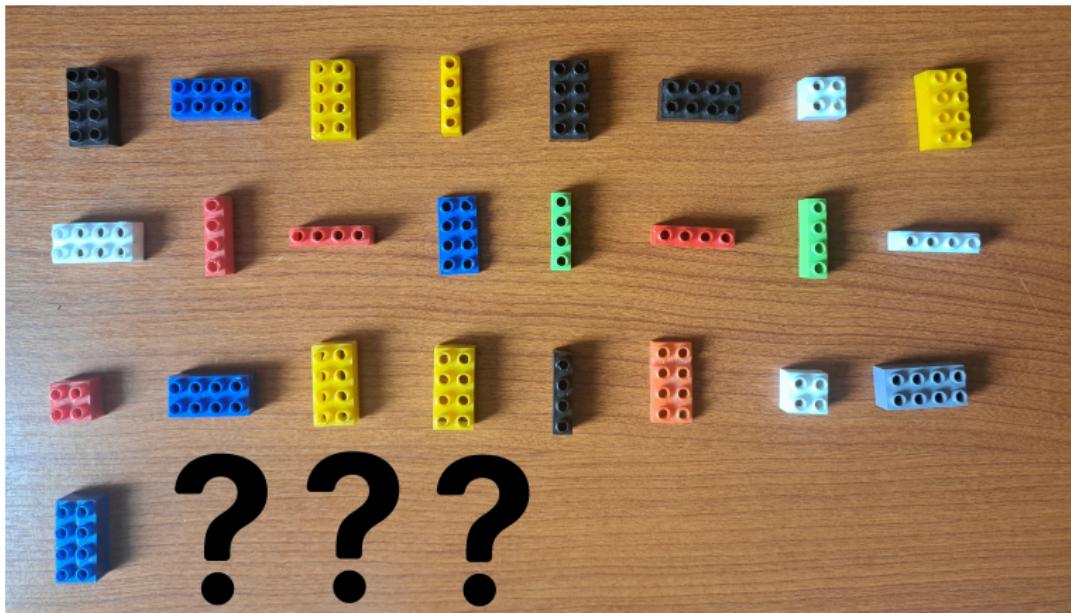


Secuencias

- Datos ordenados: históricos del clima, mercado inmobiliario, trayectoria de un cohete, lenguaje natural, etc.

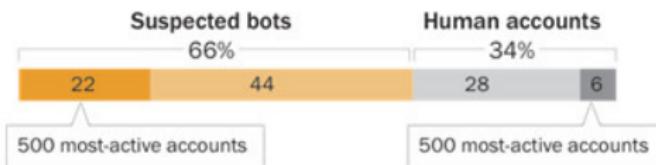


Modelando secuencias: ejemplos de tareas

- *Detección de evento*: “Ok Google”, “Hola Siri” ó “Hey Alexa”
- *Clasificación*: ¿robot o humano? dada la actividad de un visitante a un sitio web, ¿me gusta o no? IMDB Movie Reviews,...

The most-active Twitter bots produce a large share of the links to popular news and current events websites

Share of tweeted links to popular news and current events websites posted by ...



Source: Analysis of 379,841 tweeted links to 925 popular news and current events websites collected over the time period July 27–Sept. 11, 2017.
“Bots in the Twittersphere”

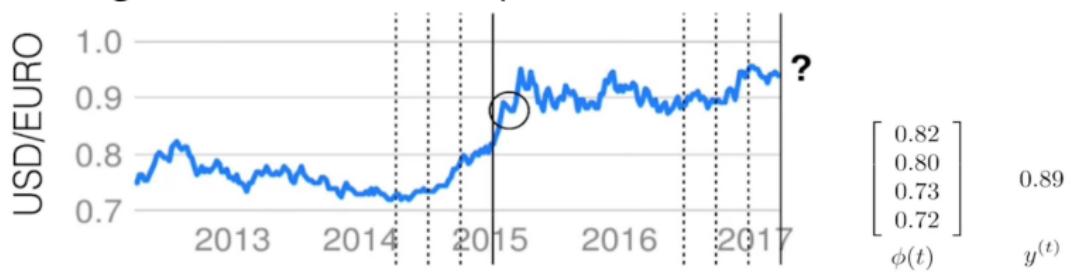
PEW RESEARCH CENTER

Modelando secuencias: ejemplos de tareas

- *Detección de anomalías:* detección de cosas inusuales



- *Forecasting* de una serie de tiempo



Modelando secuencias: ejemplos de tareas

- *Procesamiento* de lenguaje natural

This course has **been a tremendous** ...

