PONTIFICIA
UNIVERSIDAD
CATÓLICA DE
VALPARAÍSO

Introducción al Deep Learning

Redes Neuronales Recurrentes y LSTM

Dr. Ing. Gabriel Hermosilla Vigneau

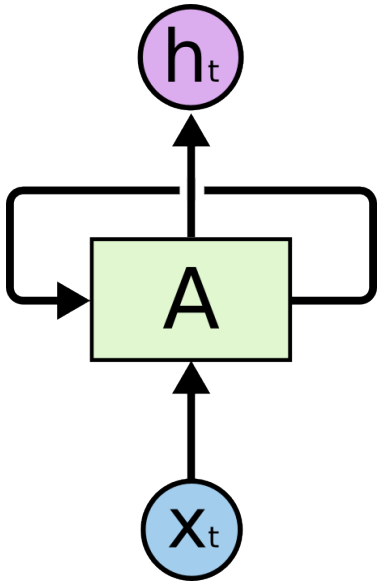
Redes neuronales recurrentes

- Los seres humanos no comienzan a pensar desde cero cada segundo. Al leer se comprende cada palabra basándose en la comprensión de las palabras anteriores, es decir los pensamientos tienen persistencia.
- Las redes neuronales tradicionales no pueden hacer esto, y parece una deficiencia importante. Por ejemplo, imagine que desea clasificar qué tipo de evento está ocurriendo en cada punto de una película. No está claro cómo una red neuronal tradicional podría usar su razonamiento sobre eventos anteriores en la película para informar a los posteriores.
- Las redes neuronales recurrentes abordan este problema. Son redes con bucles, permitiendo que la información persista.

*Construida con la información del blog de **Christopher Olah**

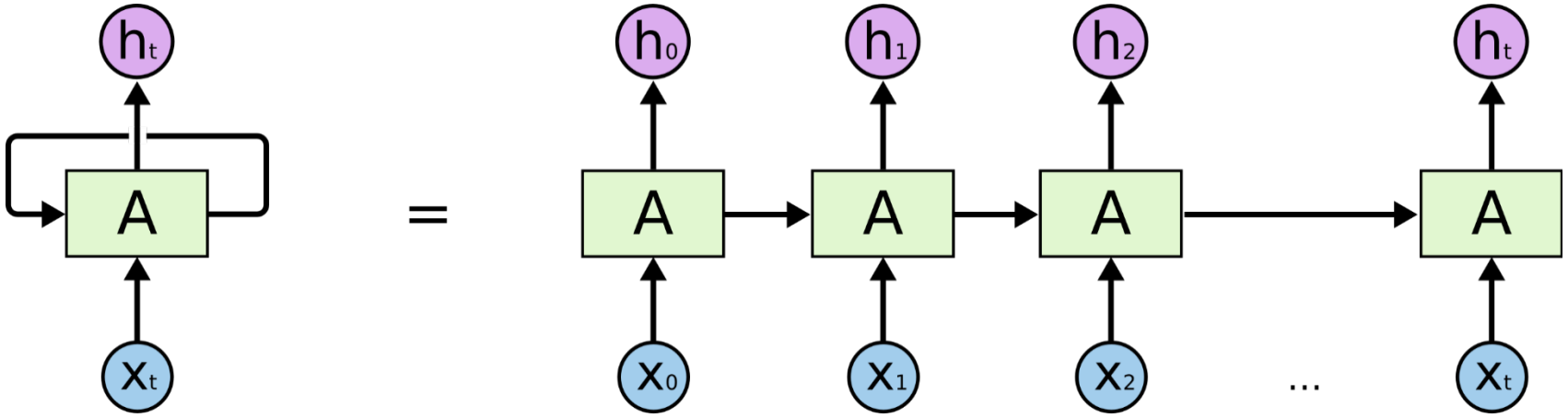
Redes neuronales recurrentes

- En el siguiente esquema, A mira alguna entrada Xt y obtiene un valor de salida ht. Existe un bucle que permite que la información pase de un paso a la siguiente red.



Redes neuronales recurrentes

- Estos bucles hacen que las redes neuronales recurrentes sean un poco diferentes. Podemos considerar una red neuronal recurrente como múltiples copias de la misma red, cada una de ellas le entrega un mensaje a su sucesor. Esto se observa en el siguiente esquema:



Redes neuronales recurrentes

- Esta naturaleza de cadena revela que las redes neuronales recurrentes están íntimamente relacionadas con secuencias y listas. Son la arquitectura natural de la red neuronal que se utiliza para dichos datos.
- En los últimos años, ha habido un éxito increíble al aplicar RNN a una variedad de problemas: reconocimiento de voz, modelado del lenguaje, traducción, subtítulos de imágenes, etc.
- Esencial para estos éxitos es el uso de las redes "LSTM", un tipo muy especial de red neuronal recurrente que funciona para muchas tareas, mucho mejor que la versión estándar. Casi todos los resultados interesantes basados en redes neuronales recurrentes se logran con ellos.

