Redes neuronales recurrentes con puertas
Gated Recurrent Neural Networks

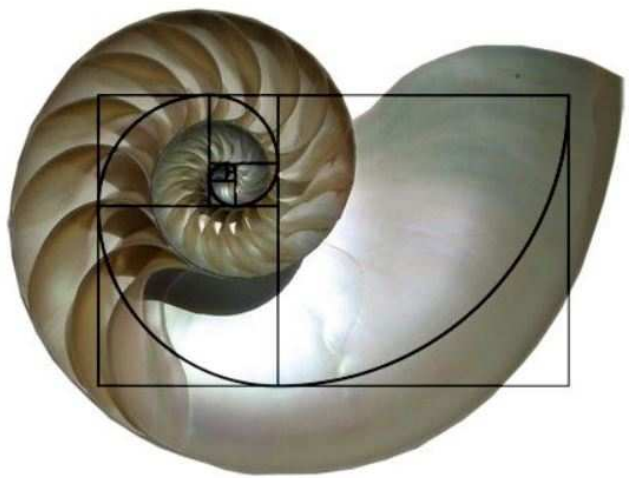
Miguel A. Gutiérrez Naranjo

Departamento de Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla

18 de Mayo, 2018

Introducción

- Cuando analizamos una fotografía toda la información se recibe de una vez. No cambia con el tiempo.
- Pero en muchas situaciones, la predicción de un valor depende de la evolución en el tiempo de los valores previos.
- En otras palabras, necesitamos herramientas específicas para tratar con información secuencial



Caveat

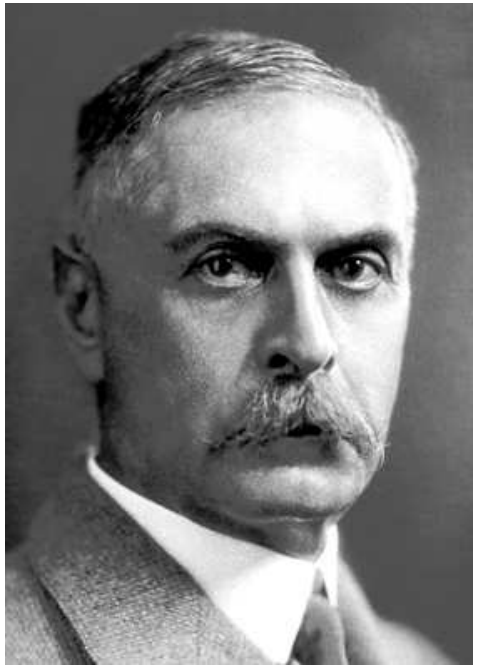
- Al aplicar técnicas de Aprendizaje Automático, asumimos *a priori* que la secuencia que analizamos *no es aleatoria*.
- La respuesta correcta de cualquier sistema de Aprendizaje Automático que tome como entrada una secuencia aleatoria debe ser

NO HE ENCONTRADO REGULARIDAD EN LOS DATOS.

Grupos sanguíneos

¿Qué significa aleatorio?

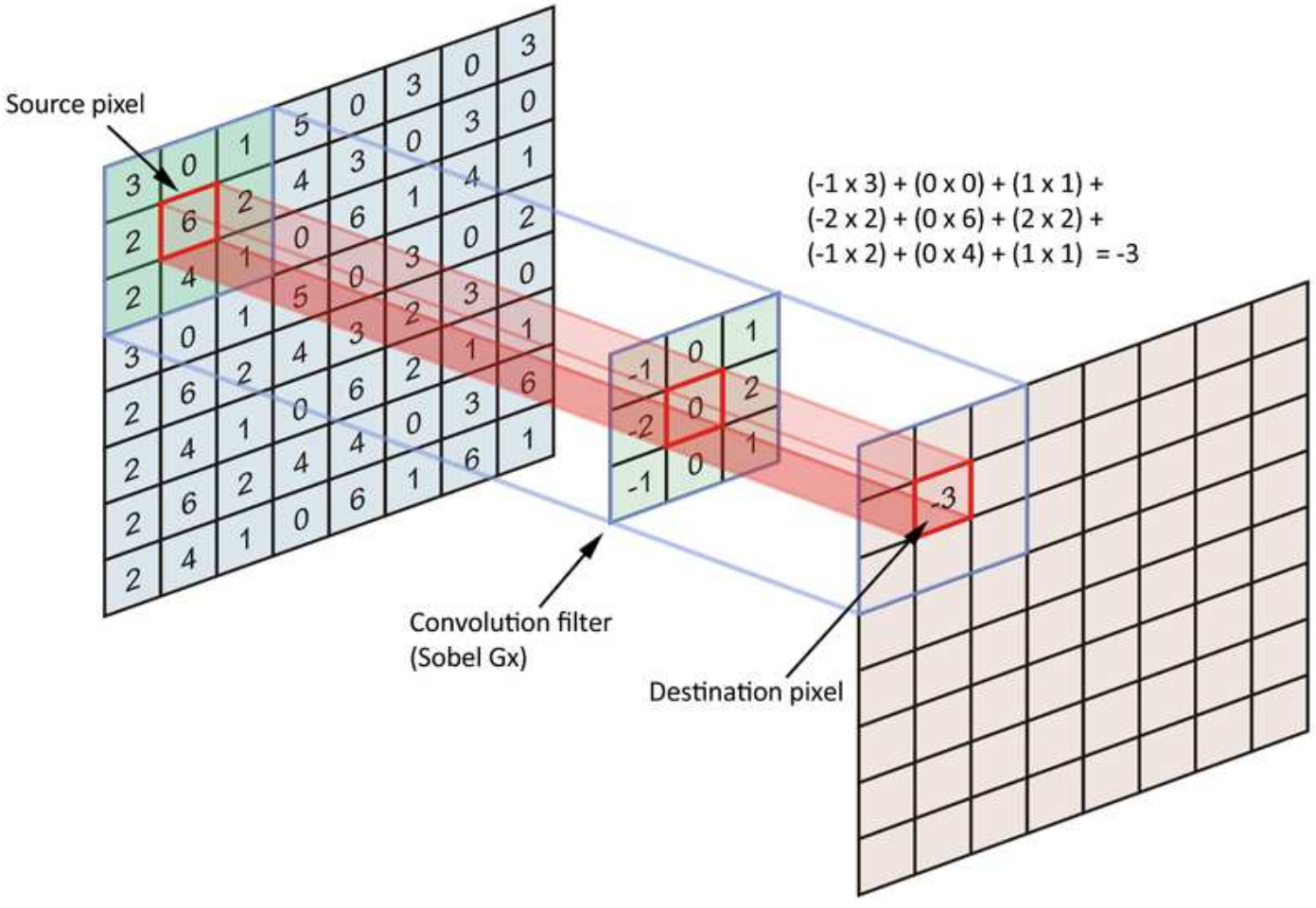
- El primer intento de transfusión sanguínea fue en 1492.
- El éxito o fracaso era aleatorio.
- Karl Landsteiner identificó los cuatro grupos *A*, *B*, *AB* y 0 en 1909.
- Recibió el Premio Nobel de Medicina en 1930.



Ejemplos de aplicación de Aprendizaje Automático a *información secuencial*

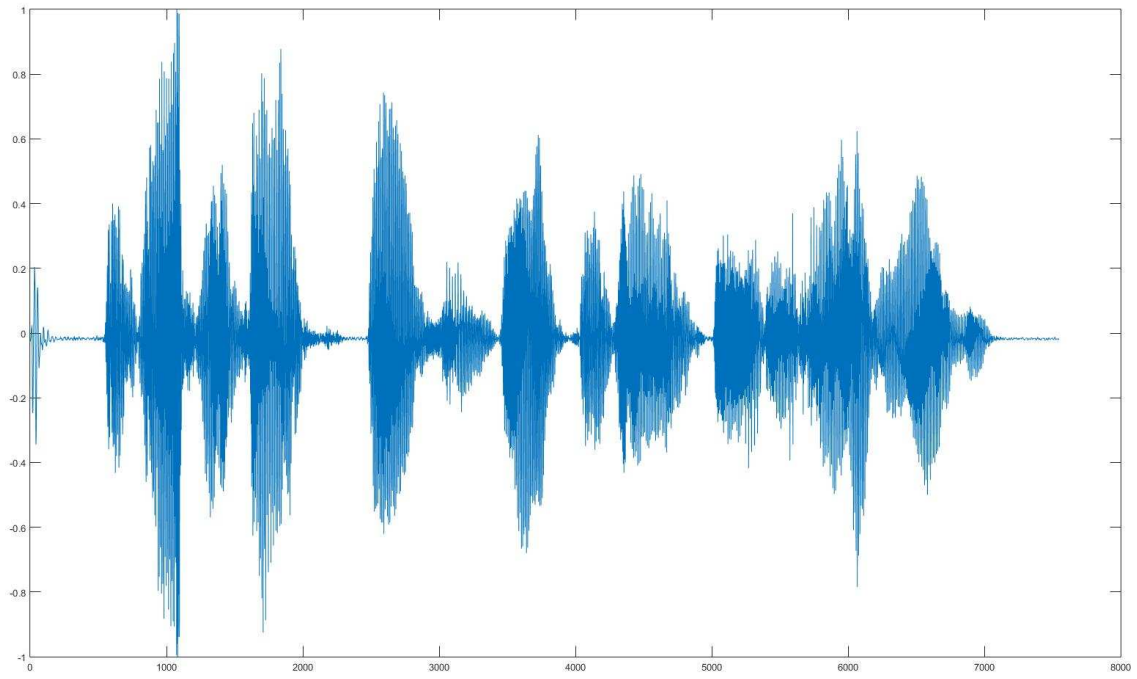
- Machine Translation
  https://arxiv.org/pdf/1409.3215.pdf
- Hand writing generation
  https://arxiv.org/pdf/1308.0850v5.pdf
- Speech Recognition in English and Mandarin
  https://arxiv.org/abs/1512.02595
- Stock Price Prediction
  https://ieeexplore.ieee.org/document/8126078/
- *Modelado de Sistemas Dinámicos (Tesis de Diego Cabrera)*
  https://idus.us.es/xmlui/handle/11441/70311
- ...