$$\begin{array}{c} \mathbf{a} \\ \left[\begin{array}{c} 1 \\ 0 \\ \vdots \\ 0 \end{array} \right] \\ \text{tremendous} \\ \mathbf{been} \\ \left[\begin{array}{c} 0 \\ 1 \\ \vdots \\ 0 \end{array} \right] \\ \phi(t) \quad y^{(t)} \end{array}$$

This course has been **a tremendous** ...

$$\begin{array}{c} \mathbf{tremendous} \\ \left[\begin{array}{c} 0 \\ \vdots \\ 1 \\ 0 \end{array} \right] \\ ? \\ \mathbf{a} \\ \left[\begin{array}{c} 1 \\ 0 \\ \vdots \\ 0 \end{array} \right] \\ \phi(t) \quad y^{(t)} \end{array}$$

- *Generación* de música e imágenes
- *Traducción* de un lenguaje natural a otro
- etc.: subtitulado de imágenes, la síntesis de audio, análisis de vídeo, entablar un diálogo,...

Modelando secuencias: codificando todo

Input:

- FFNN: vector de características

$$\mathbf{x} \in \mathbb{R}^d.$$

- Ahora: lista ordenada de vectores de características

$$\mathbf{x}_1, \dots, \mathbf{x}_T \quad \text{donde} \quad \mathbf{x}_t \in \mathbb{R}^d$$

$$y \quad t = 1, 2, \dots, T$$

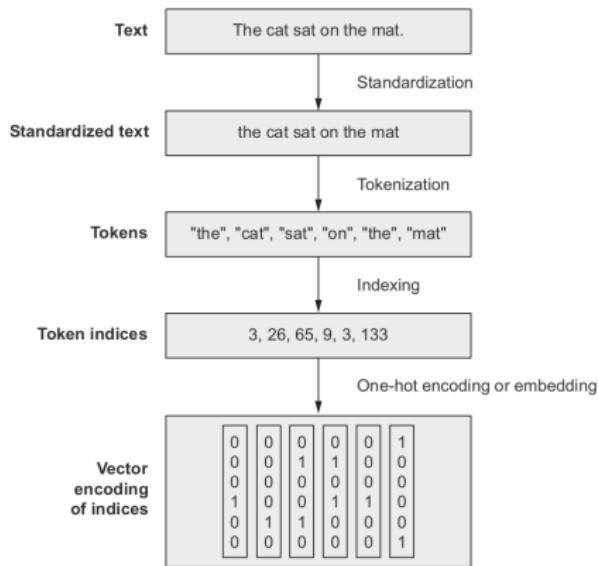
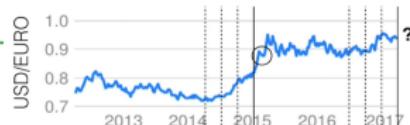


Figure 11.1 From raw text to vectors

Modelando secuencias: descripción general

Modelos SIN memoria para secuencias 1D

El modelo autorregresivo de orden p , AR(p)¹



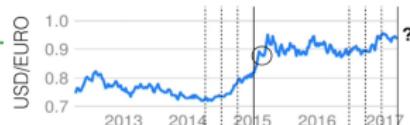
$$X_t = a_1 X_{t-1} + a_2 X_{t-2} + \cdots + a_p X_{t-p} + W_t.$$

¹El modelo puede recordar las últimas p observaciones, pero no aprendió la dinámica.