Redes neuronales recurrentes

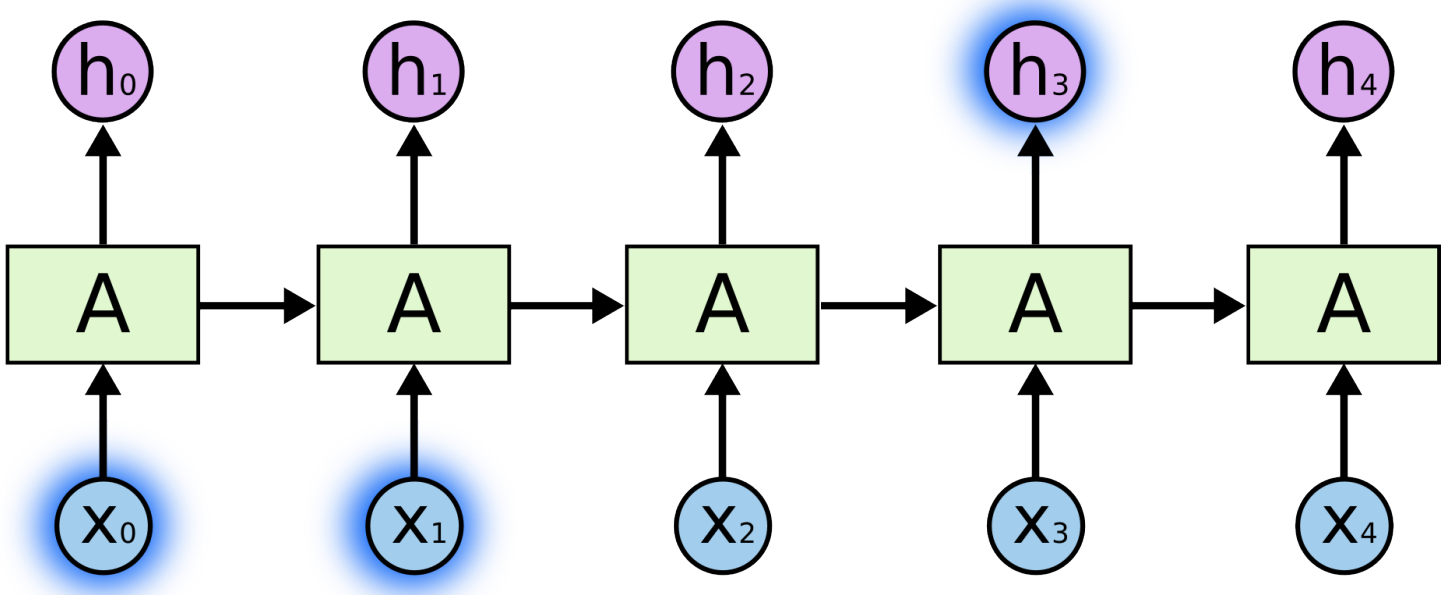
El problema de las dependencias a largo plazo

- Una de las apelaciones de las RNN es la idea de que podrían ser capaces de conectar información previa a la tarea presente, como el uso de frames de video anteriores podría informar la comprensión del cuadro actual. Si los RNN pudieran hacer esto, serían extremadamente útiles. ¿Es posible realizar esto?
- A veces, solo necesitamos mirar información reciente para realizar la tarea presente. Por ejemplo, considere un modelo de lenguaje que intente predecir la siguiente palabra basándose en los anteriores. Si estamos tratando de predecir la última palabra en "las nubes están en el ***cielo*** ", no necesitamos ningún otro contexto, es bastante obvio que la próxima palabra será cielo. En tales casos, donde la brecha entre la información relevante y el lugar que se necesita es pequeña, los RNN pueden aprender a usar la información pasada.

Redes neuronales recurrentes

El problema de las dependencias a largo plazo

- Ejemplo de dependencias cortas:



Redes neuronales recurrentes

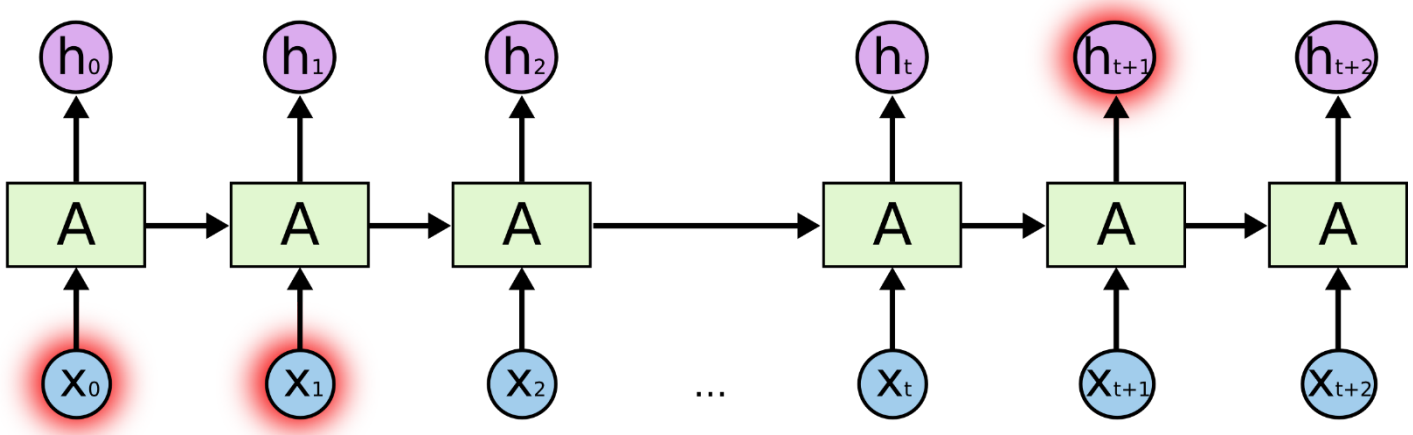
El problema de las dependencias a largo plazo

- También hay casos en los que necesitamos más contexto. Considere intentar predecir la última palabra en el texto "Crecí en Francia ... hablo ***francés con*** **fluidez**". La información reciente sugiere que la siguiente palabra probablemente sea el nombre de un idioma, pero si queremos restringir qué idioma, necesitamos el contexto de **Francia**, desde más atrás. Es totalmente posible que la brecha entre la información relevante y el punto donde se necesita sea muy grande.
- Desafortunadamente, a medida que aumenta la brecha, los RNN no pueden aprender a conectar la información.

Redes neuronales recurrentes

El problema de las dependencias a largo plazo

- Ejemplo de dependencias largas:



Redes neuronales recurrentes

El problema de las dependencias a largo plazo

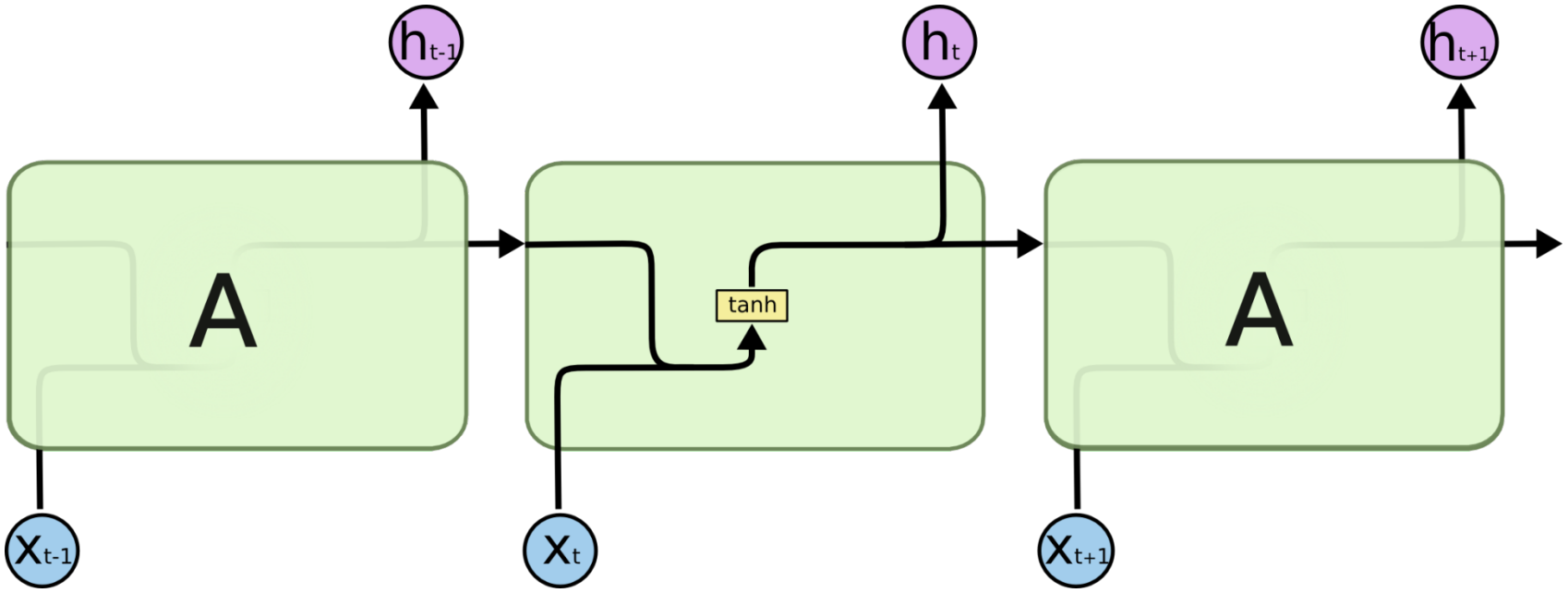
- En teoría, las RNN son absolutamente capaces de manejar tales "dependencias a largo plazo". Un ser humano podría elegir cuidadosamente los parámetros para resolver **problemas de juguetes (sencillos)** de esta forma. Lamentablemente, en la práctica, los RNN no parecen ser capaces de aprenderlos.
- El problema fue explorado en profundidad por Hochreiter (1991) y Bengio, et al. (1994) , quienes encontraron algunas razones bastante fundamentales por las que podría ser difícil.
- ¡Afortunadamente, los LSTM no tienen este problema!