Una primera aproximación: *Usar Convolución*



Una primera aproximación: *Convolución 2D*

Tratamos la representación gráfica de la secuencia de valores como una imagen

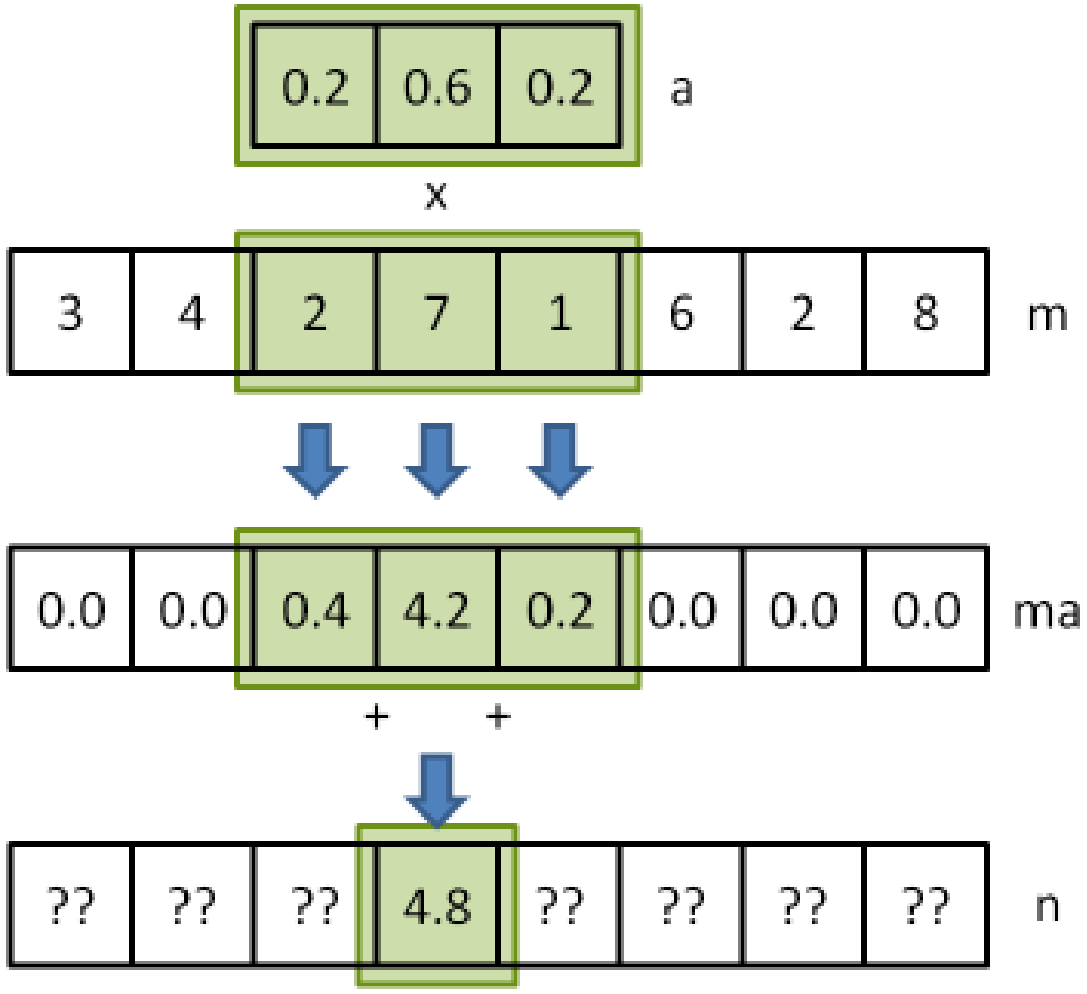


Una primera aproximación: *Convolución*

- Información local
- Uso de parámetros compartidos (Mismo *kernel*)
- La salida de la convolución es función de un (pequeño) conjunto de valores contiguos.
- ¿Puedo utilizar estas ideas para el análisis de secuencias?

Una primera aproximación: *Convolución 1D*

Tratamos la sucesión de datos como una secuencia 1D



Caveat

Martillo de oro[1] *:* Cuando la única herramienta que tienes es un martillo, todo problema comienza a parecerse a un clavo

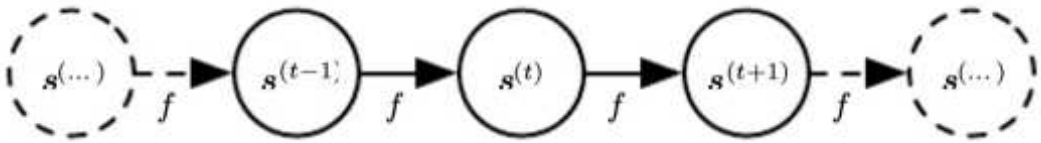
[1] https://es.wikipedia.org/wiki/Martillo_de_oro

Grafos computacionales

- Representación clásica de un sistema dinámico

$$
\begin{aligned}
s^{(0)} & \\
s^{(1)} &= f(s^{(0)}; \theta) \\
s^{(2)} &= f(s^{(1)}; \theta) \\
&= f(f(s^{(0)}; \theta); \theta)
\end{aligned}
$$

- En general $s^{(t)} = f(s^{(t-1)}; \theta)$



Asumimos que $s^{(t+1)}$ depende *exclusivamente* de $s^{(t)}$

Grafos computacionales

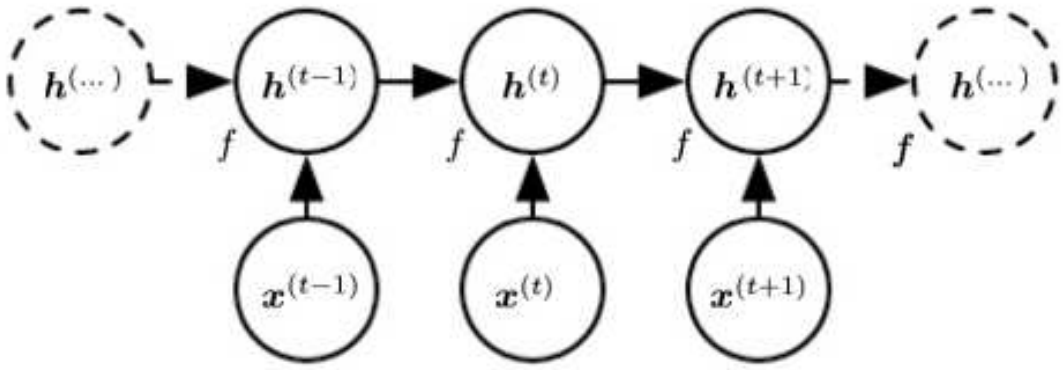
- No es realista pensar que $s^{(t+1)}$ depende *exclusivamente* de $s^{(t)}$

$$
\begin{aligned}
s^{(0)} & \\
s^{(1)} &= f_0(s^{(0)}) \\
s^{(2)} &= f_1(s^{(1)}, s^{(0)}) \\
s^{(3)} &= f_2(s^{(2)}, s^{(1)}, s^{(0)}) \\
\ldots & \\
s^{(t+1)} &= f_t(s^{(t)}, \ldots, s^{(1)}, s^{(0)})
\end{aligned}
$$

Grafos computacionales

- Introducimos la noción de estado $h^{(t)}$

$$\begin{aligned} h^{(t)} &= g_t(x^{(t)}, \ldots, x^{(1)}, x^{(0)}) \\ &= f(x^{(t)}, h^{(t-1)}; \theta) \end{aligned}$$



Caveat

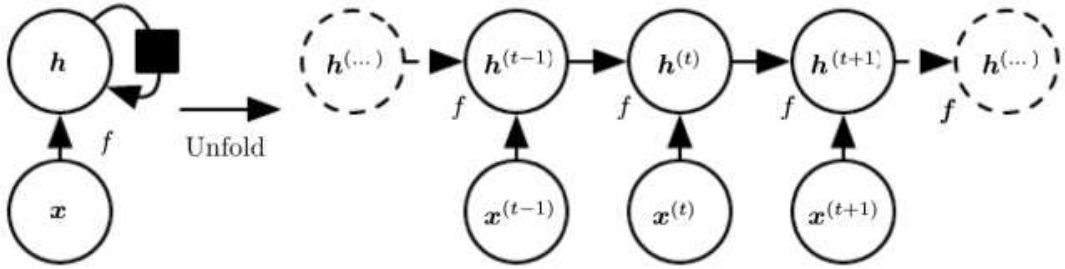
$$h^{(t)} = f(h^{(t-1)}, x^{(t)}; \theta)$$

Asumimos como cierto *a priori*

- . . . que el *estado oculto* $h^{(t)}$ guarda *toda la información* útil de la secuencia $x^{(t)}, \ldots, x^{(1)}, x^{(0)}$
- . . . que la *misma* función de transición f con los *mismos parámetros* θ nos sirve para todos los valores de t.

Grafos computacionales

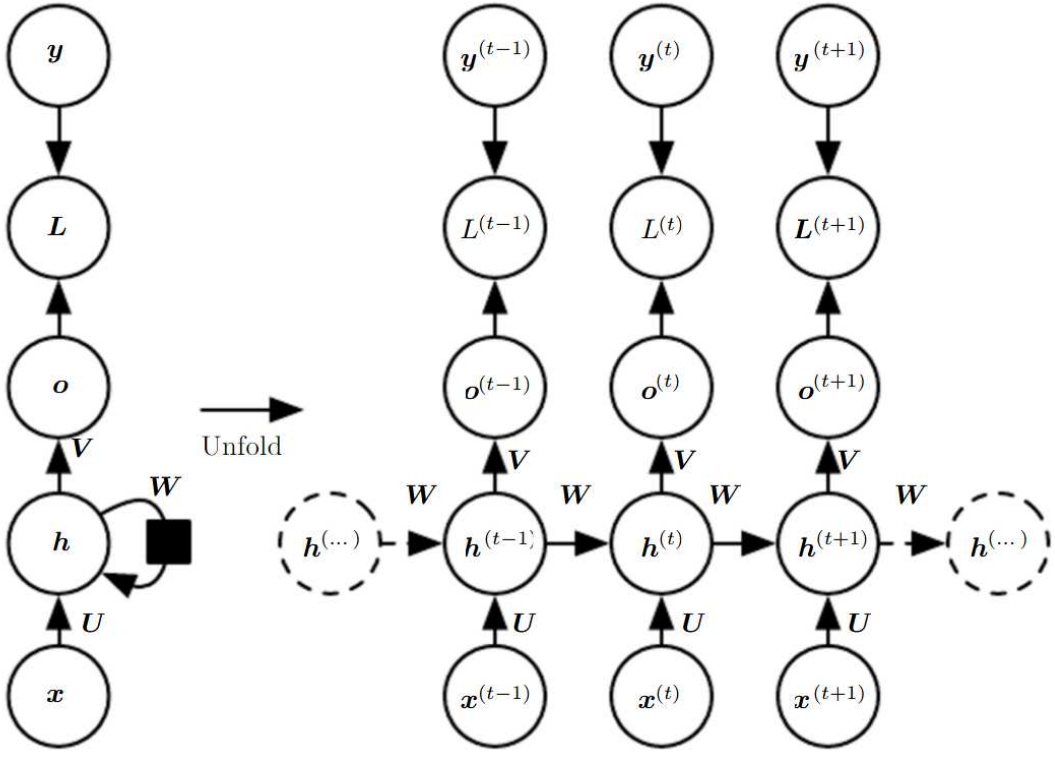
Representación compacta



- El cuadrado negro representa un retraso de una unidad de tiempo en la computación.
- Usando esta idea de los sistemas dinámicos, ¿podemos diseñar redes neuronales para *aprender* a partir de secuencias?

Redes neuronales recurrentes: Versión 1

Versión 1: Producen una salida en cada unidad de tiempo y tienen conexiones entre las unidades ocultas.



