Redes Neutrales Recurrentes (RNN)

Intuición

Luis R. Furlán

Octubre de 2021

RNN

Antecedentes

- Hemos visto el uso de redes neuronales (NN) para problemas de:
  - Clasificación
  - Regresión
- Nos falta ver cómo trabajar información secuencia con NN

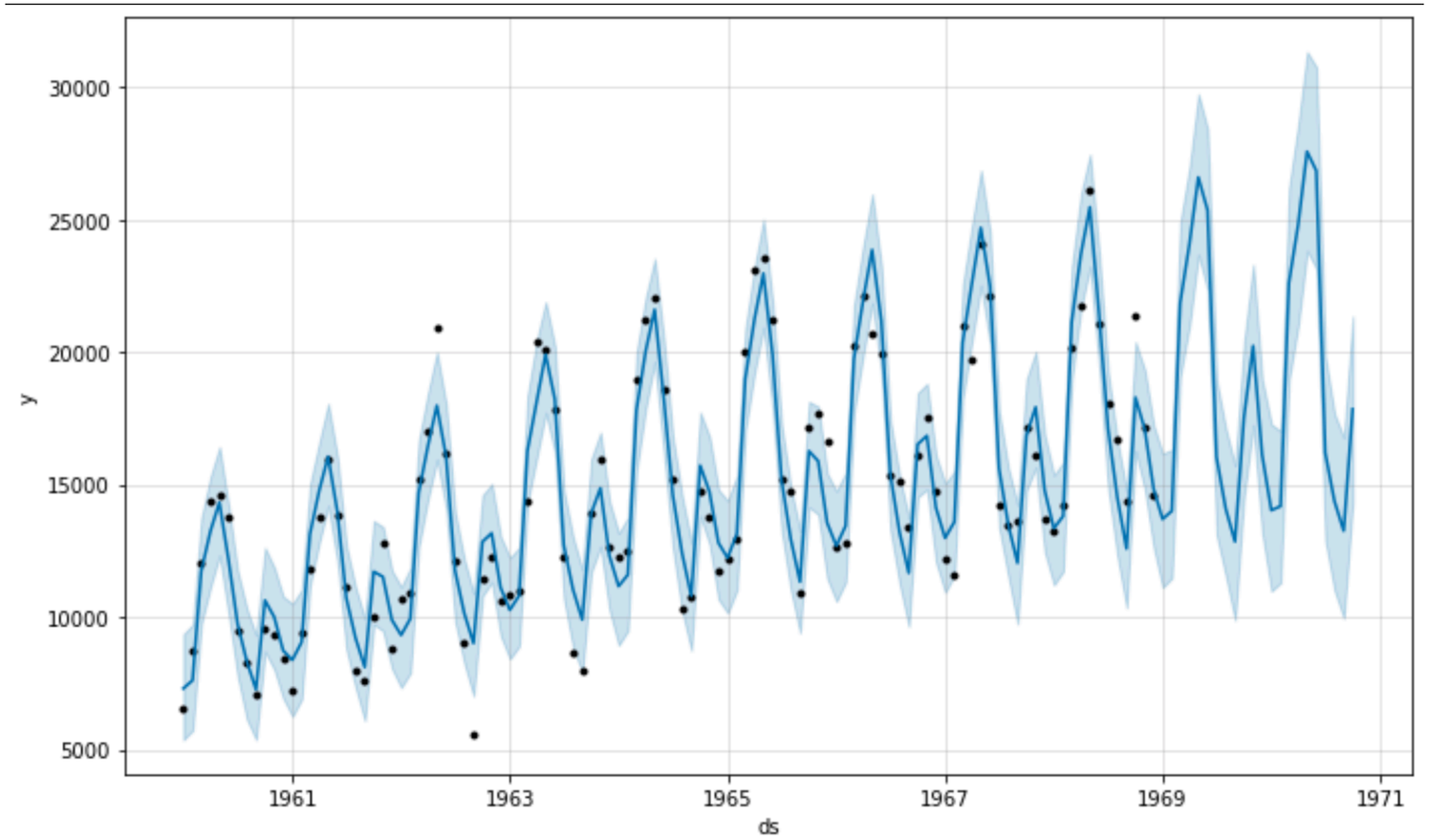
RNN

Antecedentes

- Así como las CNN son más efectivas para datos de imágenes 2-D
- Las RNN son más efectivas con datos secuenciales:
  - Datos de venta con fechas
  - Secuencia de textos
  - Datos de latidos del corazón
  - etc.

RNN

Series de Tiempo



Recordarán esta gráfica: Venta de Autos en Canadá

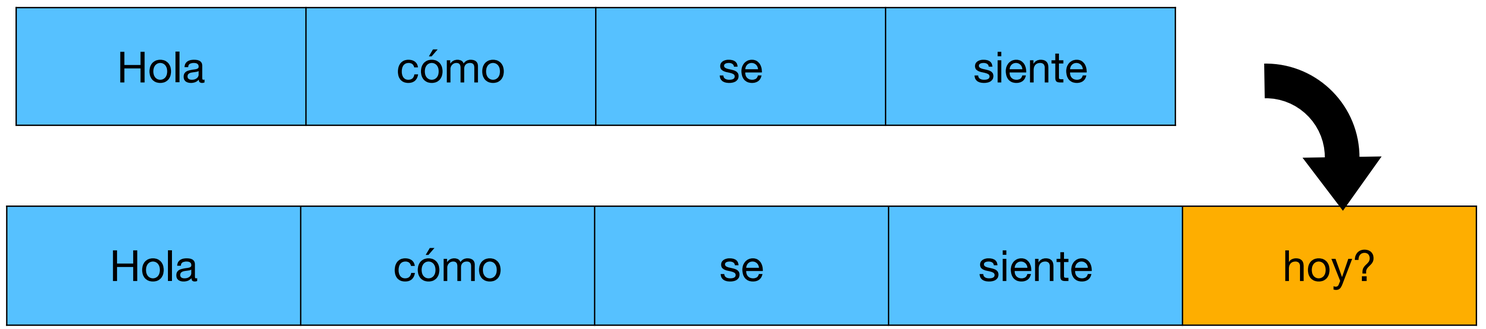
Los puntos negros eran los datos históricos, La línea azul es la predicción.

La línea azul va más allá de los puntos porque, justamente, queríamos predecir más allá de los datos históricos

De eso es lo que se trata…predecir el futuro

RNN

Secuencias



Vamos a darle a la red la secuencia de palabras: Hola cómo se siente

Queremos que la red prediga la palabra: hoy

Una RNN puede hacer esto

RNN

Vistazo general

- Vamos a ver
  - Teoría de las RNN
  - Teoría de Long-Short Term Memory (LSTM) y Gated Recurrent Units (GRU)
  - Implementación básica de una RNN
  - Series de tiempo con una RNN
  - Ejercicio

RNN

Secuencias

- Son colecciones de datos pero con un orden específico
- Ejemplos:
  - Datos de series de tiempo (Ventas)
  - Oraciones
  - Audio
  - Trayectorias de automóviles
  - Música

RNN

Secuencias

- Imaginemos una secuencia simple:
  - [1, 2, 3, 4, 5, 6]
- ¿Seremos capaces de predecir una secuencia similar desplazada un paso al futuro?
  - [2, 3, 4, 5, 6, 7]
- Es una posibilidad, pero no podemos estar 100% seguros porque es al futuro!

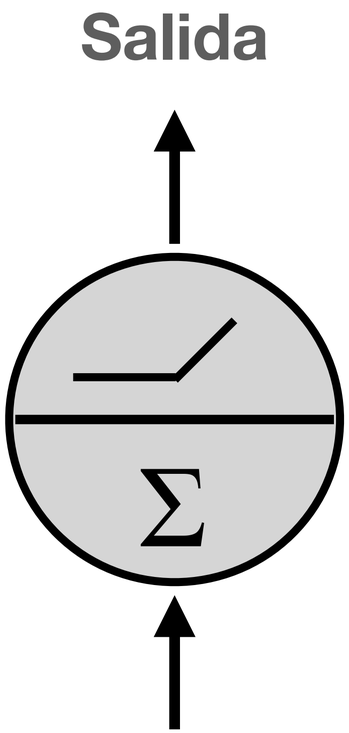
RNN

Secuencias

- Para hacer esto correctamente, necesitamos que…de alguna forma…una neurona pueda “conocer” de su propia historia de salida
- Una forma fácil de hacer esto, es de re-alimentar su entrada con su salida!

RNN

Neurona normal en una NN de propagación hacia el frente

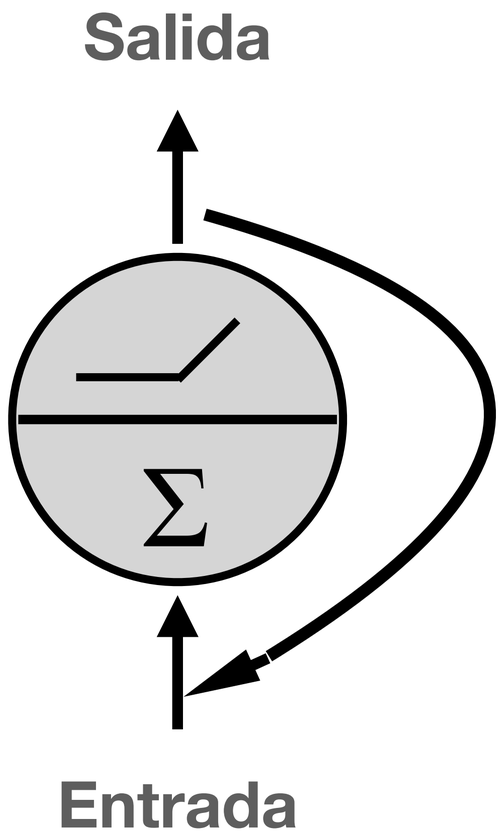


Función de activación

Agregación de entradas

RNN

Neurona recurrente

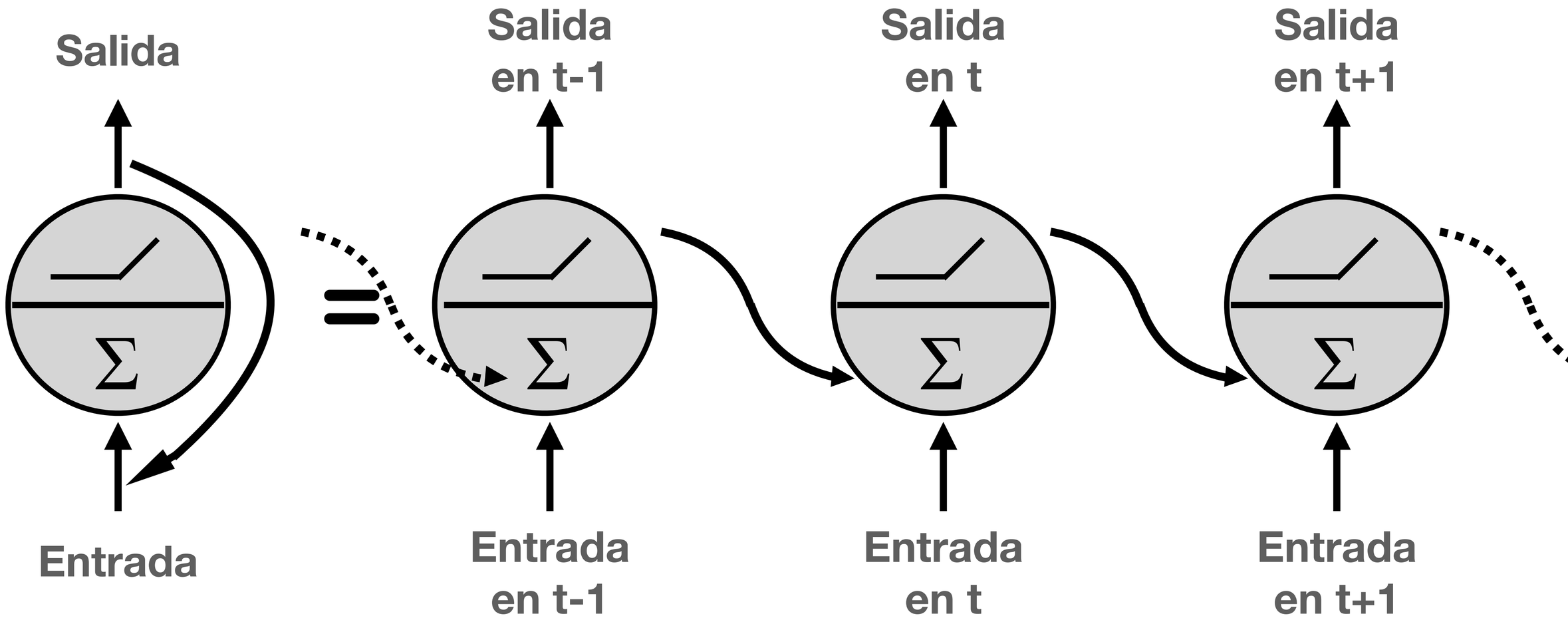


Pasa la salida de vuelta a la entrada

Al pasar el tiempo, ¿cómo se verá?

RNN

Neurona recurrente “desenrollada” (unroll)

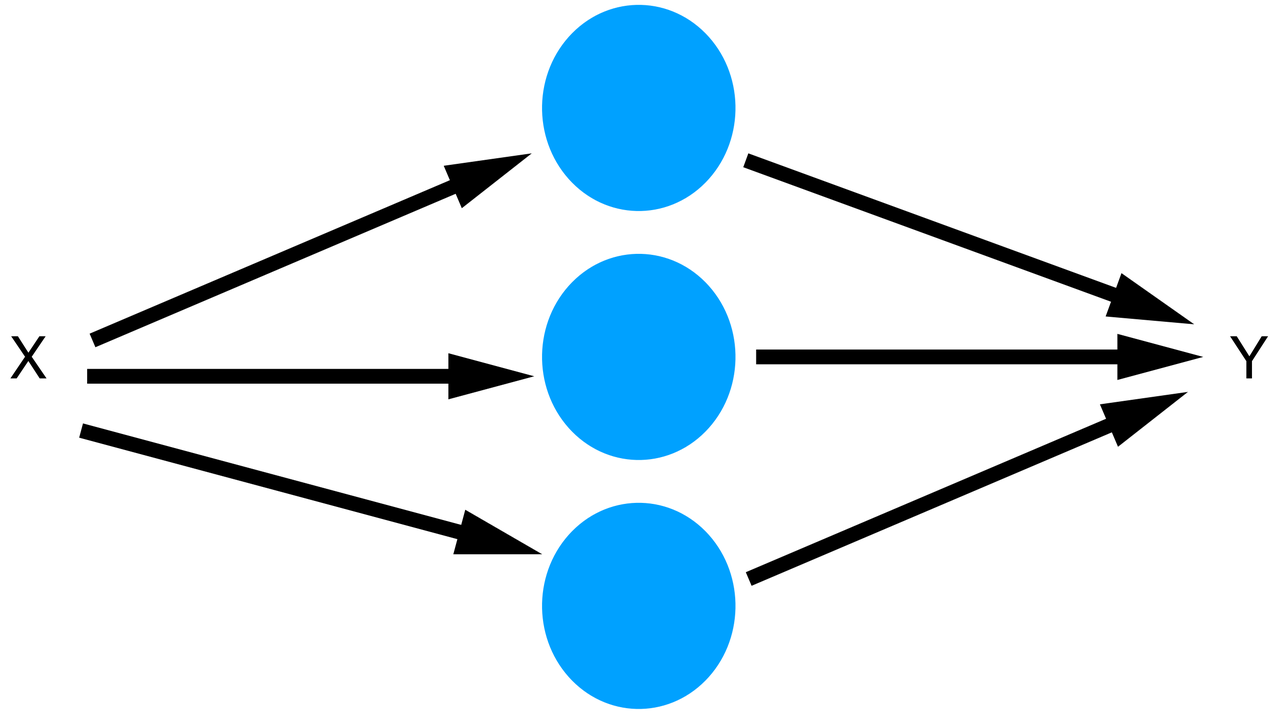


RNN

- A las celdas que son una función de entradas de pasos en tiempo anteriores, se les llama *cellulas de memoria (memory cells)*
- Las RNN también son flexibles en sus entradas y salidas, para ambas secuencias y valores vectoriales
- También se pueden crear capas enteras de neuronas recurrentes…