Modelando secuencias: descripción general

Modelos SIN memoria para secuencias 1D

El modelo autorregresivo de orden p , AR(p)¹



$$X_t = a_1 X_{t-1} + a_2 X_{t-2} + \cdots + a_p X_{t-p} + W_t.$$

El modelo autorregresivo con entradas exógenas, ARX(p)

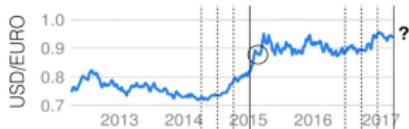
$$X_t = a_1 X_{t-1} + \cdots + a_p X_{t-p} + b_0 u_t + b_1 u_{t-1} + \cdots + b_q u_{t-q}$$

¹El modelo puede recordar las últimas p observaciones, pero no aprendió la dinámica.

Modelando secuencias: descripción general

Modelos SIN memoria para secuencias 1D

El modelo autorregresivo de orden p , AR(p)¹



$$X_t = a_1 X_{t-1} + a_2 X_{t-2} + \cdots + a_p X_{t-p} + W_t.$$

El modelo autorregresivo con entradas exógenas, ARX(p)

$$X_t = a_1 X_{t-1} + \cdots + a_p X_{t-p} + b_0 u_t + b_1 u_{t-1} + \cdots + b_q u_{t-q}$$

El modelo autorregresivo de media móvil, ARMA(p,q)

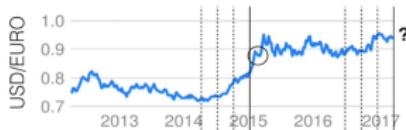
$$X_t = a_1 X_{t-1} + \cdots + a_p X_{t-p} + W_t + b_1 W_{t-1} + \cdots + b_q W_{t-q}$$

¹El modelo puede recordar las últimas p observaciones, pero no aprendió la dinámica.

Modelando secuencias: descripción general

Modelos SIN memoria para secuencias 1D

El modelo autorregresivo de orden p , AR(p)¹



$$X_t = a_1 X_{t-1} + a_2 X_{t-2} + \cdots + a_p X_{t-p} + W_t.$$

El modelo autorregresivo con entradas exógenas, ARX(p)

$$X_t = a_1 X_{t-1} + \cdots + a_p X_{t-p} + b_0 u_t + b_1 u_{t-1} + \cdots + b_q u_{t-q}$$

El modelo autorregresivo de media móvil, ARMA(p,q)

$$X_t = a_1 X_{t-1} + \cdots + a_p X_{t-p} + W_t + b_1 W_{t-1} + \cdots + b_q W_{t-q}$$

El modelo autorregresivo integrado de media móvil, ARIMA(p,d,q)

...

¹El modelo puede recordar las últimas p observaciones, pero no aprendió la dinámica.

Modelando secuencias: descripción general

Modelos CON memoria para secuencias: *hidden state*

Hay un estado oculto \mathbf{h}_t ,

- Tiene una dinámica interna

Modelando secuencias: descripción general

Modelos CON memoria para secuencias: *hidden state*

Hay un estado oculto \mathbf{h}_t ,

- Tiene una dinámica interna
- El *hidden state* hace un resumen/sumario/compendio de las observaciones pasadas

Modelando secuencias: descripción general

Modelos CON memoria para secuencias: *hidden state*

Hay un estado oculto \mathbf{h}_t ,

- Tiene una dinámica interna
- El *hidden state* hace un resumen/sumario/compendio de las observaciones pasadas
- Se actualiza con la predicción

Modelando secuencias: descripción general

Modelos CON memoria para secuencias: *hidden state*

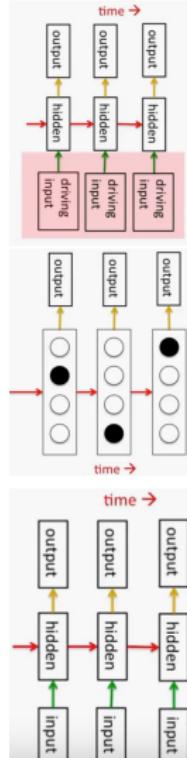
Hay un estado oculto \mathbf{h}_t ,

- Tiene una dinámica interna
- El *hidden state* hace un resumen/sumario/compendio de las observaciones pasadas
- Se actualiza con la predicción
- Se asume que la dinámica que genera las observaciones es estacionaria, para el entrenamiento a partir de datos históricos, normalmente se crean ejemplos muestreando ventanas aleatoriamente.

