \cdots, X_d$. Para cada componente se calcula la media empírica μ_j y la desviación estándar empírica σ_j del conjunto de entrenamiento. La muestra normalizada se define entonces como

$$Z_j = \frac{X_j - \mu_j}{\sigma_j}, \quad j = 1, \cdots, d$$

De esta forma, las variables normalizadas presentan media aproximadamente cero y varianza unitaria, lo cual contribuye a evitar escalas desbalanceadas entre los distintos atributos.

Esta normalización resulta particularmente adecuada en el contexto de las redes LSTM y GRU, ya que los valores centrados de cero permiten que la función de activación tanh, utilizada en las secciones 4.2 y 4.3, opere en una región óptima para favorecer gradientes estables y un entrenamiento eficiente.

5.2. Hiperparámetros

La selección de hiperparámetros para los dos modelos se realizó de forma empírica. Dado el alcance del problema, el tamaño del conjunto de datos y el propósito introductorio del laboratorio, se optó por no incorporar técnicas adicionales como regularización explícita, búsqueda sistemática de hiperparámetros o variaciones en los esquemas de inicialización. En su lugar, se entrenó un único modelo por arquitectura y se utilizaron los mismos hiperparámetros tanto para la red LSTM como para la red GRU, con el fin de facilitar la comparación directa entre ambas implementaciones.

5.2.1. Inicialización de pesos

Todos los parámetros de las dos arquitecturas se inicializaron de manera homogénea utilizando la inicialización de Xavier en su variante normal *Glorot & Bengio* [2010]. Esta elección contrasta con la inicialización por defecto empleada en Keras , donde los pesos de las capas recurrentes pueden seguir esquemas distintos según su función interna.

Formalmente, si N denota la dimensión de entrada (fan-in) asociada a cada matriz de pesos, los valores de sus elementos se muestrean a partir de una distribución Gaussiana

$$W \sim \mathcal{N}(0, \sigma), \quad \sigma = \sqrt{\frac{2}{N}}$$

Este esquema permite mantener la varianza de las activaciones aproximadamente constante a lo largo de la red, evitando que las unidades recurrentes saturen prematuramente en regiones de gradiente cercano a cero.

Por su parte, los vectores de sesgo se inicializan en cero, siguiendo la práctica estándar para este tipo de arquitecturas.

5.2.2. Función de pérdida

Como función de pérdida se empleó el error cuadrático medio (MSE) por ser una medida estándar y adecuada para problemas de regresión con datos numéricos continuos.

Sea m el número de series (muestras). Denotando por $Y_i^{\langle t \rangle}$ el valor observado de la i-ésima muestra en el tiempo t y por $\hat{Y}_i^{\langle t \rangle}(\phi)$ la predicción correspondiente obtenida a partir de los parámetros ϕ. Como se presenta en (4-3), definimos la pérdida promedio como

$$\mathcal{L}[\phi] = \frac{1}{2mT_x} \sum_{i=1}^{m} \sum_{t=1}^{T_x} \left(Y_i^{\langle t \rangle} - \hat{Y}_i^{\langle t \rangle}(\phi) \right)^2$$

En donde el factor de normalización $1/2mT_x$ es conveniente para simplificar el gradiente durante el entrenamiento, ya que cancela el término proveniente de la derivada del cuadrado.

Es relevante observar que en este caso particular de estudio, cada muestra produce una única predicción, por lo que la expresión se reduce a un único término por serie. La función de pérdida entonces queda simplemente como sigue

$$\mathcal{L}[\phi] = \frac{1}{2m} \sum_{i=1}^{m} \left(Y_i^{\langle T_x \rangle} - \hat{Y}_i^{\langle T_x \rangle}(\phi) \right)^2$$

5.2.3. Mini lotes

Para mejorar la eficiencia computacional durante el entrenamiento y aportar un nivel controlado de regularización implícita, los modelos se entrenaron utilizando mini lotes de tamaño 64. El uso de mini lotes permite equilibrar el compromiso entre la alta varianza del aprendizaje puramente estocástico y el costo computacional del descenso por gradiente a partir del conjunto completo. En particular, el gradiente se estima a partir de un subconjunto representativo del conjunto de entrenamiento, lo que introduce un nivel moderado de ruido beneficioso para evitar mínimos locales poco profundos y reducir el riesgo de sobreajuste.

Formalmente, si el conjunto de entrenamiento es

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$$

se particiona en mini lotes

$$\mathcal{B}_k = \{(x_i, y_i)\}_{i \in I_k}, \quad |\mathcal{B}_k| = 64$$

en donde los índices I_k se seleccionan aleatoriamente en cada época. Con este mini lote con $|\mathcal{B}_k| = 64$ muestras, la pérdida utilizada en cada paso de optimización es

$$\mathcal{L}_{\mathcal{B}_k}[\phi] = \frac{1}{2|\mathcal{B}_k|} \sum_{i \in \mathcal{B}_k} \left(Y_i^{\langle T_x \rangle} - \hat{Y}_i^{\langle T_x \rangle}(\phi) \right)^2 \tag{5-1}$$

cuyo gradiente proporciona una estimación estocástica del gradiente total.

5.2.4. Optimizador

Como optimizador se ha utilizado el algoritmo de *estimación de momento adaptativo* (Adam) introducido por *Kingma & Ba* [2017], método ampliamente popular en este tipo de modelos complejos debido a su estabilidad y rapidez de convergencia. El algoritmo combina la idea del momentum con la estrategia de promedios móviles de los cuadrados de los gradientes (RMSprop), obteniendo así un proceso de optimización balanceado y adaptable a distintas escalas de actualización.

