Descripción de las redes LSTM

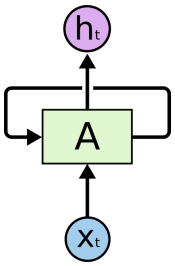
Publicado el 27 de agosto de 2015

Redes neuronales recurrentes

Los humanos no comienzan su pensamiento desde cero cada segundo. A medida que lee este ensayo, comprende cada palabra en función de su comprensión de las palabras anteriores. No tiras todo y empiezas a pensar desde cero otra vez. Tus pensamientos tienen persistencia.

Las redes neuronales tradicionales no pueden hacer esto y parece una gran deficiencia. Por ejemplo, imagine que desea clasificar qué tipo de evento está sucediendo 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 los posteriores.

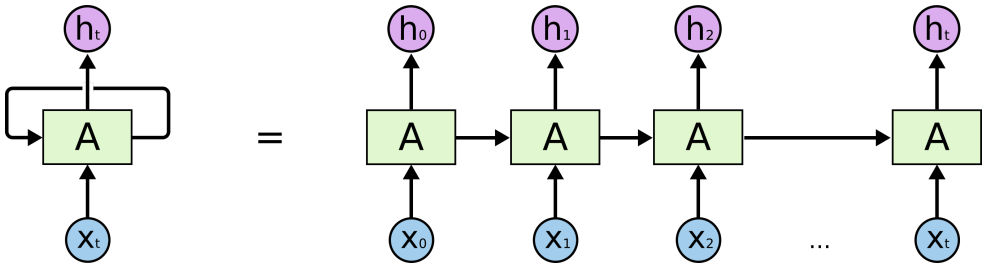
Las redes neuronales recurrentes abordan este problema. Son redes con bucles en ellas, lo que permite que la información persista.



Las redes neuronales recurrentes tienen bucles.

En el diagrama anterior, un trozo de red neuronal,AA, mira alguna entradax_tXty genera un valor$h_t$ht. Un bucle permite que la información pase de un paso de la red al siguiente.

Estos bucles hacen que las redes neuronales recurrentes parezcan algo misteriosas. Sin embargo, si piensa un poco más, resulta que no son tan diferentes de una red neuronal normal. Se puede pensar en una red neuronal recurrente como múltiples copias de la misma red, cada una de las cuales pasa un mensaje a un sucesor. Considere lo que sucede si desenrollamos el ciclo:



Una red neuronal recurrente desenrollada.

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 tales datos.

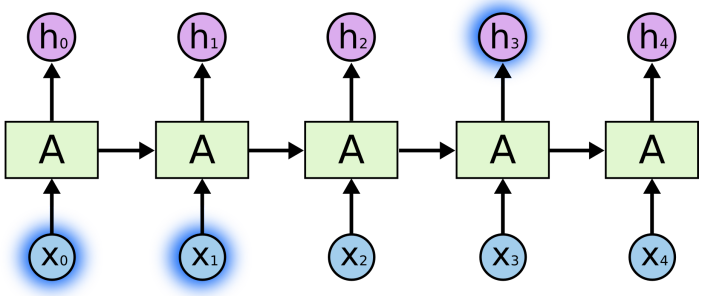
¡Y ciertamente se usan! En los últimos años, ha habido un éxito increíble al aplicar RNN a una variedad de problemas: reconocimiento de voz, modelado de lenguaje, traducción, subtítulos de imágenes... La lista continúa. Dejaré la discusión de las hazañas asombrosas que se pueden lograr con RNN en la excelente publicación de blog de Andrej Karpathy, The Unreasonable Effectiveness of Recurrent Neural Networks . Pero realmente son bastante sorprendentes.

Esencial para estos éxitos es el uso de "LSTM", un tipo muy especial de red neuronal recurrente que funciona, para muchas tareas, mucho mejor que la versión estándar. Casi todos los resultados emocionantes basados en redes neuronales recurrentes se logran con ellos. Son estos LSTM los que explorará este ensayo.