Modelando secuencias: descripción general

Modelos CON memoria para secuencias: Ejemplos

1. Sistemas dinámicos lineales (LDS): estados ocultos con dinámica lineal, con ruido Gaussiano.
2. Modelo oculto de Márkov (HSMM): dinámica controlada por una matriz de transición entre los estados ocultos. Primeros sistemas prácticos de reconocimiento de voz en los años 1980 y 1990.
3. Redes Neuronales Recurrentes (RNN): estado oculto distribuido con dinámica no lineal. Con la cantidad suficiente de memoria y de datos. RNN pueden calcular cualquier cosa que sea calculada por tu computadora



RNN arquitectura

1. Una salida en cada paso de tiempo

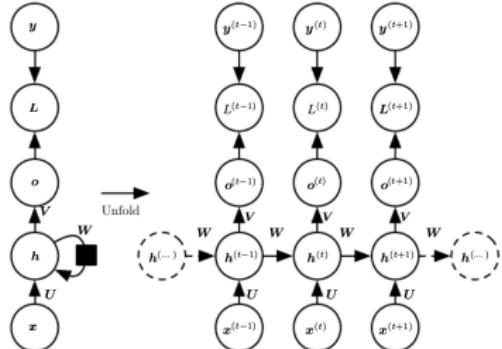


Figure 10.3: The computational graph to compute the training loss of a recurrent network

$$h^{(t)} = \tanh(b + Wh^{(t-1)} + Ux^{(t)}), \quad o^{(t)} = c + Vh^{(t)}$$

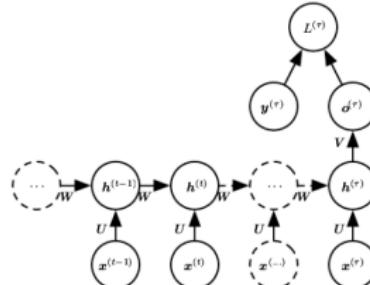


Figure 10.5: Time-unfolded recurrent neural network with a single output at the end

Entrenamiento

- Podemos pensar una RNN como FFNN con capas con pesos compartidos.
- Los pesos compartidos se introducen a través de restricciones lineales en los pesos.
- Calculamos los gradientes como antes y luego los modificamos para que satisfagan las restricciones.

Entrenamiento

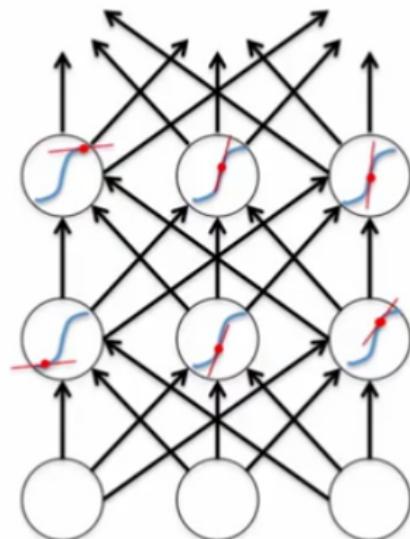
- Podemos pensar una RNN como FFNN con capas con pesos compartidos.
- Los pesos compartidos se introducen a través de restricciones lineales en los pesos.
- Calculamos los gradientes como antes y luego los modificamos para que satisfagan las restricciones.
- El algoritmo de entrenamiento puede pensarse en el dominio temporal como:
 1. El *forward pass* apila las actividades de todas las unidades a cada paso de tiempo.
 2. El *backward pass* quita las actividades de la pila para calcular el error de las derivadas en cada paso de tiempo.
 3. Luego del backward pass sumamos todas las derivadas en todos los tiempos diferentes para cada peso.