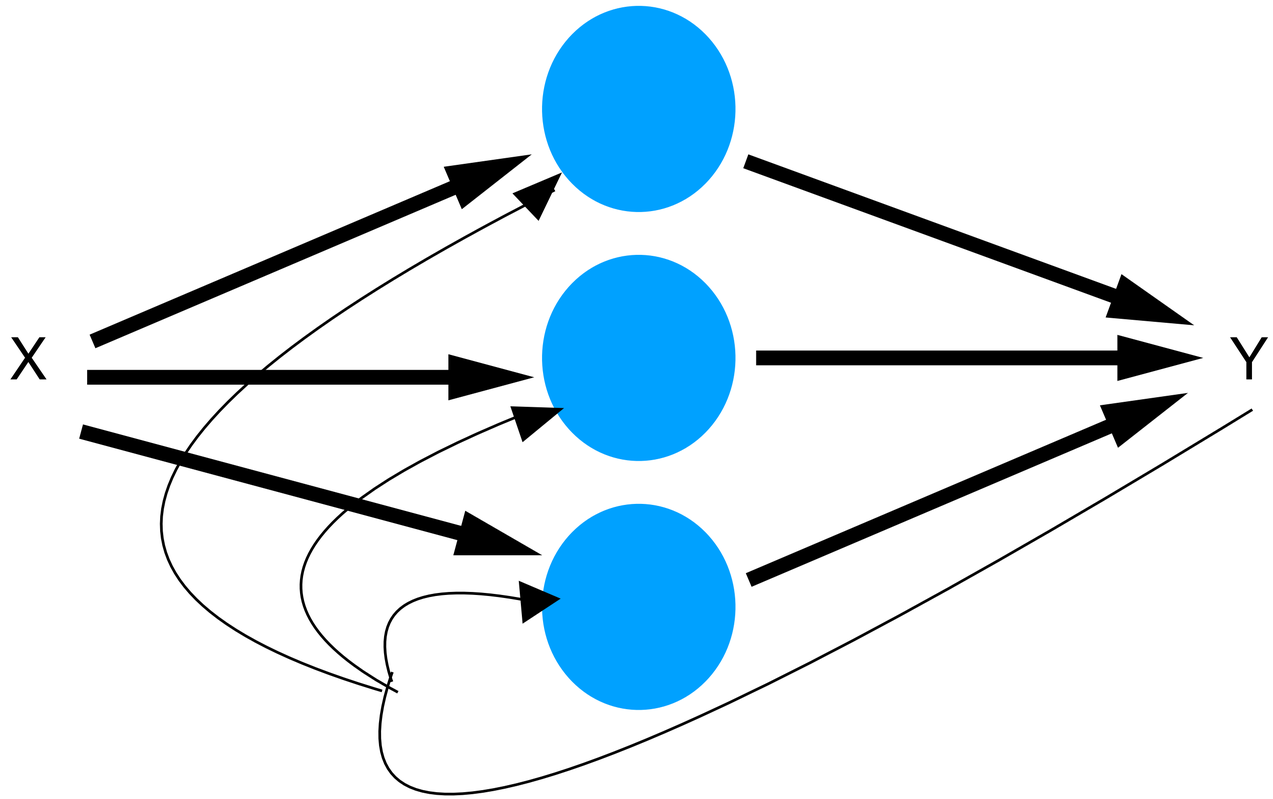
RNN

Capa de ANN con 3 neuronas



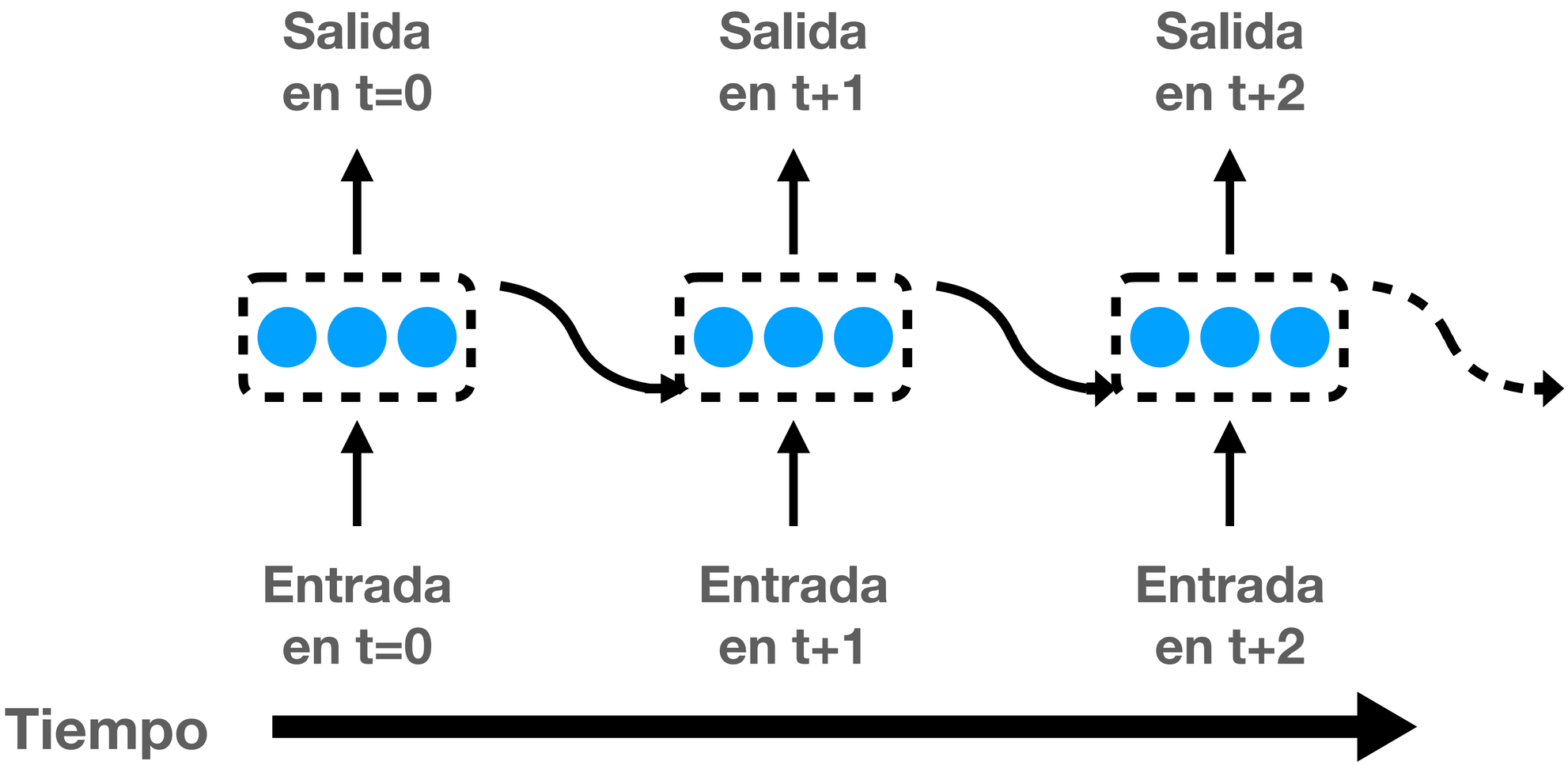
RNN

Capa de RNN con 3 neuronas



RNN

Capa “desenrollada”

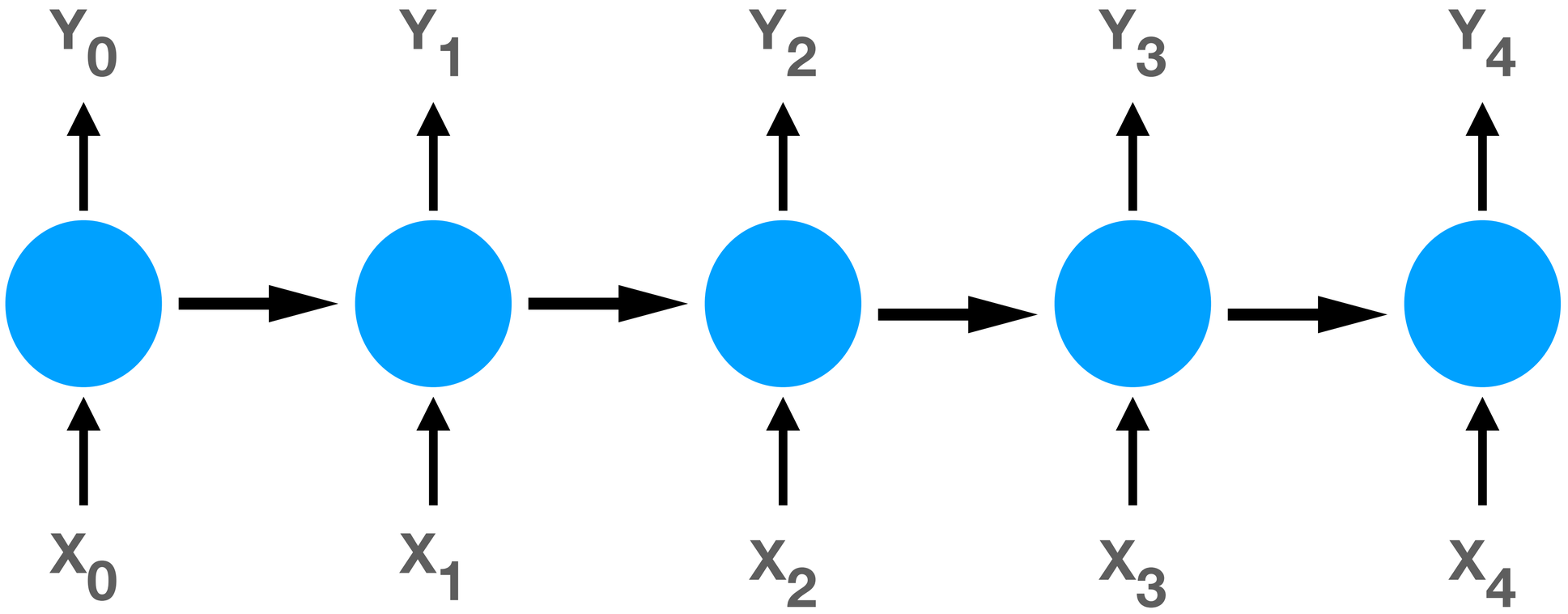


RNN

- Son muy flexibles en las entradas y salidas
- Veamos unos ejemplos

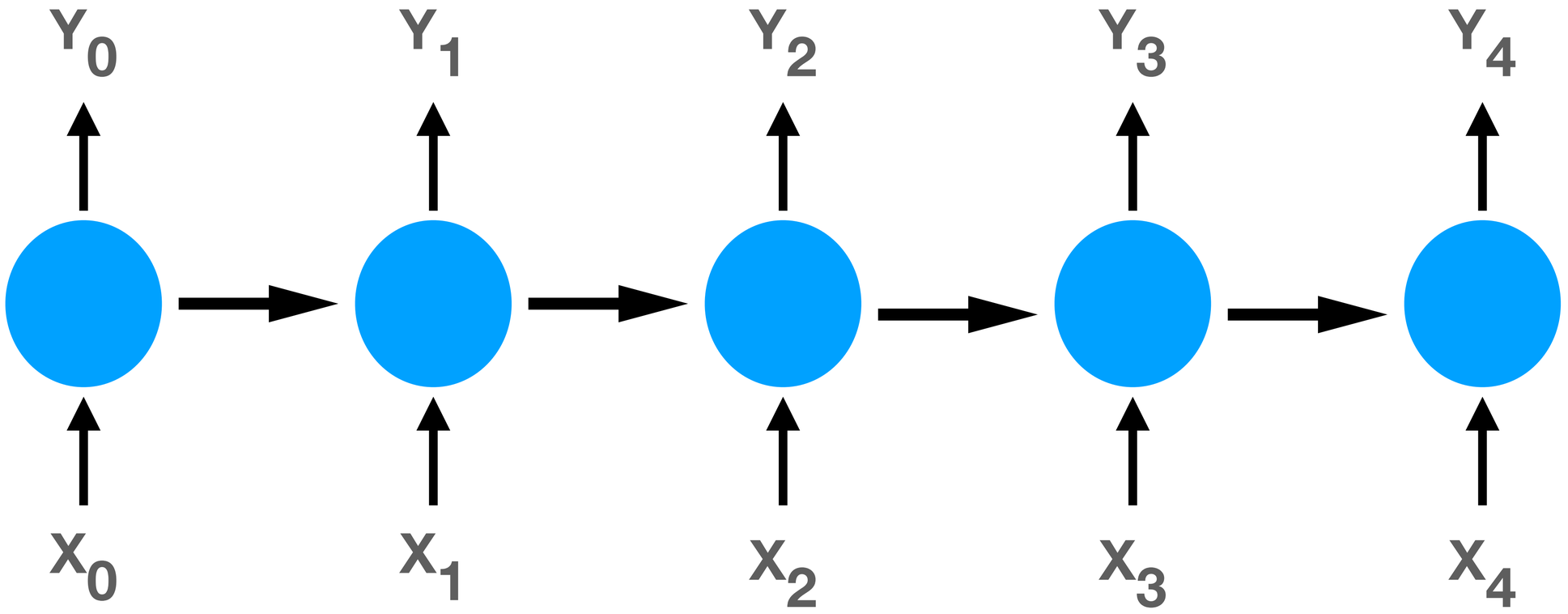
RNN

Modelo: Secuencia a secuencia (muchos a muchos)



RNN

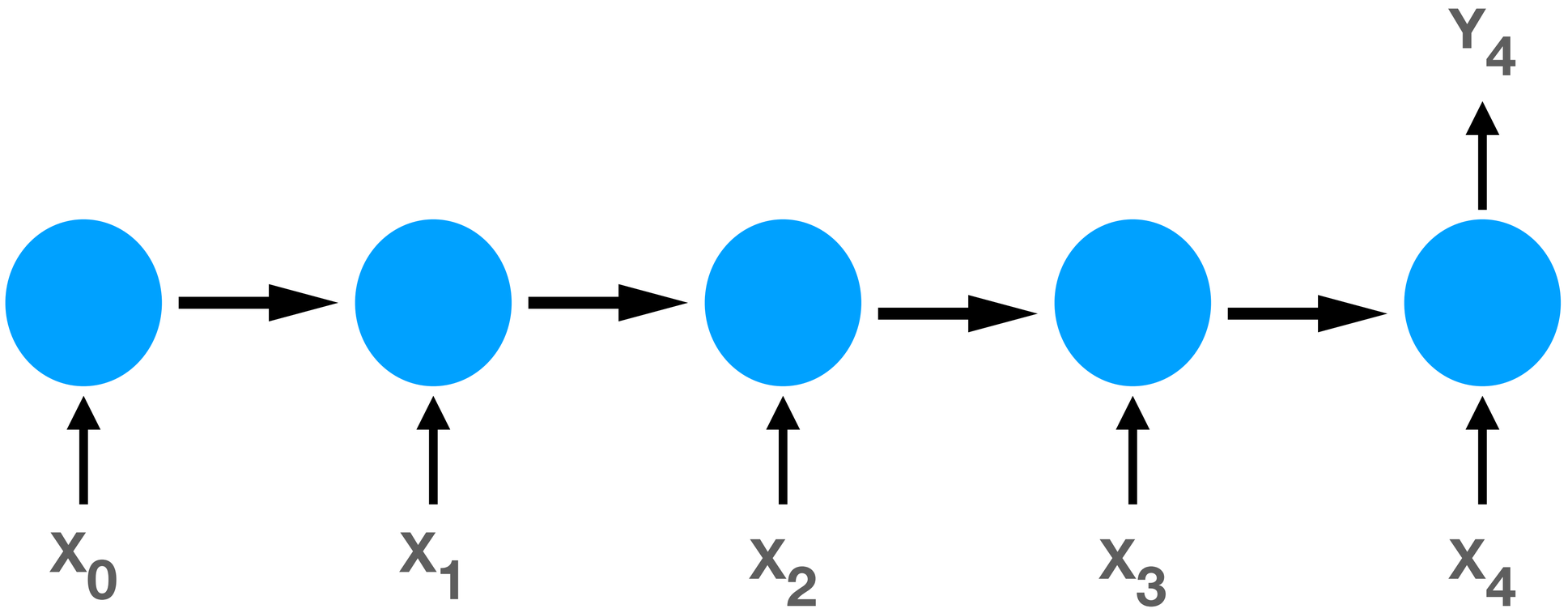
Ejemplo: dadas las 5 palabras previas, predecir las siguientes 5



