{desafío}
latam_

Refactorización de Redes Neuronales _

Motivación

Preliminares

Ya conocemos los elementos fundacionales de una red neuronal:

- Sabemos que la unidad básica es una neurona que recibe impulsos y los reexpresa.
- Estas neuronas se pueden ensamblar generando capas ocultas que facilitan la representación distribuida.
- En la medida que concatenamos múltiples capas, tenemos mayores chances de capturar de buena manera un fenómeno complejo.
- También conocemos el rol de la regularización y la agenda de entrenamiento para perfilar de mejor manera cómo nuestra red puede tener un buen desempeño.

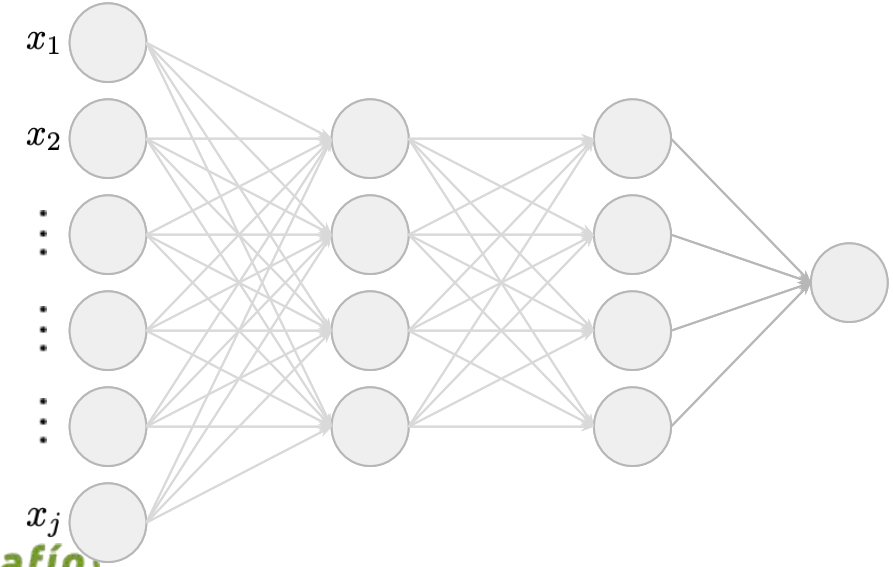
Weight Decay y Regularización

Preliminares

- **Un acápite:** La diferencia entre regularización y weight decay en las redes neuronales.
- **Objetivo:** Evitar la sobreestimación de pesos en una red neuronal.
- **Weight Decay:** Parámetro de escalamiento en la regla de actualización de los pesos en la fase de backpropagation.
- **Regularización L2:** Penalización en la función de costo.
- El comportamiento entre ambos métodos es idéntico (condicional al learning rate y lambda) cuando implementamos gradiente estocástico.
- En métodos adaptativos como Adam, AdaDelta y RMSprop, el rol del weight decay es más relevante, dado que reacciona de buena manera frente a la actualización del learning rate.

