Análisis de Series Temporales

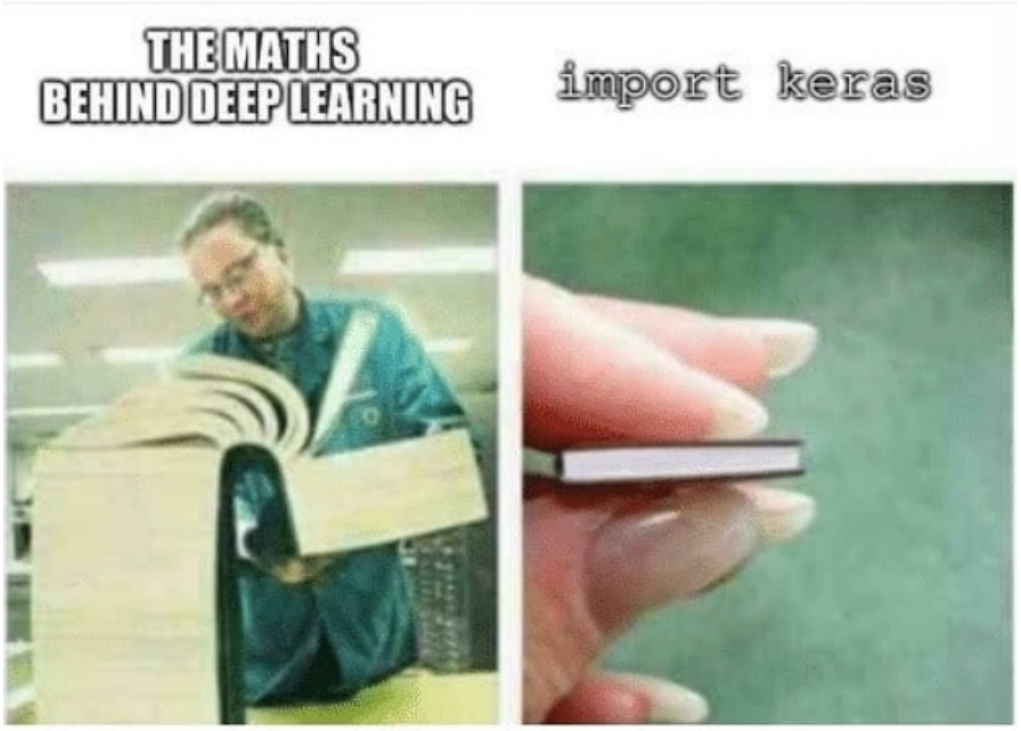
Clase 7 - Forecasting de Series Temporales con Redes Neuronales

Rodrigo Del Rosso
RDelRosso-ext@austral.edu.ar

03 de Diciembre de 2021

Agenda

- Introducción
- Modelo Autorregresivo de Redes Neuronales $NNAR(p, k)$
- Long-Short Term Memory $LSTM$



Machine learning be like

Introducción

El problema de Machine Learning más simple que involucra una secuencia es un problema uno a uno.

En este caso, se dispone de una entrada de datos (tensor) para el modelo, que genera una predicción.

- Regresión Lineal
- Clasificación
- Clasificación de Imágenes con Red Convolucional

Una red neuronal recurrente (RNN) se ocupa de los problemas de secuencia porque sus conexiones forman un ciclo dirigido. En otras palabras, pueden retener el estado de una iteración a la siguiente utilizando su propia salida como entrada para el siguiente paso.

Modelo Autorregresivo de Redes Neuronales

Modelo Autorregresivo de Redes Neuronales (NNAR)

