EDEM
Escuela de Empresarios

Time Series Forecasting

RNNs y LSTMs

National University of Singapore (NUS)
Félix Fuentes

Curso 2020/2021 – Edición II

Fecha 27/03/2021

Índice

Redes neuronales recurrentes

Problemas temporales

¿Cómo abordamos problemas con componente temporal?

- ¿Cómo podemos predecir las compras de un determinado producto en un periodo concreto de tiempo?
- ¿Y las personas que viajan con una determinada aerolinea?
- ¿Podríamos predecir el valor de una determinada acción en bolsa?
- ¿Y predecir la demanda de un determinado producto para avisar a nuestros proveedores?

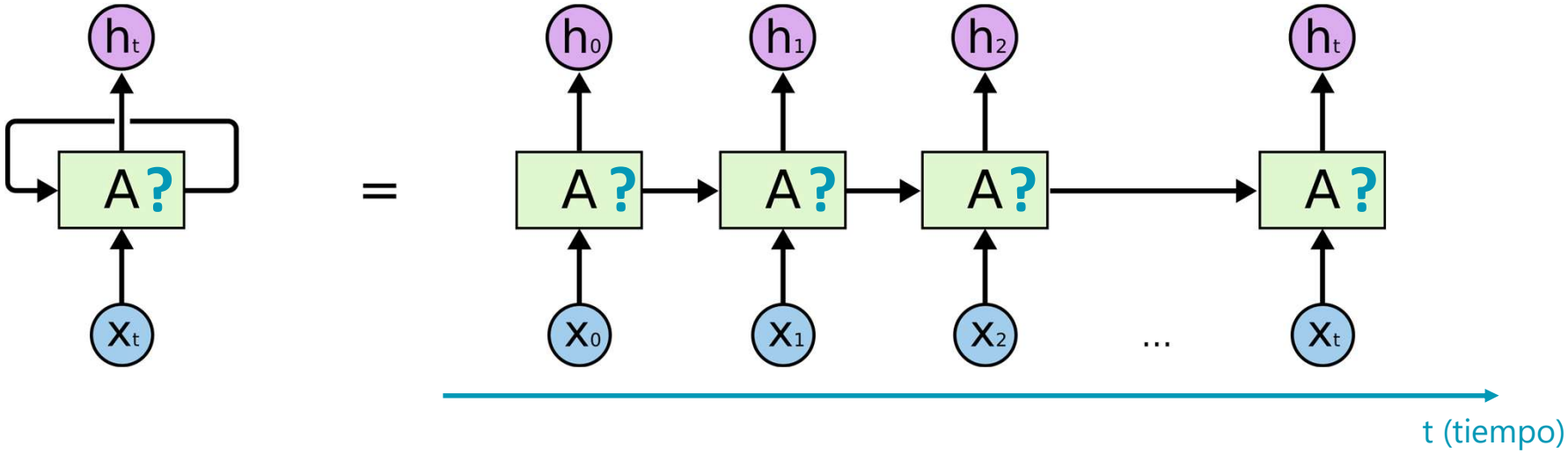
Necesitamos algún algoritmo con **MEMORIA**.

Las redes neuronales NO la tienen.

Redes neuronales recurrentes

¿Qué son las Redes Neuronales Recurrentes?

En su esencia, conjuntos de capas y “puertas” que deciden que información guardar y cual no a lo largo del tiempo.



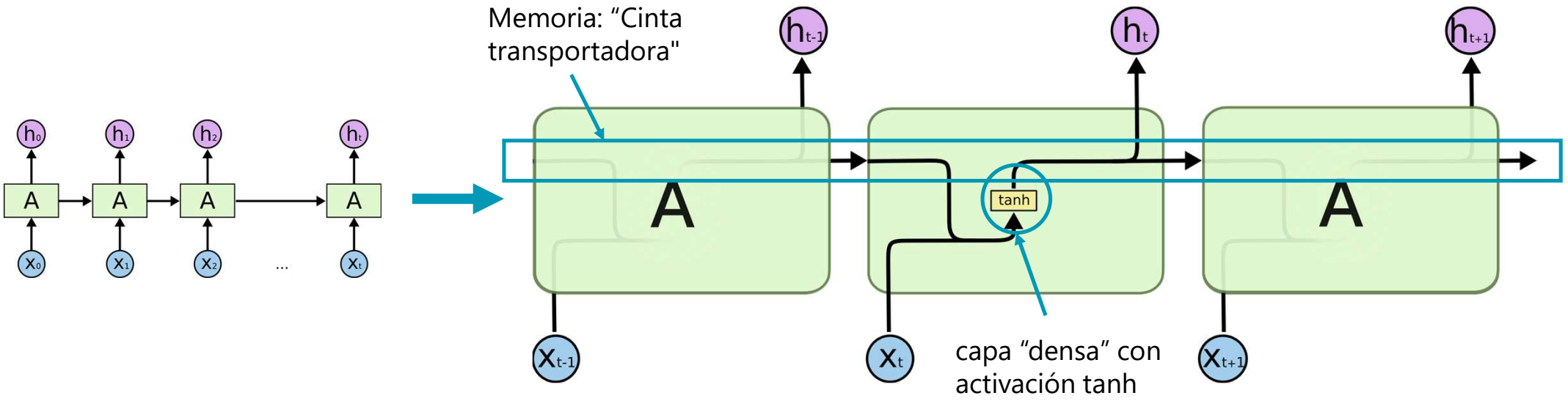
El diagrama básico expresa la red de forma comprimida. Si lo “desenrollamos” obtenemos la arquitectura real de una RNN. Pero... ¿qué significa?

http://colah.github.io/posts/2015-08-Understanding-LSTMs/
Video recomendado explicando la intuición: https://www.youtube.com/watch?v=UNmqTiOnRfg
Para ampliar: http://karpathy.github.io/2015/05/21/rnn-effectiveness/ (Video)

Redes neuronales recurrentes

¿Qué son las Redes Neuronales Recurrentes?

¿Qué es A?