El algoritmo mantiene dos estimaciones: un promedio móvil del gradiente, denominado momento de primer orden, y un promedio móvil de los cuadrados del gradiente, correspondiente al momento de segundo orden.

Ambas estimaciones se corrigen para compensar el sesgo introducido durante las primeras iteraciones. Formalmente, para un parámetro genérico θ y su gradiente $\nabla_\theta \mathcal{L}_t$[3], las actualizaciones están dadas por:

$$v_t = \beta_1 v_{t-1} + (1 - \beta_1) \nabla_\theta \mathcal{L}_t$$
$$s_t = \beta_2 s_{t-1} + (1 - \beta_2) (\nabla_\theta \mathcal{L}_t)^2$$

en donde v_t y s_t son los momentos de primer orden y segundo orden, respectivamente. Para corregir el sesgo inicial se utilizan:

$$\hat{v}_t = \frac{v_t}{1 - \beta_1^t} \qquad\qquad \hat{s}_t = \frac{s_t}{1 - \beta_2^t}$$

Finalmente, la actualización del parámetro toma la forma:

$$\theta_t \longleftarrow \theta_{t-1} - \eta \frac{\hat{v}_t}{\sqrt{\hat{s}_t} + \varepsilon} \tag{5-2}$$

en donde η es la tasa de aprendizaje y ε un término de regularización numérica.

Para ambos modelos se utilizan como parámetros:

- Tasa de aprendizaje: $\eta = 0{,}001$.

[3] Es importante aclarar que la notación $\mathcal{L}_t$, así como el resto de notaciones en el resto de esta sección de la forma ς_t, se refiere a un instante de tiempo t en las épocas de entrenamiento. No se debe confundir con un instante de la ventana de los datos de entrada.

- Tasa de decaimiento del momento: $\beta_1 = 0{,}9$.
- Tasa de decaimiento del segundo momento: $\beta_2 = 0{,}999$.
- Regularizador numérico: $\varepsilon = 0{,}00000001$.

5.2.5. Número de unidades/neuronas del estado oculto

Para el modelo se seleccionaron $n_h = 78$ unidades en el estado oculto. Este valor fue elegido tras realizar pruebas preliminares sobre modelos en Keras. Asimismo, este tamaño permite capturar suficientemente las dependecias temporales sin generar un aumento significativo en el tiempo de entrenamiento.

5.3. Contenido del laboratorio

El laboratorio está distribuido en 7 secciones

1. Dataset
2. Problema de gradientes en redes neuronales recurrentes
3. Construyendo una LSTM
    a) Construyendo el forward pass de una celda
    b) Construyendo el forward pass del modelo
    c) Construyendo la retropropagación de una celda
    d) Construyendo la repropropagación del modelo
    e) Entrenamiento del modelo
    f) Predicción del modelo
4. Construyendo una GRU
    a) Construyendo el forward pass de una celda
    b) Construyendo el forward pass del modelo
    c) Construyendo la retropropagación de una celda
    d) Construyendo la repropropagación del modelo
    e) Entrenamiento del modelo
    f) Predicción del modelo

5. Implementaciones con Keras
6. Predicciones con intervalos de confianza (opcional)
7. Bibliografía

En cada uno se da una explicación teórica y matemática de lo que se quiere implementar, posteriormente hay celdas para implementar con autocalificadores -véase la Figura **5-3**,**5-4**,**5-5**-. En la primera

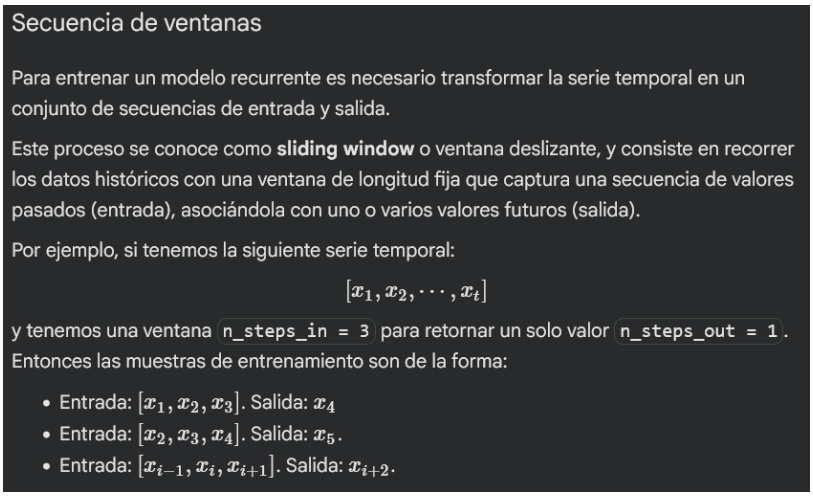


Figura 5-3: Pantallazo realizado a la explicación de la sección *Secuencia de ventanas* del laboratorio.

```python
def create_sequences(data, n_steps_in, n_steps_out):
    """
    Crea secuencias de entrada-salida para LSTM usando sliding window
    Argumentos:
    data -- array 1D con los datos de inflación
    n_steps_in -- número de pasos temporales de entrada (ventana)
    n_steps_out -- número de pasos a predecir (tamaño de la salida).

    Retorna:
    X -- array con secuencias de entrada, shape (n_samples, n_steps_in)
    Y -- array con valores objetivo, shape (n_samples, n_steps_out)
    """
    X, Y = [], []
    ### START CODE HERE ###
    # Recorremos la serie aplicando la ventana deslizante
    for i in range( ??? ):
        # Secuencia de entrada: ventana de n_steps_in valores. X.append(...)
        X.append( ??? )
        # Valor objetivo: el siguiente valor después de la ventana. Y.append(..
        Y.append( ??? )
    ### END CODE HERE ###

    return np.array(X), np.array(Y)
```