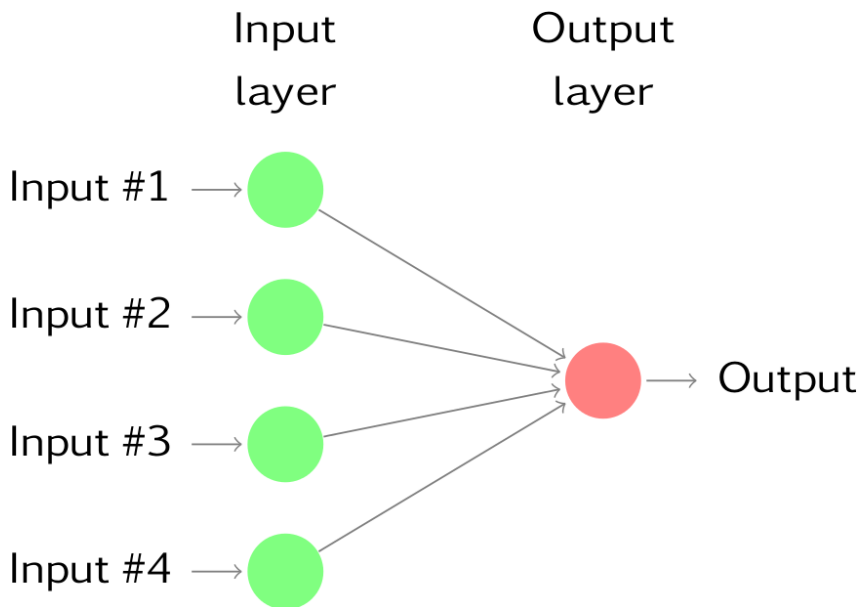
Las Redes Neuronales artificiales son métodos de pronóstico que se basan en modelos matemáticos simples del cerebro. **Permiten modelizar relaciones complejas no lineales entre la variable de respuesta y sus predictores**.

Se puede pensar en una red neuronal como una red de "neuronas" que están organizadas en capas. Los **predictores (o entradas)** forman la capa inferior y los **pronósticos (o salidas)** forman la capa superior. También puede haber capas intermedias que contengan "neuronas ocultas".

Las redes más simples no contienen capas ocultas y son equivalentes a regresiones lineales.

Modelo Autorregresivo de Redes Neuronales (NNAR)

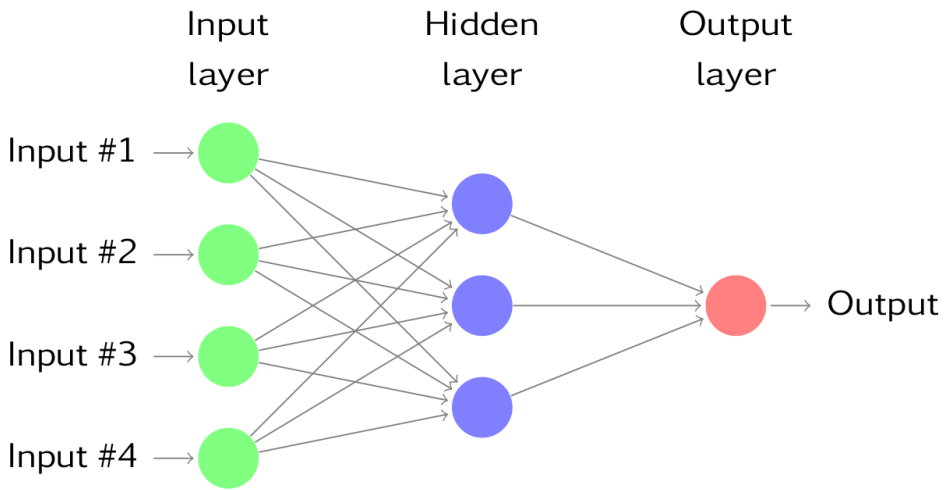
La siguiente figura muestra la versión de red neuronal de una regresión lineal con cuatro predictores.



Los coeficientes adjuntos a estos predictores se denominan "pesos". Los pronósticos se obtienen mediante una combinación lineal de las entradas. Los pesos se seleccionan en el marco de la red neuronal utilizando un "algoritmo de aprendizaje" que minimiza una "función de costo" como el MSE.

Modelo Autorregresivo de Redes Neuronales (NNAR)

Una vez que agregamos una capa intermedia con **neuronas ocultas**, la red neuronal se vuelve no lineal.



Esto se conoce como **multilayer feed-forward network**, donde cada capa de nodos recibe entradas de las capas anteriores. Las salidas de los nodos en una capa son entradas para la siguiente capa. Las entradas de cada nodo se combinan mediante una combinación lineal ponderada. Luego, el resultado es modificado por una función no lineal antes de ser emitido.

Modelo Autorregresivo de Redes Neuronales (NNAR)

Por ejemplo, en el diagrama anterior, las entradas en la neurona oculta j en la figura anterior se combinan linealmente para dar,

$$z_j = b_j + \sum_{k=1}^{4} w_{k,j} x_k \quad \textit{donde} \quad j = 1, 2, 3$$

Cuando se usa el uso de una red neuronal de retroalimentación para pronosticar una serie de tiempo, las entradas pueden ser valores observados previamente de la serie de tiempo y_t.