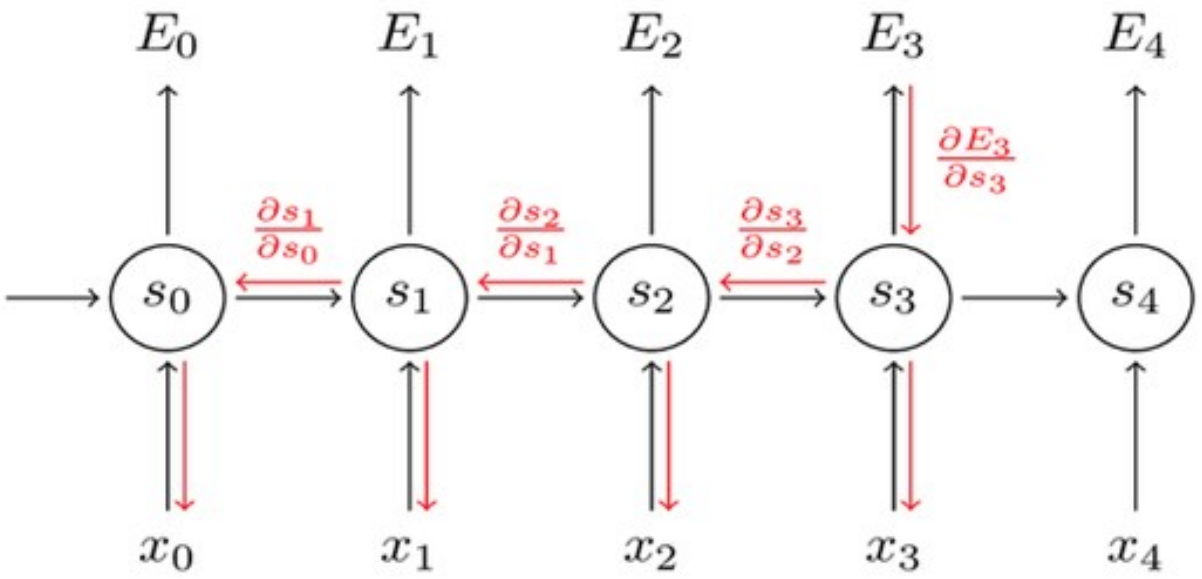
Ya no tenemos únicamente una entrada, sino que “**arrastramos**” un “**estado**” de una entrada a otra → ¡ya tenemos **memoria**!

Recordad que estamos tratando con **problemas temporales**.

Redes neuronales recurrentes

¿Qué son las Redes Neuronales Recurrentes?

¿Cómo aprenden las RNN?

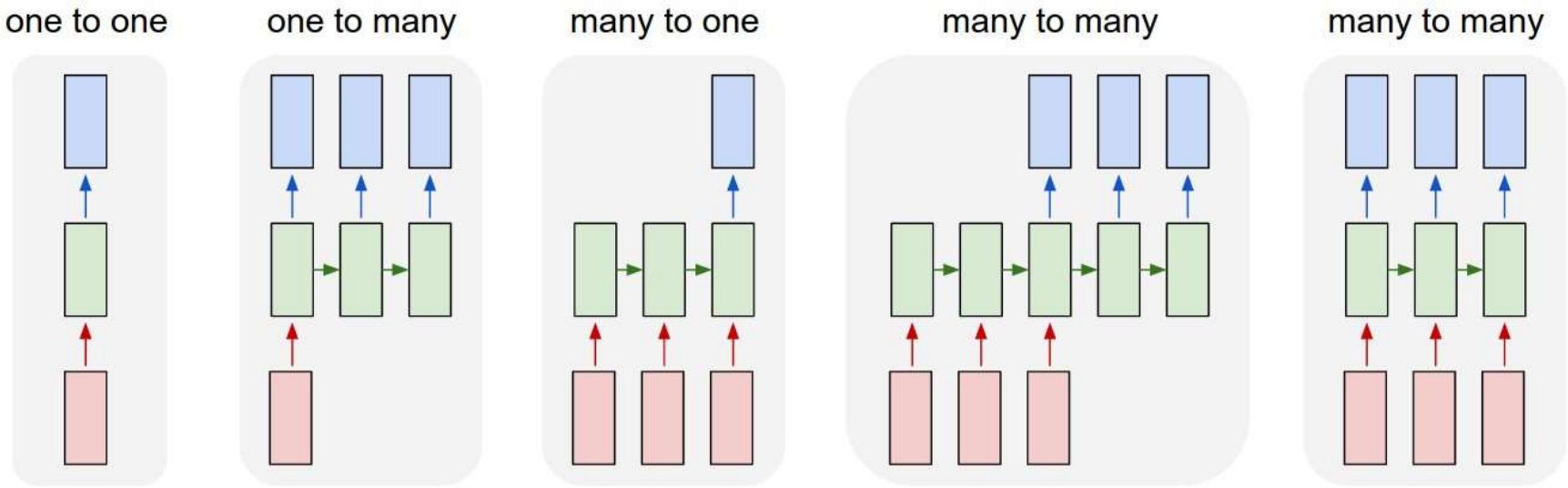


Back Propagation Through Time

Redes neuronales recurrentes

¿Qué son las Redes Neuronales Recurrentes?

¿Qué tipos de problemas nos permiten abordar las RNN?



Redes neuronales recurrentes

RNN – Ventajas e inconvenientes

Ventajas

- En general, RNNs dan una buena solución a problemas de predicción de series temporales
- Su desempeño no se ve demasiado afectado por *missing values*.
- Pueden encontrar patrones complejos en las series
- Dan buenos resultados prediciendo incluso más allá de unos pocos instantes
- Modelan las secuencias de forma que cada muestra se puede considerar independiente de las demás (permite *shuffle*)

Inconvenientes

- Cuando se entrenan en secuencias muy largas sufren el conocido como vanishing gradient or exploding gradient problem
- Las RNNs básicas tienen poca memoria y no pueden tener en cuenta un histórico muy largo
- Su entrenamiento es dificilmente paralelizable y además es muy costoso computacionalmente