Figura 5-4: Pantallazo realizado al código de la sección *Secuencia de ventanas* del laboratorio.

sección se explica el conjunto de datos, se hace la carga y la preparación de los mismos para el uso posterior. Como celdas de ejercicios el usuario debe terminar una función para generar la secuencias de datos llamada `create_sequences` y una función para redimensionar la forma de los tensores utilizados como datos llamada `reshape_data`.

```
Datos aleatorios generados:
[0.37454012 0.95071431 0.73199394 0.59865848 0.15601864 0.15599452
 0.05808361 0.86617615 0.60111501 0.70807258 0.02058449 0.96990985
 0.83244264 0.21233911 0.18182497 0.18340451 0.30424224 0.52475643
 0.43194502 0.29122914]

Primer secuencia de entrada (X):
[0.37454012 0.95071431 0.73199394 0.59865848 0.15601864]

Primer secuencia de salida (Y):
[0.15599452 0.05808361]
Test Aprobado
```

Figura 5-5: Pantallazo realizado al autocalificable de la sección *Secuencia de ventanas* del laboratorio.

En la segunda sección simplemente se hace una pequeña explicación del desvanecimiento de gradientes, no hay celdas para programar.

En la tercera sección se construye el modelo LSTM, esta primera sección, así como el diseño del resto del laboratorio, está fuertemente basada en el laboratorio de redes neuronales recurrentes del curso *Ng & deeplearning.ai* [2018]. En la primera parte se explica la arquitectura de cada bloque, con ello el usuario debe terminar una función para realizar la propagación en la predicción del modelo, llamada `lstm_cell_forward`. En la segunda parte se explica el funcionamiento del modelo completo, como ejercicio el usuario debe terminar una función para realizar toda la predicción completa llamada `lstm_forward`. En la siguiente parte se explica el proceso de retropropagación, el usuario debe calcular las derivadas de cada uno de los parámetros en la función llamada `lstm_cell_backward`. En la siguiente subsección se explica la retropropagación completa, el usuario debe completar la función `lstm_backward` para esto, después se explicará la inicialización de los parámetros, el usuario debe completar la función para hacer esto llamada `initialize_lstm_parameters`. En la siguiente parte se explica el proceso de entrenamiento, el usuario debe completar la función `model_lstm_adam` y después entrenarlo. En la última parte de la sección se explica el proceso de predicción, el usuario debe completar la función `predict_LSTM`, después de esto hay algunas celdas para obtener las predicciones con su gráfica.

En la cuarta sección se construye el modelo GRU, la ruta de esta sección es la misma a la anterior. Sin embargo, las celdas calificadoras se limitan a simplemente las funciones `gru_cell_forward`, `gru_forward`, `gru_cell_backward`, `gru_backward` y `model_gru_adam`.

En la quinta sección simplemente se menciona que la librería Keras ofrece funciones para invocar capas LSTM y GRU, después de esto se ejecutan celdas para entrenar dos modelos con estas arquitecturas. Aquí no hay celdas calificables. Después de eso se realizan las predicciones de todos los modelos y se hace una gráfica comparativa entre las cuatro comparaciones.

En la sexta sección opcional, se explica el método de bootstraping para obtener intervalos de confianza sobre las predicciones de los modelos implementados desde cero. El usuario deberá terminar la función llamada `bootstrap_prediction_intervals` y tras esto, realizar dos gráficas comparativas entre los modelos LSTM y los modelos GRU.

La séptima sección simplemente contiene las referencias bibliográficas.

Capítulo 6

Resultados y Discusión

En este capítulo se presentan los principales resultados derivados del entrenamiento y validación de las arquitecturas LSTM y GRU implementadas. Cada resultado es discutido inmediatamente después de su exposición, con el propósito de analizar su significado, su coherencia con la metodología empleada y su relevancia en el contexto del problema aplicado. Se incluyen las curvas de entrenamiento, las predicciones obtenidas y una comparación entre las implementaciones manuales y las basadas en Keras.

6.1. Entrenamiento de los modelos

Los modelos fueron entrenados durante 200 épocas utilizando los hiperparámetros descritos en la sección anterior. La evolución de la función de pérdida definida en (5-1) puede observarse en la Figura **6-1**. Adicionalmente, en la Tabla **6-1** se presenta un resumen de los valores correspondientes cada veinte épocas, lo cual permite apreciar la tendencia global del proceso de entrenamiento.

En primer lugar, se observa que el modelo GRU reduce la función de pérdida con mayor rapidez y, en todas las épocas analizadas, presenta valores inferiores a los del modelo LSTM. No obstante, al finalizar las 200 épocas ambas implementaciones convergen a un valor cercano a $0{,}0045$. En contraste, los modelos entrenados con Keras alcanzan valores relativamente más altos, con una convergencia alrededor de $0{,}0099$; en este caso el modelo LSTM de Keras converge más rápido que las implementaciones manuales.

