Predicción del precio del bitcoin mediante redes neuronales recurrentes

|  |
| --- |
| *Daniel Ramos Hoogwout*  **Tutor**: *Victor Troster* |
| Treball de fi de Màster Universitari en Analisi de Dades Massives en Economia i Empresa (MADM)  Universitat de les Illes Balears 07122 Palma de Mallorca  dramoshoogwout@gmail.com |

Resumen

En los últimos años se ha incrementado exponencialmente la utilización de criptomonedas, una moneda virtual que pretende sustituir al dinero fiduciario emitido por los bancos centrales. Este tipo de activos ha crecido tanto en número como en capitalización al nivel de poder compararse incluso con índices bursátiles, por ello el objetivo de este estudio es intentar predecir el precio del Bitcoin, la criptomoneda por excelencia, mediante un tipo de redes neuronales recurrentes, en concreto, *Long Short-Term* Memory (LSTM), uno de los algoritmos de *deep learning* más avanzados que existen que gracias a su celda de memoria permiten almacenar información del pasado para hacer predicciones más precisas.

Abstract

In recent years, the use of cryptocurrencies has increased exponentially, a virtual currency that aims to replace fiduciary money issued by the central banks. This type of asset has grown both in number and in capitalization to the level of being able to be compared even with stock market indexes, for this reason the objective of this study is to try to predict the price of Bitcoin, the cryptocurrency by excellence, through a type of recurrent neural network, specifically, Long Short-Term Memory (LSTM), one of the most advanced deep learning algorithms to date, that thanks to its memory cell, allows to store past information to make more precise predictions.

Palabras clave: Predicción, Series Temporales, Redes Neuronales Recurrentes, LSTM, Bitcoin, Cripto.

1. Introducción

Las redes neuronales recurrentes representan uno de los algoritmos más avanzados que existen en el mundo del *deep learning* supervisado. Como ya su propio nombre indica una red neuronal intenta imitar un cerebro humano para resolver problemas, en este caso, la red neuronal recurrente intenta imitar el lóbulo frontal, la parte del cerebro humano que se encarga de la memoria a corto plazo. Haciendo uso de esta memoria el algoritmo es capaz de reaccionar a situaciones futuras con información del pasado y por este motivo hace que este tipo de algoritmos sean tan potentes para predecir series temporales.

Las redes neuronales recurrentes, en concreto, la variante de Long Short-Term Memory (LSTM) son redes neuronales muy adecuadas para datos de series temporales ya que gracias a su estructura incluye una celda de memoria que permite mantener información por periodos prolongados en el tiempo venciendo así al problema del gradiente descendiente, un problema inherente de las redes neuronales recurrentes clásicas.

Este tipo de redes se han aplicado con resultados notables en aplicaciones como reconocimiento de voz, lenguaje natural, síntesis de voz… Por ello se va a utilizar este tipo de redes neuronales en este trabajo para intentar predecir el precio del bitcoin haciendo uso de datos de series temporales con su precio pasado entre otros indicadores para poder hacer una predicción lo más precisa posible.

* 1. Redes Neuronales Recurrentes

Una red neuronal recurrente es un tipo de red neuronal que dentro de su estructura puede retroalimentarse, es decir, utiliza su propio *feedback* para influenciar la entrada actual y la salida.



Figura 1: Diagrama de la red neuronal recurrente

Por tanto, la salida de una red neuronal recurrente depende de los elementos previos de la propia secuencia. Esto genera una característica distintiva, la red neuronal recurrente comparte el mismo parámetro de peso dentro de cada capa de la red.[1]