Esto se denomina **Neural Network Autoregression**. Por ejemplo, podría utilizarse los últimos cuatro valores anteriores para forecastear el próximo valor de la serie temporal $(y_{t-1}, y_{t-2}, y_{t-3}, y_{t-4})$. Los nodos ocultos (**hidden nodes**) serían dados por,

$$z_j = b_j + \sum_{k=1}^{4} w_{k,j} y_{t-k}$$

En la capa oculta (hidden layer), se aplica una función de activación (activation function) a los nodos ocultos (hidden nodes).

Modelo Autorregresivo de Redes Neuronales (NNAR)

La función de activación es una función no lineal, como una función sigmoidea/logística,

$$\phi(z) = \frac{1}{1+e^{-z}}$$

que se aplica para dar la entrada a la siguiente capa de la red neuronal.

La aplicación de esta función tiende a reducir el efecto de valores de entrada extremos, lo que hace que toda la red sea más robusta al efecto de valores atípicos.

La capa de salida de la red neuronal es otra combinación lineal de los nodos ocultos después de aplicar la función de activación $\phi(z)$.

$$y_t = a + \sum_{k=1}^{3} h_j \phi(z_j)$$

O en términos de los rezagos pasados se tiene,

$$y_t = a + \sum_{k=1}^{3} h_j \phi(b_j + \sum_{k=1}^{4} w_{k,j} y_{t-k})$$

Modelo Autorregresivo de Redes Neuronales (NNAR)

Importante - Parámetros

Los parámetros (o pesos) $a, h_1, h_2, h_3, b_1, b_2, b_3, w_{1,1}, w_{1,2}, \ldots, w_{3,3}, w_{4,3}$ **aprenden de los datos**.

Los pesos a estimar generalmente están restringidos para evitar que sean demasiado grandes. El parámetro de tuning que restringe los pesos se conoce como el "decay parameter" y, a menudo, es igual a 0,1.

Los pesos "aprenden" eligiendo primero valores aleatorios para ellos inicialmente, y estos pesos elegidos al azar se actualizan luego utilizando los datos observados o series temporales. Debido a que los pesos se eligen inicialmente al azar, existe un elemento de aleatoriedad en las predicciones producidas por una red neuronal. Por lo tanto, la red generalmente se entrena varias veces utilizando diferentes puntos de partida aleatorios, y luego se promedian los resultados. El número de capas ocultas y el número de nodos en cada capa oculta deben especificarse de antemano. Se pueden utilizar métodos de validación cruzada para ayudar a elegir valores "óptimos" para la complejidad del modelo.

Modelo Autorregresivo de Redes Neuronales (NNAR)

Con datos de series de tiempo, los valores rezagados se pueden usar como entradas a una red neuronal, al igual que en un modelo $AR(p)$. A esto se denomina un modelo autorregresivo de red neuronal (modelo $NNAR(p, k)$), el cual tiene dos componentes p y k,

Solamente se considerará **Feed-Forward Neural Networks** con una capa oculta, y se utilizará la notación $NNAR(p, k)$, donde,

- p denota el número de valores rezagados que se utilizan como entradas.
- k denota el número de nodos ocultos que están presentes.

Ejemplos

El modelo $NNAR(4, 3)$ que se muestra arriba, es una red neuronal con las últimas cuatro observaciones como entradas para predecir la salida y_t, y con tres neuronas en la capa oculta.

Ahora bien, un modelo $NNAR(p, 0)$ es equivalente a un modelo $ARIMA(p, 0, 0)$, pero sin las restricciones sobre los parámetros para asegurar la estacionariedad, es decir,

$$y_t = a + \sum_{p=1}^{4} w_p y_{t-p} + \epsilon_t$$

Modelo Autorregresivo de Redes Neuronales (NNAR)

Si el conjunto de datos es **estacional**, se expresa como $NNAR(p, P, k)$, donde P denota la cantidad de lags estacionales.

- El valor de p se elige en función de los criterios de información.
- Las redes neuronales tienen un componente aleatorio inherente.

Por lo tanto, se sugiere que el modelo de red neuronal se ejecute varias veces, siendo 20 el requisito mínimo.

Luego, el resultado final se presenta como media o mediana. También se sabe que las redes neuronales no funcionan bien con los datos de tendencias. Por lo tanto, debemos eliminar la tendencia o diferenciar los datos antes de ejecutar el modelo de red neuronal.