Redes LSTM

- Las redes Long Short Term Memory, generalmente llamadas "LSTM", son un tipo especial de RNN, capaz de aprender dependencias a largo plazo. Fueron introducidos por Hochreiter y Schmidhuber (1997) , y fueron refinados y popularizados por muchas personas en la actualidad. Funcionan tremendamente bien en una gran variedad de problemas, y ahora son ampliamente utilizados.
- Los LSTM están diseñados explícitamente para evitar el problema de dependencia a largo plazo. Recordar información durante largos períodos de tiempo es prácticamente su comportamiento predeterminado, ¡no es algo que les cueste aprender!
- Todas las redes neuronales recurrentes tienen la forma de una cadena de módulos repetitivos de la red neuronal. En las RNN estándar, este módulo de repetición tendrá una estructura muy simple, como una sola capa de tanh.

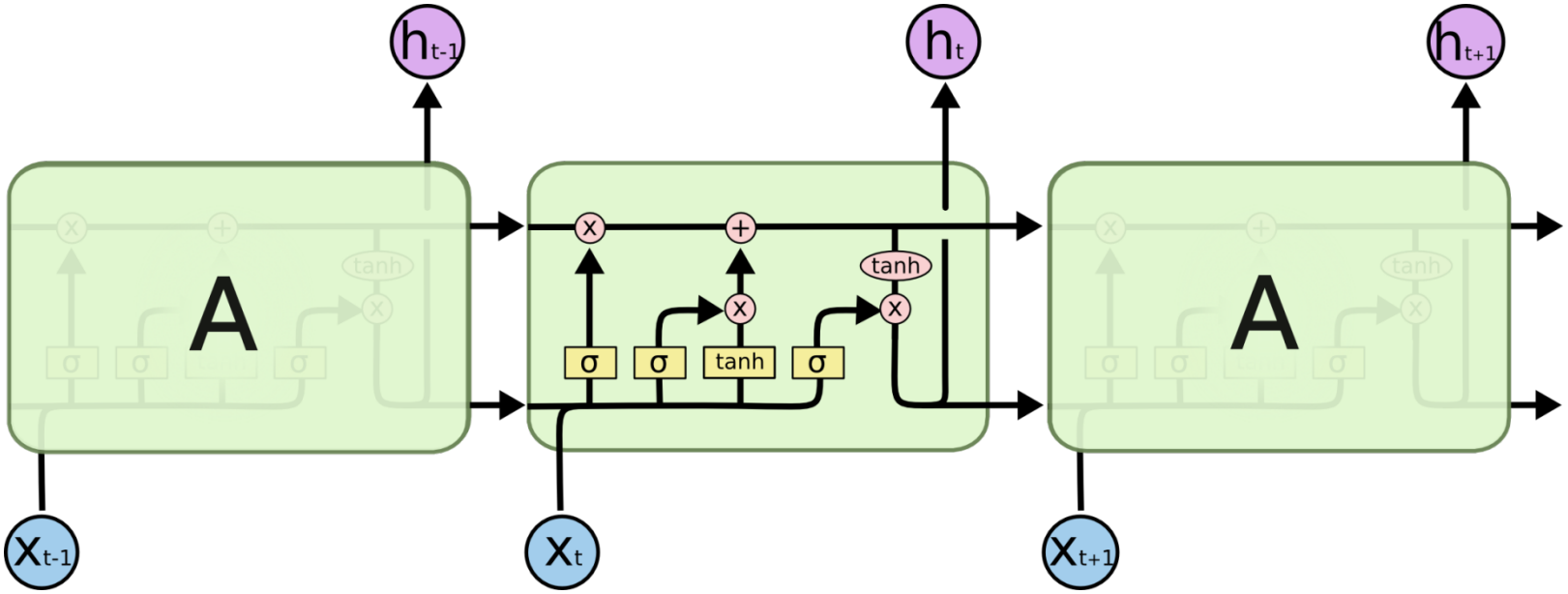
Redes LSTM

- **El módulo de repetición en un RNN estándar contiene una sola capa.**



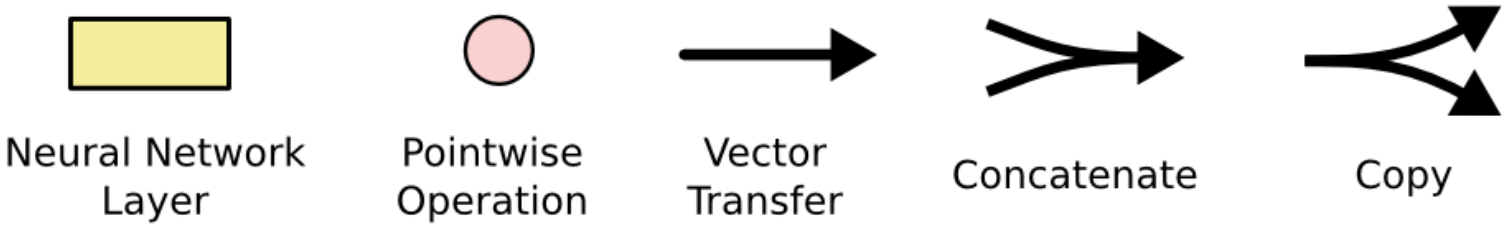
Redes LSTM

- Los LSTM también tienen esta estructura tipo cadena, pero el módulo de repetición tiene una estructura diferente. En lugar de tener una sola capa de red neuronal, hay cuatro que interactúan de una manera muy especial.



Redes LSTM

- Por ahora no nos preocuparemos por los detalles de lo que está pasando. Luego vamos a revisar el diagrama LSTM paso a paso más tarde. Por ahora, solo intentemos sentirnos cómodos con la notación que usaremos.