Limitantes de las redes feed forward

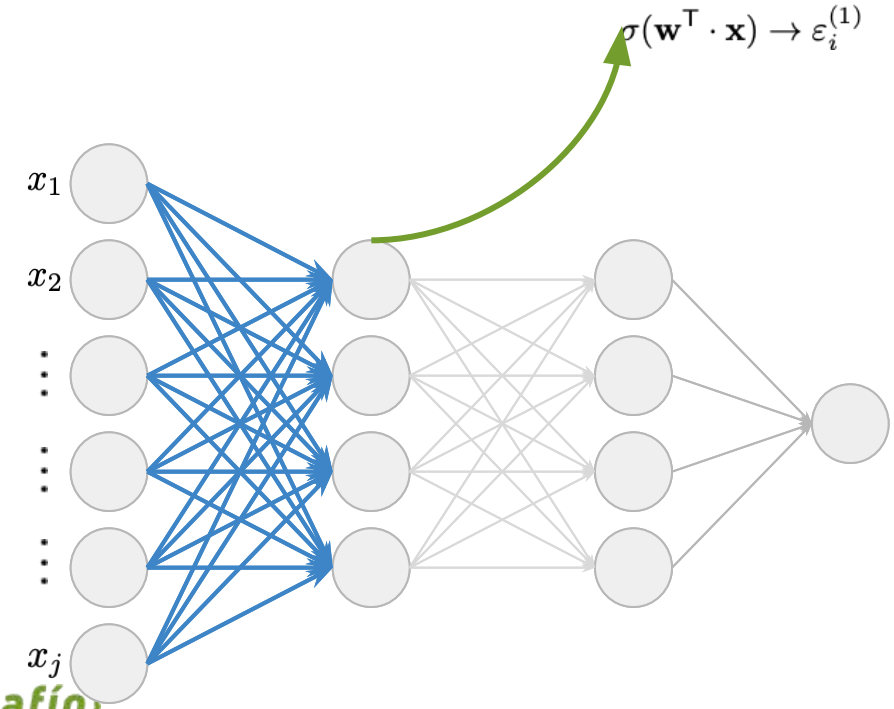
Limitantes de feed forward



¿Qué sabemos?

Una neurona recibe pesos de múltiples atributos y emite una señal que es procesada posteriormente por otras neuronas.

Limitantes de feed forward

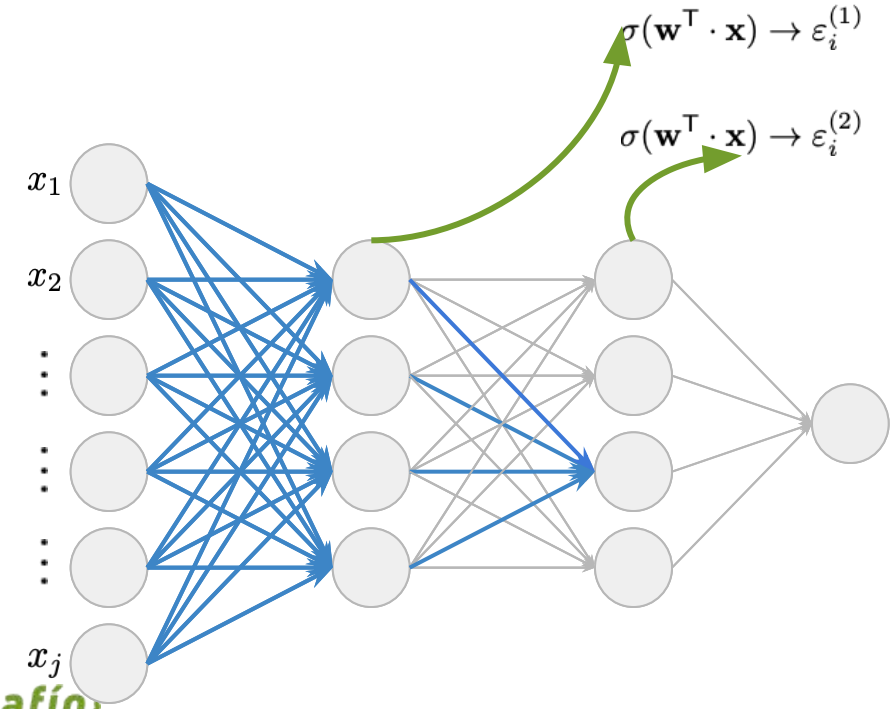


¿Qué sabemos?

Una neurona recibe pesos de múltiples atributos y emite una señal que es procesada posteriormente por otras neuronas.

Esto no es problemático si estamos trabajando con datos “transversales” (una cantidad finita de observaciones en un momento específico.