Problema de los gradientes: el *backward pass* es linear

- En el forward pass usamos funciones que aplastan (como la función logística) lo que previene la explosión de los vectores de actividades

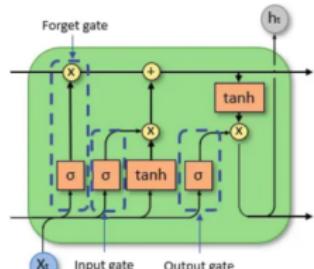
Problema de los gradientes: el *backward pass* es linear

- En el forward pass usamos funciones que aplastan (como la función logística) lo que previene la explosión de los vectores de actividades
- El backward pass es completamente lineal, si los errores en la capa final se duplican, todos los errores en las derivadas se duplican.
 - Si los pesos son pequeños, los gradientes decaen exponencialmente.
 - Si los pesos son grandes, los gradientes crecen exponencialmente.
 - Este efecto puede verse en una RNN entrenada en una secuencia larga (100 pasos de tiempo).
 - FFNN hacen frente a estos errores porque sólo tienen algunas capas ocultas.



LSTM

- Hochreiter & Schmidhuber (1997) resolvieron el problema de hacer que una RNN recuerde cosas por mucho tiempo (cientos de pasos de tiempo).
- Diseñaron una celda de memoria usando unidades logísticas y lineales con interacciones multiplicativas.
- La información accede a la celda cuando la *write gate* está en *on*.
- La información permanece en la celda en tanto la *keep gate* está en *on*.
- Se puede usar backpropagation a través de este circuito porque las funciones logísticas tienen buenas derivadas.



Chris Bishop wrote a terrific textbook on neural networks in 1995 and has a deep knowledge of the field and its core ideas. His many years of experience in explaining neural networks have made him extremely skilled at presenting complicated ideas in the simplest possible way and it is a delight to see how this is applied to the revolutionary new developments in the field. Geoffrey Hinton

With the recent explosion of deep learning and AI as a research topic, and the quickly growing importance of AI applications, a modern textbook on the topic was badly needed. "The New Bishop" masterfully fills the gap, combining a solid theoretical foundation and a wealth of practical examples, as well as how to apply all of this to various application areas. Yann LeCun

This excellent and very educational book will bring the reader up to date with the main concepts and advances in deep learning with a solid anchoring in probability. These concepts are powering current industrial AI systems and are likely to form the basis of further advances towards artificial general intelligence. Yoshua Bengio



Gating

- En una RNN de una sola capa el hidden state siempre se actualiza con el resultado de la operación

$$\mathbf{h}^{(t)} = \tanh(\mathbf{b} + \mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)})$$

- Introducimos un vector de compuerta de la misma dimensión que el hidden state, que determina la cantidad de información que se va a sobrescribir en el siguiente estado. Una RNN con compuerta de una sola capa se puede escribir como

$$\mathbf{g}_t = \text{sigmoid}(\mathbf{W}^g \mathbf{h}^{(t-1)} + \mathbf{W}^x \mathbf{x}^{(t)})$$

$$\mathbf{h}^{(t)} = (1 - g_t) \odot \mathbf{h}^{(t-1)} + g_t \odot \tanh(\mathbf{b} + \mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)})$$

Gating: LSTM

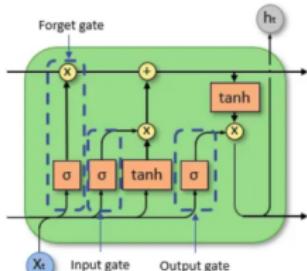
$$\mathbf{f}_t = \text{sigmoid} \left(W^{f,h} \mathbf{h}_{t-1} + W^{f,x} \mathbf{x}_t \right) \quad \text{forget gate}$$

$$\mathbf{i}_t = \text{sigmoid} \left(W^{i,h} \mathbf{h}_{t-1} + W^{i,x} \mathbf{x}_t \right) \quad \text{input gate}$$

$$\mathbf{o}_t = \text{sigmoid} \left(W^{o,h} \mathbf{h}_{t-1} + W^{o,x} \mathbf{x}_t \right) \quad \text{output gate}$$