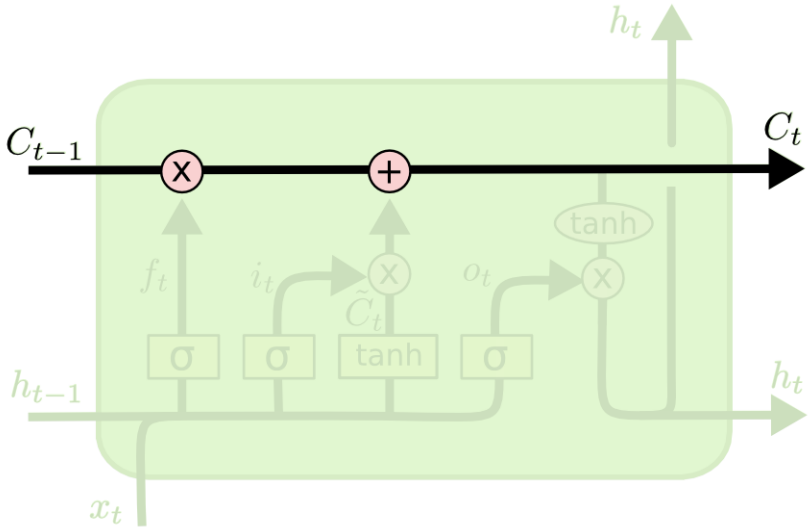
Redes LSTM

- En el diagrama anterior, cada línea transporta un vector completo, desde la salida de un nodo hasta las entradas de otros. Los círculos rosados representan operaciones puntuales, como la adición de vectores, mientras que los cuadros amarillos son capas de red neuronal.
- Las líneas que se fusionan denotan concatenación, mientras que una línea de bifurcación denota el contenido que se está copiando y las copias que van a diferentes ubicaciones.
- Todo el proceso anterior le permite a la red aprender dependencias de largo plazo.

Redes LSTM

La idea central detrás de LSTMs

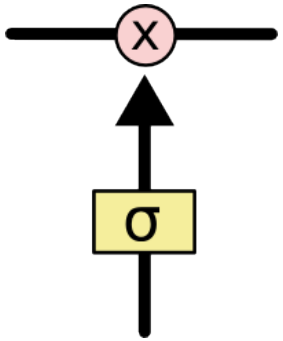
- La clave para los LSTM es el estado de la celda, la línea horizontal que recorre la parte superior del diagrama.
- El estado de la célula es algo así como una cinta transportadora. Corre hacia abajo por toda la cadena, con solo algunas interacciones lineales menores. Es muy fácil que la información fluya sin cambios.



Redes LSTM

La idea central detrás de LSTMs

- La LSTM tiene la capacidad de eliminar o agregar información al estado de la celda, cuidadosamente regulado por estructuras llamadas compuertas.
- Las puertas son una forma de permitir que la información fluya. Se componen de una capa de red neuronal sigmoide y un producto punto.