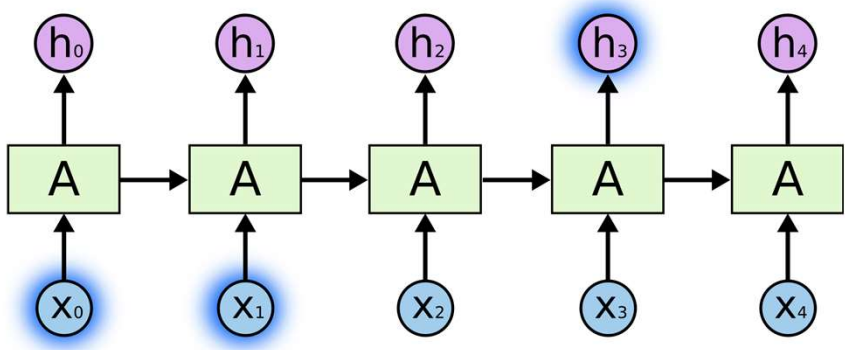
https://sensioai.com/blog/034_rnn_intro

Redes neuronales recurrentes

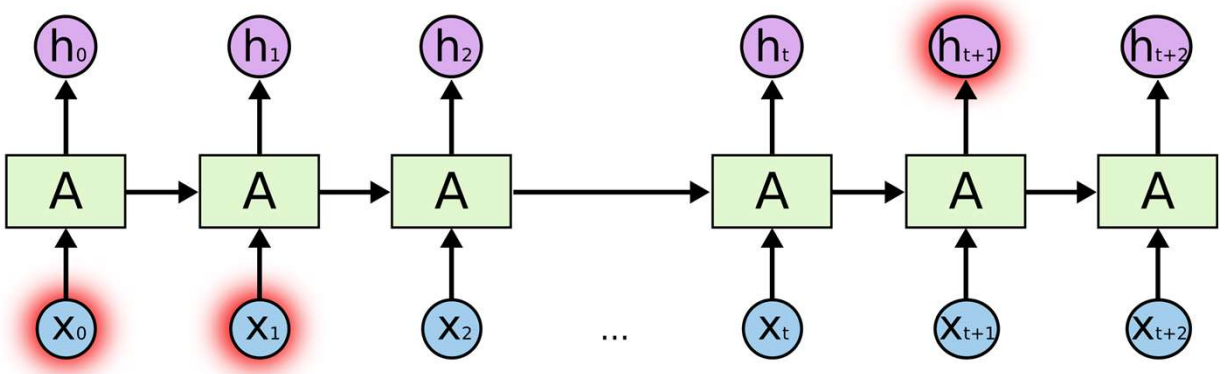
El problema de la memoria

Las RNN básicas sufren de problemas de memoria...

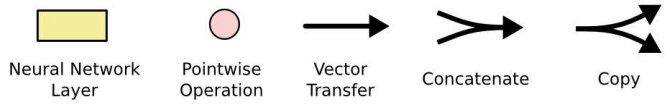
Ejemplo 1



Ejemplo 2



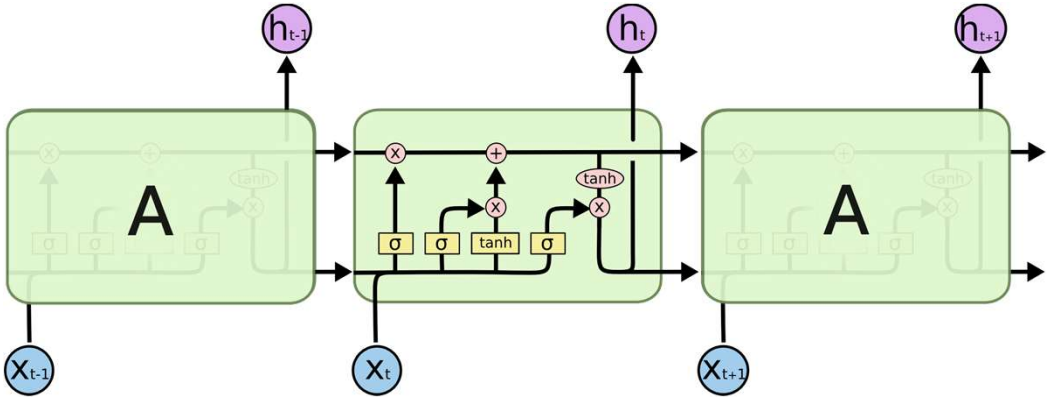
Redes neuronales recurrentes



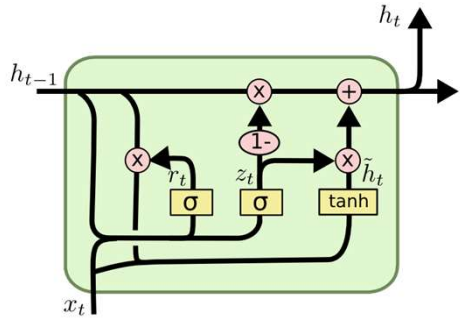
El problema de la memoria

...que se solventan con otras implementaciones un poco más complejas.

Long Short-Term Memory (LSTM) networks



Gated Recurrent Unit (GRU) networks

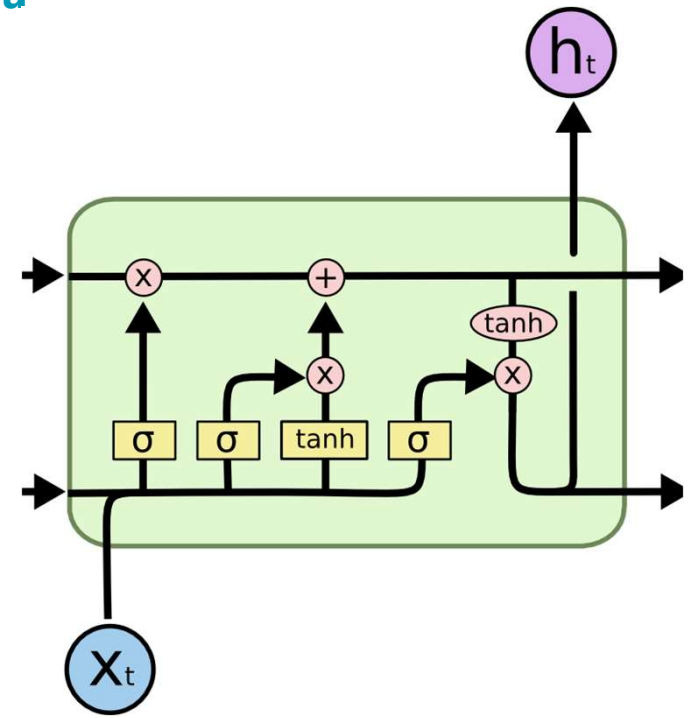


Redes neuronales recurrentes

Long Short Term Memory networks (LSTMs) – Estructura

Igual que las RNN, pero incorporan mecanismos para elegir qué "recuerdan" y que "olvidan":