Existen diversas razones que pueden explicar estas diferencias. En primer lugar, como se discutió en la sección 5.2.1, Keras emplea esquemas de inicialización de pesos distintos a los utilizados en la implementación manual, lo cual puede influir tanto en la velocidad de convergencia como en la tendencia a evitar el sobreajuste. En segundo lugar, Keras realiza el cómputo mediante optimizaciones internas altamente eficientes —incluyendo operaciones vectorizadas, kernels en bajo nivel y rutinas

Resumen Historial Función de Pérdida				
Época	LSTM	GRU	LSTM (Keras)	GRU (Keras)
20	0,0208	0,0113	0,0538	0,0758
40	0,0123	0,0080	0,0382	0,0435
60	0,0092	0,0060	0,0234	0,0246
80	0,0068	0,0053	0,0175	0,0187
100	0,0070	0,0050	0,0135	0,0149
120	0,0056	0,0045	0,0129	0,0133
140	0,0059	0,0047	0,0121	0,0118
160	0,0050	0,0045	0,0111	0,0137
180	0,0049	0,0044	0,0094	0,0116
200	0,0047	0,0043	0,0095	0,0106

Tabla 6-1: Resumen de los valores obtenidos de la función de pérdida durante el entrenamiento.

numéricamente estables– que difieren de las operaciones implementadas directamente en NumPy. Estas diferencias en la infraestructura computacional pueden generar disparidades en la trayectoria de entrenamiento y en el valor final de la función de pérdida.

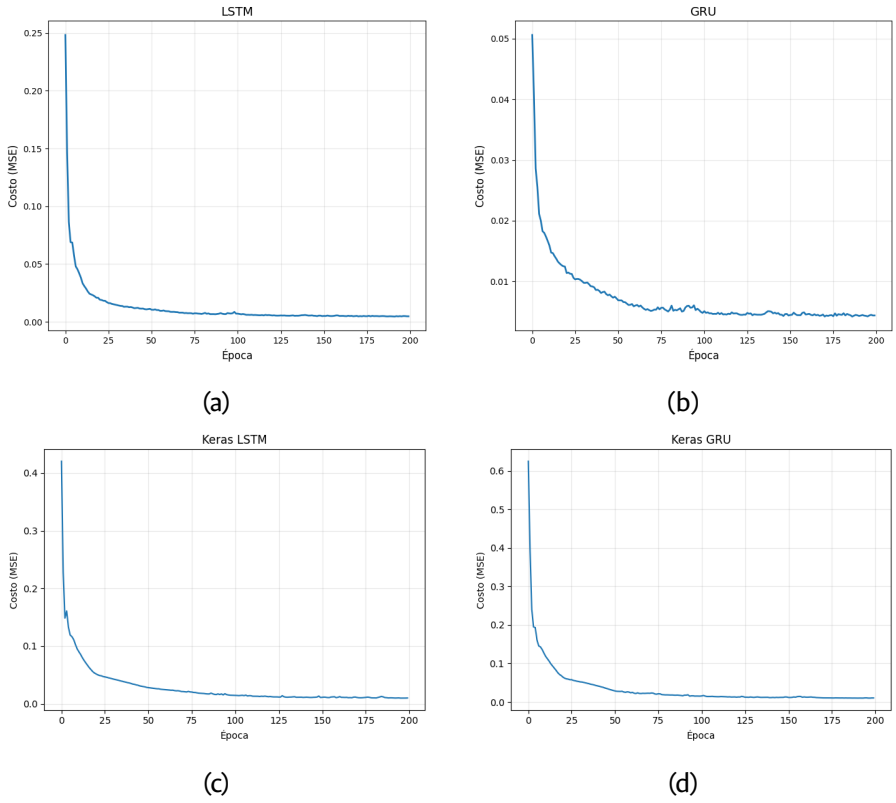


Figura 6-1: Historial de la función de pérdida a lo largo de las 200 épocas. (a) Modelo LSTM, (b) Modelo GRU, (c) Modelo LSTM en Keras, (d) Modelo GRU en Keras.

6.2. Predicción de los modelos

Tras el entrenamiento, se realiza la predicción sobre los datos de prueba y para los dos años posteriores. Los resultados generales pueden observarse en la Figura **6-2**. En la Tabla **6-2** se presentan los valores estimados por cada modelo en el conjunto de prueba y en la Tabla **6-3** se muestra el error asociado a dichas predicciones.

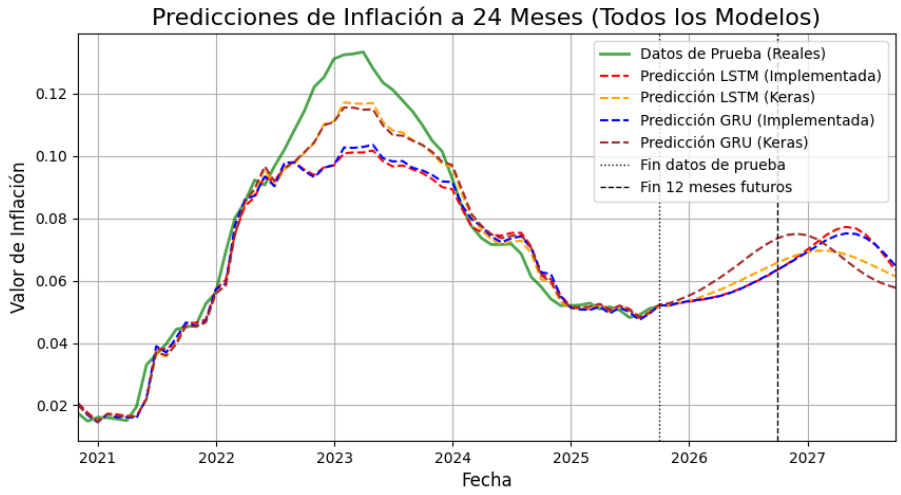


Figura 6-2: Predicciones realizadas sobre el conjunto de prueba y sobre 24 meses posteriores de todos los modelos.