Redes LSTM

La idea central detrás de LSTMs

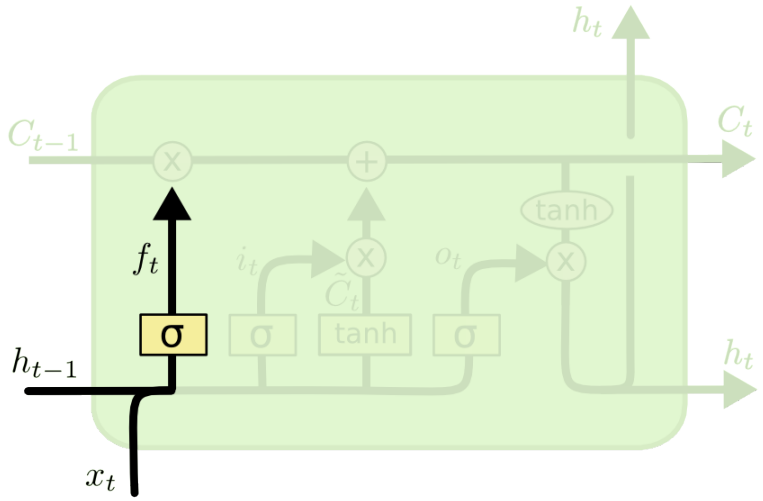
- La capa sigmoide produce números entre cero y uno, que describen la cantidad de cada componente que debe dejarse pasar. Un valor de cero significa "no dejar pasar nada", mientras que un valor de uno significa "dejar pasar todo".

- Un LSTM tiene tres de estas puertas, para proteger y controlar el estado de la celda.

Redes LSTM

Paso a paso de las LSTMs (1)

- El primer paso en nuestro LSTM es decidir qué información vamos a eliminar del estado de la celda. Esta decisión es tomada por una capa sigmoide llamada la "capa de puerta de olvido (forget gate layer)". Esta se representa por x_t y h_{t-1}, y genera un número entre 0 y 1 para cada número en la celda de estado C_{t-1}. Un número 1 representa que se debe mantener la información mientras que un 0 que se debe deshacer la información.

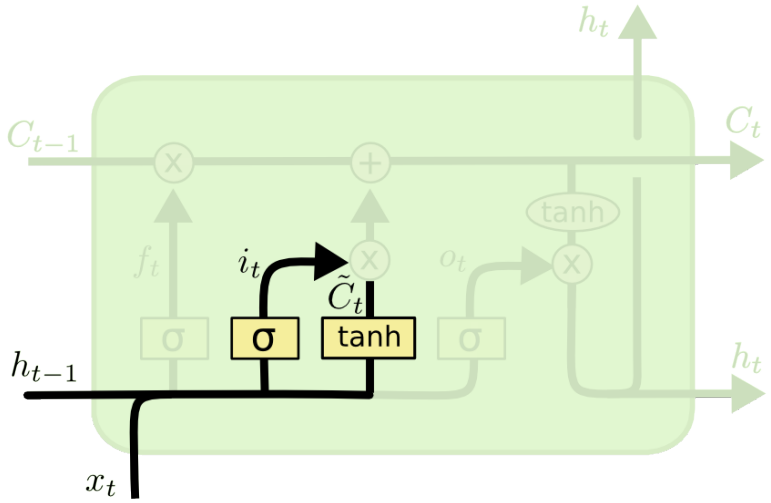


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Redes LSTM

Paso a paso de las LSTMs (2)

- El siguiente paso es decidir qué información nueva vamos a almacenar en el estado de la celda. Esto tiene dos partes. Primero, una capa sigmoide llamada "capa de puerta de entrada" decide qué valores actualizaremos.
- A continuación, una capa tanh crea un vector de nuevos valores candidatos, $\widetilde{C}_t$, que podría agregarse al estado. En el siguiente paso, combinaremos estos dos para crear una actualización para el estado.



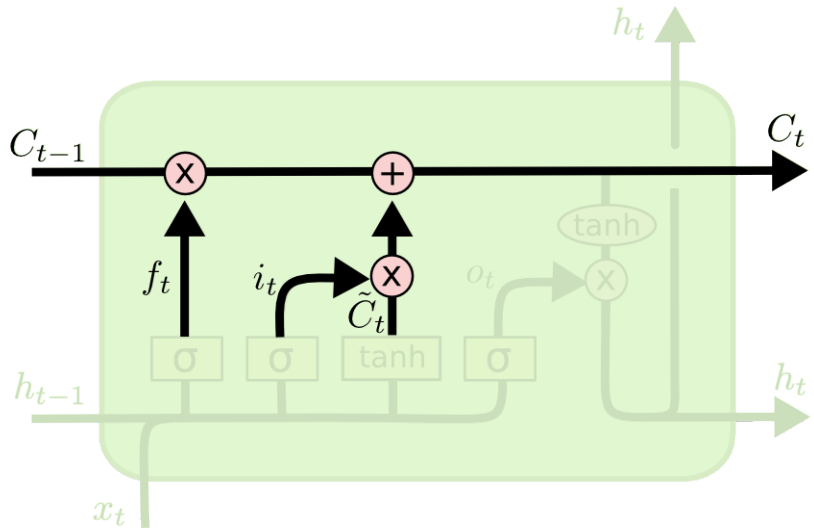
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