- **"Cinta transportadora":** mantiene en memoria los datos necesarios
- **Puerta de olvido:** elimina datos de la cinta transportadora
- **Puerta de entrada:** añade datos a la cinta transportadora
- **Puerta de salida:** decide qué datos de la cinta transportadora va a sacar la red por salida en cada instante t_n

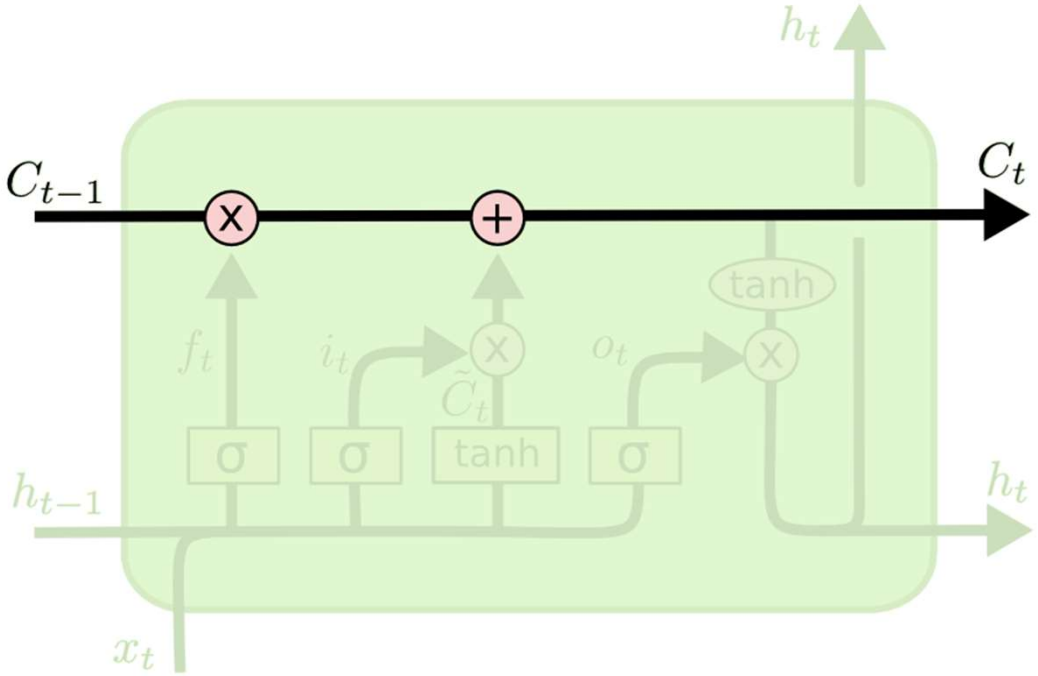


Redes neuronales recurrentes

Long Short Term Memory networks (LSTMs) – Estado

La “cinta transportadora” es el **estado** de la celda LSTM, es decir, la **memoria.**

Para cada predicción, la celda **combinará** la información en la **memoria** con la **entrada** que reciba para producir la **salida**.

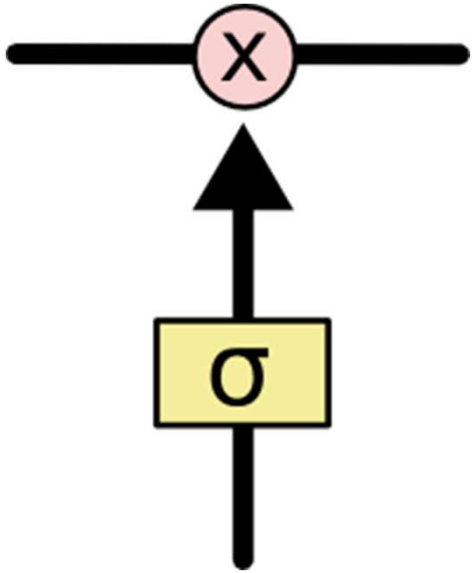


Redes neuronales recurrentes

Long Short Term Memory networks (LSTMs) - Puertas

Las **puertas** se componen de una red neuronal de una capa con función de activación sigmoide, lo cual da como salida un "**mapa de memoria**", que multiplicados por la salida del instante anterior y la nueva entrada, decide qué pasa al siguiente punto.

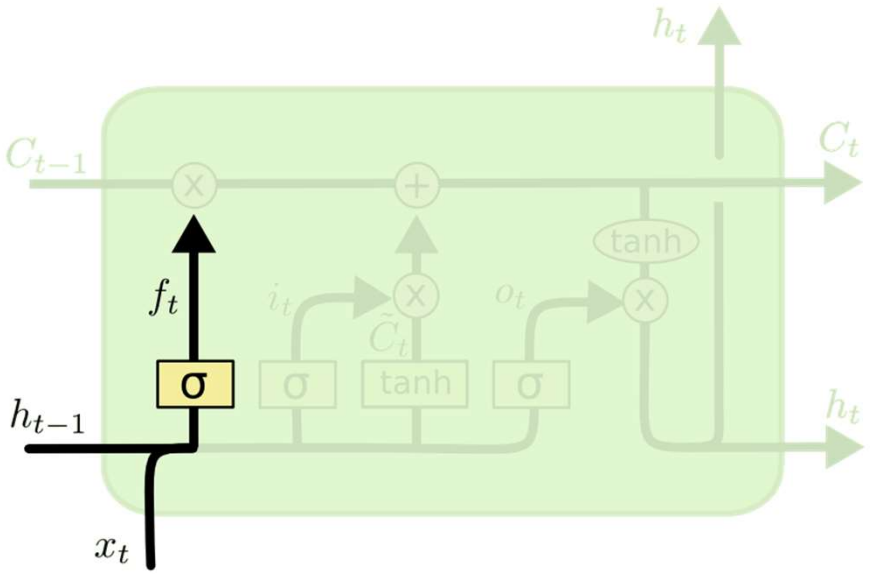
El mapa de memoria toma valores entre 0 y 1.