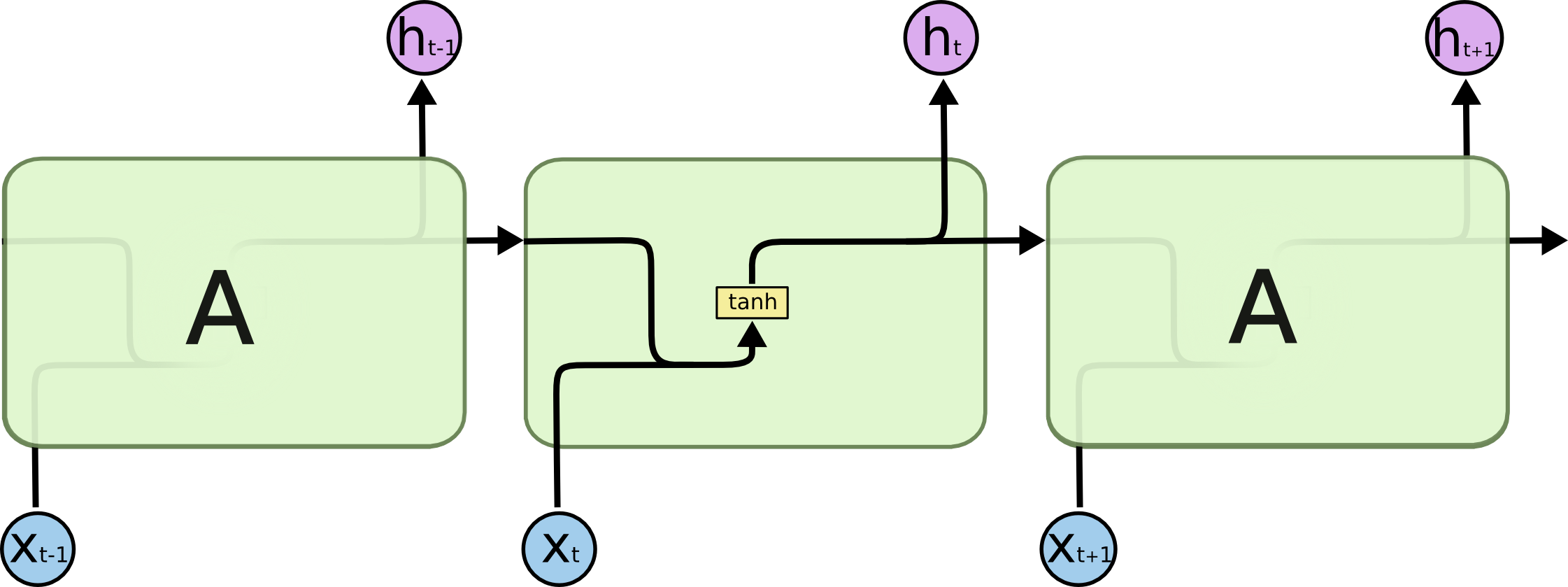


Figura 2: Estructura de un módulo RNR clásico

Fuente: Colah’s Blog [1]

Es decir, cada vez que la red actualiza sus pesos debe propagarse hacia atrás para actualizar todos y cada uno de los pesos de las capas que constituyen la red, generando así un posible desvanecimiento del gradiente.

* 1. Desvanecimiento del gradiente

Este problema que presentan las redes neuronales recurrentes fue descubierto por Josep Hochreiter.[2]

El problema radica que en la red neuronal utiliza un algoritmo de gradiente descendiente para encontrar el mínimo global de la función de costes que es la configuración óptima de la red.

En el caso de las redes neuronales recurrentes la información pasada se utiliza como input para determinar la predicción presente por lo tanto cuando se actualizan los pesos para minimizar el error se tendrá que actualizar todas las neuronas que han participado en el proceso.



Figura 2: Diagrama de RNR *Many to One*

Al principio del proceso se asignan pesos aleatorios con valores cercanos a cero y entonces se comienza a entrenar la red, al multiplicar todos los momentos pasados por un valor cercano a cero el gradiente se reduce tras cada multiplicación por lo que al final se hace cada vez más difícil actualizar los pesos de la red y aumenta el tiempo en el que se obtiene el resultado final.

* 1. LSTM

Es una variación de la red neuronal recurrente clásica que soluciona el problema del desvanecimiento del gradiente comentado en la sección anterior. Fue una de las soluciones propuestas por Josep Hochreiter y Jürgen Schmidhuber[1], esta evolución añade celdas de entrada, salida y olvido.



Figura 3: Estructura de un módulo LSTM

Fuente: Adaptado de [3]

Gracias a esta nueva arquitectura se separa las celdas de memoria (C) y los valores de salida de la red (h) evitando así el desvanecimiento del gradiente. Cabe recordar que en la figura anterior C, h y X son vectores, es decir, son capas completas de neuronas.