Modelo Autorregresivo de Redes Neuronales (NNAR)

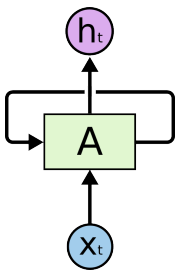
Importante

- Cuando se trata de pronósticos, la red se aplica de forma iterativa. Para pronosticar un paso adelante, simplemente se utilizan las entradas históricas.
- Sin embargo, para pronosticar dos pasos adelante, primero tenemos que realizar un pronóstico de un paso adelante para obtener $\hat{y}_{T+1}$ y luego tratar este pronóstico como un valor histórico para obtener el próximo pronóstico $\hat{y}_{T+2}$.
- Este proceso se repite iterativamente para obtener un pronóstico de h pasos adelante $\hat{y}_{T+h}$

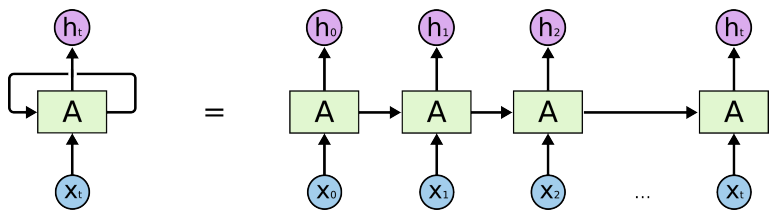
Long-Short Term Memory

Long Short-Term Memory (LSTM)

LSTM fue introducido por **Hochreiter y Schmidhuber** en 1997 como una red neuronal recurrente (RNN) que se entrena mediante la retropropagación a través del tiempo y supera el problema del gradiente de desaparición (vanishing gradient problem). Que sea recurrente significa que mediante bucles permiten que la información persista.



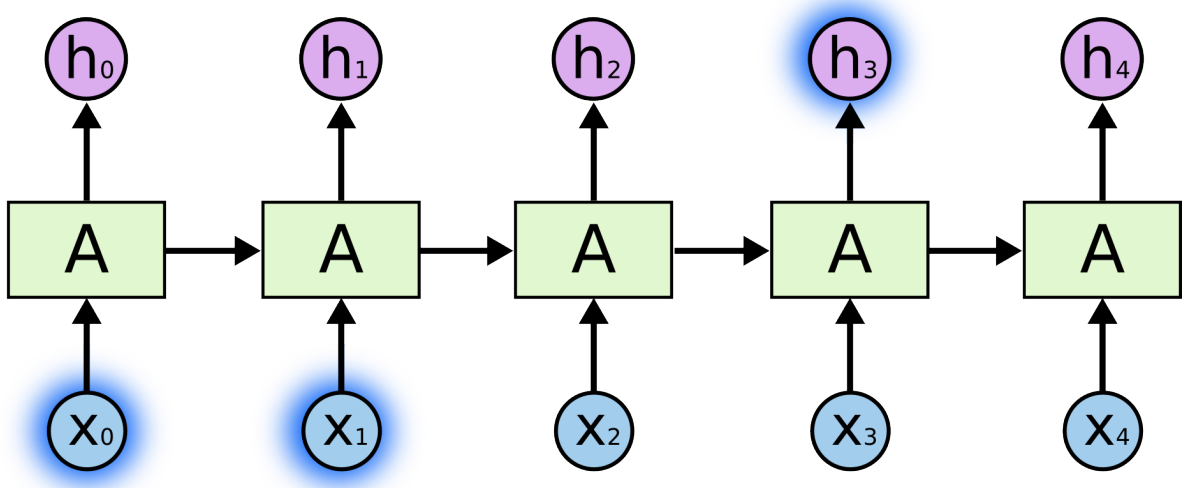
Una red neuronal recurrente se puede considerar como múltiples copias de la misma red, cada una de las cuales pasa un mensaje a un sucesor. Considere lo que sucede si desenrollamos el ciclo:



Long Short-Term Memory (LSTM)

Uno de los atractivos de los RNN es la idea de que podrían conectar la información anterior a la tarea actual.