El problema de las dependencias a largo plazo

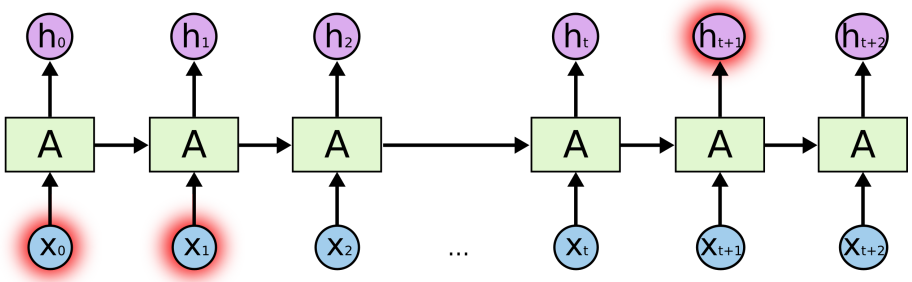
Uno de los atractivos de las RNN es la idea de que podrían conectar información previa con la tarea actual, como el uso de cuadros de video anteriores que podrían informar la comprensión del cuadro actual. Si los RNN pudieran hacer esto, serían extremadamente útiles. Pero pueden? Eso depende.

A veces, solo necesitamos mirar información reciente para realizar la tarea actual. Por ejemplo, considere un modelo de lenguaje que intente predecir la siguiente palabra en función de las anteriores. Si estamos tratando de predecir la última palabra en "las nubes están en el *cielo* ", no necesitamos más contexto, es bastante obvio que la siguiente palabra será cielo. En tales casos, donde la brecha entre la información relevante y el lugar donde se necesita es pequeña, los RNN pueden aprender a usar la información pasada.



Pero también hay casos en los que necesitamos más contexto. Considere tratar de predecir la última palabra en el texto "Crecí en Francia... hablo *francés* con fluidez ". Información reciente sugiere que la siguiente palabra probablemente sea el nombre de un idioma, pero si queremos acotar qué idioma, necesitamos el contexto de Francia, desde más atrás. Es muy posible que la brecha entre la información relevante y el punto donde se necesita se vuelva muy grande.

Desafortunadamente, a medida que crece esa brecha, los RNN se vuelven incapaces de aprender a conectar la información.



En teoría, los RNN son absolutamente capaces de manejar tales "dependencias a largo plazo". Un ser humano podría elegir cuidadosamente los parámetros para resolver problemas de juguetes de esta forma. Lamentablemente, en la práctica, los RNN no parecen poder aprenderlos. El problema fue explorado en profundidad por Hochreiter (1991) [alemán] y Bengio, et al. (1994) , quien encontró algunas razones bastante fundamentales por las que podría ser difícil.

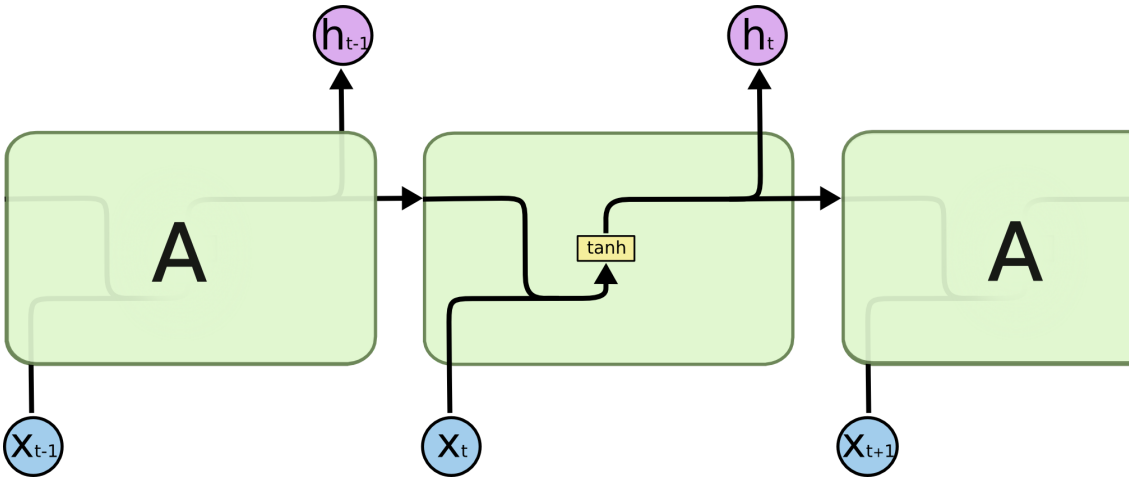
¡Afortunadamente, los LSTM no tienen este problema!