1. Metodología

En esta sección se expone la metodología utilizada para obtener el modelo final expuesto en la siguiente sección.

* 1. Construcción de la red neuronal

Para este análisis se va a utilizar una estructura compuesta por 4 capas de entrada interconectadas conteniendo 50 neuronas en cada capa dándole así una elevada dimensionalidad para que el modelo pueda capturar correlaciones en el precio del bitcoin a un nivel mucho más complejo que si el modelo fuese más sencillo ya que no podría capturar correlaciones más avanzadas.

Tras cada capa de entrada se le añade una capa de *dropout*, una capa de olvido que se utiliza para desactivar aleatoriamente neuronas de la capa anterior y solo un cierto porcentaje de ellas pase la información hacia delante para prevenir el sobreajuste.

* 1. Robustez/Complejidad de entranamiento

Falta completar esta sección

* 1. Modelo inicial

Se comienza con un modelo sencillo para establecer una base sobre la cual introducir cambios y mejorar el poder predictivo del modelo. El modelo inicial solo toma como input los precios de apertura pasados. Cabe destacar que los pesos iniciales en las redes neuronales se asignan de manera completamente aleatorio, por tanto, es muy importante fijar esta aleatoriedad con una semilla para poder replicar los resultados y compararlos con otros modelos.

* 1. Hiperparametrización

En el proceso de entrenar una red neuronal hay varios parámetros que pueden alterarse para obtener mejores resultados.

En este caso los más importantes son el *batch size*, los *epochs* y el periodo el cual la red neuronal puede observar valores pasados.

El *batch size* es el número de muestras para trabajar antes de actualizar los parámetros internos del modelo. Es decir, el dataset se va a dividir en muestras de 32 unidades tras las cuales los parámetros del modelo se actualizarán [1].

Un buen tamaño por defecto es 32 ya que usar tamaños pequeños se logra mejorar la estabilidad y el rendimiento de generalización. [Masters D, Luschi C] aparte de que el proceso de converge rápidamente para el coste computacional que supone.

Los *epochs* definen el número de veces que el algoritmo de aprendizaje va a digerir el dataset de entreno. Este valor va a ser seleccionado automáticamente por el propio algoritmo de aprendizaje ya que se ha especificado un máximo de 100 *epochs* con una paciencia de un 10%, es decir, si tras 10 *epochs* no hay una mejora en el modelo, el algoritmo para el entrenamiento. Esto es importante ya que al aprender relativamente rápido el modelo puede sobre ajustarse con facilidad al aumentar el número de *epochs*.

Finalmente, el periodo en el cual tiene visión la red neuronal se ha establecido en 30, 60, 90 y 120 días. Esto significa que, para hacer la predicción de un día, la red neuronal tendrá acceso a la información de los x días anteriores definidos por este periodo.

* 1. Modelo final

El modelo final se escoge comparando los resultados de los distintos modelos creados a partir de la combinación de los hiperparámetros anteriormente descritos. Para evaluar los modelos se han usado dos medidas, en concreto, la raíz del error cuadrático medio (RMSE) y el error absoluto medio (MAE).

La raíz error cuadrático medio mide la magnitud medio del error penalizando sobre todo los errores más elevados. Esto es deseable ya que separarse mucho del precio real implica la posibilidad de tener perdidas elevadas. En cambio, el error medio absoluto es el promedio de las diferencias absolutas entre la predicción y la observación real donde todas las diferencias tienen el mismo peso.

* 1. Multistep forecasting

Generalmente las predicciones se suelen hacer sobre un periodo de tiempo, es decir, se predice un periodo de tiempo de una sola vez.

En cambio, en este estudio se toma una aproximación diferente, se hace una predicción de múltiples pasos, es decir, se predice el precio del bitcoin para cada día utilizando los precios reales de los últimos x días definidos por el hiperparámetro periodo. En otras palabras, cada día se ha predicho sobre los valores reales anteriores, en vez de predecir sobre predicciones anteriores.