Predicciones por Fecha					
Primeros dos años					
Modelo	10/20	04/21	10/21	04/22	10/22
Dato Real	0.0175	0.0195	0.0458	0.0923	0.1222
LSTM	0.0207	0.0164	0.0465	0.0867	0.0939
GRU	0.0205	0.0159	0.0454	0.0873	0.0932
LSTM (keras)	0.0200	0.0164	0.0453	0.0890	0.1042
GRU (keras)	0.0202	0.0164	0.0452	0.0896	0.1044
Últimos dos años					
Modelo	04/23	10/23	04/24	10/24	04/25
Dato Real	0.1282	0.1048	0.0716	0.0541	0.0516
LSTM	0.1018	0.0926	0.0747	0.0604	0.0508
GRU	0.1036	0.0939	0.0741	0.0621	0.0494
LSTM (keras)	0.1171	0.1008	0.0732	0.0593	0.0499
GRU (keras)	0.1150	0.1013	0.0749	0.0595	0.0503

Tabla 6-2: Predicciones realizadas cada seis meses en los datos de prueba por los cuatro modelos.

En primer lugar, se observa que los modelos logran ajustar de manera adecuada el comportamiento general de los datos de prueba. Sin embargo, en el intervalo donde la inflación alcanza su máximo

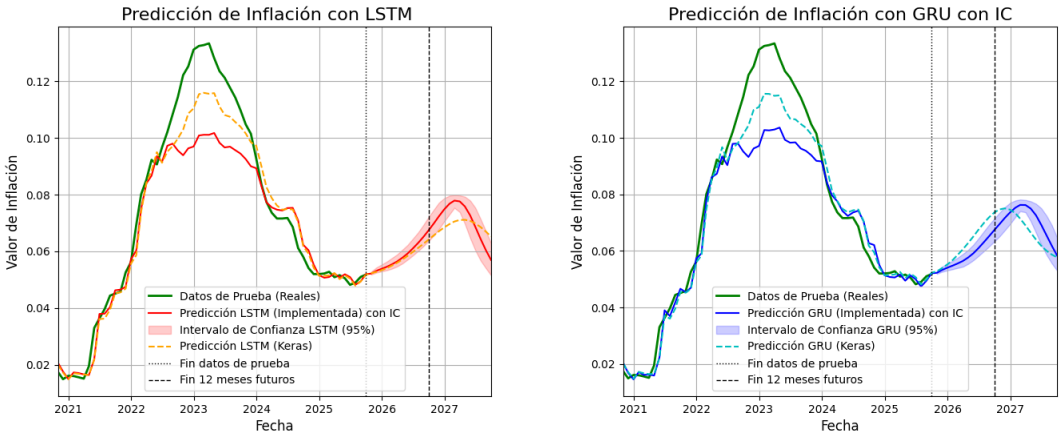
histórico —aproximadamente entre agosto de 2022 y noviembre de 2023— ninguno de los modelos consigue reproducir este pico con precisión. Los modelos implementados en Keras se aproximan ligeramente mejor en esta región, lo cual explica que sus errores sean más bajos. Este comportamiento es esperable, dado que dicho intervalo corresponde a un comportamiento atípico de la inflación que no aparece en el resto del conjunto de entrenamiento.

Modelo	Error (MSE)
LSTM	0.00017
GRU	0.0016
LSTM (keras)	0.00005
GRU (keras)	0.00006

Tabla 6-3: Errores (MSE) por modelo

Por otra parte, la predicción realizada a 24 meses más allá de los datos disponibles arroja resultados adicionales. Los cuatro modelos coinciden en proyectar un incremento de la inflación hasta valores cercanos al $8\,\%$. No obstante, la forma de la trayectoria varía de manera notable: los modelos implementados desde cero producen curvas prácticamente idénticas y de crecimiento moderado, mientras que el modelo GRU de Keras anticipa un incremento más acelerado y alcanza el pico con mayor rapidez. En contraste, el modelo LSTM de Keras presenta una pendiente más suave y proyecta el menor crecimiento entre los modelos analizados.

Finalizando el laboratorio, también se obtuvieron intervalos de confianza para las predicciones de los modelos implementados desde cero. En la Figura **6-3** se pueden observar las predicciones de ambas arquitecturas dentro de sus respectivos intervalos de confianza, así como la predicción generada por Keras.



(a) (b)

Figura 6-3: Predicciones con intervalos de confianza entre modelos de (a) LSTM y (b) GRU.

En el caso de las dos arquitecturas LSTM, durante el primer año la predicción de Keras permanece dentro del intervalo de confianza del modelo implementado manualmente, lo cual evidencia la similitud en el comportamiento de ambos enfoques. Sin embargo, para las arquitecturas GRU la situación es distinta: la predicción de Keras difiere notablemente de la obtenida con el modelo implementado desde cero y, en casi ningún momento, se encuentra contenida en su intervalo de confianza. Este contraste resulta especialmente interesante, pues mientras la versión LSTM de Keras coincide razonablemente con el intervalo del modelo manual, su contraparte GRU no exhibe la misma consistencia.