Redes neuronales recurrentes

Long Short Term Memory networks (LSTMs) – Puerta de olvido

Puerta de olvido: elimina datos de la cinta transportadora

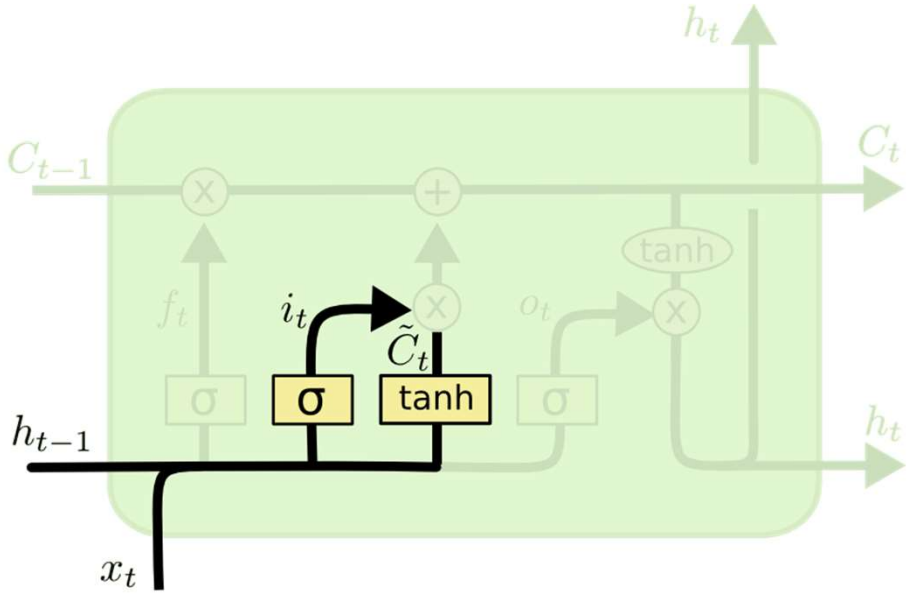


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Redes neuronales recurrentes

Long Short Term Memory networks (LSTMs) – Puerta de entrada

Puerta de entrada: añade datos a la cinta transportadora



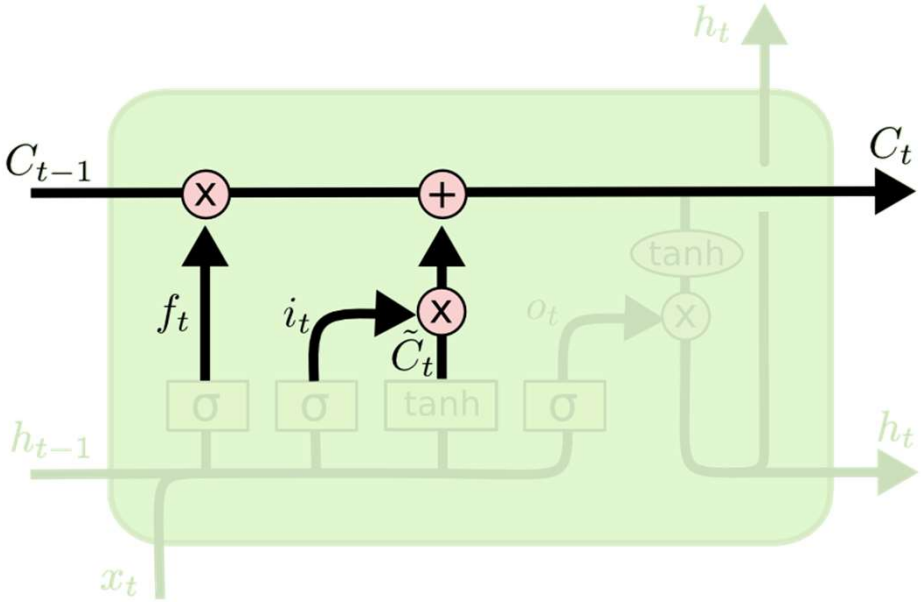
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Redes neuronales recurrentes

Long Short Term Memory networks (LSTMs) – Actualización del estado

Cuando ha decidido qué información de la cinta transporadora (**estado**) olvida y qué añade, ejecuta la **actualización.**

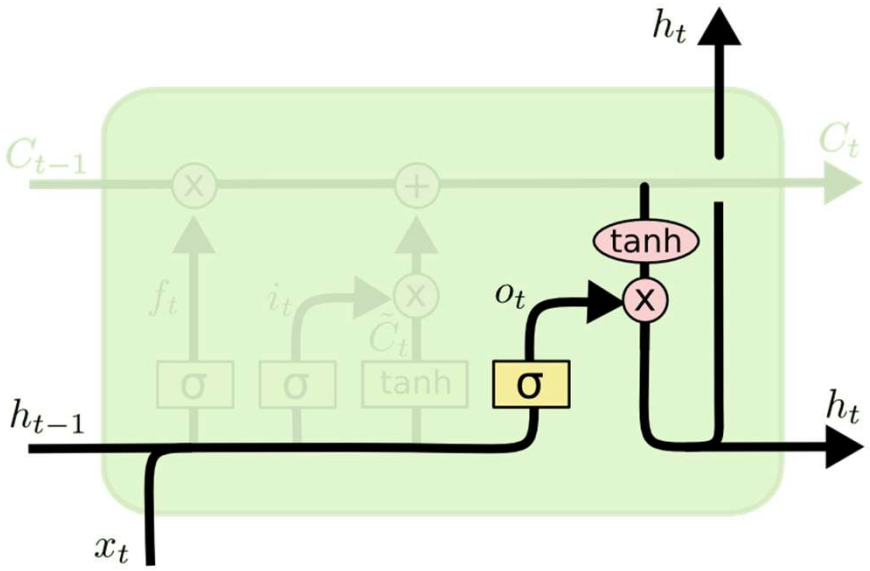


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Redes neuronales recurrentes

Long Short Term Memory networks (LSTMs) – Puerta de olvido

Puerta de salida: decide qué datos de la cinta transportadora va a sacar la red por salida en cada instante t_n



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Redes neuronales recurrentes

Ejemplos

Notebook 1. Predicción de demanda en vuelos.