Figura 4: Predicción en múltiples pasos

Tal y como se puede observar en la figura anterior se hace una predicción en múltiples pasos donde el modelo puede observar los valores pasados de los últimos 90 días y en base a estos hace una predicción y a continuación repite el proceso siempre utilizando el precio real.

1. Análisis Empírico

En esta sección se aplican las redes neuronales recurrentes explicadas en la sección anterior para predecir el precio de apertura del Bitcoin durante un período de tres meses.

El modelo ha ido evolucionando desde un modelo inicial donde solamente se tenía en cuenta el precio de apertura del bitcoin para utilizarlo como modelo de partida e ir ajustando y añadiendo información hasta llegar al modelo que presenta más adelante.

* 1. Datos

Para la muestra de entrenamiento se ha escogido el período que abarca desde 17 de septiembre de 2015 hasta el 30 de noviembre de 2019. Se ha optado por este período ya que comienza desde el dato más antiguo disponible en Yahoo Finance hasta tres meses antes de que el efecto de la pandemia fuese global, considerando marzo como el inicio de la pandemia de COVID-19. Por tanto, la muestra de testo comienza desde el 1 de diciembre de 2019 hasta el 29 de febrero de 2020.

Finalmente se ha escogido como muestra de validación el periodo que abarca desde el 1 de marzo de 2021 hasta el 27 de noviembre de 2021, se utiliza para dar una estimación imparcial de la habilidad predictora del modelo.

Yahoo Finance ofrece varios datos referentes a la cotización de activos de los cuales se han escogido el precio de apertura y de cierre diarios para entrenar la red neuronal. En un principio también se había incluido un índice bursátil, en concreto, el IBEX 35 ya que en los últimos meses se ha comentado mucho la correlación creciente entre el bitcoin y los índices bursátiles, desafortunadamente la inclusión del índice empeoró mucho las predicciones y se ha optado por eliminarlo de este análisis.

* 1. Matriz de entrada

La entrada de información a la red neuronal está compuesta por una matriz tridimensional que contiene el precio de apertura y de cierre del bitcoin con 90 *timesteps*¸ es decir, en cada momento del tiempo la red neuronal será capaz de observar 90 días previos al día actual. Previamente se había hecho uso de 60 días como período base, pero el incremento a 3 meses supuso una mejora considerable en el modelo.



Figura 4: Matriz de Entrada

Antes de introducir estos datos en la red neuronal estos se han normalizado a valores entre 0 y 1 para simplificar la entra a la red neuronal, evitando así que una variable domine sobre otra en el caso de que sus magnitudes fuesen diferentes.

* 1. Estructura

La red neuronal que procesa esta matriz está compuesta por una entrada, cuatro capas con cincuenta neuronas cada una intercaladas por una capa de *dropout* que permite que se desactiven un 20% de las neuronas, por tanto, solo el 80% de la información pasará a la siguiente capa previniendo así el sobreajuste. Finalmente hay una capa de salida que proporciona la predicción del precio de apertura del bitcoin.



Figura 5: Estructura de la red neuronal

Se ha optado por alimentar la red neuronal en *batches* de 32 unidades ya que es un valor relativamente pequeño que ayuda a converger de manera rápida a costa de introducir ruido en el proceso de entrenamiento.

En un principio el número de *epochs* o en otras palabras el número de veces que el algoritmo de aprendizaje ha digerido todo el *dataset* de entrenamiento se había situado en 100. Utilizando el parámetro de *Early Stopping* del paquete de *keras* se ha utilizado una paciencia de 10 *epochs* para monitorizar si el modelo dejaba de minimizar el gradiente desciendiente en las siguientes 10 iteraciones ya que al usar batches de un tamaño de 32 unidades el modelo aprende con mucha rapidez y puede ocasionar overfitting.

En este modelo, el *Early Stopping* se activa a los 46 *epochs*, es decir, desde el *epoch* 36 no hay una disminución de la pérdida del gradiente descendiente del modelo y por tanto selecciona los pesos del *epoch* 36.