Redes neuronales recurrentes: Versión 1

- Son *universales* (cualquier función computable por una máquina de Turing puede ser computada por una red recurrente de este tipo de un tamaño finito)
- L es una función que mide la diferencia entre la salida de la red $\hat{y}^{(t)}$ y la salida esperada $y^{(t)}$
- La salida $\hat{y}^{(t)}$ se calcula del siguiente modo (U, V y W son matrices de pesos y b y c son los vectores de sesgo *(bias)*)

$$
\begin{aligned}
a^{(t)} &= b + Wh^{(t-1)} + Ux^{(t-1)} \\
h^{(t)} &= tanh(a^{(t)}) \\
o^{(t)} &= c + Vh^{(t)} \\
\hat{y}^{(t)} &= softmax(o^{(t)})
\end{aligned}
$$

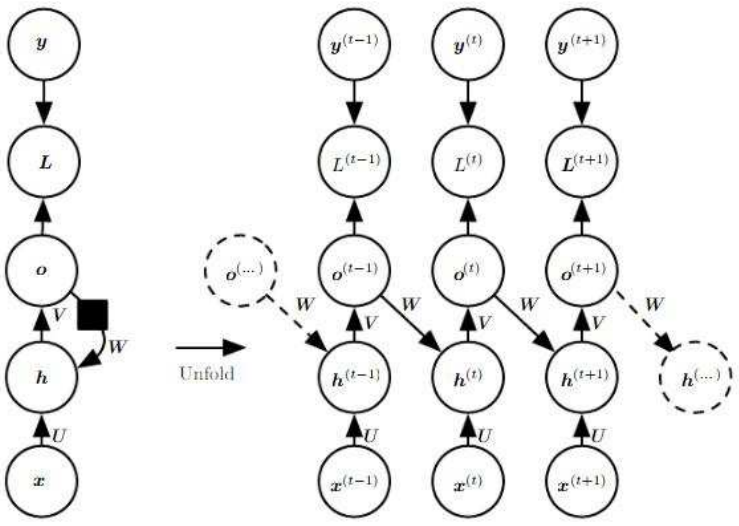
Redes neuronales recurrentes: Versión 1

- A partir de la representación *unfolded* podemos ver que el entrenamiento de la red consiste en el ajuste de los pesos de las matrices U, V y W y los vectores b y c.
- El entrenamiento se realiza con una variante de la retropropagación conocido como BPTT

  Back-propagation through time
- Funciona muy bien, pero es *extremadamente costoso* computacionalmente.

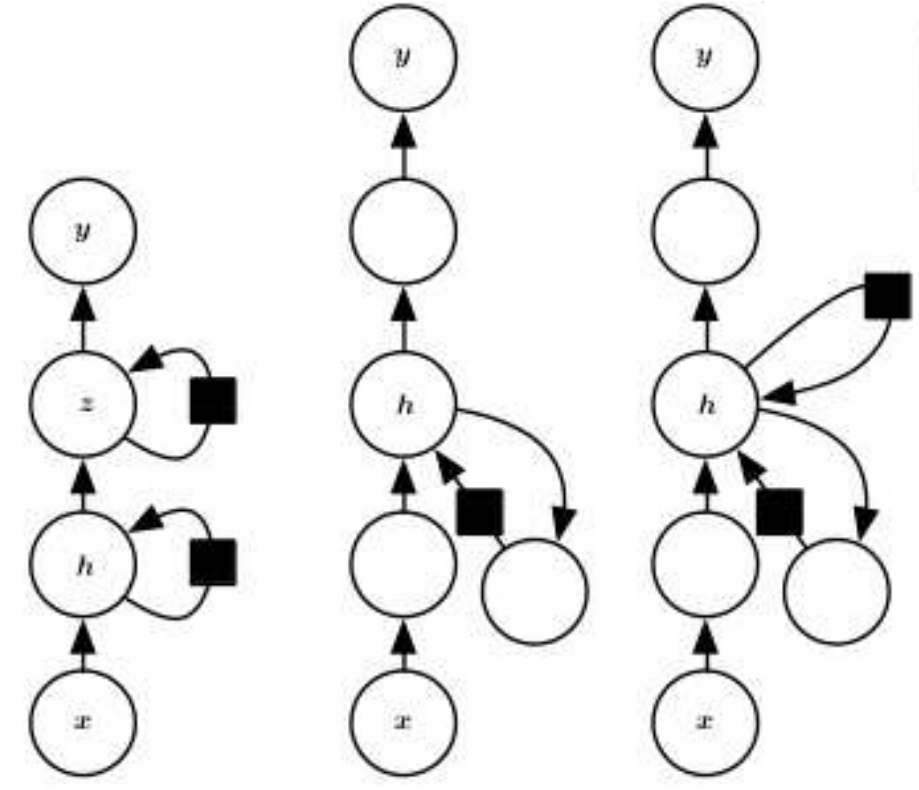
Otras posibles arquitecturas: Versión 2

Versión 2: Producen una salida en cada unidad de tiempo y sólo tienen conexiones desde la salida $o^{(t-1)}$ a las unidades ocultas h^t.



- No son universales
- Usan el método Teacher Forcing para entrenar.

Muchas otras posibles arquitecturas



Muchas otras posibles arquitecturas

Hidra de Lerna[2]

[2] https://es.wikipedia.org/wiki/Hidra_de_Lerna

Muchas otras posibles arquitecturas

- Redes recurrentes bidireccionales: Combina dos redes, cada una de ellas abordando la secuencia de entrada en una dirección.
- Arquitecturas Encoder-Decoder: Un *encoder* toma la entrada y produce un contexto C y un decoder trata de obtener la salida a partir de C.
- Redes neuronales Recursivas: El grafo de computación se formaliza como un árbol.
- ...

Muchas otras posibles arquitecturas

. . . sí, vale, pero ¿Cuál es la mejor?

The *no free lunch* Theorem

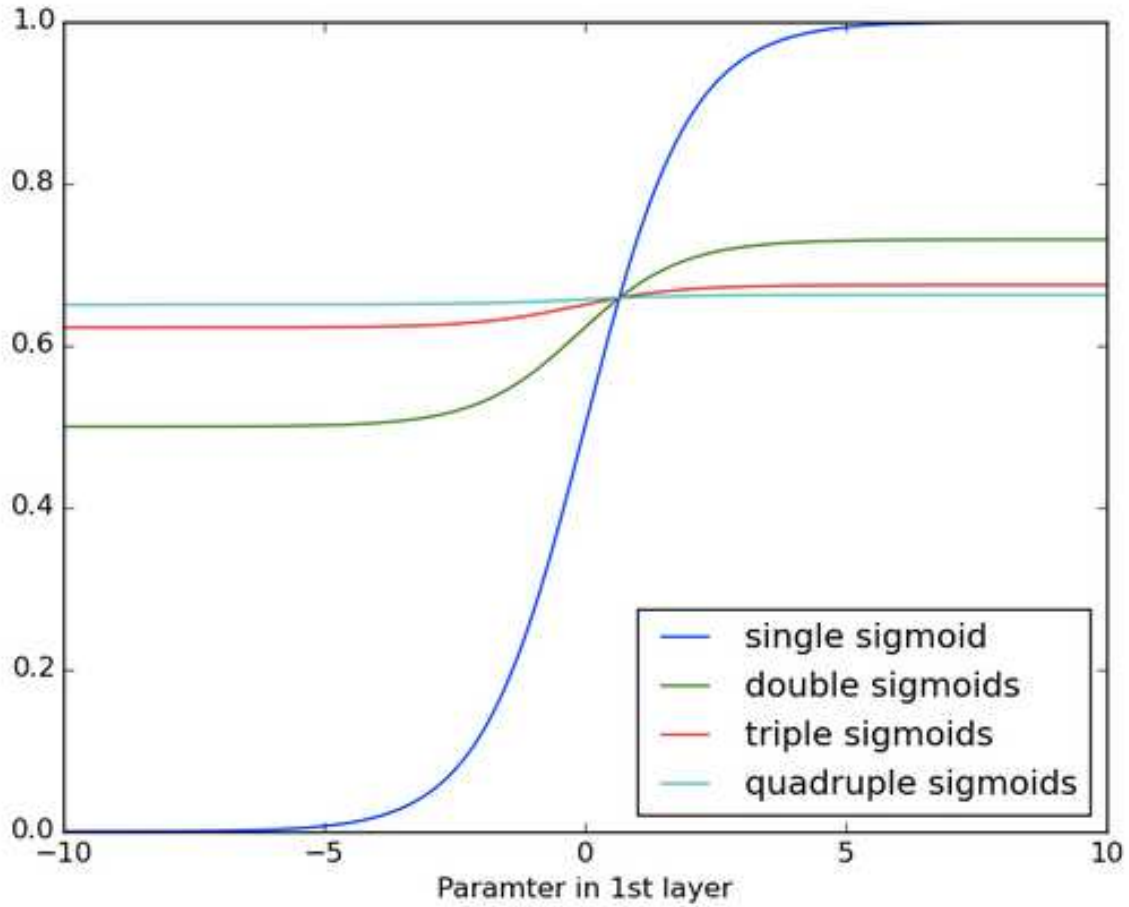
- David H. Wolpert y William G. Macready. *No Free Lunch Theorems for Optimization*. IEEE Transactions on Evolutionary Computation, Vol.1, num. 1, pp.67-82.[3]
- De manera informal . . .
- . . . . . *siempre puedo encontrar un conjunto de entrenamiento donde el sistema de aprendizaje A se comporta mejor que el sistema B.*

[3] https://ti.arc.nasa.gov/m/profile/dhw/papers/78.pdf

The Vanishing Gradient Problem

Usual en las dependencias a largo plazo

$$(-\infty, \infty) \overset{\sigma}{\Longrightarrow} (0, 1) \overset{\sigma}{\Longrightarrow} (0.5, 0.7301) \overset{\sigma}{\Longrightarrow} (0.6224, 0.6750) \overset{\sigma}{\Longrightarrow} (0.6507, 0.6626)$$