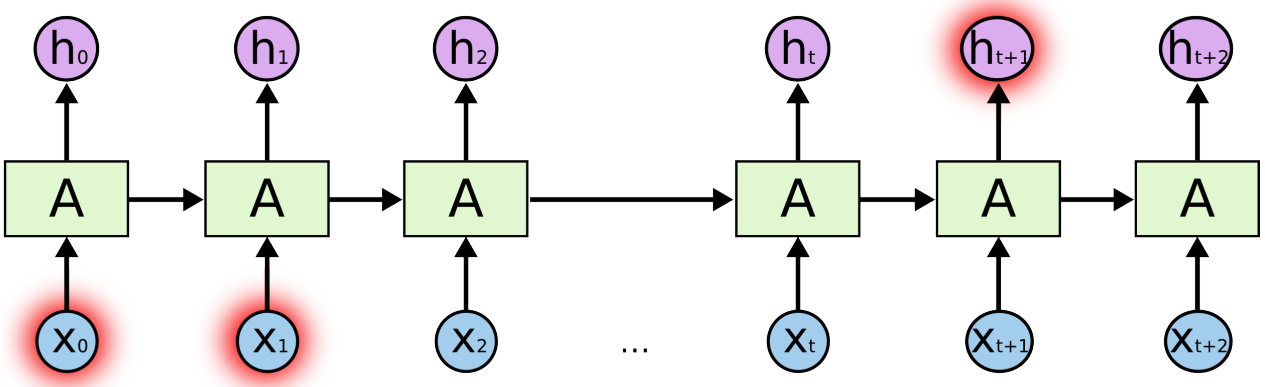
Donde la brecha entre la información relevante y el lugar donde se necesita es pequeña, los RNN pueden aprender a usar la información pasada.



También hay casos en los que se necesita más contexto y es muy posible que la brecha entre la información relevante y el punto donde se necesita se vuelva muy grande.

Long Short-Term Memory (LSTM)

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

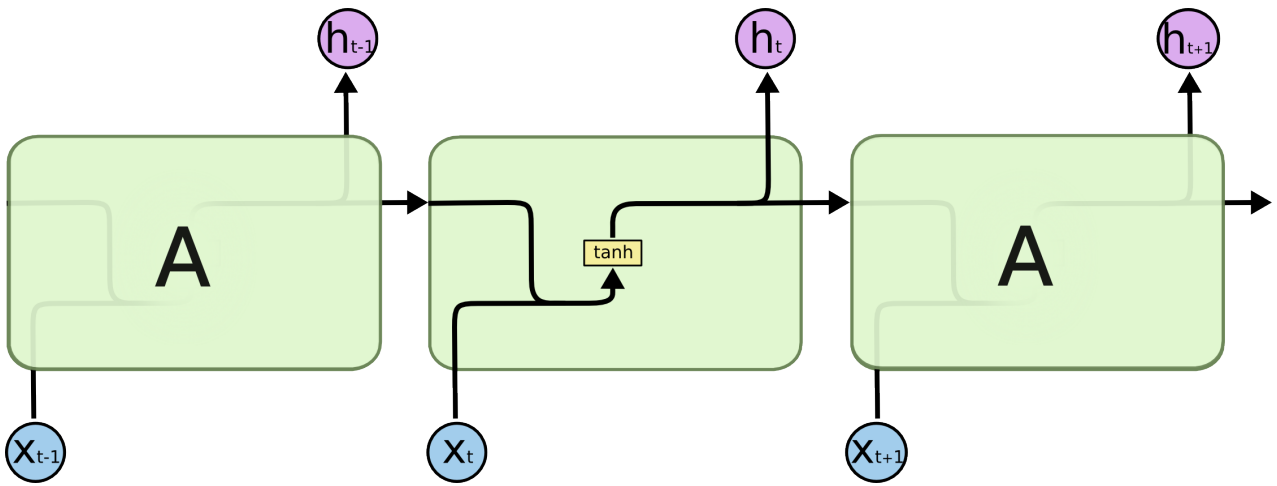


En teoría, los RNN son absolutamente capaces de manejar tales "dependencias a largo plazo". *¡Afortunadamente, los LSTM no tienen este problema!*

Long Short-Term Memory (LSTM)

Esta red está diseñada explícitamente para **evitar el problema de dependencia a largo plazo**.