Notebook 2. Predicción de valores en bolsa.

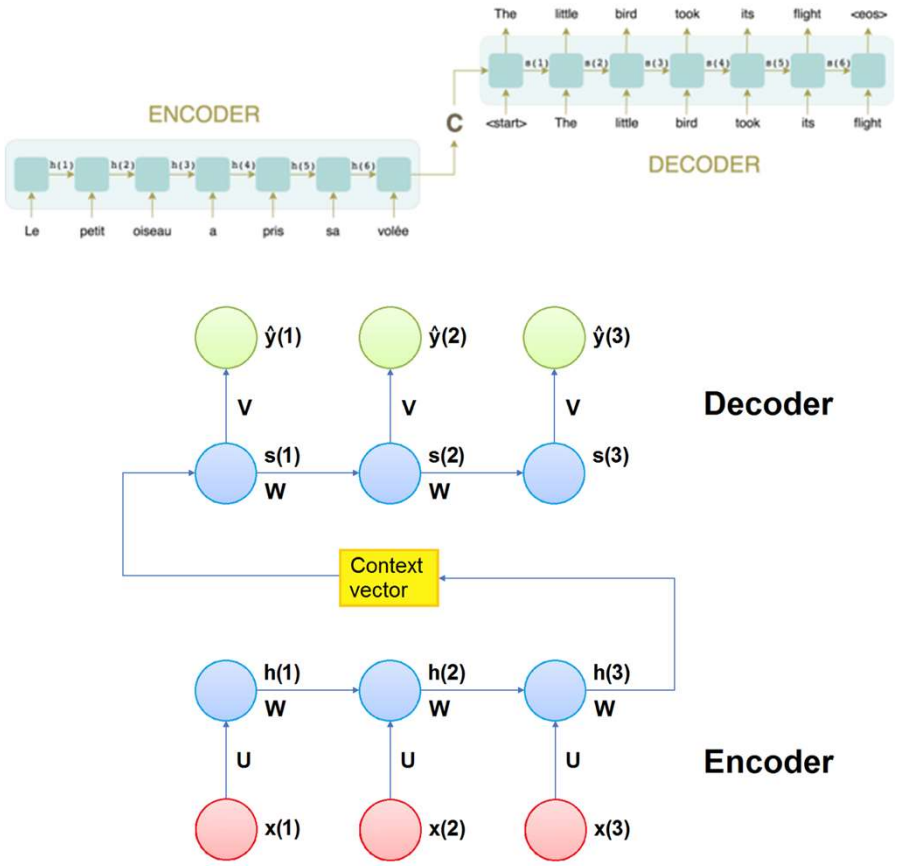
Para ampliar: Predicción de series temporales con TF

Redes neuronales recurrentes

Arquitectura Encoder-decoder

- Este tipo de arquitectura se introdujo para lidiar con problemas del tipo many-to-many o **sequence-to-sequence**.
- Permite **diferente longitud** de entrada y de salida.
- El **codificador procesa** la secuencia de entrada y la codifica en un un **context vector.**
- El **decodificador genera** la secuencia de salida utilizando la información del **context vector.**
- Si la secuencia es **muy larga**, el **context vector** puede **no** ser **suficiente** para codificarla.

¿Solución? La **atención.**

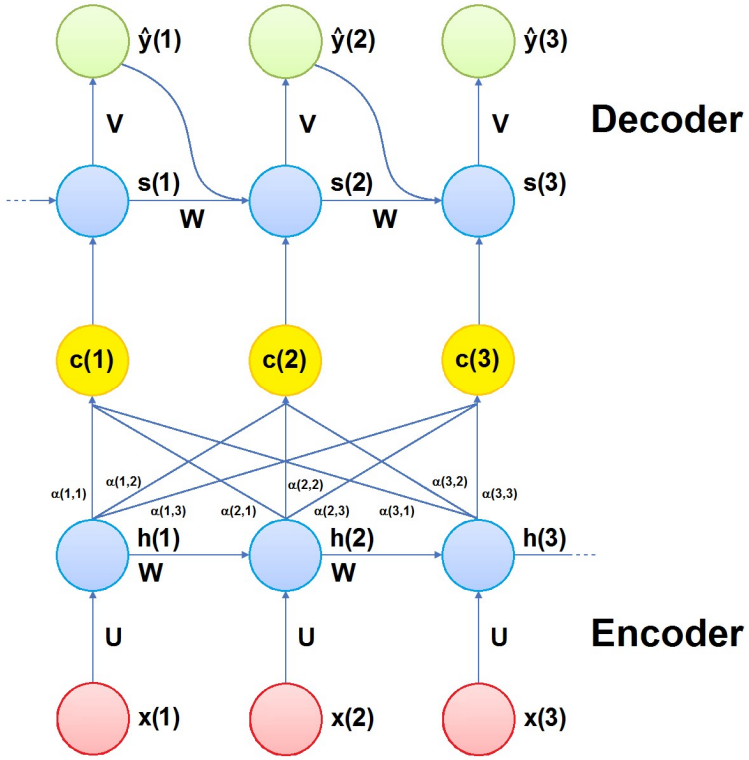


https://towardsdatascience.com/time-series-forecasting-with-deep-learning-and-attention-mechanism-2d001fc871fc
https://blog.dataiku.com/decoding-nlp-attention-mechanisms-to-understand-transformer-models