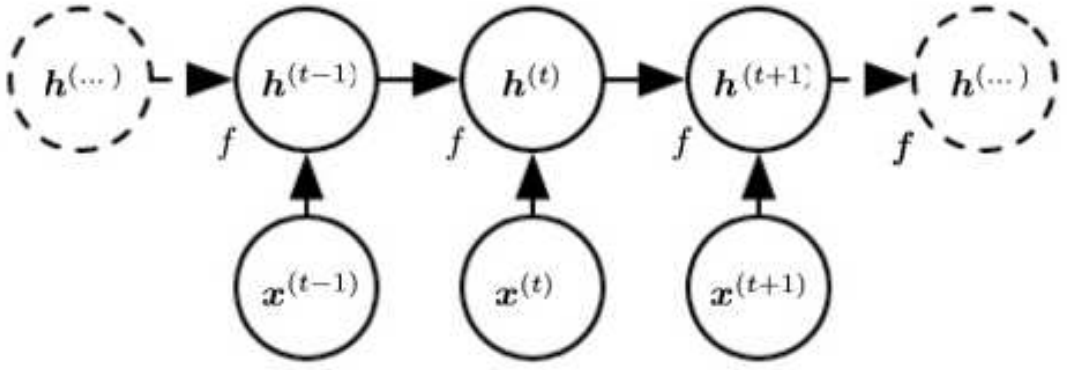


The Long Short-Term Memory Model

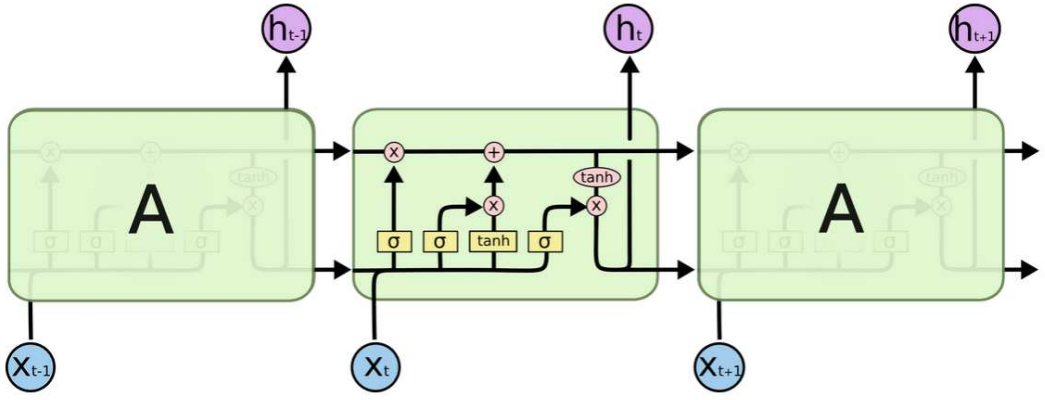
- ¿Cómo podemos introducir bucles en las representaciones compactas de los grafos computacionales evitando el *vanishing gradient problem*?
- Debemos crear grafos a través del tiempo de manera que las derivadas que necesitamos para el cálculo de los gradientes *no desaparezcan (vanish)*
- *Idea clave:* haciendo que el peso del bucle (que llamamos antes W) esté controlado por otra unidad (llamada *puerta del olvido (forget gate)*).

Unidades LSTM

Grafo computacional desenrollado (unfolded) de una red recurrente



Grafo computacional desenrollado (unfolded) de una red LSTM



Unidades LSTM

- Entrada:
  - x_t el valor de la secuencia en el instante t.
  - h_{t-1} la salida de la unidad LSTM en el paso anterior
  - C_{t-1} el estado de la unidad LSTM en el paso anterior.
- Salida
  - Nuevo estado, que es combinación del estado anterior C_{t-1} y el *candidato* a nuevo estado $\tilde{C}_t$. Podemos ver f_t e i_t como el *grado de influencia* de C_{t-1} y $\tilde{C}_t$ en el estado final.

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

  - La salida de la unidad

$$h_t = o_t * tanh(C_t)$$

Unidades LSTM

- Entrada: x_t, h_{t-1}, C_{t-1}
- Salida: $C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$ $\qquad$ $h_t = o_t * tanh(C_t)$

Nuevo Candidato:

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

- Puede usarse otra función de activación distinta a *tanh*
- $W_C[h_{t-1}, x_t]$ es una abreviatura para $W_C^1 * h_{t-1}$ y $W_C^2 * x_t$ donde W_C^1 son los *recurrent weights* y W_C^2 son los *input weights*.
- $[h_{t-1}, x_t]$ es la concatenación de los vectores h_{t-1} y x_t y b_C es el vector *bias*.

Unidades LSTM

- Cálculo del nuevo estado

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

- f_t es la forget gate

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

- i_t es la external input gate

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

Unidades LSTM

- Una vez calculado el nuevo estado

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

- La salida de la unidad LSTM es

$$h_t = o_t * tanh(C_t)$$

- o_t es la output gate

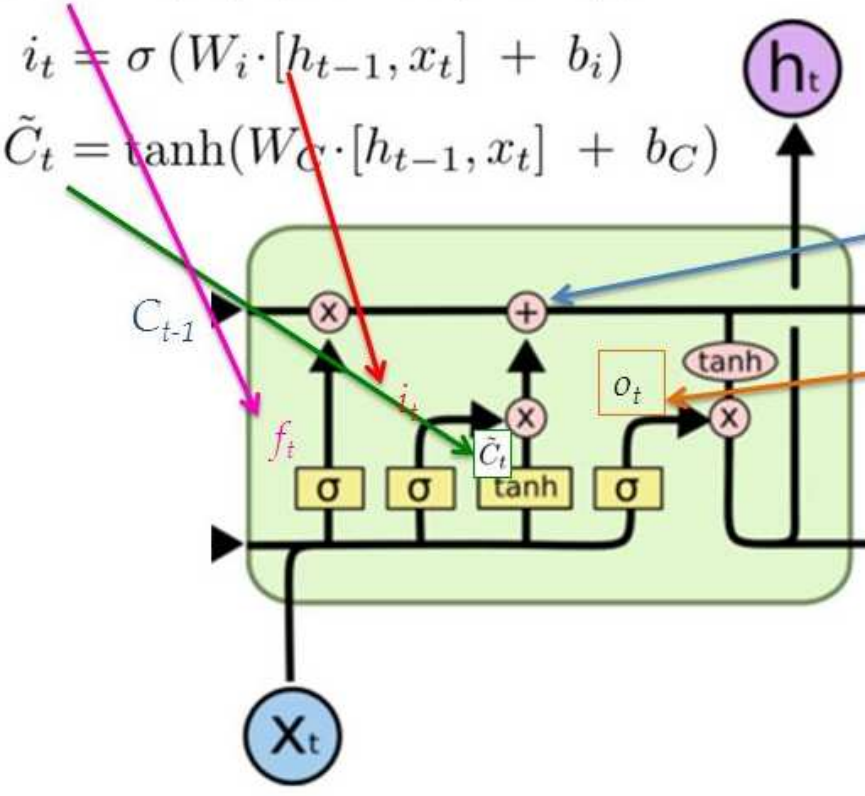
$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

Unidades LSTM

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

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

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



(2)

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

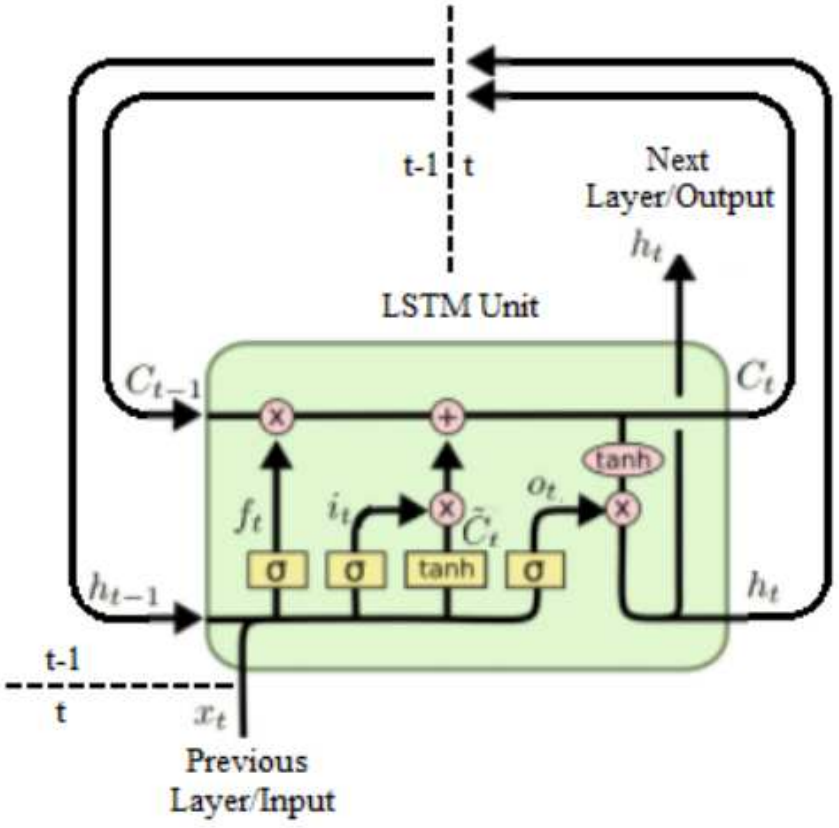
(3)

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

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

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Unidades LSTM



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Long Short-Term Memory Model

- El modelo LSTM ha sido aplicado con éxito en los sistemas predictivos basados en información secuencial.
- Fue presentado en 1997 en por Sepp Hochreiter y Jürgen Schmidhuber[4]
- Necesita ajustar W_C, W_f, W_i, W_o, b_C, b_f, b_i, b_o.
- ¿Son necesarios todos los parámetros? ¿No podemos simplificar la arquitectura de las unidades?

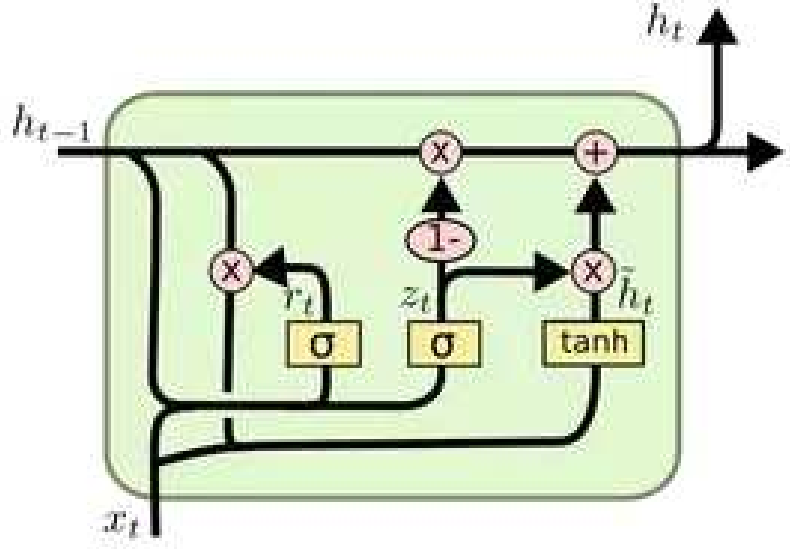
[4] S. Hochreiter and J. Schmidhuber. *Long Short-Term Memory*. Neural Computation 9(8): 1735-1780, 1997.

Gated Recurrent Units

- Presentadas por Kyunghyun Cho *et al.*[5].
- Su rendimiento en el modelado de música polifónica (entre otros campos de aplicación) es similar a las LSTM.
- Incluso parece que se comportan mejor que las LSTM para conjuntos pequeños.
- Pueden verse como una simplificación de las LSTM.

[5] https://arxiv.org/abs/1406.1078

Gated Recurrent Units



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

- z_t es la *update gate*.
- r_t es la *reset gate*.
- h_t es a la vez *estado y salida*.