Redes LSTM

Las redes de memoria a largo plazo, generalmente llamadas simplemente "LSTM", son un tipo especial de RNN, capaces de aprender dependencias a largo plazo. Fueron introducidos por Hochreiter & Schmidhuber (1997) , y muchas personas los refinaron y popularizaron en sus siguientes trabajos. [1] Funcionan tremendamente bien en una gran variedad de problemas y ahora se utilizan ampliamente.

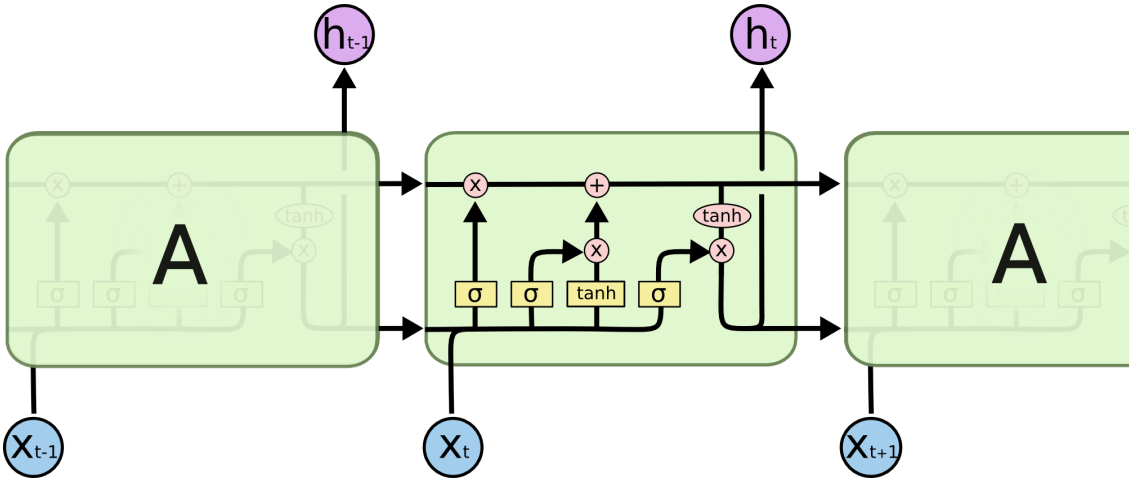
Los LSTM están diseñados explícitamente para evitar el problema de la dependencia a largo plazo. Recordar información durante largos períodos de tiempo es prácticamente su comportamiento predeterminado, ¡no es algo que les cueste aprender!

Todas las redes neuronales recurrentes tienen la forma de una cadena de módulos repetidos de red neuronal. En RNN estándar, este módulo repetitivo tendrá una estructura muy simple, como una sola capa de tanh.



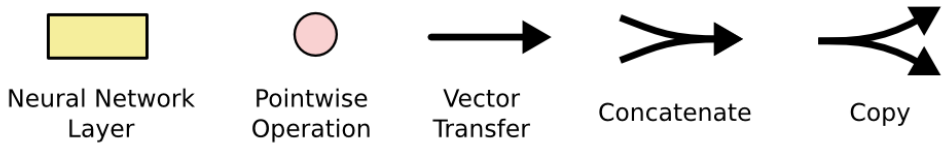
El módulo de repetición en un RNN estándar contiene una sola capa.

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



El módulo repetitivo en un LSTM contiene cuatro capas que interactúan.

No se preocupe por los detalles de lo que está pasando. Veremos el diagrama LSTM paso a paso más adelante. Por ahora, tratemos de sentirnos cómodos con la notación que usaremos.

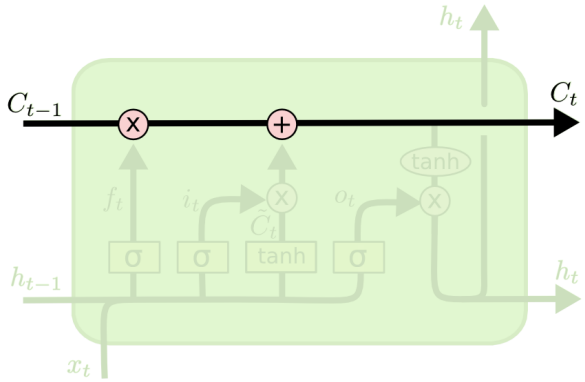


En el diagrama anterior, cada línea lleva un vector completo, desde la salida de un nodo hasta las entradas de los demás. Los círculos rosas representan operaciones puntuales, como la suma de vectores, mientras que los cuadros amarillos son capas de redes neuronales aprendidas. Las líneas que se fusionan denotan concatenación, mientras que una línea que se bifurca denota que su contenido se copia y las copias se dirigen a diferentes ubicaciones.