6.3. Consideraciones sobre la evaluación del laboratorio

Si bien el laboratorio ofrece una estructura teórica computacional coherente, es importante señalar que, hasta la fecha de escrito este documento, no se cuenta con evidencia de su aplicación en contextos docentes reales ni con retroalimentación proveniente de estudiantes o profesores. Es por esto, que la validación pedagógica del recurso aún no ha sido evaluado, dado que dicha etapa requiere un proceso de implementación institucional que excede los alcances de este trabajo.

No obstante, es de esperar que en un futuro este laboratorio pueda ser utilizado en el curso *Redes Neuronales, Arquitecturas y Aplicaciones*, ofrecido por la Universidad Nacional de Colombia para la Maestría en Actuaría y Finanzas.

Capítulo 7

Conclusiones

El desarrollo de las implementaciones construidas manualmente permitió evidenciar que los modelos obtenidos son capaces de reproducir con notable fidelidad el comportamiento de las implementaciones provistas por Keras.

El laboratorio construido integra explicaciones matemáticas detalladas, implementaciones paso a paso y ejercicios autocalificables, lo que facilita que el estudiante no solo ejecute código, sino que entienda de manera profunda el razonamiento detrás del funcionamiento de las redes recurrentes. Este enfoque permite que el usuario explore desde la formulación de las ecuaciones fundamentales, hasta la interpretación del comportamiento de los modelos en un caso real, fortaleciendo la comprensión conceptual y la capacidad de análisis.

La aplicación práctica y los resultados obtenidos sobre la predicción de la inflación en Colombia ilustran la capacidad y viabilidad de emplear LSTM y GRU como herramientas exploratorias en contextos financieros y actuariales. Este trabajo en particular, presenta algunos resultados que podrían ser de interés en sectores como el de la economía.

Con el objetivo de garantizar la reproducibilidad de estos resultados y poner a disposición de la comunidad académica esta herramienta, el código fuente y el cuaderno interactivo se encuentran disponible en el repositorio del proyecto: github.com/MateoOrtiz001/Laboratory-LSTM-GRU-Inflation.

Capítulo 8

Recomendaciones y planes a futuro

El desarrollo del laboratorio permitió consolidar una base conceptual y computacional para el estudio de las redes LSTM y GRU dentro del contexto aplicado. Sin embargo, existen diversas líneas de ampliación que podrían fortalecer tanto la rigurosidad del laboratorio como su alcance pedagógico y práctico.

Primeramente, aunque las implementaciones manuales y las basadas en Keras presentan comportamientos comparables, sería valioso profundizar en el estudio de la estabilidad numérica y la eficiencia computacional de cada aproximación. Extender la comparación hacia diferentes inicializadores, tamaños de red o estrategias de regularización permitiría caracterizar de forma más precisa sus ventajas y limitaciones en escenarios reales. De igual manera, el laboratorio podría ampliarse incorporando otros modelos recurrentes o secuenciales que actualmente son ampliamente utilizados en la industria, tales como las arquitecturas bidireccionales, las variantes profundas, las variantes peephole para LSTM o incluso modelos más recientes como las redes basadas en atención o Transformers adaptados para series temporales. Esto no solo enriquecería la perspectiva del estudiante, sino que permitiría establecer contrastes metodológicos entre distintas familias de modelos.

Otra posible línea de trabajo futuro consiste en diversificar los estudios de caso. Aunque la predicción de la inflación en Colombia representa un ejemplo concreto y representativo, el laboratorio podría incluir otros problemas típicos de la actuaría y las finanzas, como la modelación de siniestralidad, la estimación de reservas, el análisis de tasas de interés o la predicción de precios de activos financieros.

Finalmente, se sugiere explorar la posibilidad de estructurar el laboratorio como un recurso modular, permitiendo su integración dentro de cursos de aprendizaje automático, redes neuronales, ciencias actuariales o finanzas cuantitativas. Una organización más granular facilitaría su adaptación a distintos niveles de profundidad, desde una perspectiva introductoria hasta un enfoque más avanzado orientado al análisis crítico y la experimentación computacional.

Anexo A

Apéndice 1. Funciones de Activación

En este corto capítulo definiremos brevemente tres de las funciones de activación -transformaciones no lineales- más utilizadas en la aplicación de *Redes Neuronales*.

A.1. ReLU

La *Unidad Lineal Rectificada* (ReLU de sus siglas en inglés) es una función de activación introducida por *Nair & Hinton* [2010], definida como

$$ReLU(x) = \begin{cases} x, & \text{si } x > 0 \\ 0, & \text{en otro caso.} \end{cases} \tag{A-1}$$

Geométricamente, dado $x \in \mathbb{R}^n$, la función $ReLU$ parte el hiperplano, convirtiendo la parte negativa en constante cero y manteniendo la parte positiva con pendiente.

Esta función es bastante popular por su fácil interpretación y buen rendimiento en la práctica, sobre todo en *Redes Neuronales Superficiales* o en capas intermedias de una *Red Neuronal Profunda*, en donde, dada una combinación lineal de k términos -en términos de la capa sería k neuronas- esta función de activación permite generar aproximaciones a funciones más complejas a través de pequeños trozos de funciones lineales. De esta forma se puede observar más claramente la idea del Teorema de Aproximación Universal (*Hornik* [1991]).