Gated Recurrent Units

. . . y muchas, muchas, muchas variantes

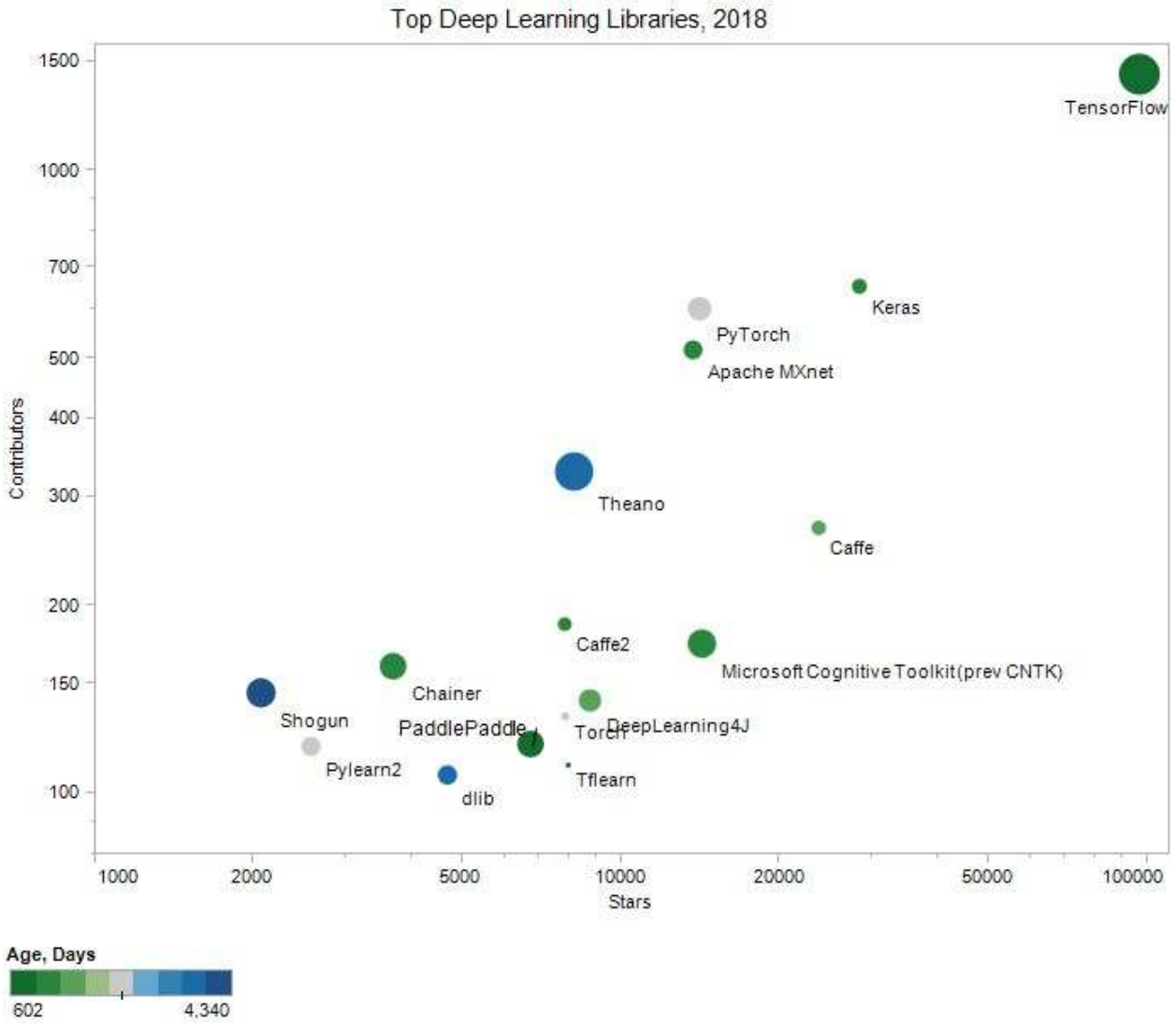
Análisis de secuencias

. . . sí, vale, pero ¿funciona?

Experimentos

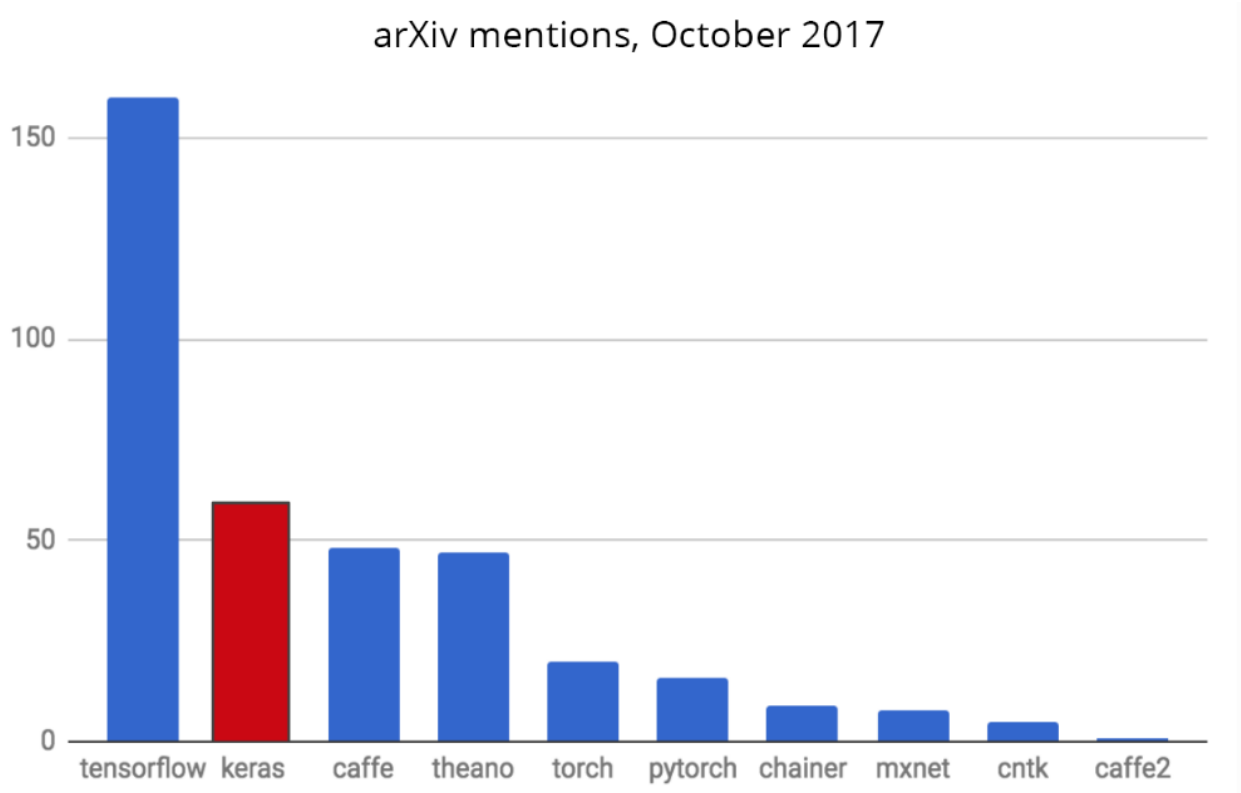
- *Keras* significa cuerno en griego. Es una referencia a una imagen literaria que hace referencia a los espíritus de los sueños (Oneiroi), entre los que están los que engañan a los hombres con falsas visiones y los anuncian un futuro que se hará realidad.
- Puede usar en *backend* Tensorflow (Google), Theano (Universidad de Montreal) o CNTK (Microsoft)
- Muy bien documentado
- *Hacer las cosas tan simples como se pueda*.

Top 16 open source DL libraries



https://www.kdnuggets.com/2018/04/top-16-open-source-deep-learning-libraries.html.

Keras en *arxiv* en 2017



Colab Research

https://colab.research.google.com/

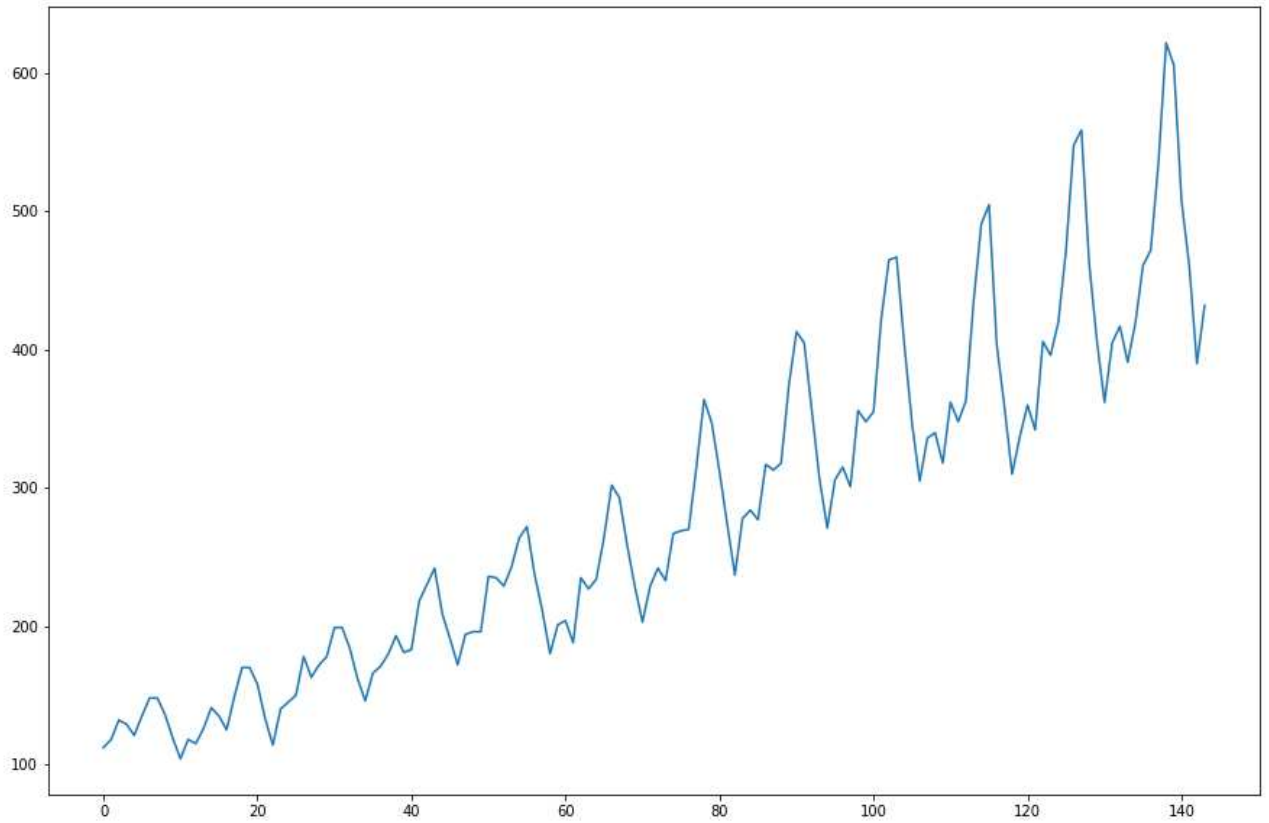
- Acceso gratuito con cuenta en Google.
- Acceso a NVIDIA GPU Tesla K80
  - 24 GB de memoria GDDR5
  - 480 GB/s de ancho de banda de memoria
  - 4992 núcleos de procesamiento paralelo CUDA

Un primer ejemplo

- Basado en el post https://tinyurl.com/y7pn2bb8
- Problema: Predecir el número de pasajeros internacionales a partir de un registro histórico de datos.
- El *dataset* consta de los registros mensuales de pasajeros (en unidades de mil) desde enero de 1949 a Diciembre de 1960 (12 años, 144 registros). Los valores oscilan entre 104 y 622.