Redes neuronales recurrentes

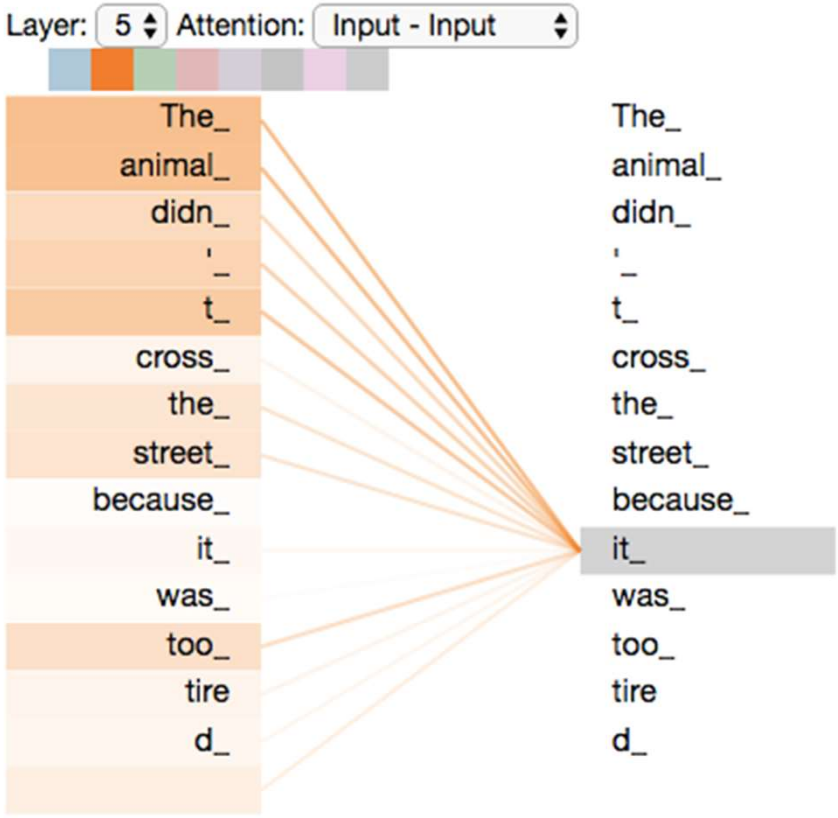
Arquitectura Encoder-decoder con atención

- Permite al decodificador fijarse en determinada información codificada por el codificador al decodificar y generar la secuencia de salida
- Se genera un context vector diferente para cada timestep del decodificador, en función del "estado" anterior del decodificador y todos los "estados" anteriores del codificador, asignándoles pesos
- Da mayor peso (**atención**) a los elementos más importantes
- Da buenos resultados incluso con largas secuencias
- El modelo final es más interpetable



Redes neuronales recurrentes

Arquitectura Encoder-decoder con atención

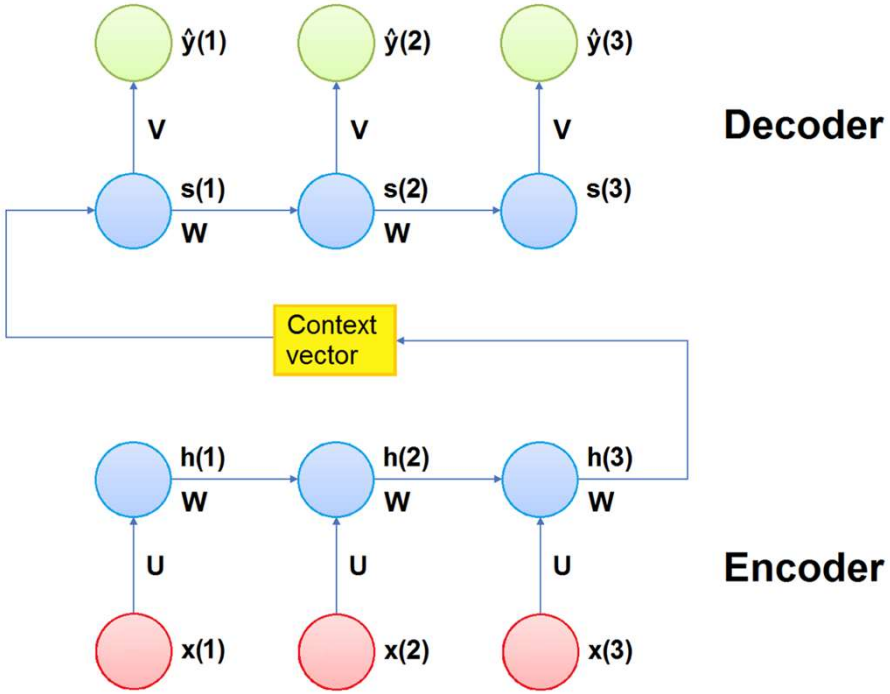


https://towardsdatascience.com/time-series-forecasting-with-deep-learning-and-attention-mechanism-2d001fc871fc

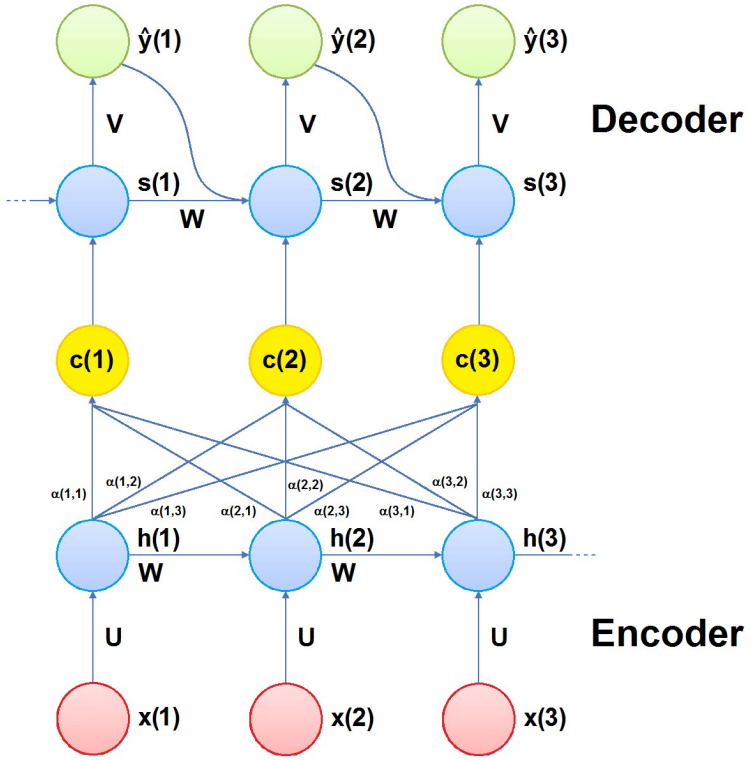
Redes neuronales recurrentes

Arquitectura Encoder-decoder con atención

Encoder-decoder



Encoder-decoder con atención



https://towardsdatascience.com/time-series-forecasting-with-deep-learning-and-attention-mechanism-2d001fc871fc