La idea central detrás de los LSTM

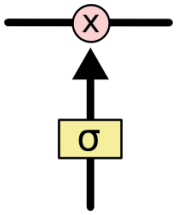
La clave de los LSTM es el estado de la celda, la línea horizontal que atraviesa la parte superior del diagrama.

El estado de la celda es como una cinta transportadora. Se ejecuta directamente a lo largo de toda la cadena, con solo algunas interacciones lineales menores. Es muy fácil que la información fluya sin cambios.



El LSTM tiene la capacidad de eliminar o agregar información al estado de la celda, cuidadosamente regulado por estructuras llamadas puertas.

Las puertas son una forma de dejar pasar información opcionalmente. Están compuestos por una capa de red neuronal sigmoidea y una operación de multiplicación puntual.



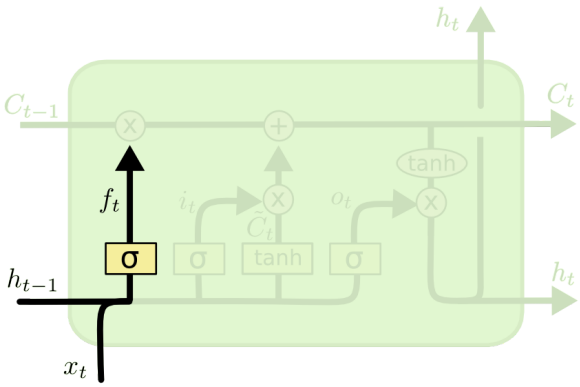
La capa sigmoidea genera números entre cero y uno, que describen la cantidad de cada componente que se debe dejar pasar. Un valor de cero significa "no dejar pasar nada", mientras que un valor de uno significa "¡dejar pasar todo!"

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

Recorrido paso a paso de LSTM

El primer paso en nuestro LSTM es decidir qué información vamos a desechar del estado de la celda. Esta decisión la toma una capa sigmoidea llamada "capa de puerta de olvido". mirah_{t-1}ht−1yx_tXt, y genera un número entre0y1para cada número en el estado de la celdaC_{t-1}Ct−1. A1representa "guardar esto por completo", mientras que un 0representa "deshacerse completamente de esto".

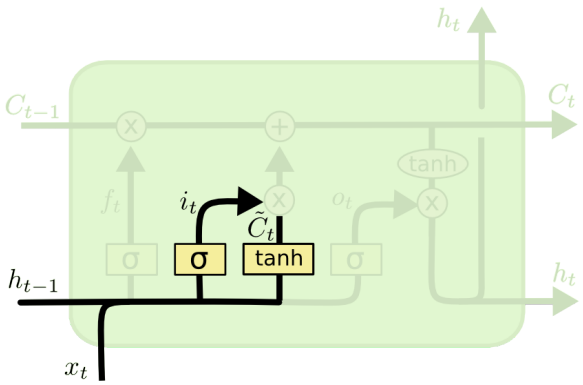
Volvamos a nuestro ejemplo de un modelo de lenguaje que intenta predecir la siguiente palabra basándose en todas las anteriores. En tal problema, el estado de la celda podría incluir el género del sujeto presente, de modo que se puedan usar los pronombres correctos. Cuando vemos un tema nuevo, queremos olvidar el género del tema anterior.



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

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

En el ejemplo de nuestro modelo de lenguaje, nos gustaría agregar el género del nuevo sujeto al estado de la celda, para reemplazar el antiguo que estamos olvidando.