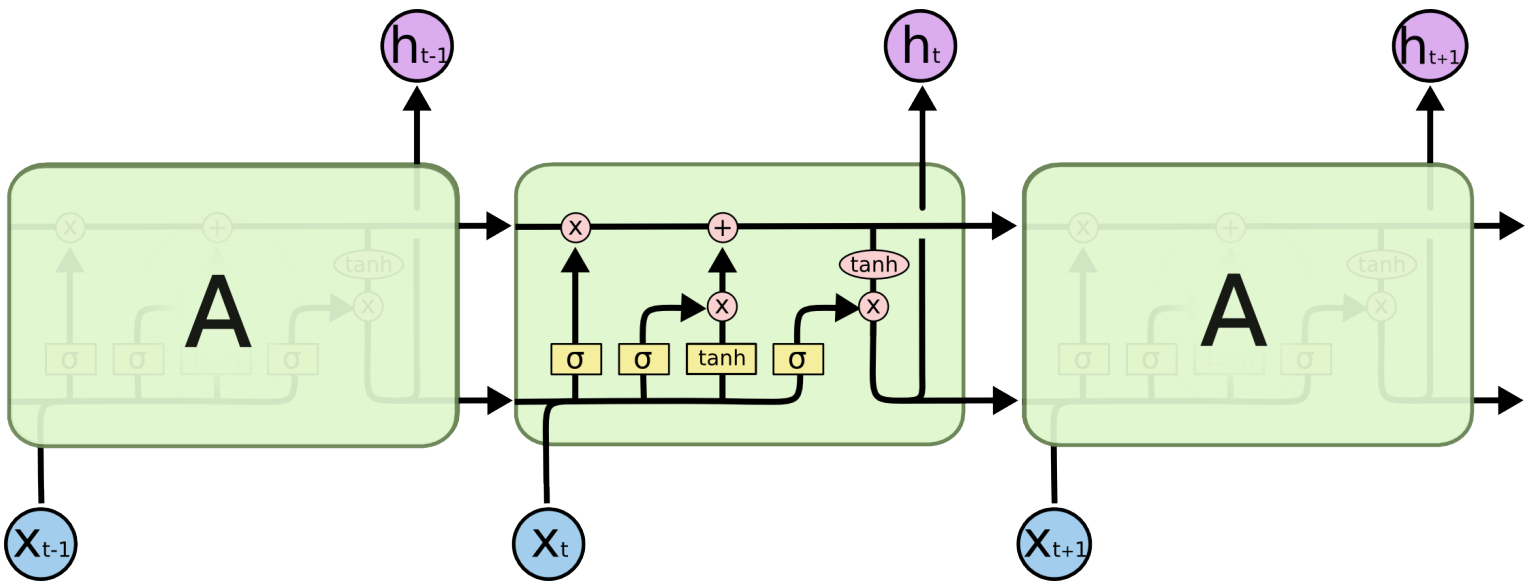
Recordar información durante largos períodos de tiempo es prácticamente su comportamiento predeterminado, no es algo que les cueste aprender.



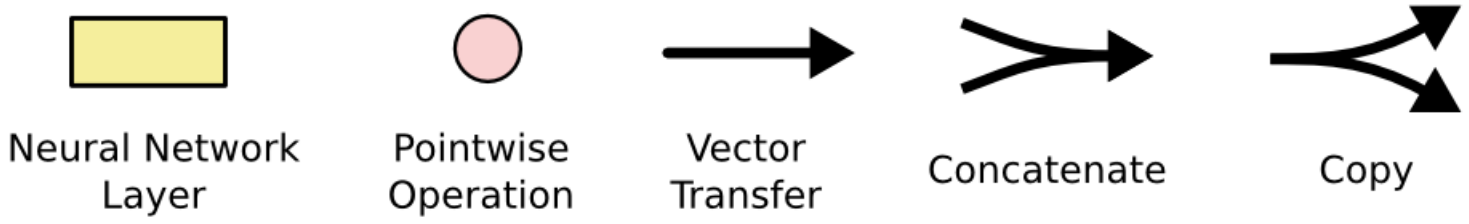
Long Short-Term Memory (LSTM)

Los LSTM también tienen esta **estructura similar a una cadena**, pero el módulo repetido tiene una estructura diferente.

En lugar de tener una sola capa de red neuronal, hay cuatro que interactúan de una manera muy especial.