Si se recibe una pregunta de 5 palabras, predecir la respuesta de 5 palabras

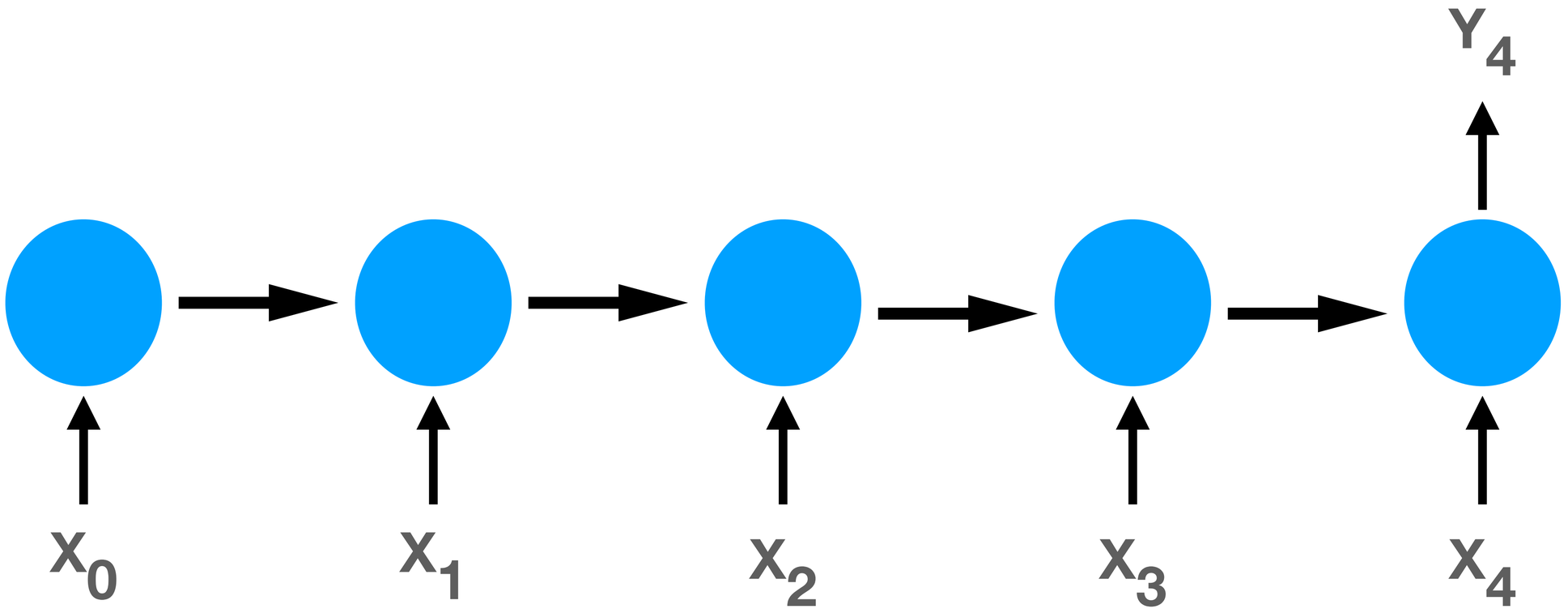
RNN

Modelo: Secuencia a vector (muchos a uno)



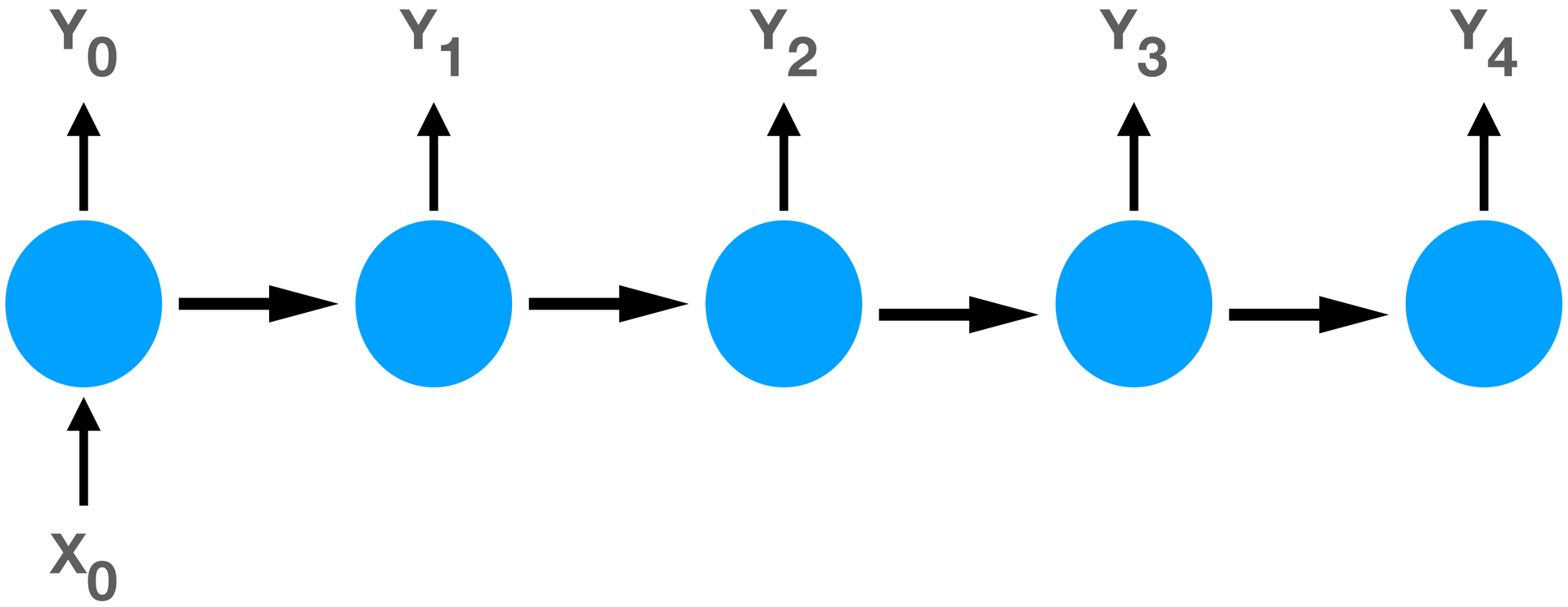
RNN

Ejemplo: dadas las 5 palabras previas, predecir la siguiente



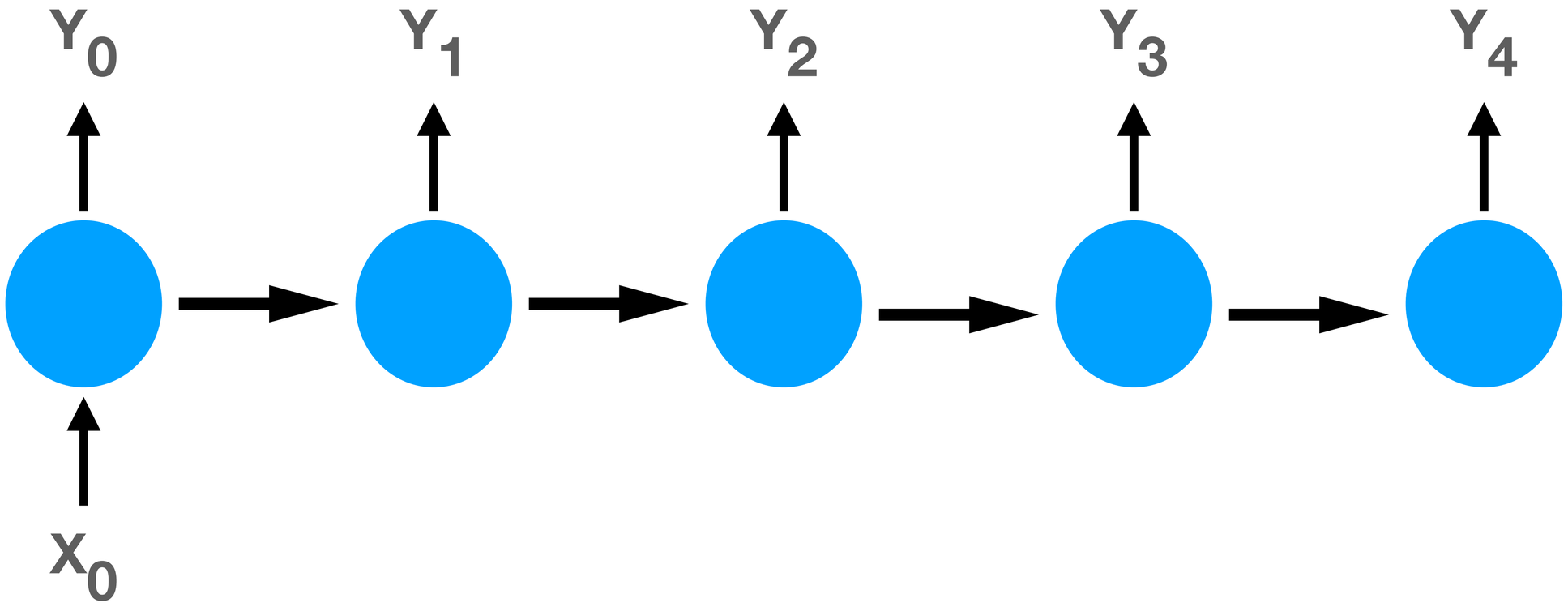
RNN

Modelo: Vector a secuencia (uno a muchos)



RNN

Ejemplo: dada una palabra, predecir las siguientes 5



RNN

Desventajas

- En realidad, solamente recuerda la salida anterior
- Sería fabuloso si pudiera recordar una historia más larga, no solo a corto plazo
- Si vemos el diagrama “desenrollado” solo se recibe la entrada del paso anterior
- Lo que esto causa es que se va perdiendo la historia más antigua

RNN

Desventajas

- También surge el problema de las gradientes “desvanecientes”
- Antes de pasar a ver las unidades de plazo largo-corto (Long Short Term Units - LSTM), veamos el asunto de las gradientes “desvanecientes”

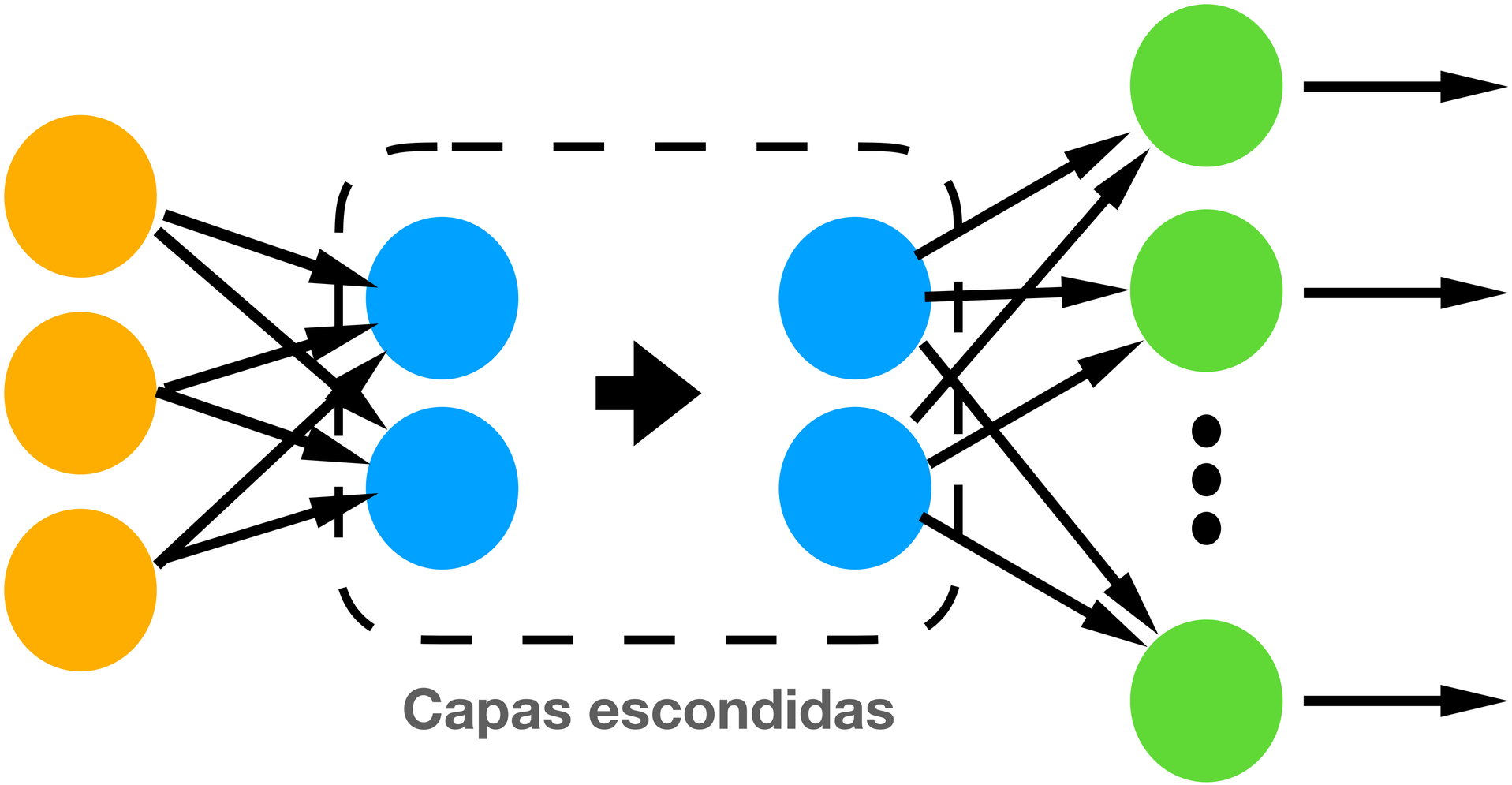
RNN

Gradientes “desvanecientes”

- Conforme nuestras redes se hacen más profundas y complejas, surgen dos inconvenientes:
  - Gradientes que “explotan”
  - Gradientes que desaparecen “desvanecientes”
- Recordemos que la gradiente se utiliza en nuestros cálculos para ajustar los pesos y sesgos en las redes