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

Redes LSTM

Paso a paso de las LSTMs (3)

- Ahora es el momento de actualizar el estado de la celda anterior, C_{t-1}, en el nuevo estado de la celda llamado C_t. Los pasos anteriores ya decidieron qué se debe hacer, por lo tanto solo necesitamos hacerlo.
- Multiplicamos el estado antiguo por f_t , olvidando las cosas que decidimos olvidar antes. Entonces añadimos $i_t * \widetilde{C_t}$. Estos son los nuevos valores candidatos, en función de cuánto decidimos actualizar cada valor de estado.

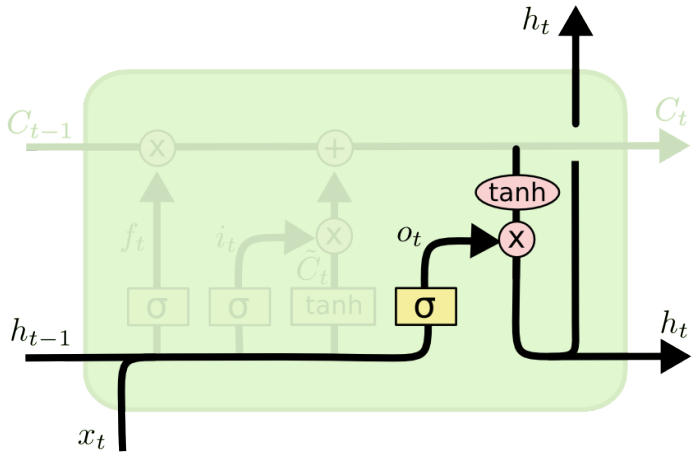


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Redes LSTM

Paso a paso de las LSTMs (4)

- Finalmente, tenemos que decidir qué vamos a producir. Esta salida se basará en nuestro estado de celda, pero será una versión filtrada. Primero, ejecutamos una capa sigmoide que decide qué partes del estado de la celda vamos a generar.
- Entonces, ponemos el estado de la celda a través de la **tanh**, para empujar los valores a estar entre -1 y 1, multiplicándolo por la salida de la puerta sigmoide, para que solo enviemos las partes que decidimos.



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

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

Redes LSTM

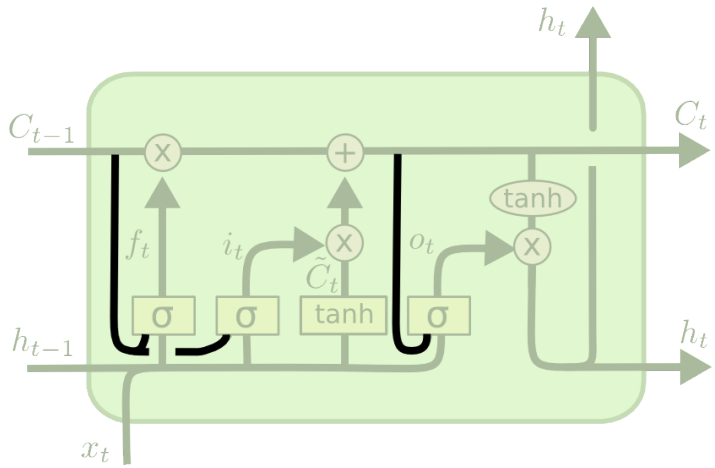
Paso a paso de las LSTMs

- Regresemos a nuestro ejemplo de un modelo de lenguaje tratando de predecir la siguiente palabra en base a todos los anteriores. En tal problema, el estado de la celda podría incluir el género del sujeto presente, de modo que se puedan usar los pronombres correctos. Cuando vemos un tema nuevo, queremos olvidar el género del tema anterior (Paso 1).
- Ahora, nos gustaría agregar el género del nuevo sujeto al estado de la celda, para reemplazar el anterior que estamos olvidando (Paso 2).
- Usando el Paso 3, es donde realmente descartamos la información sobre el género del sujeto anterior y agregamos la nueva información, tal como lo decidimos en los pasos anteriores.
- Mediante el Paso 4, dado que acaba de ver un tema, es posible que desee generar información relevante para un verbo, en caso de que eso sea lo que viene. Por ejemplo, podría mostrar si el sujeto es singular o plural, de modo que sepamos en qué forma se debe conjugar un verbo si eso es lo que sigue a continuación.