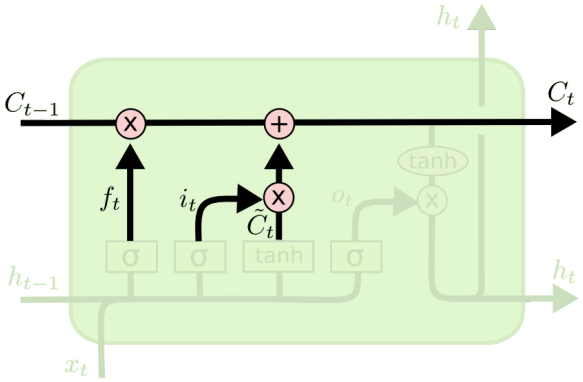


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t]\right) +$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t]$$

Ahora es el momento de actualizar el estado de la celda anterior,C_{t-1}Ct−1, en el nuevo estado celularC_tCt. Los pasos anteriores ya decidieron qué hacer, solo necesitamos hacerlo.

Multiplicamos el estado anterior porF_tFt, olvidando las cosas que decidimos olvidar antes. Luego agregamos $i_t * \tilde{C}_t$it∗C~t. Estos son los nuevos valores candidatos, escalados por cuánto decidimos actualizar cada valor de estado.

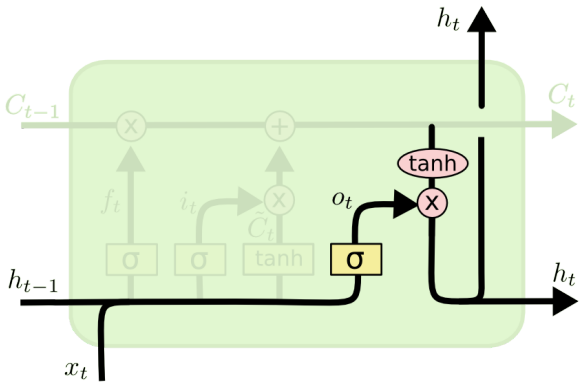
En el caso del modelo de lenguaje, aquí es donde descartaríamos la información sobre el género del sujeto anterior y agregaríamos la nueva información, como decidimos en los pasos anteriores.



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

Finalmente, tenemos que decidir qué vamos a generar. Esta salida se basará en nuestro estado de celda, pero será una versión filtrada. Primero, ejecutamos una capa sigmoidea que decide qué partes del estado de la celda vamos a generar. Luego, ponemos el estado de la celda a través$\tanh$tanh(para empujar los valores a estar entre-1−1y$1$1) y multiplíquelo por la salida de la puerta sigmoidea, de modo que solo emitamos las partes que decidimos.

Para el ejemplo del modelo de lenguaje, dado que acaba de ver un sujeto, es posible que desee generar información relevante para un verbo, en caso de que eso sea lo que sigue. Por ejemplo, podría mostrar si el sujeto es singular o plural, para que sepamos en qué forma se debe conjugar un verbo si eso es lo que sigue a continuación.

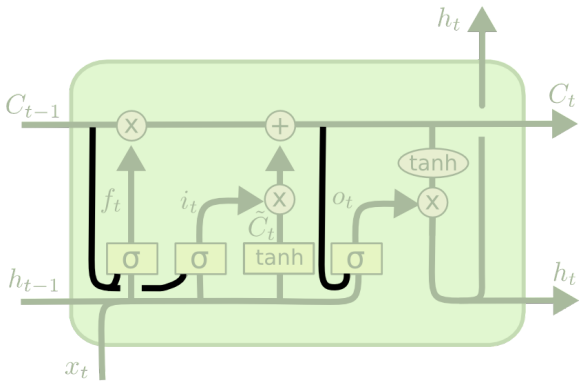


$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\right) +$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Variantes de la memoria a largo plazo

Lo que he 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 documentos que involucran LSTM usan una versión ligeramente diferente. Las diferencias son menores, pero vale la pena mencionar algunas de ellas.

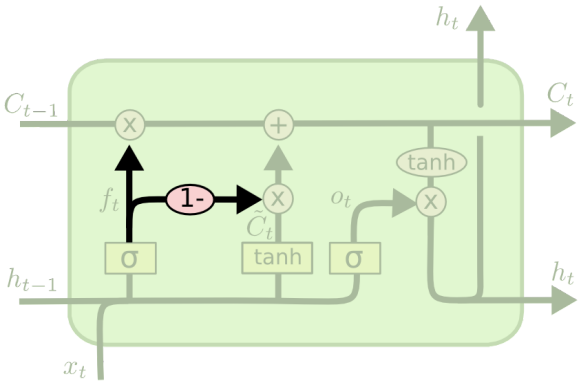
Una variante popular de LSTM, presentada por Gers & Schmidhuber (2000) , agrega "conexiones de mirilla". Esto significa que dejamos que las capas de la puerta miren el estado de la celda.