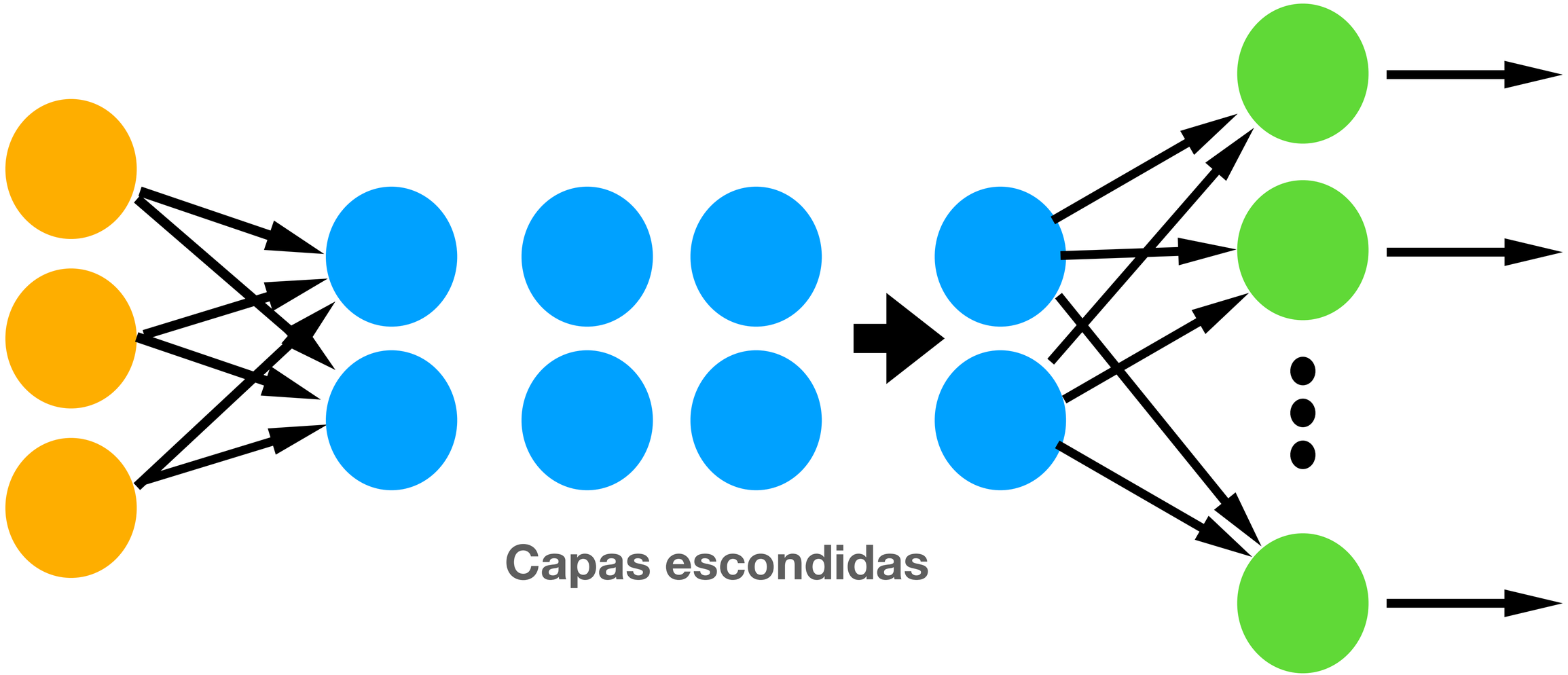
RNN

Pensémos en una red y su propagación al frente



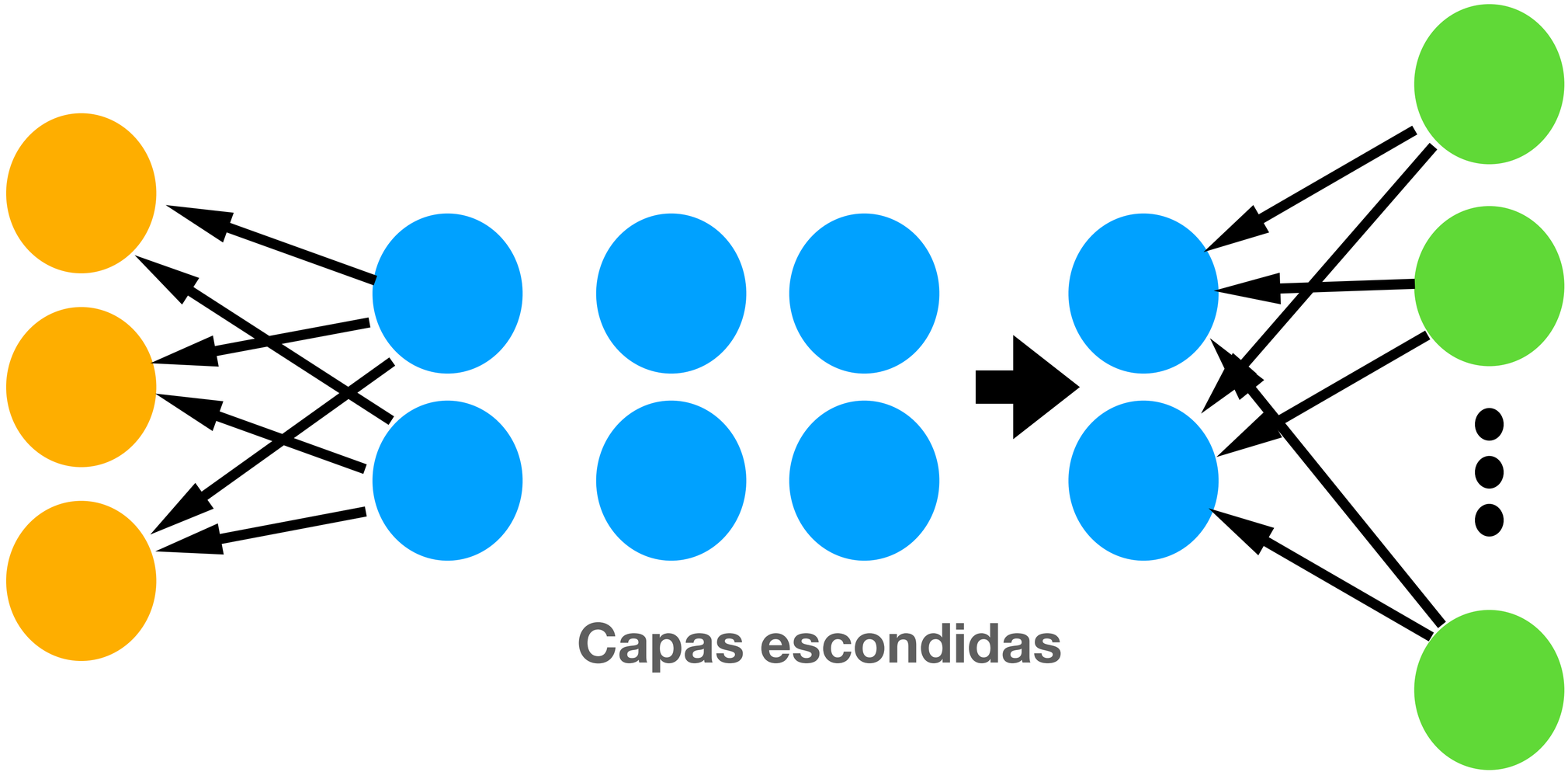
RNN

Para datos complejos, se requieren redes más profundas



RNN

El problema ocurre en la propagación hacia atrás



RNN

Propagación hacia atrás

- Retrocede de la capa de salida hasta la de entrada, propagando el error de gradiente
- Para redes profundas ocurren inconveniencias como la desaparición o, en algunos casos, la explosión de gradientes

RNN

Propagación hacia atrás

- Al ir hacia atrás, a las capas menores o primeras, las gradientes a menudo van disminuyendo, eventualmente causando que los pesos casi no cambien en las capas iniciales
- Lo opuesto también puede ocurrir, aunque no es tan frecuente, y los gradientes van aumentando hasta que “explotan”
- Esto es problemático porque lo que se quiere es que la red vaya detectando patrones, lo más cerca posible a la capa de entrada

RNN

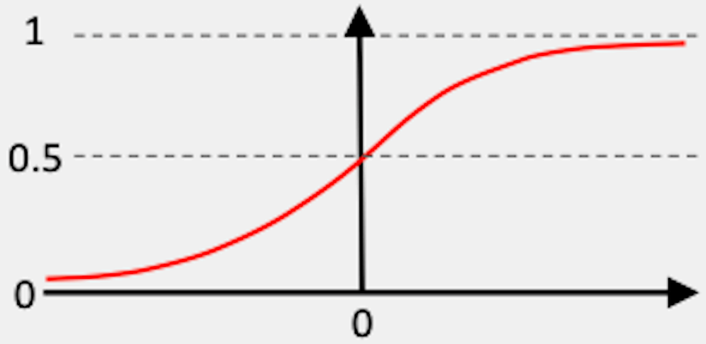
¿Porqué ocurre el desvanecimiento y explosión de gradientes?

Función de activación sigmoide

- Condensa todos los valores de entrada a un rango entre 0 y 1
- Entre más lejos la entrada esté de cero, la gradiente no cambia tanto

$$\sigma(a) = \frac{1}{1+e^{-a}}$$

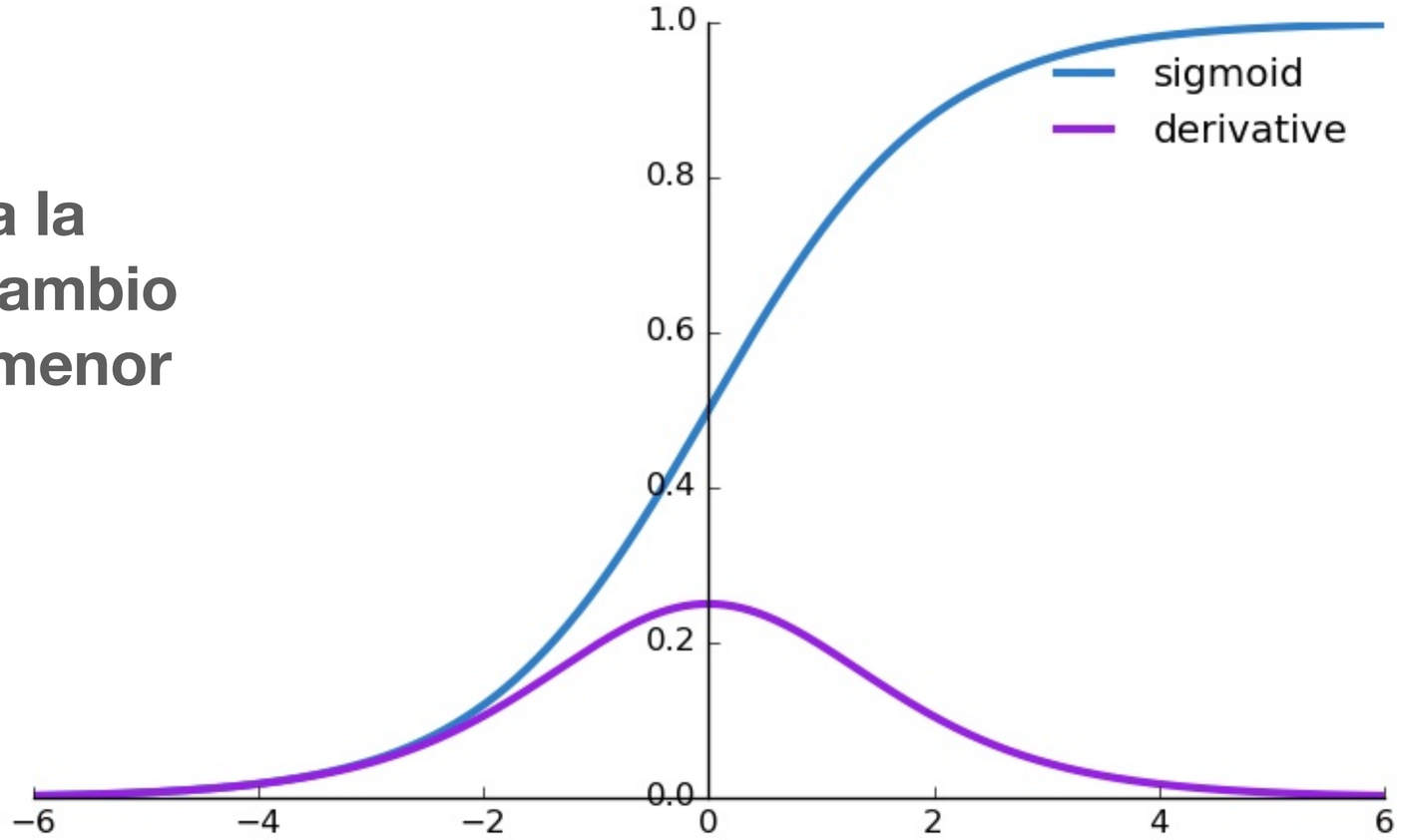
$$\frac{\partial \sigma(a)}{\partial a} = \sigma(a)(1-\sigma(a))$$



RNN

La derivada de la función sigmoide puede ser mucho menor!

Al ser pequeña la gradiente, el cambio en el peso es menor



RNN

¿Porqué ocurre el desvanecimiento y explosión de gradientes?

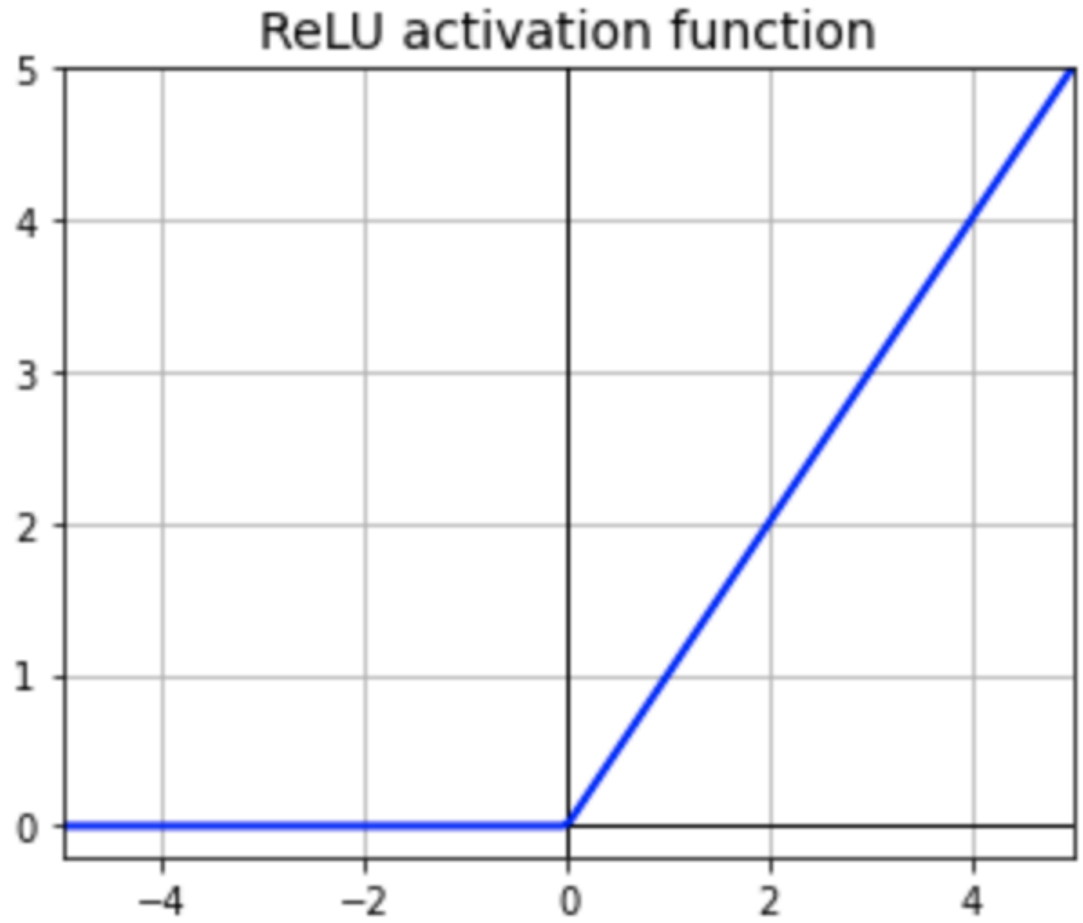
- Cuando **n** capas escondidas utilizan una activación como la función sigmoide, ocurre que **n** derivadas pequeñas son multiplicadas entre sí