Claramente esta función no es diferenciable en $x = 0$, sin embargo, en la práctica simplemente definimos su derivada como la derivada de sus partes tomando a 0 en cualquier caso. Tomemos un hiperplano $x = W\mathsf{x} + b \in \mathbb{R}^n$, entonces su derivada esta dada por

$$\frac{\partial}{\partial x} ReLU = \begin{cases} b, & \text{si } x > 0 \\ 0, & \text{en otro caso.} \end{cases}$$

A.2. Tangente Hiperbólica

La función tangente hiperbólica, denotada como tanh, es una de las funciones de activación más utilizadas en modelos recurrentes, especialmente en LSTM y GRU, fue introducida en el ámbito del aprendizaje automático por *Rumelhart et al.* [1986]. Está definida como

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Geométricamente, esta función es suave, continua y acotada en el intervalo $(-1, 1)$, lo que permite centrar sus salidas alrededor de cero. Esta característica facilita la propagación de gradientes y la estabilidad numérica en redes profundas o recurrentes.

En el contexto de Redes Neuronales, tanh es frecuentemente utilizada para generar estados intermedios o representaciones internas que oscilan entre valores negativos y positivos. Esto contrasta con funciones como $ReLU$, cuyo rango es únicamente no negativo, y permite a la red tener representaciones más equilibradas y simétricas.

La función tanh es completamente diferenciable en todo $\mathbb{R}$. Su derivada se obtiene fácilmente a partir de su definición funcional y está dada por

$$\frac{\partial}{\partial x}\tanh(x) = 1 - \tanh^2(x). \tag{A-2}$$

Esta expresión es computacionalmente eficiente y aparece con frecuencia en la retropropagación de errores dentro de arquitecturas recurrentes.

A.3. Sigmoide

La función sigmoide logística, o simplemente *sigmoide*, es una transformación no lineal ampliamente utilizada en Redes Neuronales, en especial en la construcción de compuertas dentro de arquitecturas recurrentes como las LSTM o GRU, introducido por *Rumelhart et al.* [1986] de igual forma como la tangente hiperbólica fue introducida en el ámbito del aprendizaje automático. Está definida por

$$\sigma(x) = \frac{1}{1 + e^{-x}}.$$

Geométricamente, esta función es suave, continua y acotada en el intervalo $(0, 1)$, lo cual permite interpretarla como una probabilidad o como un coeficiente de "activación parcial", propiedad que resulta fundamental en el funcionamiento de las compuertas de olvido, actualización y salida de una celda LSTM.

Su naturaleza acotada permite controlar la magnitud de las señales internas dentro de la red, evitando que crezcan sin control. Sin embargo, su región de saturación para valores muy grandes (positivos o negativos) puede provocar el conocido *desvanecimiento de gradientes*.

La función sigmoide es diferenciable en todo $\mathbb{R}$ y su derivada viene dada por

$$\frac{\partial}{\partial x}\sigma(x) = \sigma(x)(1 - \sigma(x)), \tag{A-3}$$

expresión que aparece de forma recurrente en el cálculo de gradientes de compuertas dentro del algoritmo de retropropagación a través del tiempo.

Referencias Bibliográficas

Almosova, A. & **Andresen, N.**: , 2018; Forecasting inflation with recurrent neural networks; Conference paper, T2M 2019 – Time Series and Forecasting Methods; URL https://t2m2019.sciencesconf.org/233642/Intro.pdf.

Baillargeon, J.-T.; **Lamontagne, L.** & **Marceau, E.**: , 2021; Mining actuarial risk predictors in accident descriptions using recurrent neural networks; *Risks*; **9** (1); doi:10.3390/risks9010007; URL https://www.mdpi.com/2227-9091/9/1/7.

Cai, P.; **Abdallah, A.** & **Jeganathan, P.**: , 2025; Recurrent neural networks for multivariate loss reserving and risk capital analysis; *North American Actuarial Journal*; **0** (0): 1–25; doi:10.1080/10920277.2025.2517149; URL https://doi.org/10.1080/10920277.2025.2517149.

campdav: , 2017; Tutorial: Multi-layer recurrent neural networks (lstm, rnn) for text models in python using tensorflow.; URL https://github.com/campdav/text-rnn-tensorflow.

Cárdenas-Cárdenas, J. A.; **Cristiano-Botia, D. J.**; **Martínez-Cortés, N.** *et al.*: , 2023; Colombian inflation forecast using long short-term memory approach; *Borradores de Economía; No. 1241*.

Chavhan, S.; **Raj, P.**; **Raj, P.**; **Dutta, A. K.** & **Rodrigues, J. J.**: , 2024; Deep learning approaches for stock price prediction: A comparative study of lstm, rnn, and gru models; en *2024 9th International Conference on Smart and Sustainable Technologies (SpliTech)*; IEEE; págs. 01–06.

Cho, K.; **van Merrienboer, B.**; **Bahdanau, D.** & **Bengio, Y.**: , 2014; On the properties of neural machine translation: Encoder-decoder approaches; URL https://arxiv.org/abs/1409.1259.

DeepLearning.AI: , 2018; Sequence models; Coursera Course; URL https://www.coursera.org/learn/nlp-sequence-models?specialization=deep-learning.