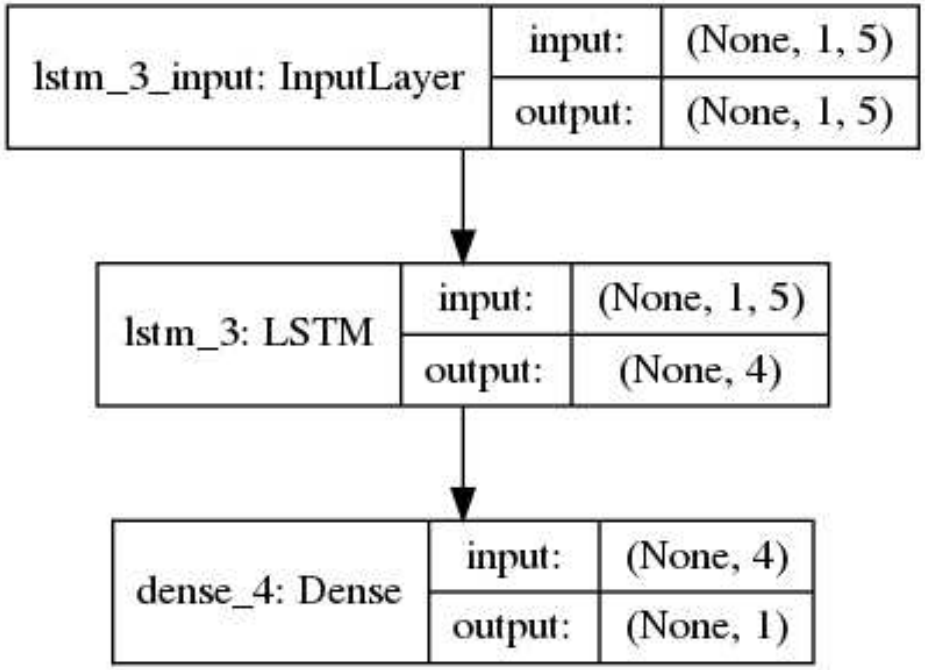
Un primer ejemplo

Evolución en doce años del número de pasajeros



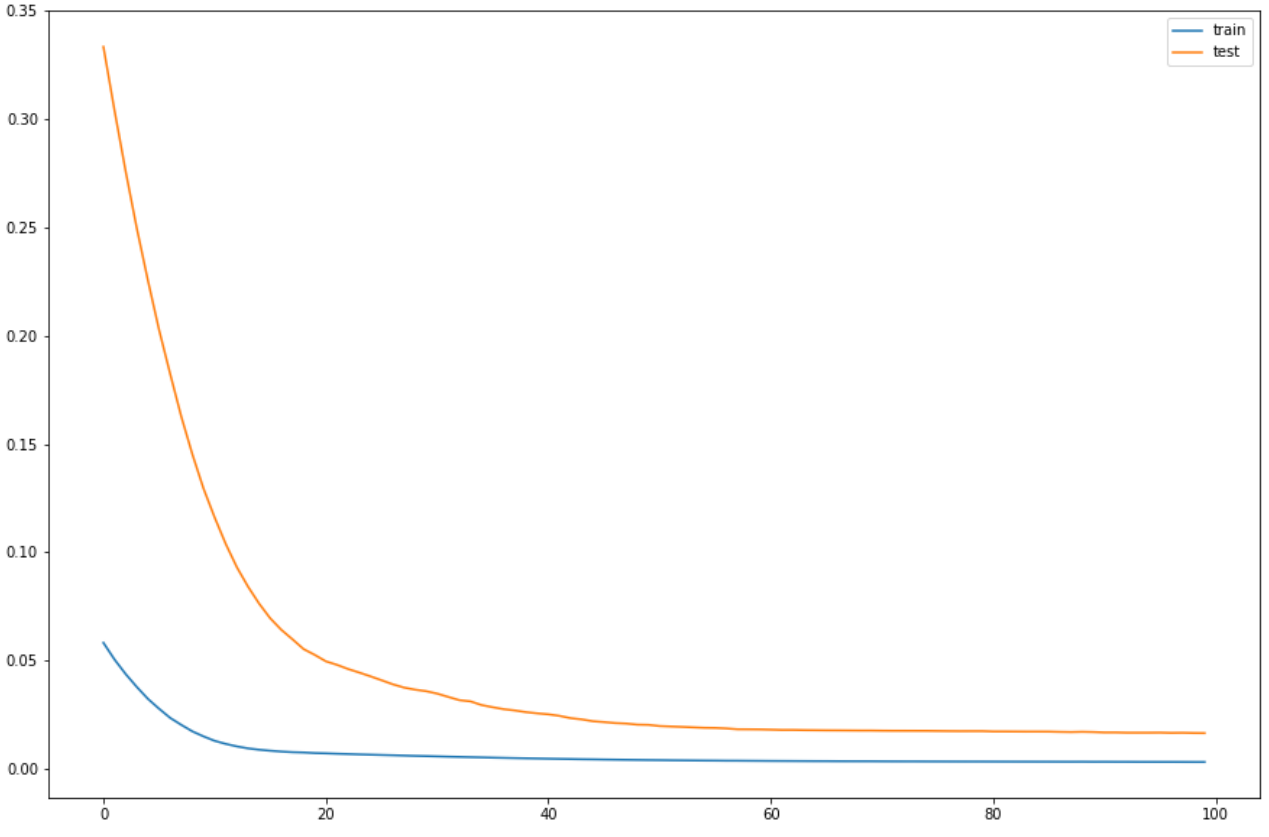
Un primer ejemplo

- Un modelo con una capa LSTM con 4 unidades y una capa densa con una neurona de salida.
- Fijamos el *look_back* a 5.
- Entrenamos con 100 *epoch* y 10 entradas por *batch*.
- Primeros dos tercios del dataset para entrenamiento y el último tercio para validación