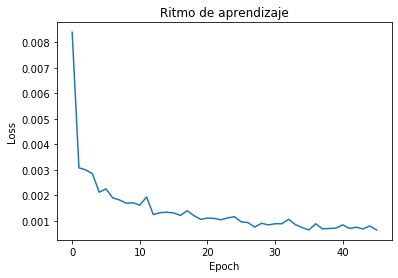


Figura 6: Ritmo de aprendizaje de la RNR

Una vez realizado el entrenamiento del modelo se predicen los siguientes 3 meses con la muestra de test y se obtiene el siguiente gráfico.

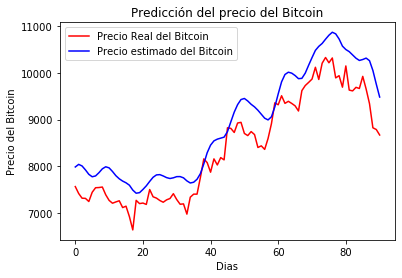
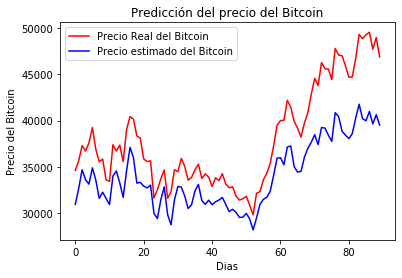


Figura 7: Predicción sobre la muestra de test

Para poder comparar esta predicción con otros modelos se estima la raíz del error cuadrático medio, así como también el error medio absoluto.

Finalmente se utiliza este modelo para predecir el precio con una muestra de validación utilizando un período futuro, en concreto, comienza a predecir junio de 2021 pudiendo así coger los 90 días anteriores para hacer una predicción.

Figura 8: Predicción sobre la muestra de validación



En este caso también se obtiene la raíz del error cuadrático medio y el error medio absoluto para así poder comparar los modelos con los resultados de la muestra de test y determinar si ha habido sobreajuste.

* 1. Resultados

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Periodo (días) | *30* | *60* | *90* | *120* |
| RMSE | 497 | 163 | 552 | 201 |
| MAE | 469 | 132 | 506 | 159 |

Tabla 1: Resultados en la muestra de test

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Periodo (días) | *30* | *60* | *90* | *120* |
| RMSE | 15803 | 10318 | 4487 | 10192 |
| MAE | 14650 | 7997 | 4005 | 9177 |

Tabla 2: Resultados en la muestra de validación

Como se puede ver en las tablas mostradas anteriormente el modelo que mejor ajuste tiene es el del periodo de 90 días, el cual tiene la puntuación más alta en el periodo de test, pero en cambio se ajusta mejor en la muestra de validación de hecho tiene una puntuación en ambos *scores* de menos de la mitad del segundo mejor modelo.

1. Conclusiones

Falta rellenar esta sección.

-El problema del gradiente para series temporales largas se puede resolver añadiendo más capas a la red.

1. Bibliografía

# Configuración de Anaconda

Para poder replicar los resultados descritos con anterioridad se tiene que configurar el entorno de desarrollo de una manera específica.

El autor de este estudio ha utilizado Anaconda para desarrollar la red neuronal, utilizando el paquete de Tensor Flow. Este paquete provoca errores en la configuración por defecto de Anaconda y es necesario crear un nuevo entorno de desarrollo en el cual se instale la versión de Numpy 1.18.5.

Anaconda instala por defecto la más reciente entre otras librerías que generan conflictos y la solución más sencilla es crear un entorno de desarrollo exclusivamente para Tensor Flow 2.3 en el cual únicamente deben instalarse la versión anteriormente comentada de Numpy, Pandas, Keras, Tensorflow y las librerías que se instalan automáticamente al seleccionar las anteriores.

# Modelos descartados

Modelo inicial

Modelo con un índice bursátil

Modelo con un periodo de 30 días

Modelo con un periodo de 60 días

Modelo con un periodo de 120 días

Referencias

[1] IBM Cloud Education, “https://www.ibm.com/cloud/learn/recurrent-neural-networks,” Recurrent Neural Networks, Sep. 14, 2020.

[2] Sepp Hochreiter, “The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions,” International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, vol. 06, no. 02, pp. 107–116, 1998.

[3] Christian Olah, “https://colah.github.io/posts/2015-08-Understanding-LSTMs/,” Understanding LSTM Networks, Aug. 27, 2015.