Notación



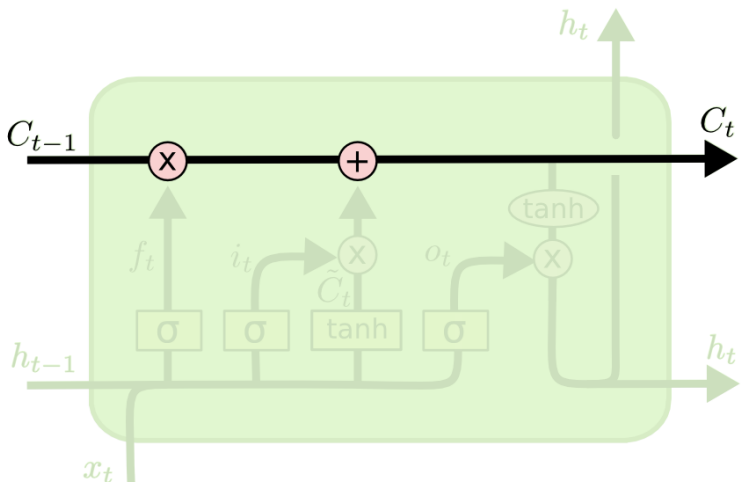
- Cada línea lleva un vector completo, desde la salida de un nodo hasta las entradas de otros.
- Los círculos rosas representan operaciones puntuales, como la suma de vectores, mientras que los cuadros amarillos son capas de redes neuronales aprendidas.
- Las líneas que se fusionan denotan concatenación, mientras que una bifurcación de líneas denota que su contenido se está copiando y las copias van a diferentes ubicaciones.

Long Short-Term Memory (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. Corre directamente a lo largo de toda la cadena, con solo algunas interacciones lineales menores.

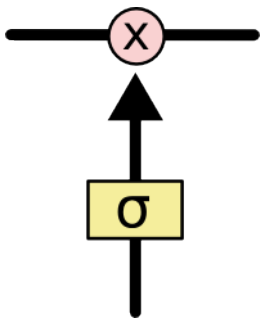
Es muy fácil que la información fluya sin cambios.



Long Short-Term Memory (LSTM)

El LSTM tiene la capacidad de eliminar o agregar información al estado de la celda, regulada cuidadosamente 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.



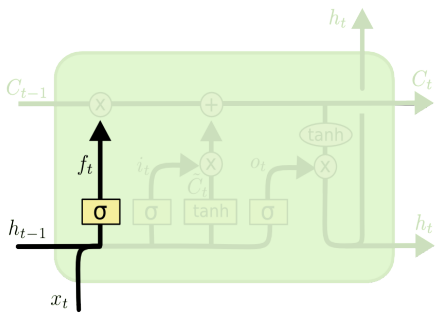
La capa sigmoidea genera números entre cero y uno, que describen cuánto de cada componente debe dejarse pasar. Un valor de cero significa "no dejar pasar nada", mientras que un valor de uno significa "dejar pasar todo". **Un LSTM tiene tres de estas puertas para proteger y controlar el estado de la celda**.

Long Short-Term Memory (LSTM)

El primer paso es decidir qué información se elimina del estado de la celda. Esta decisión se toma en la capa denominada **forget gate layer** (sigmoidea).

Esta capa mira h_{t-1} y x_t y genera un número aleatorio uniforme para cada número en el estado de la celda C_{t-1}. Donde,

- El valor 1 significa que se mantiene completamente
- El valor 0 significa que se deseche completamente



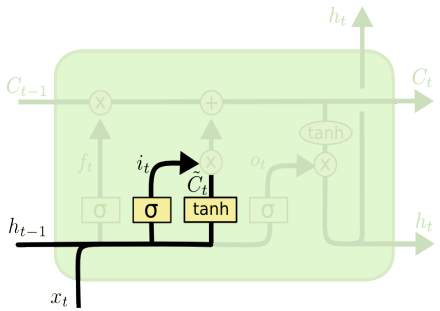
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Long Short-Term Memory (LSTM)

El siguiente paso es decidir qué nueva información almacenaremos en el estado de la celda, la cual consta de dos partes:

1) Una capa denominada **input gate layer** que decide los valores que serán actualizados.
2) Una capa **tanh** crea un vector de nuevos valores candidatos, $\hat{C}_t$, que podría agregarse al estado.

En el siguiente paso, se combinan estos dos para crear una actualización del estado.

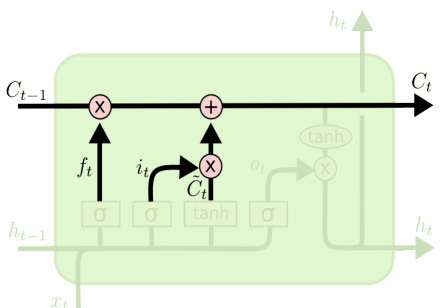


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Long Short-Term Memory (LSTM)

Ahora es el momento de actualizar el estado de la celda anterior, C_{t-1}, en el nuevo estado de la celda C_t.