Redes LSTM

Variantes

- Lo que ha descrito hasta ahora es un LSTM bastante normal. Pero no todos los LSTM son iguales a los anteriores. De hecho, parece que casi todos los artículos relacionados con LSTM utilizan una versión ligeramente diferente. Las diferencias son menores, pero vale la pena mencionar algunas de ellas.
- Una variante popular de LSTM, introducida por Gers & Schmidhuber (2000) , está agregando "peephole connections". Esto significa que dejamos que las capas de la puerta miren el estado de la celda.

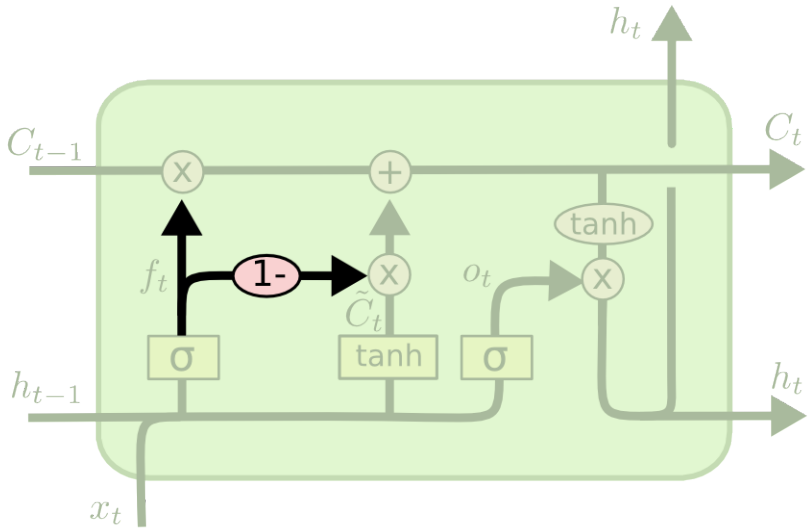


$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$

Redes LSTM

Variantes

- Otra variante es utilizar puertas de entrada y olvido acopladas. En lugar de decidir por separado qué debemos olvidar y a qué información nueva debemos agregar, tomamos esas decisiones juntas. Solo olvidamos cuando vamos a ingresar algo en su lugar. Solo ingresamos nuevos valores al estado cuando olvidamos algo más antiguo.

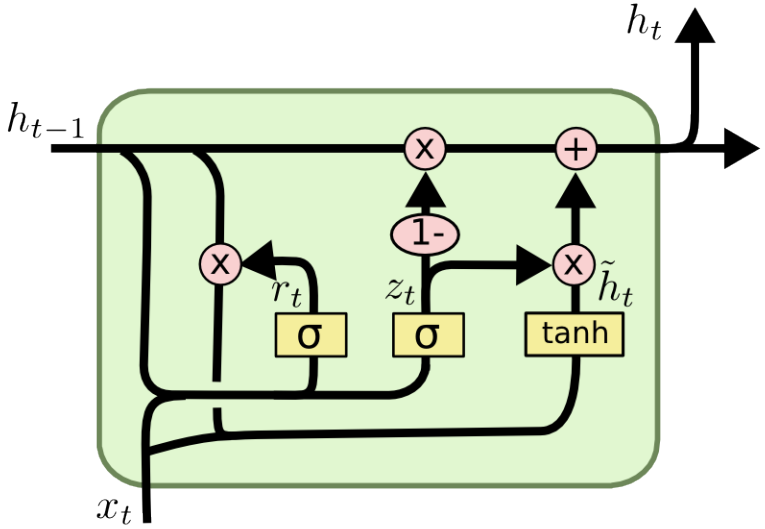


$$C_t = f_t * C_{t-1} + (1 - \boldsymbol{f_t}) * \tilde{C}_t$$

Redes LSTM

Variantes

- Una variación ligeramente más dramática en el LSTM es la Unidad Recurrente Cerrada (Gated Recurrent Unit), o GRU, introducida por Cho, et al. (2014) . Combina las puertas de olvido y entrada en una sola "puerta de actualización". También combina el estado de celda y el estado oculto, y realiza otros cambios. El modelo resultante es más simple que los modelos LSTM estándar, y se ha vuelto cada vez más popular.



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

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

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

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

Redes LSTM

Variantes

- Estas son solo algunas de las variantes LSTM más notables. Hay muchos otras, como Depth Gated RNNs de Yao, et al. (2015). También hay un enfoque completamente diferente para abordar las dependencias a largo plazo, como Clockwork RNNs de Koutnik, et al. (2014) .

- ¿Cuál de estas variantes es la mejor? ¿Las diferencias importan? Greff, et al. (2015) hacen una buena comparación de las variantes populares, encontrando que son todas iguales. Jozefowicz, et al. (2015) probó más de diez mil arquitecturas RNN, encontrando algunas que funcionaron mejor que las LSTM en ciertas tareas.