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

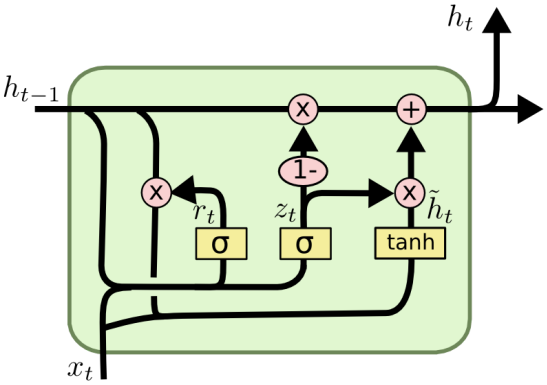
El diagrama de arriba agrega mirillas a todas las puertas, pero muchos papeles darán algunas mirillas y otras no.

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



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

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



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, \right.$$
$$h_t = (1 - z_t) * h_{t-1} + z_t *$$

Estas son solo algunas de las variantes de LSTM más notables. Hay muchos otros, como los RNN controlados por profundidad de Yao, et al. (2015) . También existe un enfoque completamente diferente para abordar las dependencias a largo plazo, como Clockwork RNN de Koutnik, et al. (2014) .

¿Cuál de estas variantes es mejor? ¿Importan las diferencias? Greff, et al. (2015) hacen una buena comparación de variantes populares y descubren que todas son casi iguales. Jozefowicz, et al. (2015) probaron más de diez mil arquitecturas RNN y encontraron algunas que funcionaron mejor que las LSTM en ciertas tareas.

Conclusión

Anteriormente, mencioné los notables resultados que la gente está logrando con las RNN. Esencialmente, todos estos se logran utilizando LSTM. ¡Realmente funcionan mucho mejor para la mayoría de las tareas!

Escritos como un conjunto de ecuaciones, los LSTM parecen bastante intimidantes. Con suerte, recorrerlos paso a paso en este ensayo los ha hecho un poco más accesibles.

Los LSTM fueron un gran paso en lo que podemos lograr con los RNN. Es natural preguntarse: ¿hay otro gran paso? Una opinión común entre los investigadores es: "¡Sí! ¡Hay un siguiente paso y es la atención!" La idea es dejar que cada paso de una RNN seleccione información para ver de una colección de información más grande. Por ejemplo, si está utilizando un RNN para crear un título que describa una imagen, podría elegir una parte de la imagen para buscar cada palabra que genera. De hecho, Xu, *et al.* (2015) hacen exactamente esto: ¡podría ser un punto de partida divertido si quieres explorar la atención! Ha habido una serie de resultados realmente emocionantes usando la atención, y parece que hay muchos más a la vuelta de la esquina...

La atención no es el único hilo interesante en la investigación de RNN. Por ejemplo, Grid LSTM de Kalchbrenner, *et al.* (2015) parecen extremadamente prometedores. Trabaje usando RNN en modelos generativos, como Gregor, *et al.* (2015) , Chung, *et al.* (2015) , o Bayer & Osendorfer (2015) , también parece muy interesante. Los últimos años han sido un momento emocionante para las redes neuronales recurrentes, ¡y los próximos prometen serlo aún más!

Expresiones de gratitud

Estoy agradecido con varias personas por ayudarme a comprender mejor los LSTM, comentar sobre las visualizaciones y brindar comentarios sobre esta publicación.

Estoy muy agradecido con mis colegas de Google por sus útiles comentarios, especialmente con Oriol Vinyals , Greg Corrado , Jon Shlens , Luke Vilnis e Ilya Sutskever . También estoy agradecido con muchos otros amigos y colegas por tomarse el tiempo para ayudarme, incluidos Dario Amodei y Jacob Steinhardt . Estoy especialmente agradecido con Kyunghyun Cho por su correspondencia extremadamente atenta sobre mis diagramas.

Antes de esta publicación, practiqué la explicación de LSTM durante dos series de seminarios que enseñé sobre redes neuronales. Gracias a todos los que participaron en ellos por su paciencia conmigo y por sus comentarios.