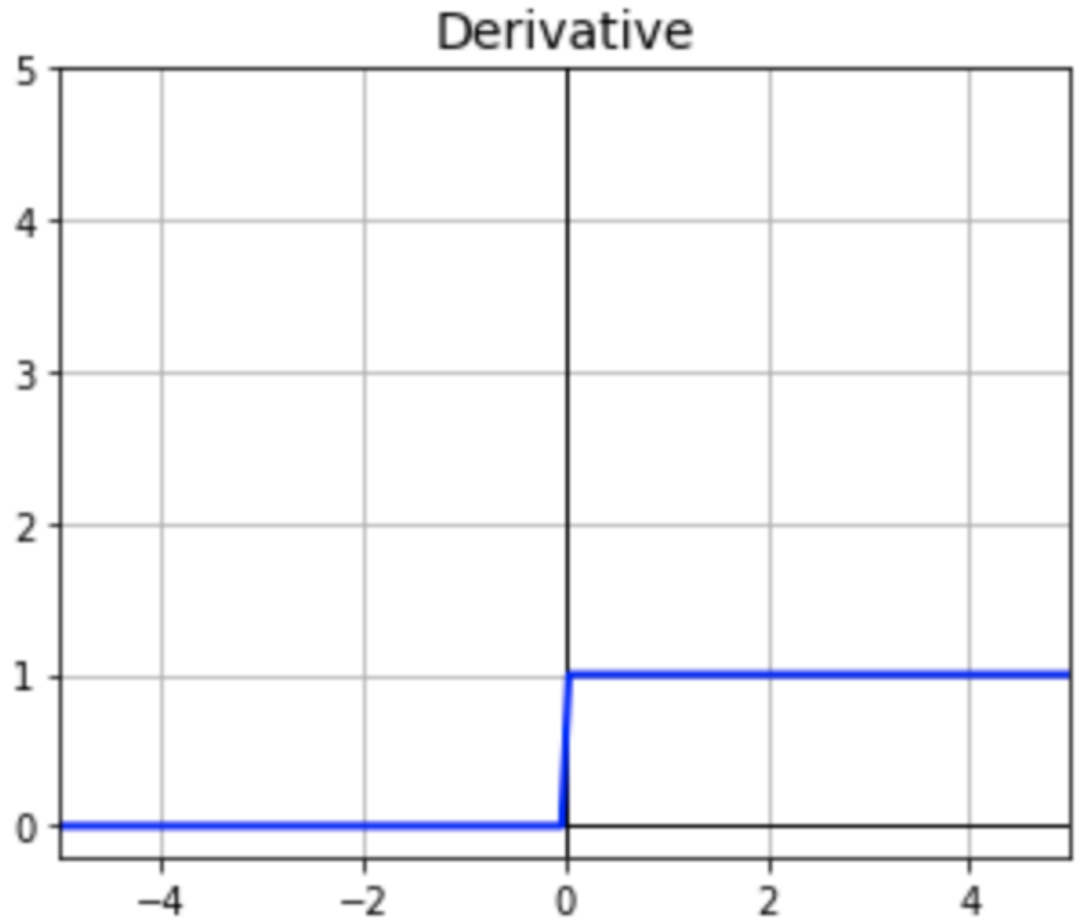
- La gradiente podría disminuir exponencialmente conforme la propagación hacia atrás vaya llegando a las capas iniciales

- Se podría pensar en cambiar la función de activación

RNN

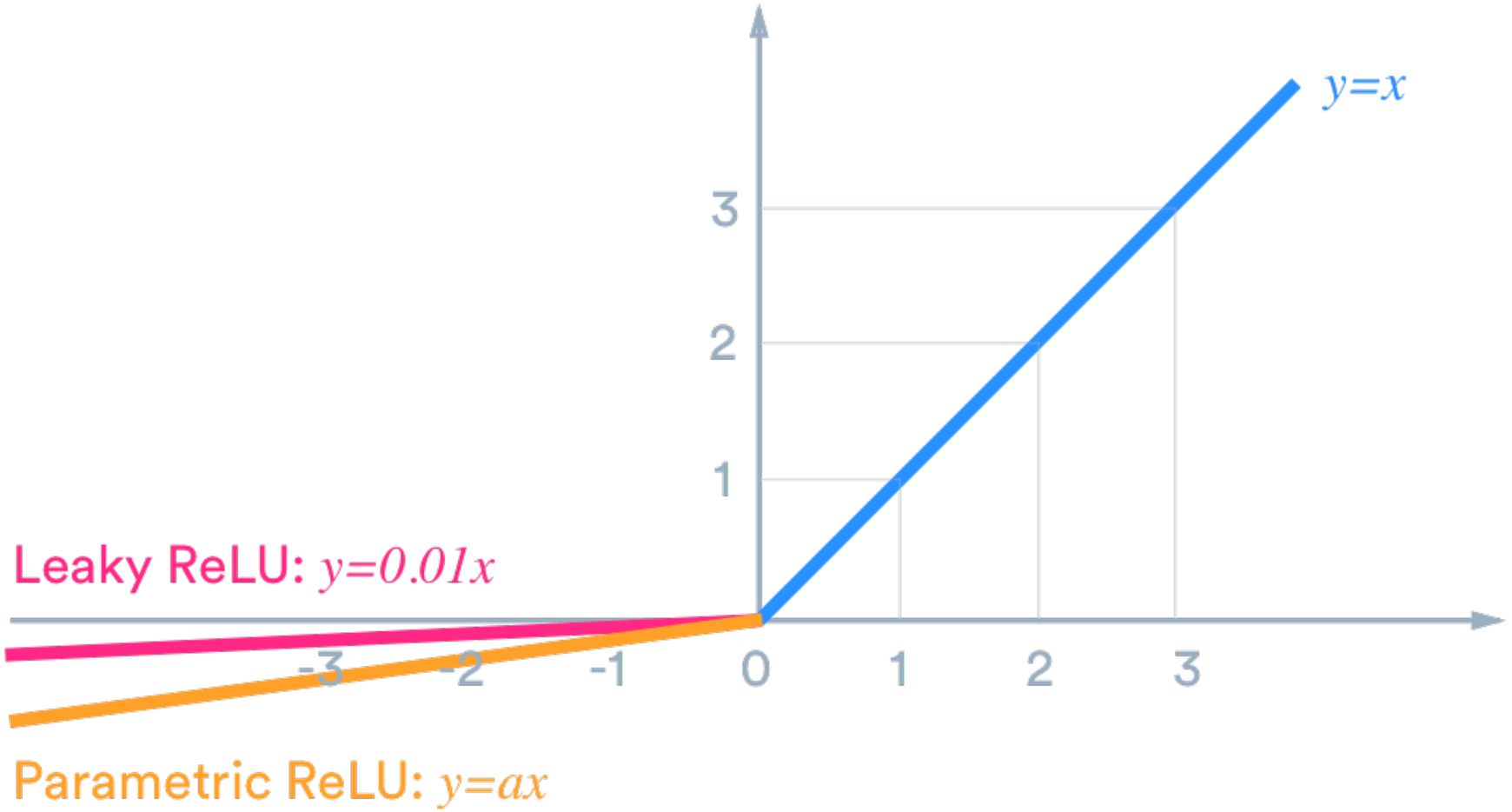
La derivada le la función ReLU es una constante!





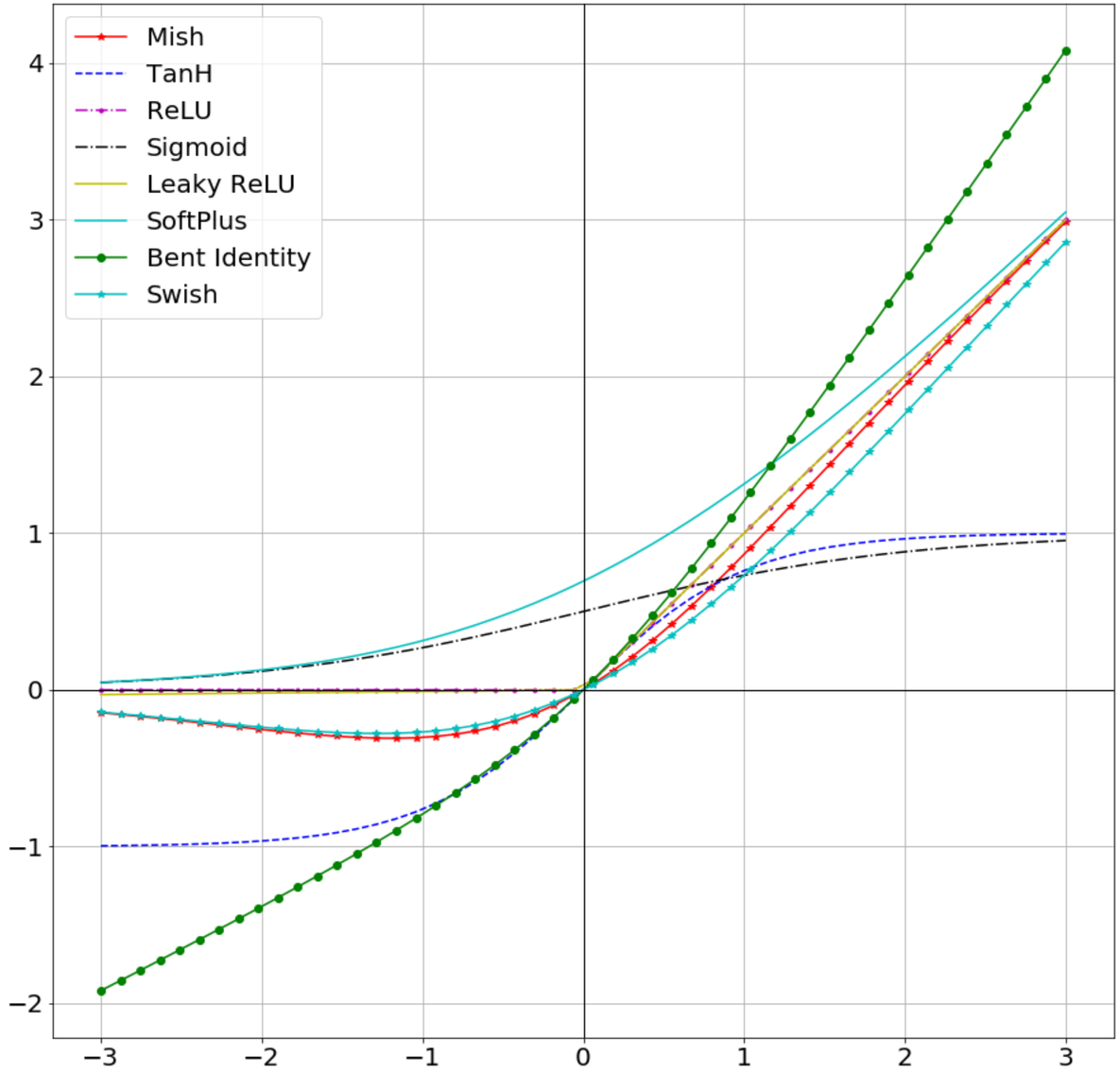
RNN

Hay variaciones de la ReLU



RNN

Otras variaciones de la ReLU



RNN

Otra solución - Normalización de Tandas

El modelo normaliza cada tanda utilizando la media y desviación estándar de cada tanda.

RNN

Otra solución - Seleccionar una inicialización diferente de los pesos

Una muy popular es la Inicialización Xavier.

RNN

Para la posible explosión de gradientes

Además de utilizar la normalización de tandas, también se utiliza la técnica…no muy elegante, pero funcional…de “recorte” de gradientes. Simplemente se predeterminan los límites y luego se recortan las gradientes si los sobrepasan.

Ejemplo: Límites entre -1 y 1

RNN

Solución

Las RNN para Series de Tiempo presentan sus propios retos. Es por eso que exploraremos unidades de neuronas especiales (LSTM) que ayudan a resolver estos problemas.

RNN

LSTM y GRU

- Todas las soluciones planteadas funcionan bien con las RNN
- Con Series de Tiempo, sin embargo, debido a lo grande de la entrada de las Series, estas podrían hacer que el entrenamiento se alargue mucho
- Una solución posible es acortar los pasos de tiempo para la predicción. En vez de cada 48 períodos, bajarlo a 24, por ejemplo
- Sin embargo, reducir los pasos hace que el modelo no sea tan bueno para predecir tendencias más largas

RNN

LSTM y GRU

- El otro tema que hemos visto es que las RNNs, luego de un tiempo corto, empiezan a “olvidar” las primeras entradas
- Con Series de Tiempo, sin embargo, debido a lo grande de la entrada de las Series, estas podrían hacer que el entrenamiento se alargue mucho
- Una solución posible es acortar los pasos de tiempo para la predicción. En vez de cada 48 períodos, bajarlo a 24, por ejemplo
- Sin embargo, reducir los pasos hace que el modelo no sea tan bueno para predecir tendencias más largas

¡Se necesita alguna forma de memoria a largo plazo para la red!

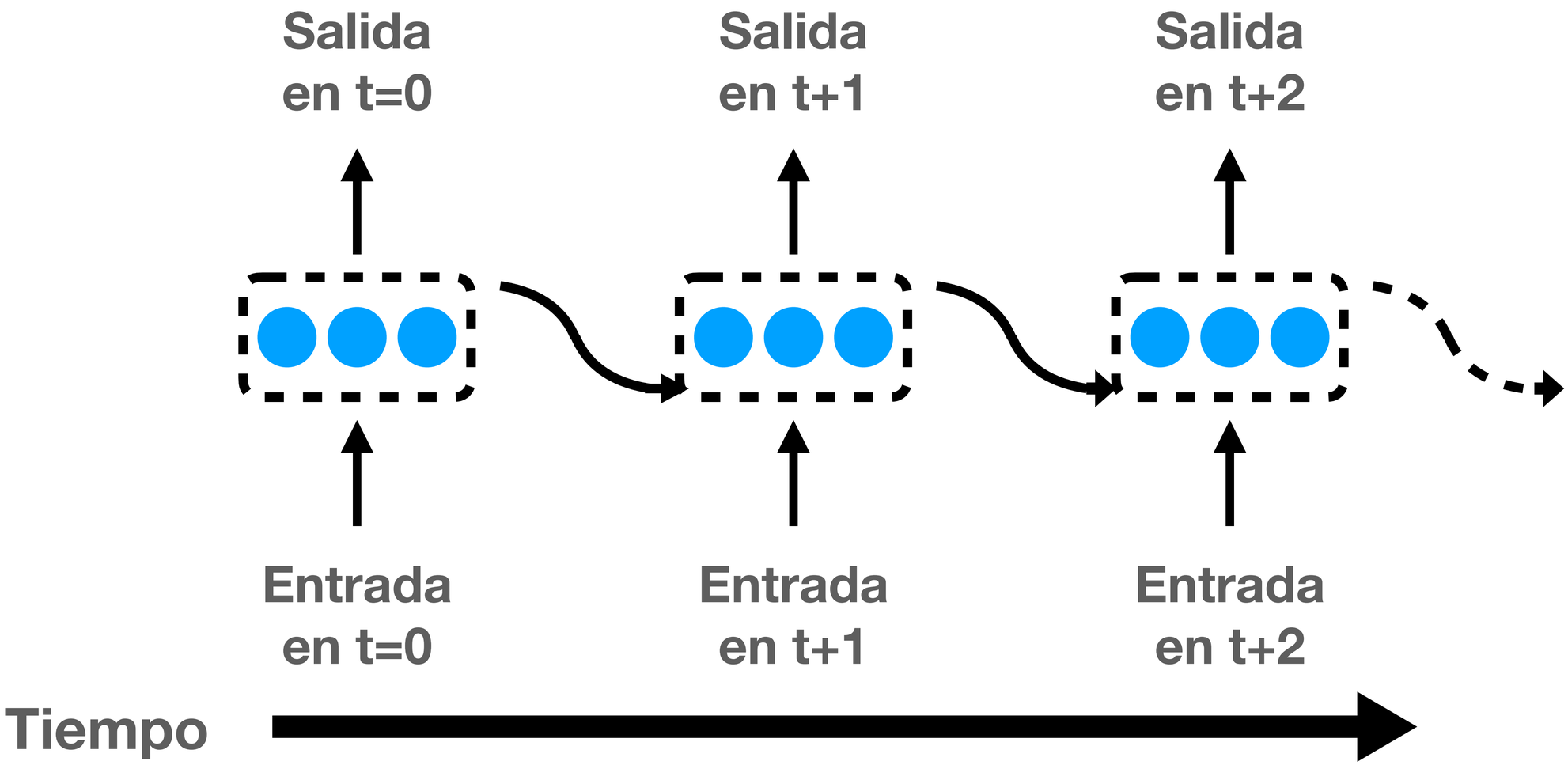
RNN

LSTM y GRU

- Las celdas, o unidades LSTM (siglas de Long Short Term Memory) se crearon para resolver estos problemas
- Veamos cómo funcionan