1) Se multiplica el estado antiguo por f_t, olvidando las cosas que se decidió olvidar antes.
2) Luego se agrega $\hat{C}_t$. El nuevo resultado representan los nuevos valores candidatos escalados según cuanto se decidió escalar cada valor de estado.

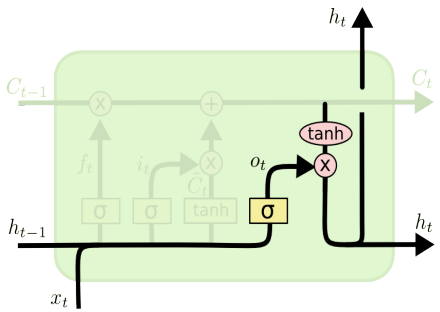


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Long Short-Term Memory (LSTM)

Finalmente se debe decidir que se va a generar. Esta salida estará basada en el estado de nuestra celda, pero será una versión filtrada.

1) Se ejecuta una capa sigmoidea que decide que partes del estado de la celda se van a generar.
2) Luego se pasa al estado celular **tanh** (empujar los valores para que estén entre -1 y 1), y multiplicarlo por la salida de la puerta sigmoidea, de modo que solamente produzcan las partes que se decidieron.



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

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

Existen variantes de la Red LSTM, pero en general es una versión estándar de dicho modelo.

Long Short-Term Memory (LSTM)

Como tal, se puede usar para crear grandes redes recurrentes que, a su vez, se pueden usar para abordar problemas de secuencia difíciles en el aprendizaje automático y lograr resultados de vanguardia.

En lugar de neuronas, las redes LSTM tienen bloques de memoria que están conectados a través de capas.

- Un bloque tiene componentes que lo hacen más inteligente que una neurona clásica y una memoria para secuencias recientes.
- Un bloque contiene puertas que administran el estado y la salida del bloque.
- Un bloque opera sobre una secuencia de entrada y cada puerta dentro de un bloque utiliza las unidades de activación sigmoidea para controlar si se activan o no, lo que condiciona el cambio de estado y la adición de información que fluye a través del bloque.

Long Short-Term Memory (LSTM)

Hay tres tipos de puertas dentro de una unidad:

- Forget Gate: decide condicionalmente qué información desechar del bloque.
- Puerta de entrada: decide condicionalmente qué valores de la entrada actualizar el estado de la memoria.
- Puerta de salida: decide condicionalmente qué generar en función de la entrada y la memoria del bloque.

Cada unidad es como una mini máquina de estado donde las puertas de las unidades tienen pesos que se aprenden durante el procedimiento de entrenamiento.

Es posible ver cómo se puede lograr un aprendizaje y una memoria sofisticados a partir de una capa de LSTM, y no es difícil imaginar cómo las abstracciones de orden superior pueden superponerse con múltiples capas de este tipo.

Long Short-Term Memory (LSTM)

Es una arquitectura de red neuronal recurrente artificial (RNN) utilizada en el campo del aprendizaje profundo. A diferencia de las redes neuronales de avance estándar, LSTM tiene conexiones de retroalimentación. No solo puede procesar puntos de datos individuales (por ejemplo, imágenes), sino también secuencias completas de datos (como entradas de voz o video).

Los modelos LSTM pueden almacenar información durante un período de tiempo.

Esta característica es extremadamente útil cuando se trata de series de tiempo o datos secuenciales. Cuando utilizamos un modelo LSTM, somos libres y podemos decidir qué información se almacenará y qué se descartará. Hacemos eso usando las "puertas". La comprensión profunda de LSTM está fuera del alcance de esta publicación, pero si está interesado en aprender más, eche un vistazo a las referencias al final de esta publicación.

Lecturas

Alonso Rodriguez. Análisis de las Series Temporales a la luz de Deep Learning. Anuario Jurídico y Económico Escurialense, LII (2019) 257-276 / ISSN: 1133-3677.

Ho. Autoregressive Models in Deep Learning.

Hyndman. Forecasting: Principles and Practice. Punto: 11.3 Neural network models.

Medium. An Autoregressive Neural Network Approach to Forecasting Bitcoin Price.

Namini et al.. A Comparison of ARIMA and LSTM in Forecasting Time Series.

Olah. Understanding LSTM Networks.

Fin