Fischer, T. & **Krauss, C.**: , 2018; Deep learning with long short-term memory networks for financial market predictions; *European Journal of Operational Research*; **270** (2): 654–669; doi:https://doi.org/10.1016/j.ejor.2017.11.054; URL https://www.sciencedirect.com/science/article/pii/S0377221717310652.

Ghojogh, B. & **Ghodsi, A.**: , 2023; Recurrent neural networks and long short-term memory networks: Tutorial and survey; URL https://arxiv.org/abs/2304.11461.

Glorot, X. & **Bengio, Y.**: , 2010; Understanding the difficulty of training deep feedforward neural networks; en *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, tomo 9 de *Proceedings of Machine Learning Research* (Editado por **Teh, Y. W.** & **Titterington, M.**); PMLR, Chia Laguna Resort, Sardinia, Italy; págs. 249–256; URL https://proceedings.mlr.press/v9/glorot10a.html.

Hochreiter, S. & **Schmidhuber, J.**: , 1997; Long short-term memory; *Neural Computation*; **9** (8): 1735–1780; doi:10.1162/neco.1997.9.8.1735; URL https://doi.org/10.1162/neco.1997.9.8.1735.

Hornik, K.: , 1991; Approximation capabilities of multilayer feedforward networks; *Neural Networks*; **4** (2): 251–257; doi:https://doi.org/10.1016/0893-6080(91)90009-T; URL https://www.sciencedirect.com/science/article/pii/089360809190009T.

Imperial College London: , 2020; Customising your models with tensorflow 2; Coursera Course; URL https://www.coursera.org/learn/customising-models-tensorflow2?specialization=tensorflow2-deeplearning.

Kingma, D. P. & **Ba, J.**: , 2017; Adam: A method for stochastic optimization; URL https://arxiv.org/abs/1412.6980.

Lindholm, M. & Palmborg, L.: , 2022; Efficient use of data for lstm mortality forecasting; en *European Actuarial Journal*, tomo 12; págs. 749–778; doi:https://doi.org/10.1007/s13385-022-00307-3.

Meier, D. & **Schelldorfer, J.**: , 2019; Lee and carter go machine learning: Recurrent neural networks; URL https://htmlpreview.github.io/?https://github.com/JSchelldorfer/ActuarialDataScience/blob/master/6%20-%20Lee%20and%20Carter%20go%20Machine%20Learning%20Recurrent%20Neural%20Networks/CHEmort_rnn.html.

Nair, V. & **Hinton, G. E.**: , 2010; Rectified linear units improve restricted boltzmann machines; en *Proceedings of the 27th International Conference on Machine Learning (ICML)*; págs. 807–814.

Ng, A. & **deeplearning.ai**: , 2018; Sequence models; https://www.coursera.org/learn/nlp-sequence-models; coursera: Accedido el 10 de Septiembre de 2025.

Packt: , 2025; Deep learning: Recurrent neural networks with python specialization; URL https://www.coursera.org/specializations/packt-deep-learning-recurrent-neural-networks-with-python.

Pamukcu, T. E.: , 2023; Rnn and lstm tutorial for beginners; Kaggle; URL https://www.kaggle.com/code/moonglow22/rnn-and-lstm-tutorial-for-beginners/notebook.

Paranhos, L.: , 2021; Predicting inflation with neural networks. working paper.; coventry: University of Warwick. Department of Economics. Warwick economics research papers series (WERPS) (1344). (Unpublished).

Perla, F.; **Richman, R.**; **Scognamiglio, S.** & **Wüthrich, M. V.**: , 2021; Time-series forecasting of mortality rates using deep learning; *Scandinavian Actuarial Journal*; **2021** (7): 572–598; doi:10.1080/03461238.2020.1867232; URL https://doi.org/10.1080/03461238.2020.1867232.

Prince, S. J.: , 2023; *Understanding Deep Learning*; The MIT Press; URL http://udlbook.com.

Richman, R. & **Wuthrich, M. V.**: , 2019; Lee and carter go machine learning: Recurrent neural networks; *SSRN*.

Rumelhart, D. E.; **Hinton, G. E.** & **Williams, R. J.**: , 1986; Learning representations by back-propagating errors; *Nature*; **323**: 533–536.

Sako, K.; **Mpinda, B. N.** & **Rodrigues, P. C.**: , 2022; Neural networks for financial time series forecasting; *Entropy*; **24** (5); doi:10.3390/e24050657; URL https://www.mdpi.com/1099-4300/24/5/657.

Shen, G.; **Tan, Q.**; **Zhang, H.**; **Zeng, P.** & **Xu, J.**: , 2018; Deep learning with gated recurrent unit networks for financial sequence predictions; *Procedia Computer Science*; **131**: 895–903; doi:https://doi.org/10.1016/j.procs.2018.04.298; URL `https://www.sciencedirect.com/science/article/pii/S1877050918306781`; recent Advancement in Information and Communication Technology:.

Yan, H. & **Ouyang, H.**: , 2018; Financial time series prediction based on deep learning; *Wireless Pers Commun*; **102**; doi:https://doi.org/10.1007/s11277-017-5086-2.

Zhou, L.: , 2024; Intelligent insurance actuarial model under machine learning and data mining; en *2024 IEEE 2nd International Conference on Image Processing and Computer Applications (ICIPCA)*; págs. 1726–1731; doi:10.1109/ICIPCA61593.2024.10708777.

Ömer Berat Sezer: , 2018; Lstm rnn tutorials with demo; URL `https://github.com/omerbsezer/LSTM_RNN_Tutorials_with_Demo`.