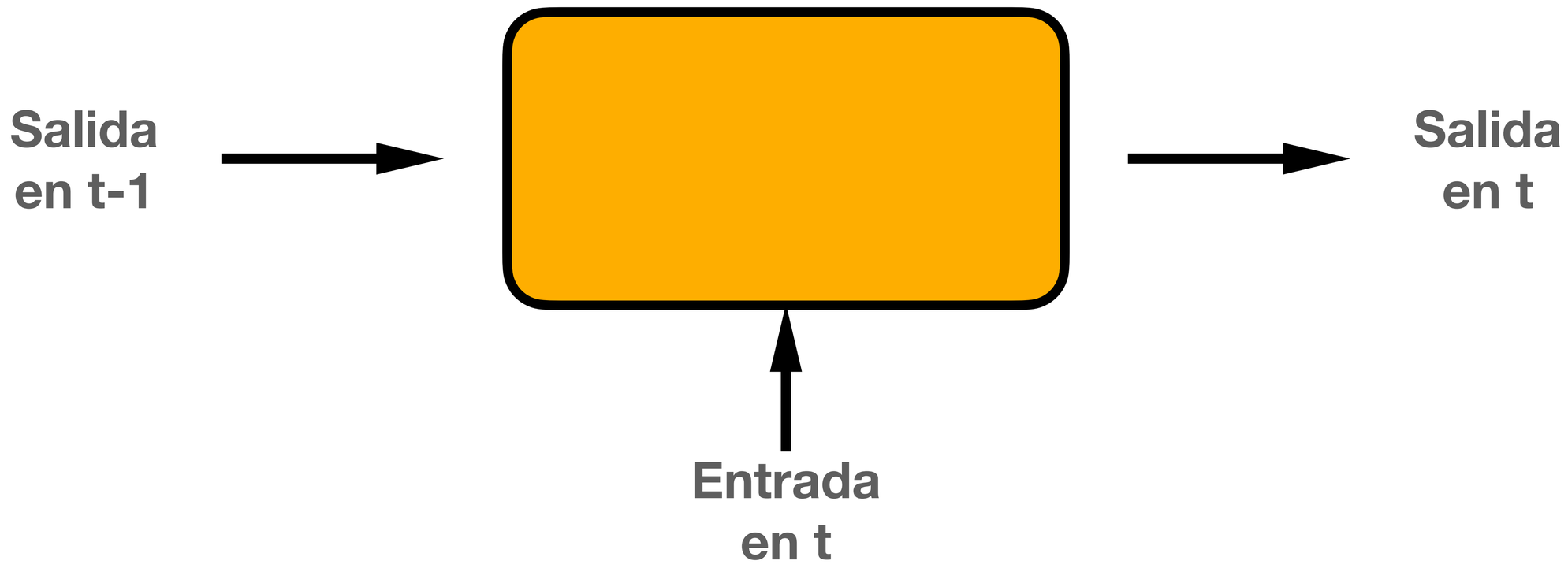
RNN

Recordemos cómo funciona una RNN típica



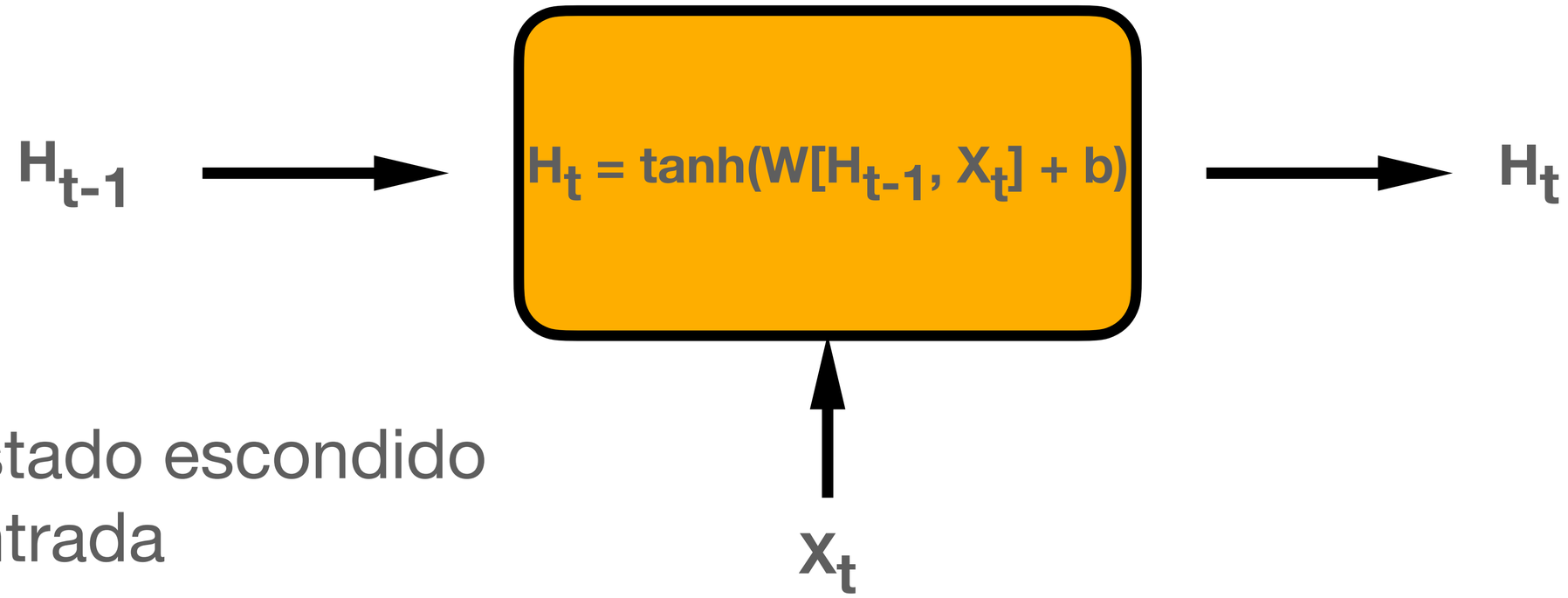
RNN

Una sola celda



RNN

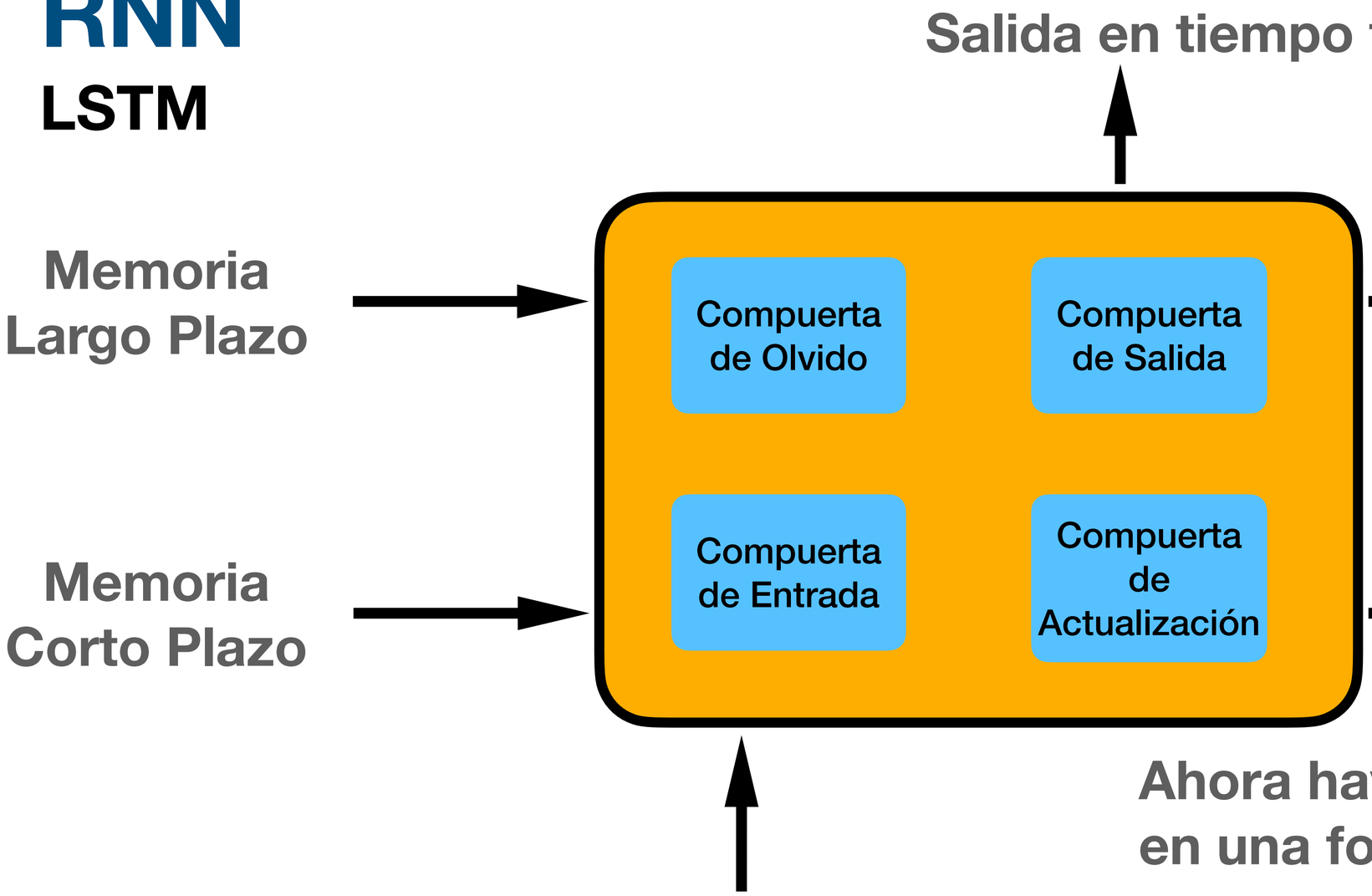
Una sola celda



H - estado escondido
X - Entrada

RNN

LSTM

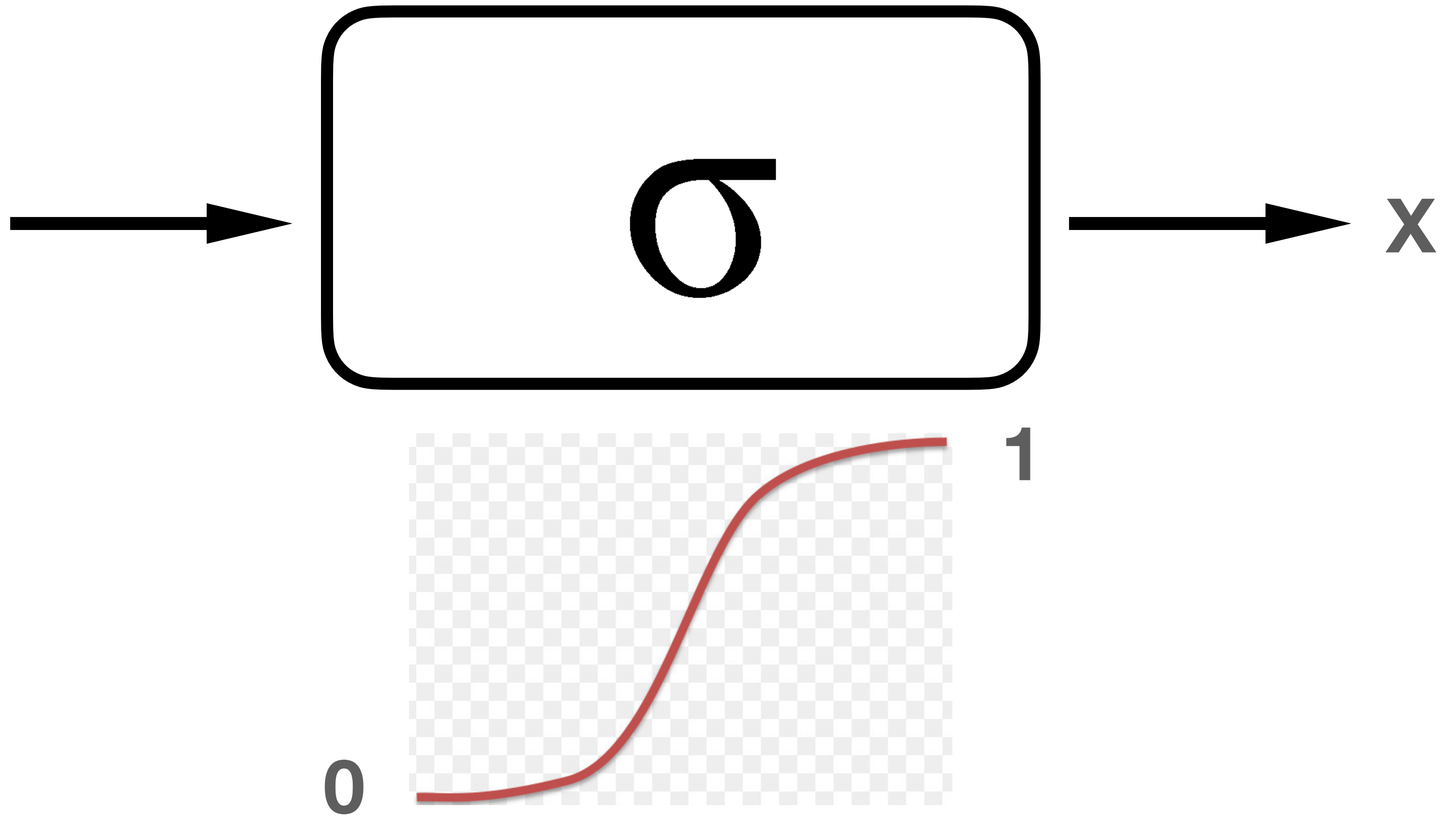


Ahora hay 4 capas interactuando en una forma especial

- Olvido - decide qué olvidar
- Entrada - decide qué dejar entrar

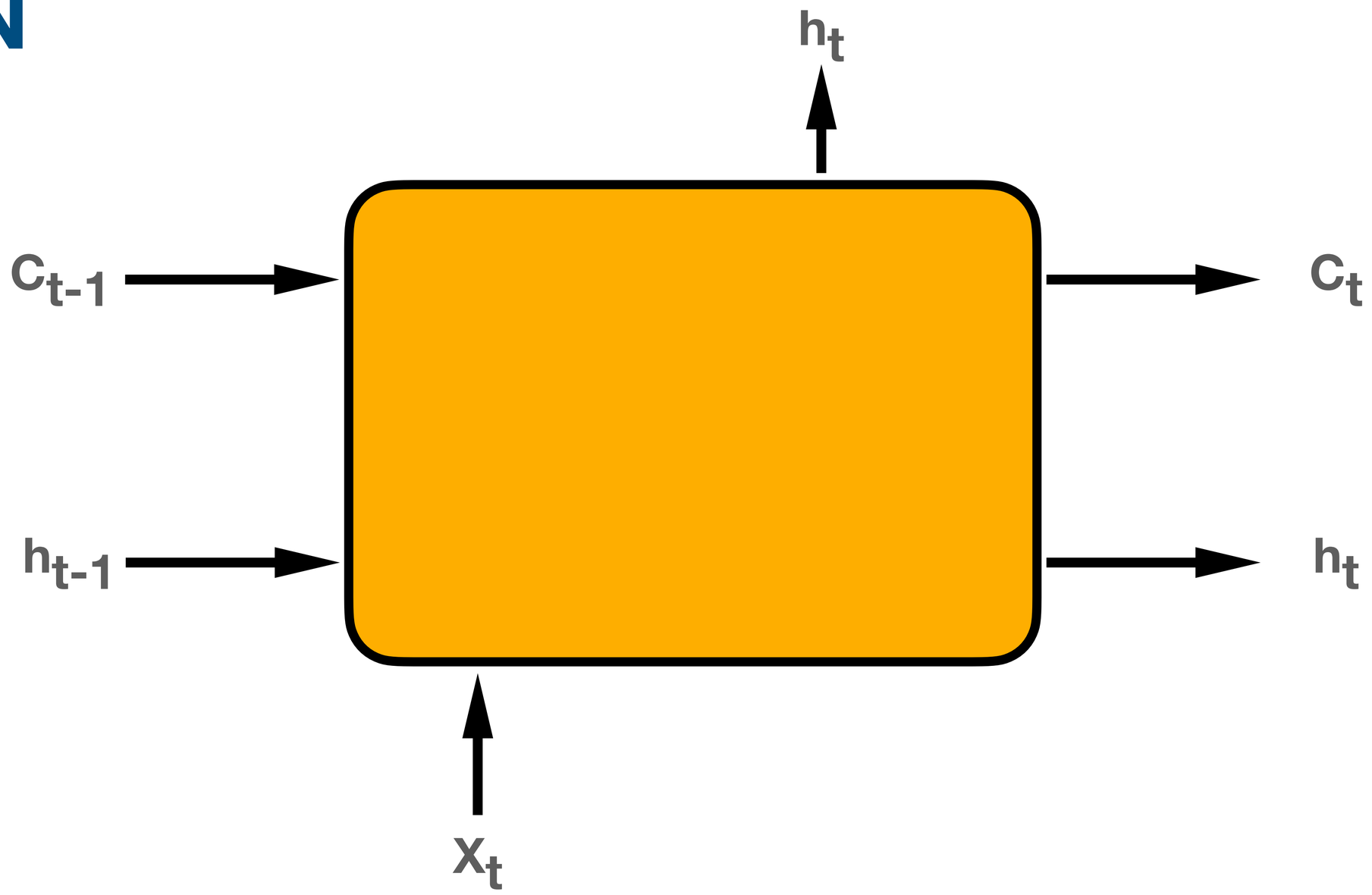
RNN

Las compuertas opcionalmente dejan pasar la información



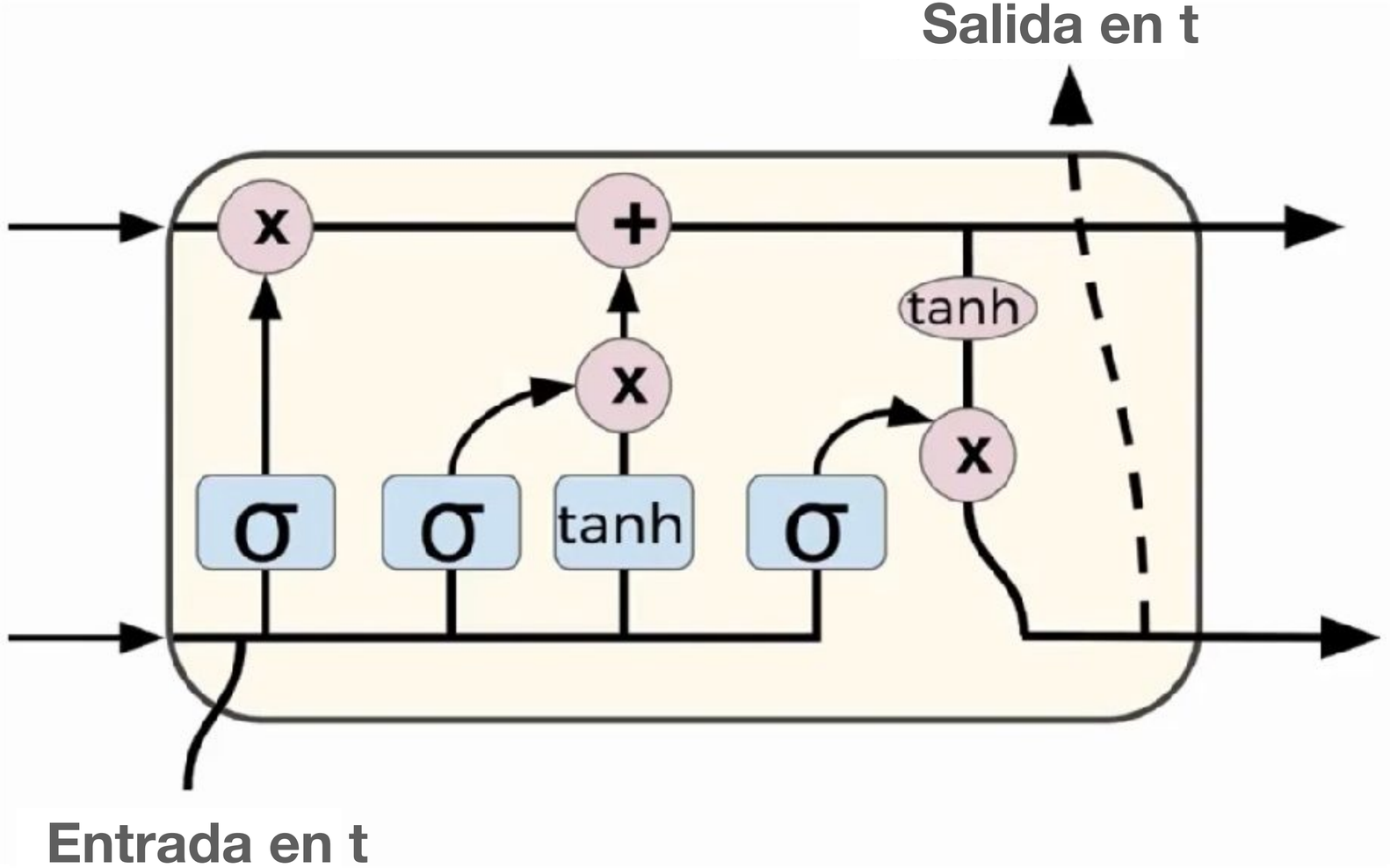
RNN

LSTM



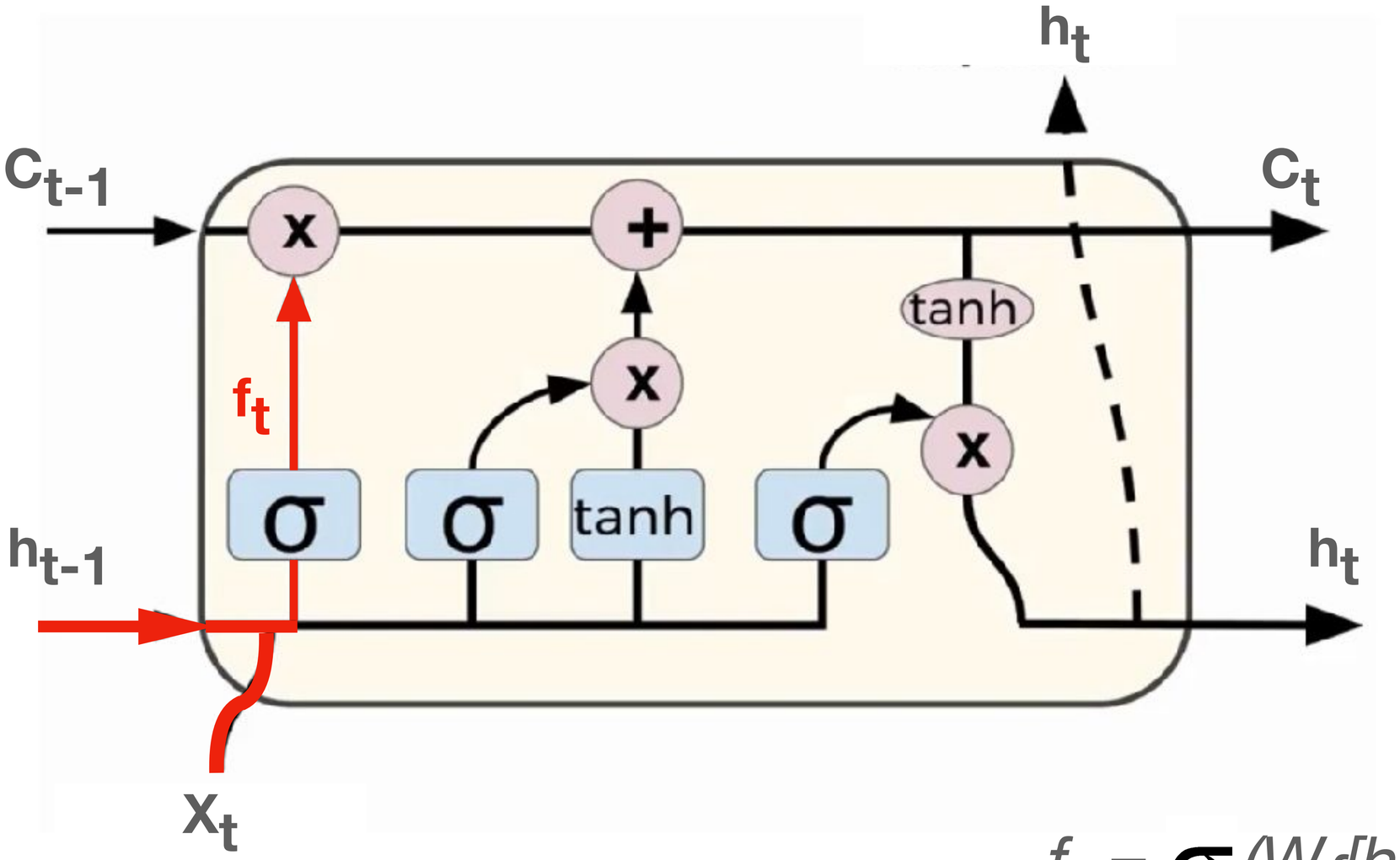