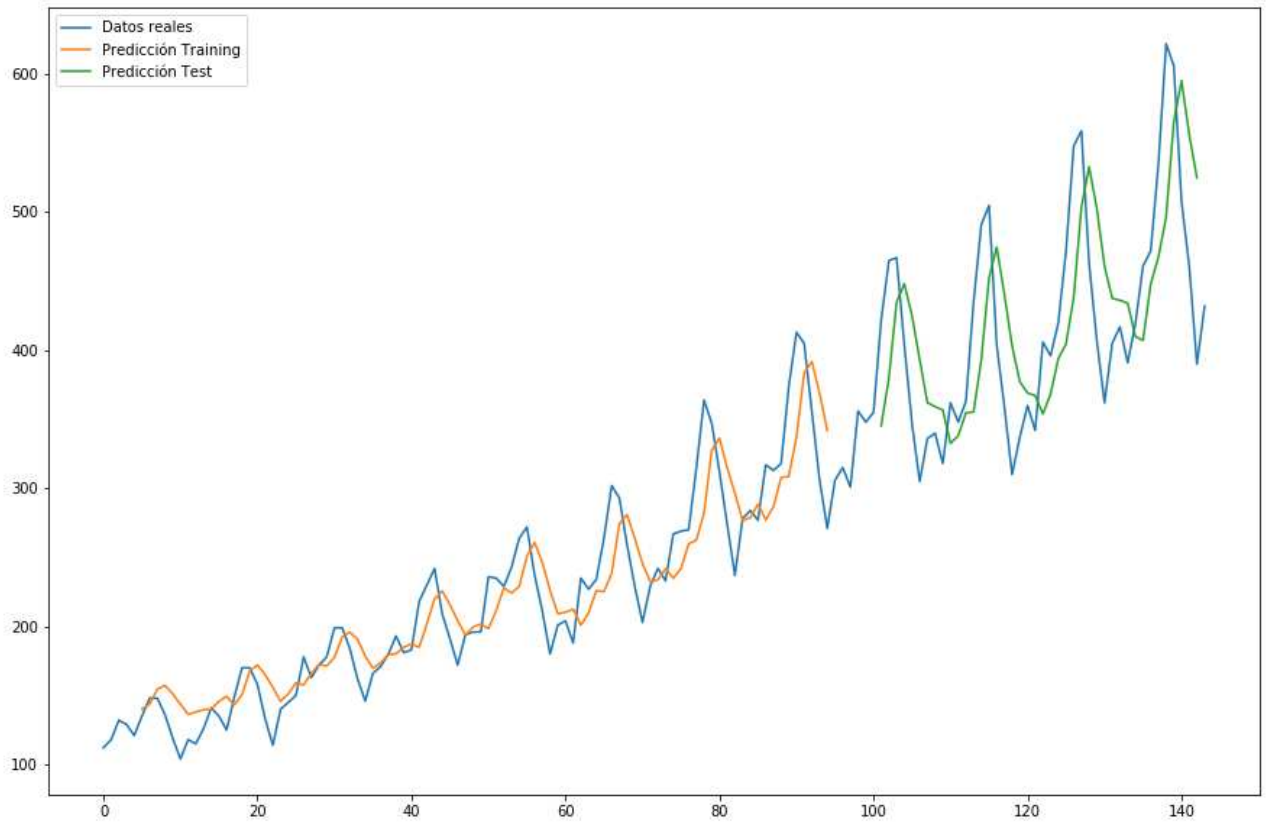
Un primer ejemplo

Usamos como *loss* el error cuadrático medio



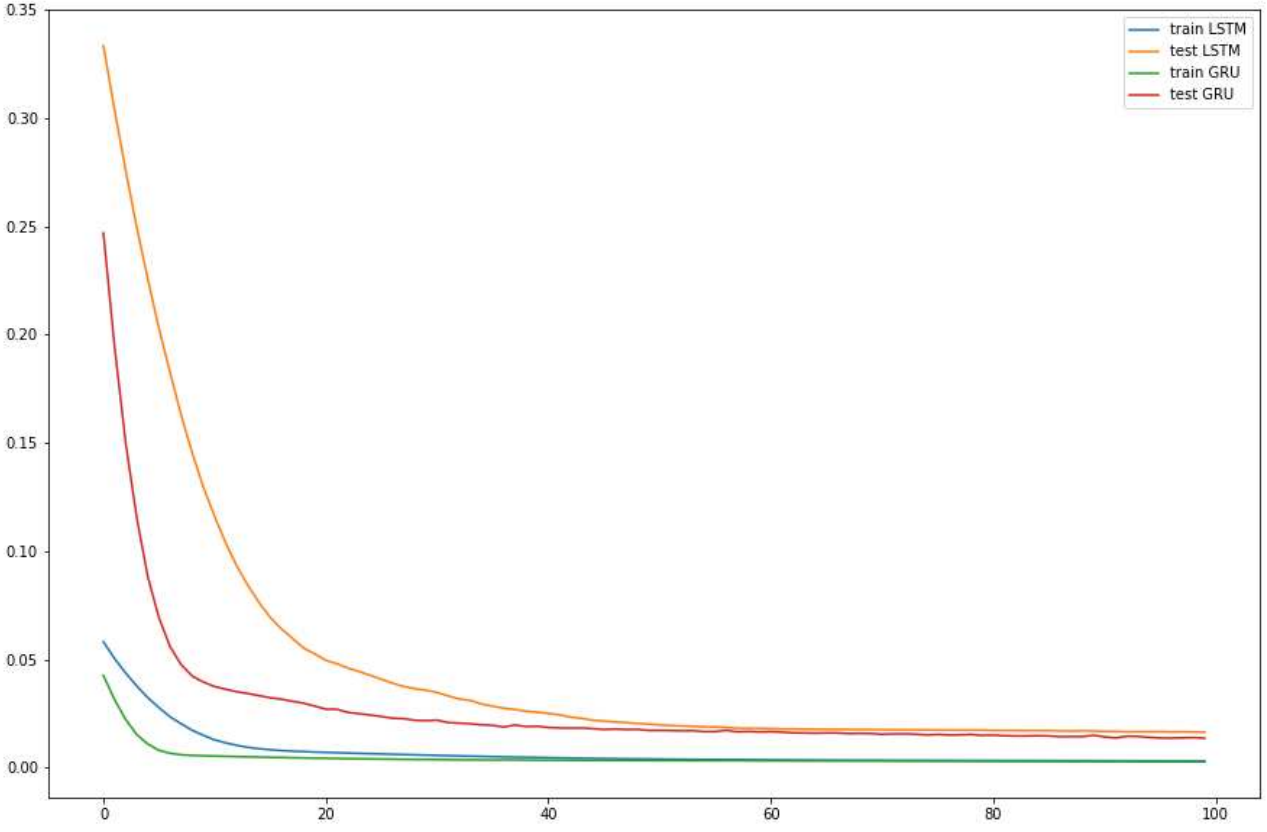
Un primer ejemplo

Train Score: 29.16 RMSE Test Score: 66.58 RMSE



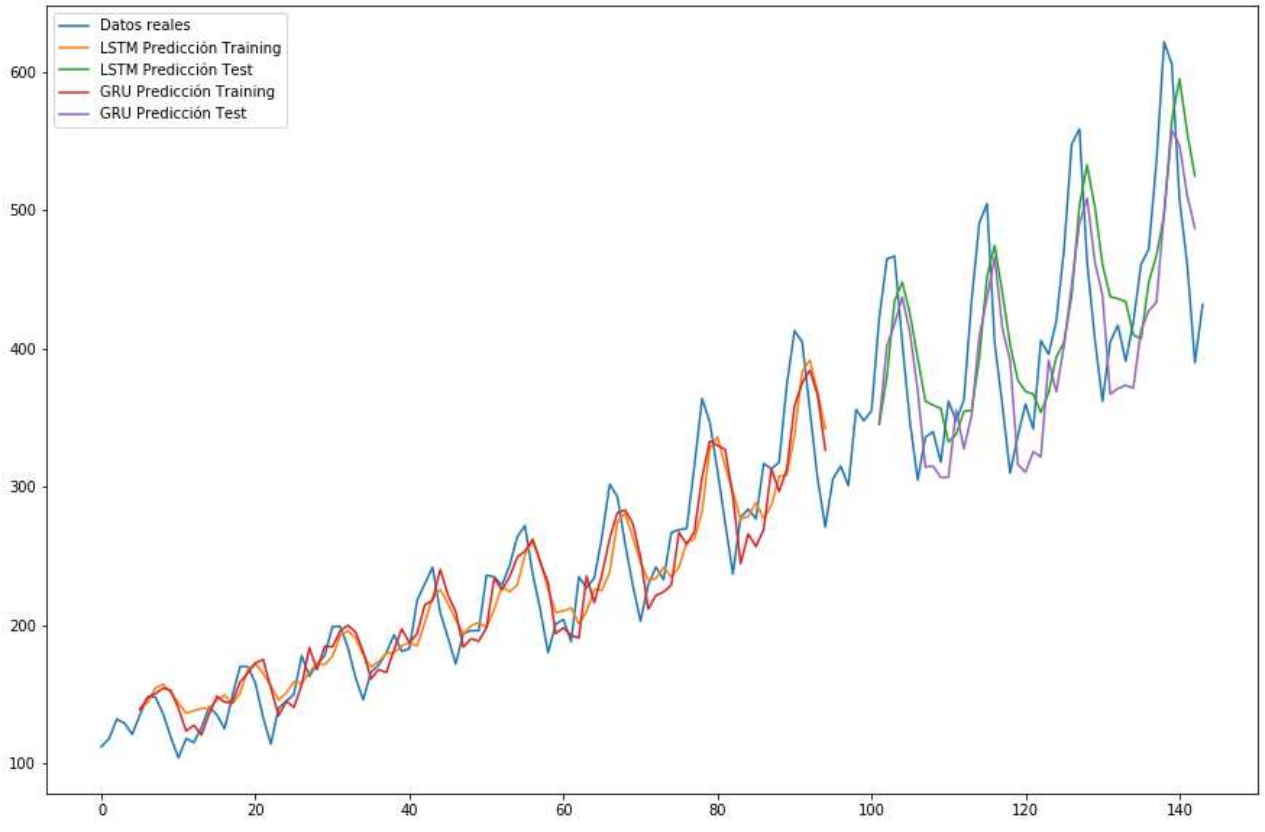
LSTM vs. GRU

Repetimos el mismo experimento con unidades GRU



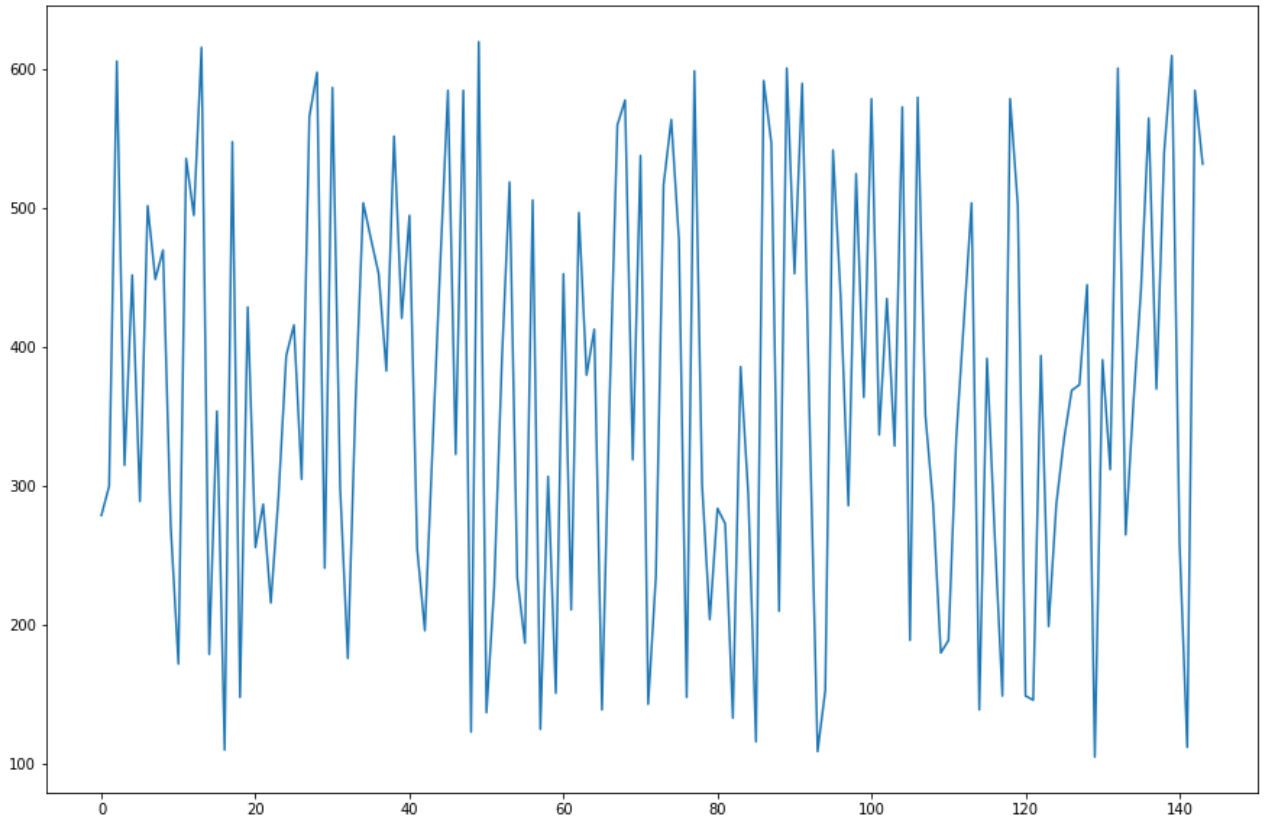
LSTM vs. GRU

LSTM : *TrainScore* : 29.16*RMSE* *TestScore* : 66.58*RMSE*
GRU : *TrainScore* : 27.32*RMSE* *TestScore* : 60.62



Datos aleatorios

Repetimos el experimento con 144 registros aleatorios en el mismo intervalo (104,622)



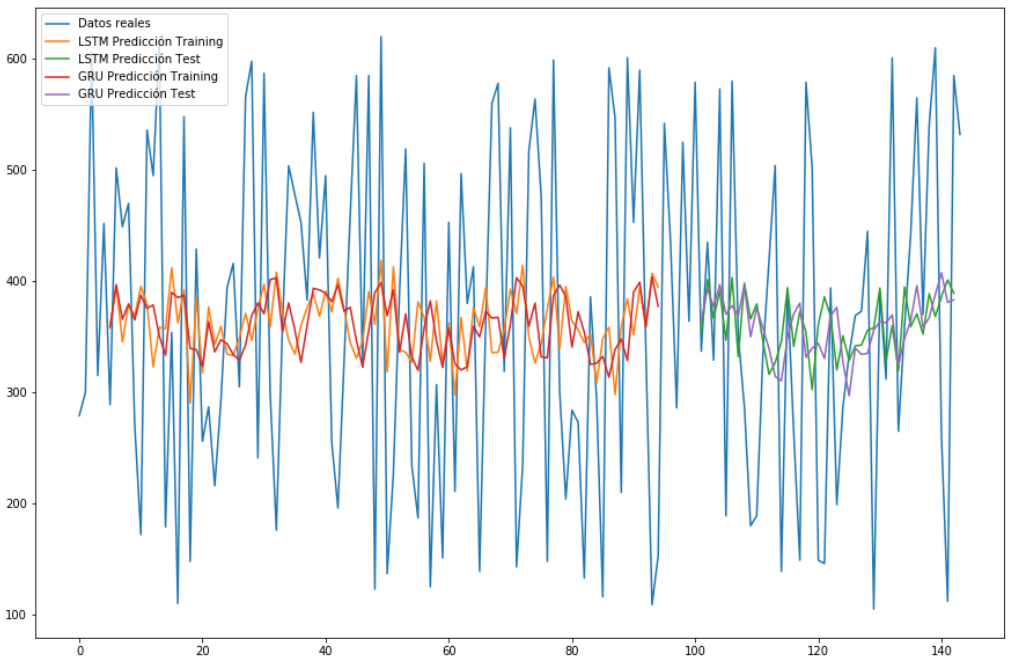
LSTM vs. GRU

- Datos estructurados:

  LSTM : *TrainScore* : 29.16*RMSE* *TestScore* : 66.58*RMSE*
  GRU : *TrainScore* : 27.32*RMSE* *TestScore* : 60.62

- Datos aleatorios:

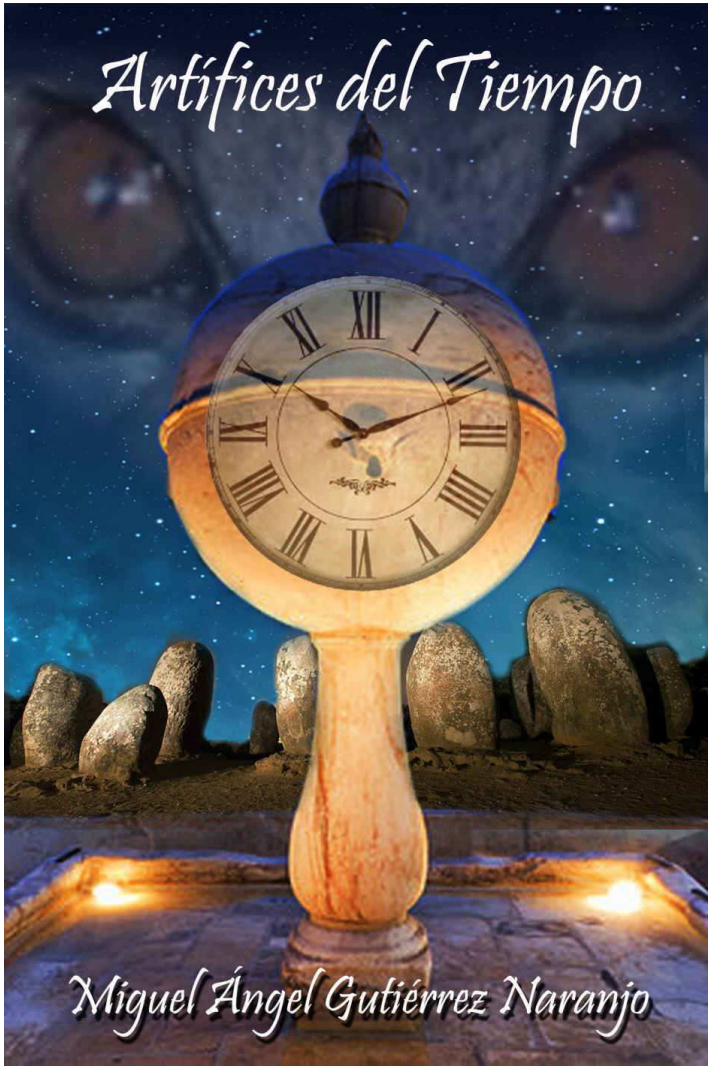
  LSTM : *TrainScore* : 158.43*RMSE* *TestScore* : 148.73*RMSE*
  GRU : *TrainScore* : 159.30*RMSE* *TestScore* : 146.85*RMSE*



Un ejemplo práctico

¿Puede aprender una red neuronal mi estilo literario?

- Autor:
  Miguel A. Gutiérrez Naranjo
- Año: 2015
- Web:
  https://tinyurl.com/ydcgg9fz
- *Dada una secuencia de 30 palabras ¿podemos predecir la siguiente?*



Un ejemplo práctico

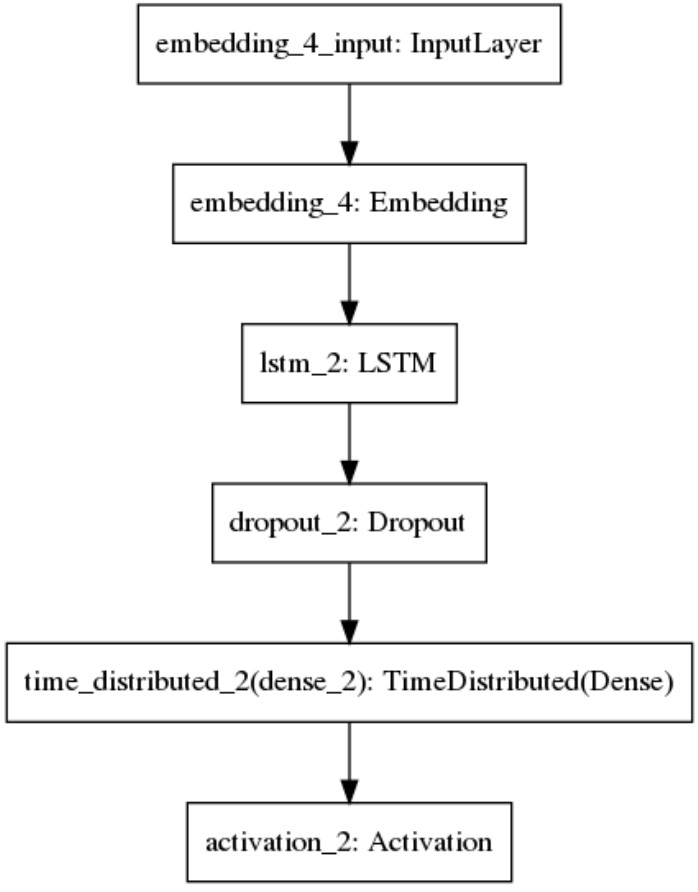
- Raw Dataset: 8883 primeras de la novela
- Preprocesado: Eliminación de títulos, signos de interrogación, guiones, ...
- Clean Dataset: 8749 palabras
  - Training: 7000 palabras
  - Validation: 1749 palabras

Seguimos http://adventuresinmachinelearning.com/keras-lstm-tutorial/

Un ejemplo práctico

¿Puede aprender una red neuronal mi estilo literario?

- *Embedding* en un espacio de dim 500.
- Capa *LSTM* con 500 unidades
- *Dropout* con factor 0.5
- Capa *Time_distributed* que crea una capa densa por cada *time steps*.
- *Softmax*

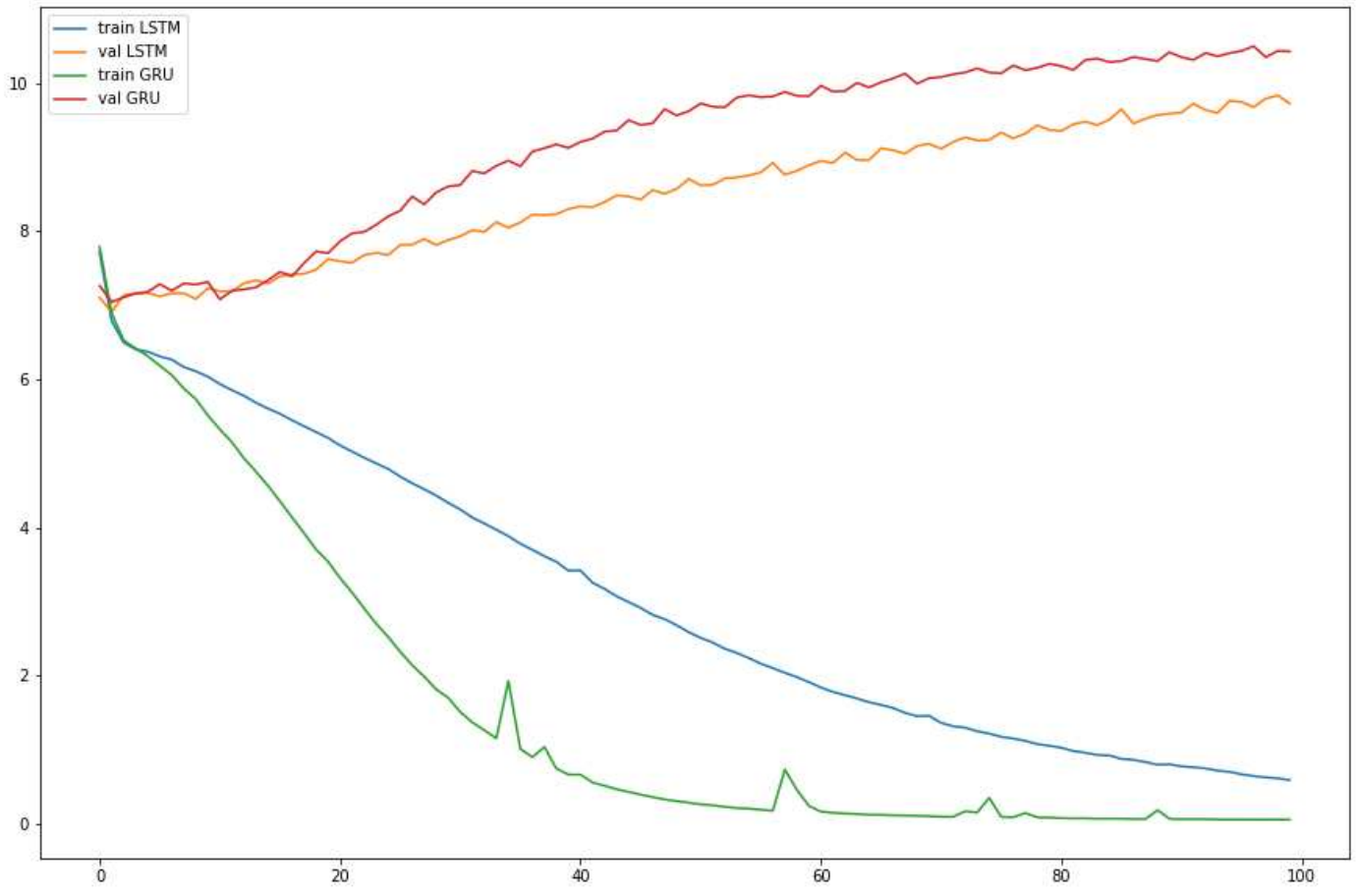


Un ejemplo práctico

Parámetros:

- Aprender a partir de secuencias de 30 palabras *(num_steps)*.
- 20 instancias en cada batch
- 100 epoch
- Experimentos con LSTM y GRU.

Comparativa LSTM y GRU



Muchas Gracias

Miguel A. Gutiérrez Naranjo

http://www.cs.us.es/~naranjo/