Redes neuronales recurrentes

Attention is all you need

Visualizar atención

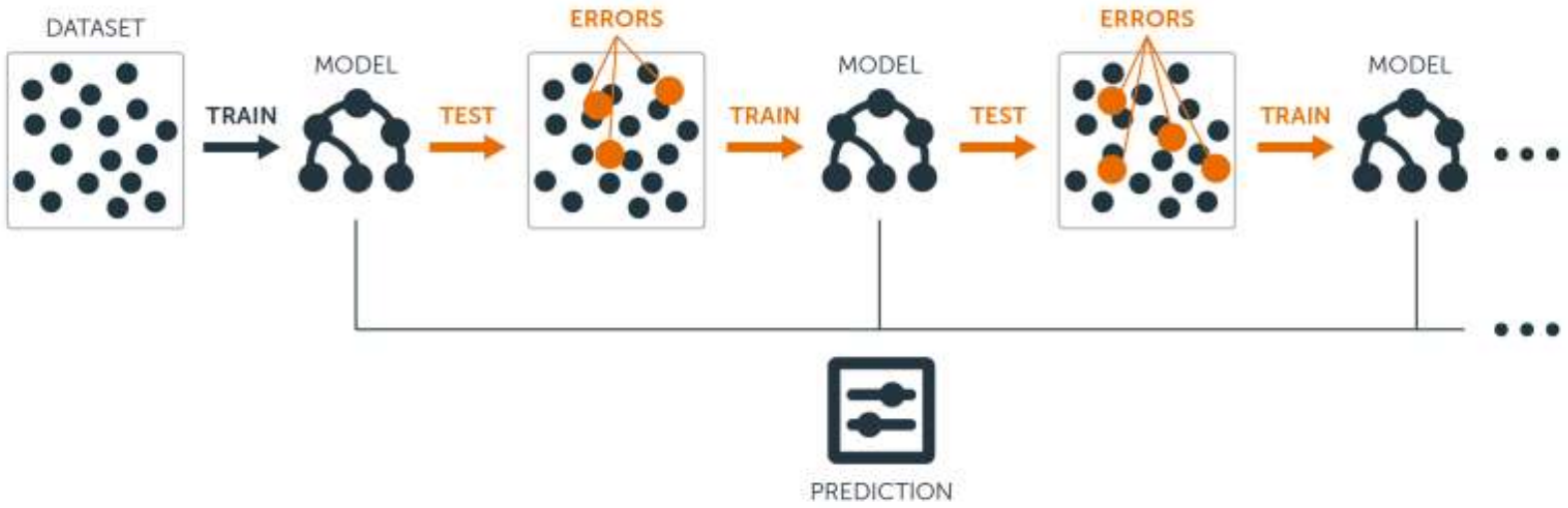
Ejemplos de LSTMs con atención

XGBoost (eXtreme Gradient Boosting)

¿Qué es y cómo funciona?

XGBoost es un método englobado dentro de "model boosting", que consiste en crear un **conjunto de modelos (ensemble)** que mejoran poco a poco utilizando los errores más grandes para ello.

Cada modelo individual intenta **corregir** los errores cometidos en la **iteración** previa mediante la **optimización** de una función de pérdidas.



https://blog.bigml.com/2017/03/14/introduction-to-boosted-trees/

XGBoost (eXtreme Gradient Boosting)

¿Qué es y cómo funciona?

Las principales características del XGBoost son las siguientes:

1. No se trata de un único modelo, sino de un **conjunto** de modelos (árboles de decisión)
2. Cada árbol no predice la variable objetivo, sino el **residuo** o el **error** cometido por árboles previos
3. Al final, todos los árboles se **combinan** para obtener una predicción final

Se trata de un método muy **potente** y utilizado muy a menudo en competiciones donde los datos son **estructurados** o **tabulares**, tanto para **clasificación** como para **regresión**.

Aquí tenéis la prueba: https://github.com/dmlc/xgboost/tree/master/demo#machine-learning-challenge-winning-solutions

https://blog.bigml.com/2017/03/14/introduction-to-boosted-trees/
https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

XGBoost (eXtreme Gradient Boosting)

¿Qué es y cómo funciona?

Notebook de Ejemplo

https://blog.bigml.com/2017/03/14/introduction-to-boosted-trees/
https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

Time Series Forecasting

Consejos

- **Escalar** datos a [0, 1] o [-1, 1]
- **Standard Scaling** (restar media y dividir entre la desviación estándar)
- **Power Transforming** (utilizar una función de transformación para hacer que los datos sigan una distribución más normal, normalmente se usa en *skewed data* / datos con outlier)
- Eliminación de **outliers**
- **Diferenciación** muestra-muestra o cálculo de diferencias de porcentajes (*Pairwise Diffing* / *Calculating Percentage Differences*)
- **Descomposición** estacional (*Seasonal Decomposition*): tratamos de hacer los datos estacionarios
- **Ingeniería** de **características** (*Feature engineering*)
- **Re-muestreado** de la componente **temporal**: ¿nos interesan los datos/hora? ¿datos/minuto? ¿datos/segundo?
- **Re-muestreado** de alguna **variable**: por ejemplo, *bucketizar* variable cuando alcance N unidades
- **Medidas móviles** (*Rolling Values*) (por ejemplo, medias móviles)
- **Agregación** de variables (*Aggregations*)
- **Combinaciones** de las anteriores

https://towardsdatascience.com/the-time-series-transformer-2a521a0efad3

Time Series Forecasting

Resumen

- Las **RNN** básicas permiten hacer uso de información **temporal** para efectuar posteriores predicciones, pero sufren de **problemas de memoria** cuando las **secuencias** son muy **largas**.
- Las **LSTM** implementan mecanismos para **decidir qué recordar y qué olvidar**, con lo que su memoria es más selectiva y alivian un poco los problemas de memoria de las RNNs.
- Tanto las **RNN** como las **LSTM** tienen el **mismo problema**: son **muy costosas de entrenar** (no se puede paralelizar, ya que el **instante t+1 depende del t**.
- Los ensembles de árboles para regresión (**XGBoost**) son algoritmos que funcionan bastante bien siempre que tengamos datos estructurados y entrenemos de la forma adecuada, pero tienen **problemas** cuando el **rango** de los valores a predecir es **diferente** al de entrenamiento.