RNN

LSTM la celda completa



RNN

LSTM - Primer paso, ¿qué es lo que se va a olvidar?



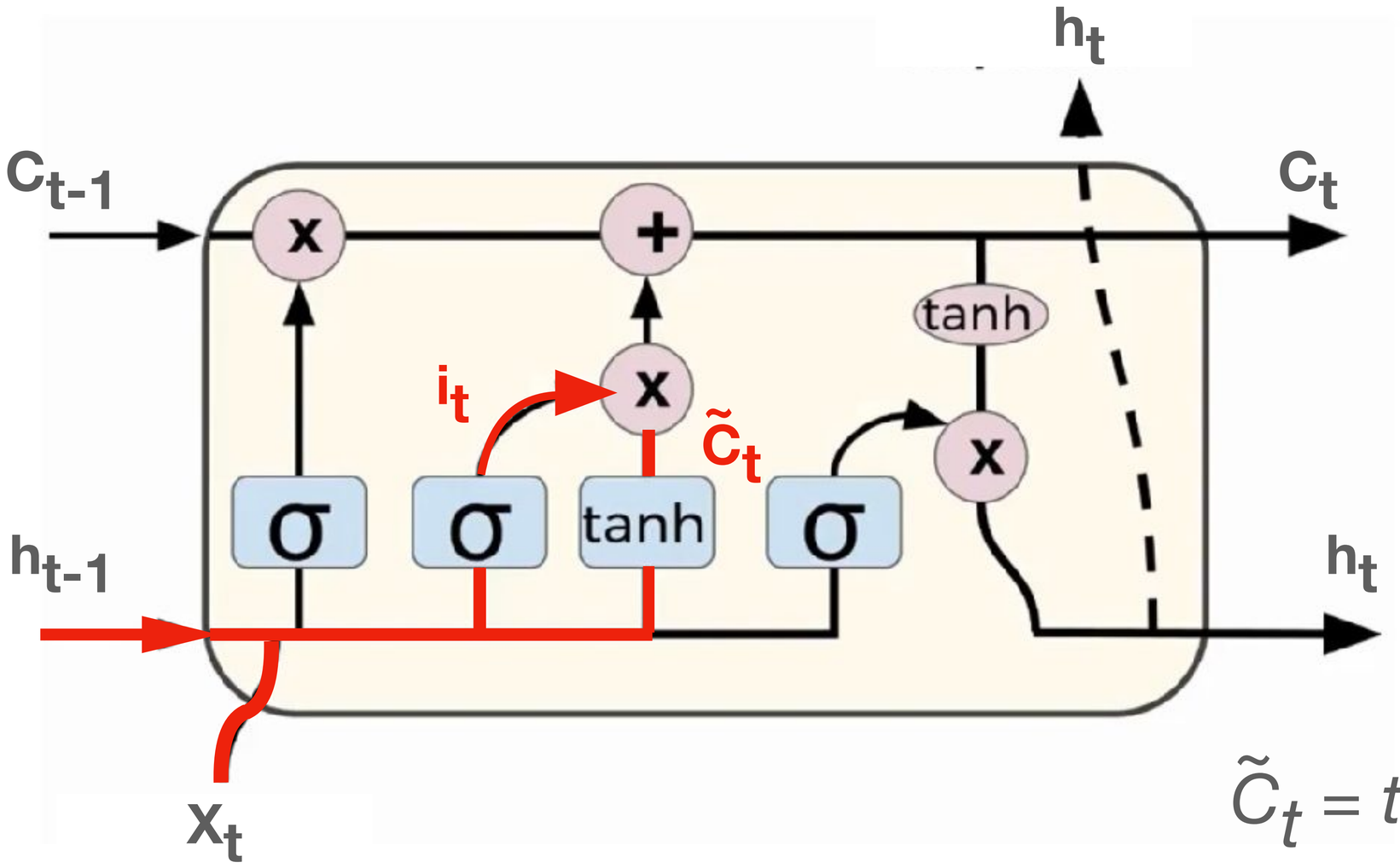
Entre más cerca es de cero, menos importancia tiene y menos peso se le dará.

Entre más cerca es de uno, tiene más importancia y más peso se le dará

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

RNN

LSTM - Segundo paso, ¿qué información nueva se almacena en el nuevo estado de la celda?



i - compuerta de entrada, decide qué valores se actualizan

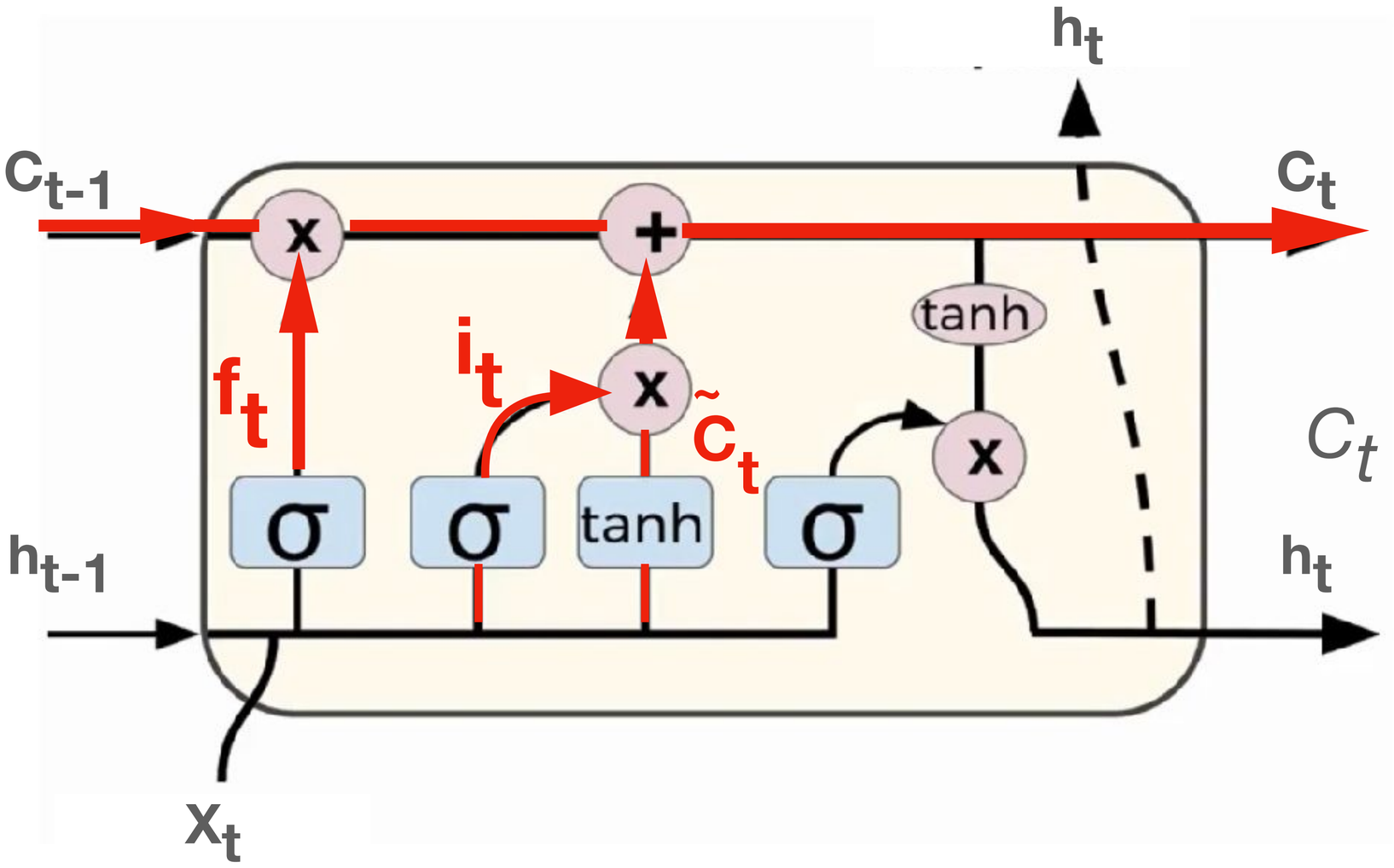
$\tilde{C}$ - compuerta de candidatos a nuevos valores