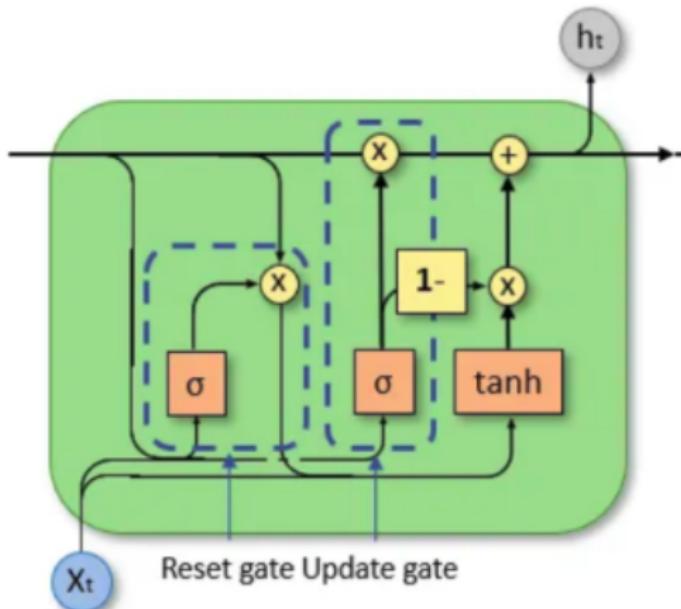
$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh \left(W^{c,h} \mathbf{h}_{t-1} + W^{c,x} \mathbf{x}_t \right) \quad \text{memory}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh (\mathbf{c}_t) \quad \text{visible state}$$



En resumen, una LSTM cell permite que la información pasada se inyecte en un momento posterior, lo que combate el problema de los gradientes.

Gating: GRU



Recurrent dropout

Dropout

- Es equivalente a usar diferentes arquitecturas para cada caso de entrenamiento con pesos compartidos. Cada modelo es fuertemente regularizado (mejor que las regularizaciones de L_2 y L_1).
- En el momento del test se usan todos los nodos pero se reescalas las activaciones.
- Dropout + Early Stopping mejora la generalización. El entrenamiento con dropout es más lento por el apagado de las neuronas, especialmente en redes profundas y se necesita más epochs para converger.
- Si tu deep neural network no está overfiteando se recomienda usar una más grande y luego usar dropout.

Recurrent dropout

- Se debe aplicar la misma máscara de dropout en cada paso de tiempo, en lugar de utilizar una máscara que varíe aleatoriamente de un paso de tiempo a otro.

Stacking recurrent layers

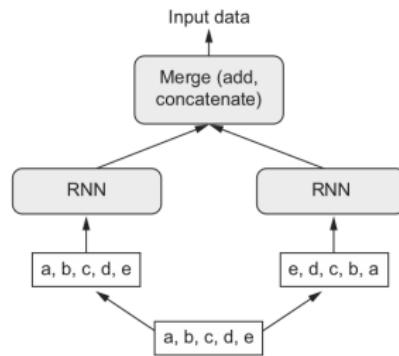
- Para mejorar la capacidad expresiva de la red. Hay que salir del *under capacity*, ir al límite del overfitting y luego regularizar.

Stacking recurrent layers

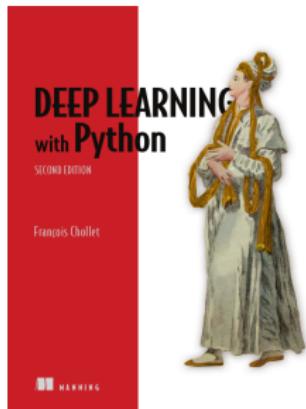
- Para mejorar la capacidad expresiva de la red. Hay que salir del *under capacity*, ir al límite del overfitting y luego regularizar.
- Cómo: aumentar el número de unidades en cada capa o sumar más capas. Ej. hasta no hace mucho tiempo, el algoritmo de Google Translate funcionaba con una pila de siete grandes capas LSTM.

Bidirectional recurrent layer

- Frecuentemente usadas en procesamiento de lenguaje natural donde la importancia de una palabra en la comprensión de una oración no depende usualmente del orden.
- Las representaciones que las RNNs producen son orden dependiente, .
- Una RNN bidireccional explota usa dos RNN regulares cada una procesando la secuencia en una dirección (cronológica y antícronológicamente), y combinas sus representaciones. Encuentra patrones que pueden ser pasados por alto por las unidireccionales.



Bibliografía



Dive into Deep Learning

ASTON ZHANG, ZACHARY C. LIPTON, MU LI, AND ALEXANDER J.
SMOLA

Deep Learning

Ian Goodfellow
Yoshua Bengio
Aaron Courville