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

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

RNN

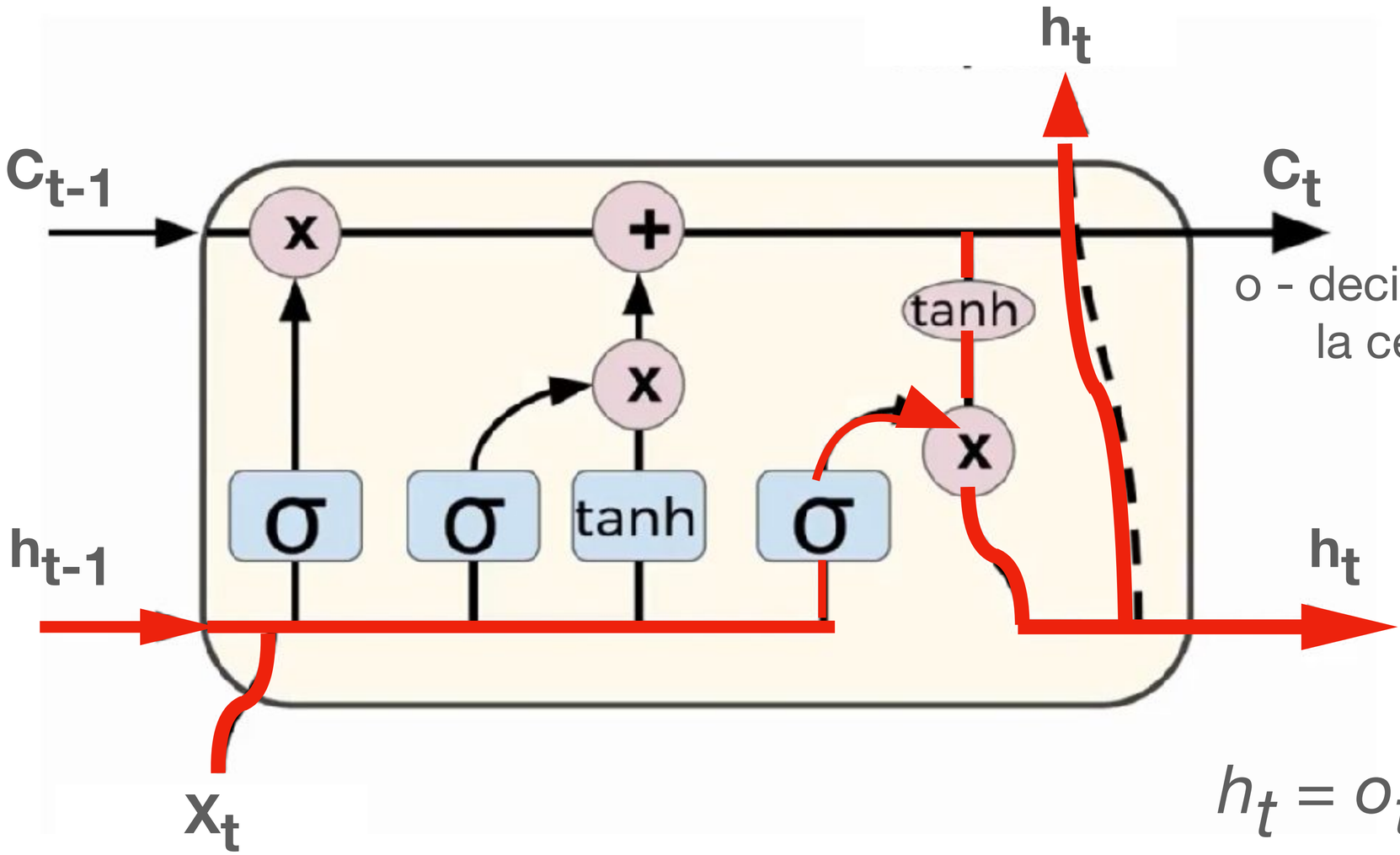
LSTM - Tercer paso, actualizar el estado anterior de la celda



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

RNN

LSTM - Cuarto paso, calcular la salida



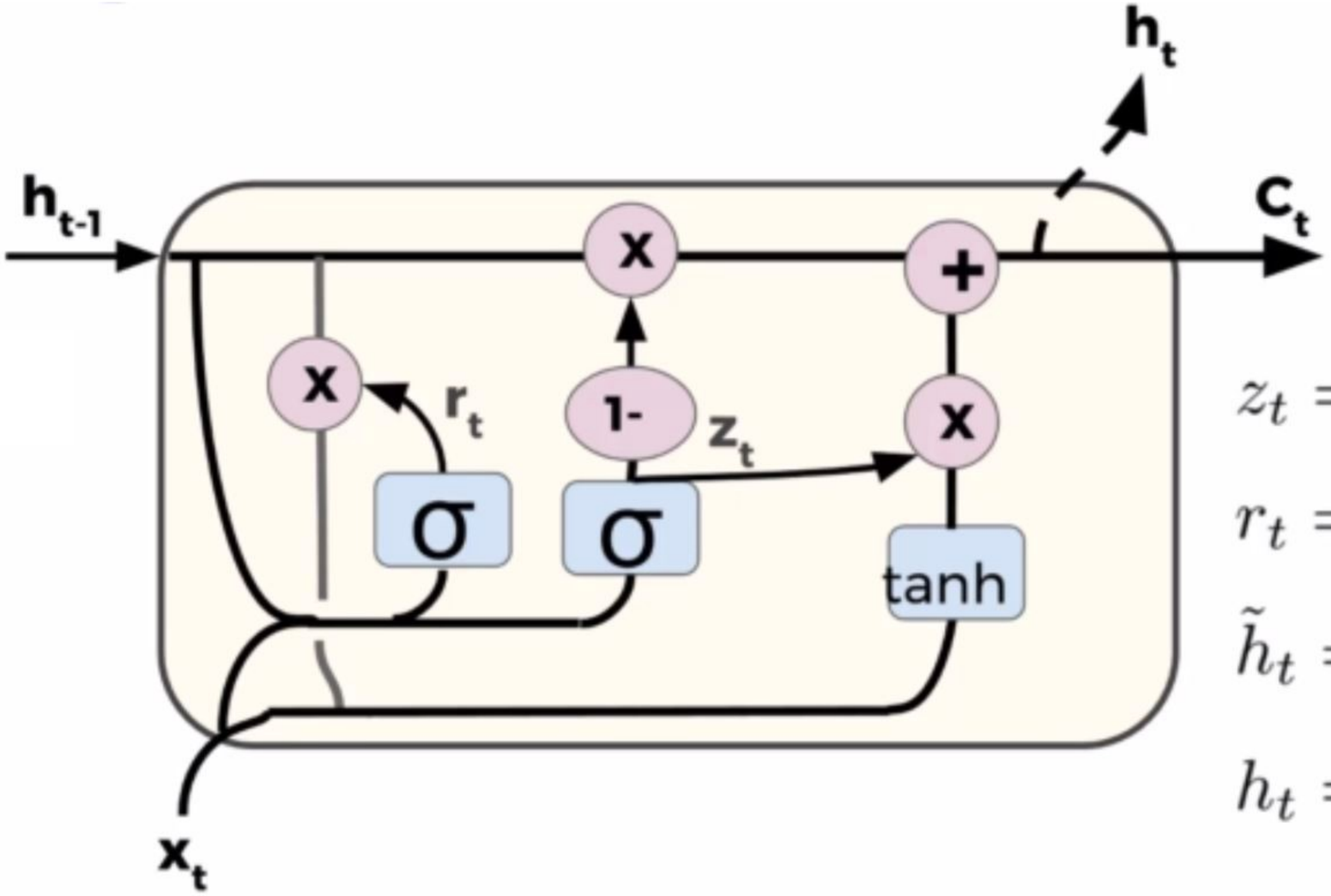
o - decide que partes del estado de la celda van a la salida

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

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

RNN

LSTM - Hay variantes de esto, por ejemplo la GRU (Gated Recurrent Unit)



Combina las compuertas de olvido y entrada para formar una de actualización

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

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

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

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

RNN

- LSTM es actualmente es más utilizado
- GRU está rápidamente creciendo en popularidad
- Vamos a trabajar con la RNN y la LSTM
- Afortunadamente no tenemos que codificar nada de lo anterior, solamente invocamos a la librería de Deep Learning

Tandas de RNN

Antes de empezar a codificar, veamos cómo se ven estas tandas

- Imaginemos una Serie de Tiempo simple
  - [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
- La separamos en dos partes
  - [0, 1, 2, 3, 4, 5, 6, 7, 8] ➡ [9]
- Dados la secuencia de entrenamiento, predecir el siguiente valor de la secuencia

Tandas de RNN

- Hay que tener en mente que generalmente podemos decidir las longitudes de la secuencia de entrenamiento y la etiqueta
  - [0, 1, 2, 3, 4] ➡ [5, 6, 7, 8, 9]

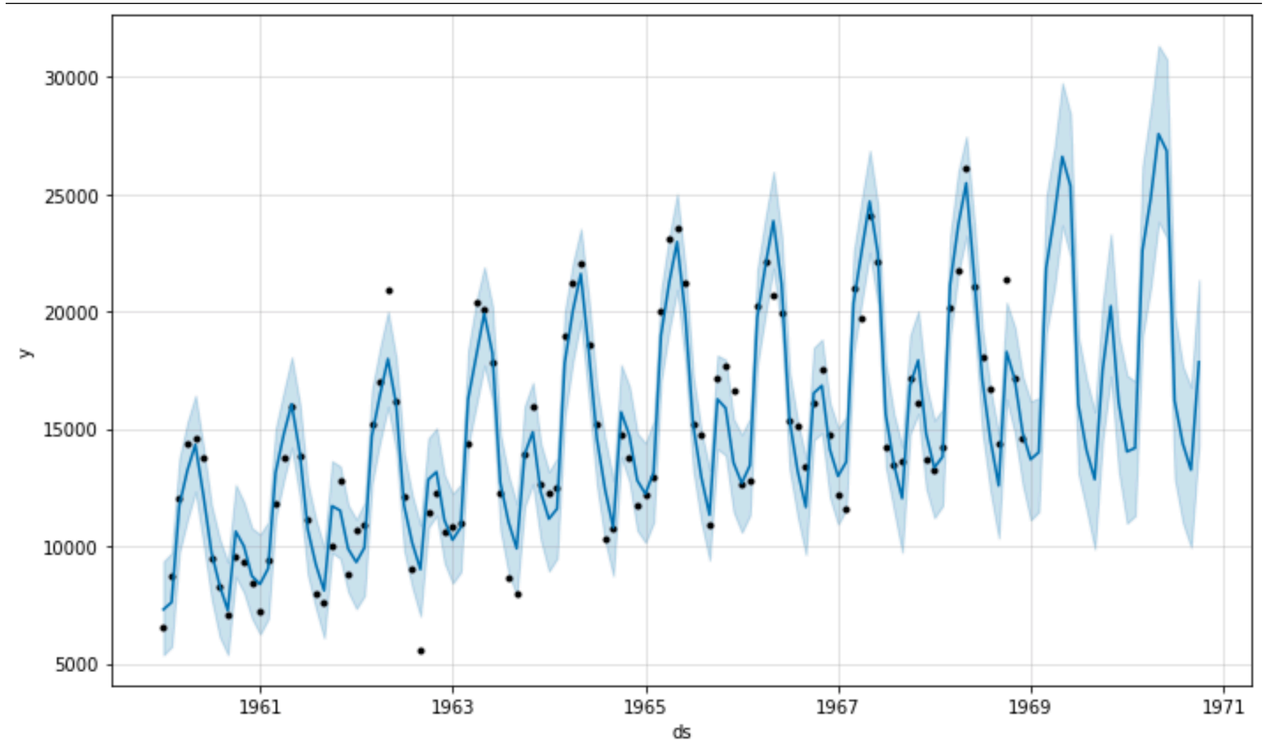
Tandas de RNN

- También podemos editar el tamaño del punto de entrenamiento, así como cuántas secuencias se deben alimentar por tanda
  - [0, 1, 2, 3] ➡ [4]
  - [1, 2, 3, 4] ➡ [5]
  - [2, 3, 4, 5] ➡ [6]

Tandas de RNN

Con la flexibilidad que hay, ¿cómo se decide la longitud de la secuencia de entrenamiento?

- No hay una respuesta definitiva, al menos, debe ser lo suficientemente larga para capturar cualquier tendencia de información que pueda ser de interés, por ejemplo:



- Venta de carros en Canadá.
- Tendencia estacional
- Se venden más en la primavera y otoño
- Se venden menos en el verano e invierno.
- Los datos eran mensuales

Por tanto, para captar la tendencia estacional deben haber, al menos, secuencias de entrenamiento de 12 meses.

Aquí es donde entra el expertaje en el dominio!

Tandas de RNN

- Esto, a menudo, requiere de experiencia y conocimiento del dominio, y
- Experimentar y usar RMSE para medir el error en las predicciones

Un buen punto para empezar es predecir solamente un dato o etiqueta en el futuro

Tandas de RNN

¿Cómo se predice con RNN?

- De nuevo imaginemos que los datos son:
  - [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
- Y que se entrenó con secuencias tales como:

  - [0, 1, 2, 3] ➡ [4]
  - [1, 2, 3, 4] ➡ [5]
  - [2, 3, 4, 5] ➡ [6]

Tandas de RNN

¿Cómo se predice con RNN?

- Se quiere predecir un paso más en el futuro…más allá del 9, y luego incorporar esa predicción en la siguiente secuencia que se usará para predecir otro valor
- Veamos un ejemplo

Tandas de RNN

¿Cómo se predice con RNN?

- Tomaríamos la última secuencia de entrenamiento y la red predicería el 10
  - [6, 7, 8, 9] → [10]
- Se percibe que el 10 sea correcto, pero no tenemos 100% de seguridad pues se está prediciendo al futuro. Si se sigue haciendo, podría resultar:
  - [7, 8, 9, 10] → [11.2]
  - [8, 9, 10, 11.2] → [12.4]
  - [9, 10, 11.2, 12.4] → [14]
  - [10, 11.2, 12.4, 14] → Predicción completada!

Es importante ver que entre más al futuro se vaya, menos confíable son los datos debido a que más y más se esta prediciendo con valores predichos

Enlaces interesantes sobre estos temas

- The Unreasonable Effects of Recurrent Neural Networks
- Exploring LSTMs
- Long short-term memory
- When to use GRU over LSTM?