Limitantes de feed forward

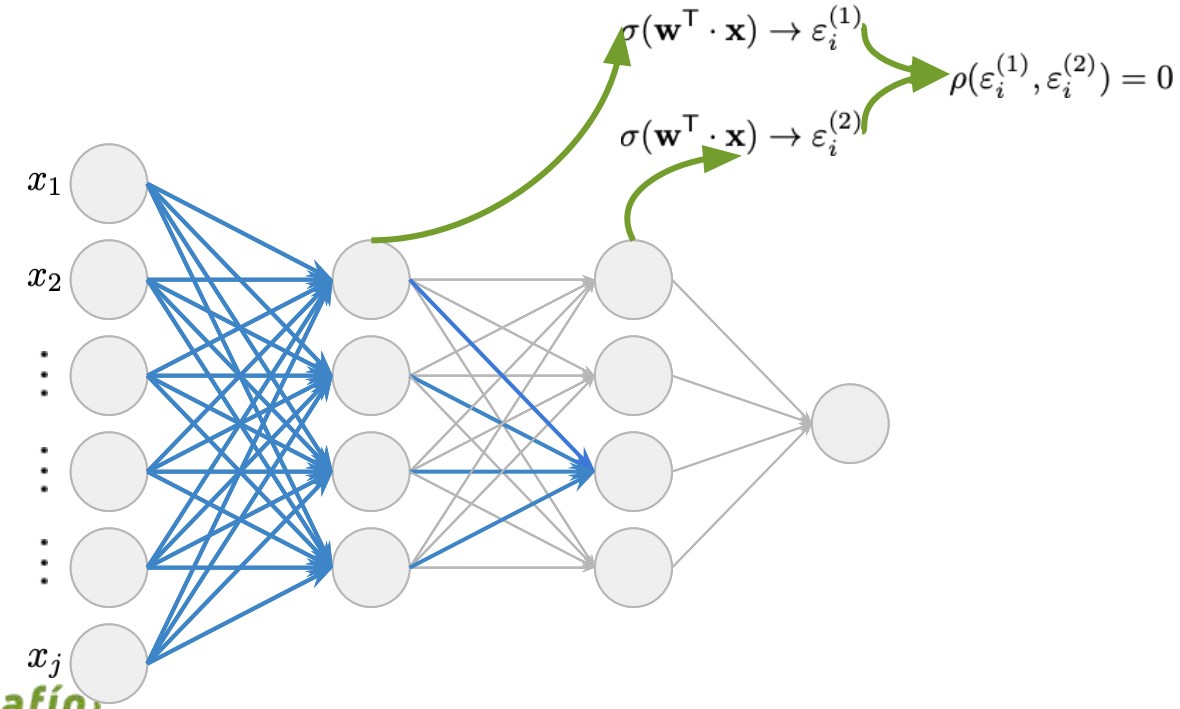


¿Qué sabemos?

Una neurona recibe pesos de múltiples atributos y emite una señal que es procesada posteriormente por otras neuronas.

Esto no es problemático si estamos trabajando con datos "transversales" (una cantidad finita de observaciones en un momento específico.

Limitantes de feed forward

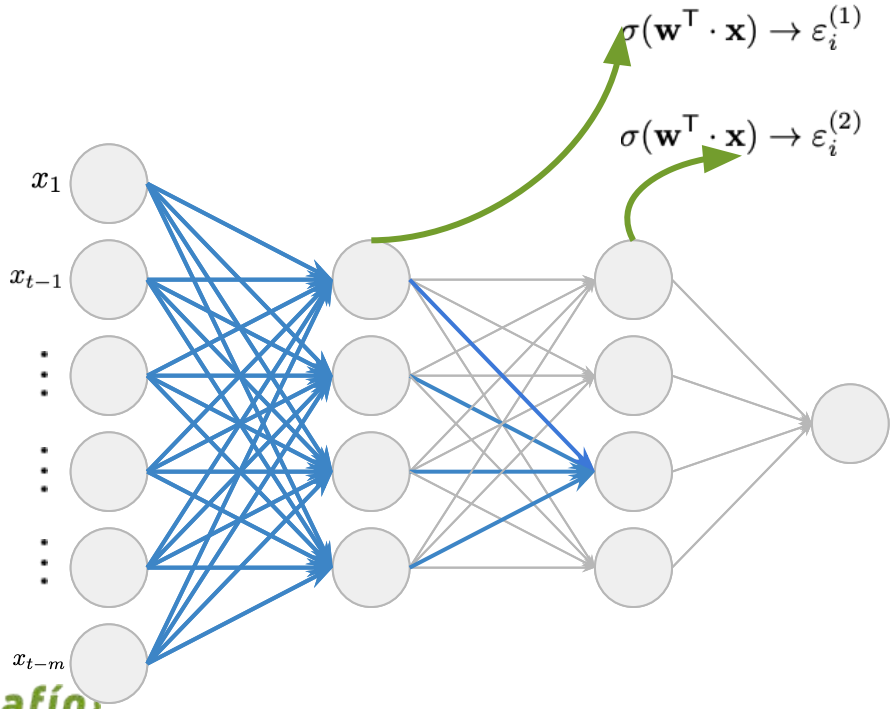


¿Qué sabemos?

Una neurona recibe pesos de múltiples atributos y emite una señal que es procesada posteriormente por otras neuronas.

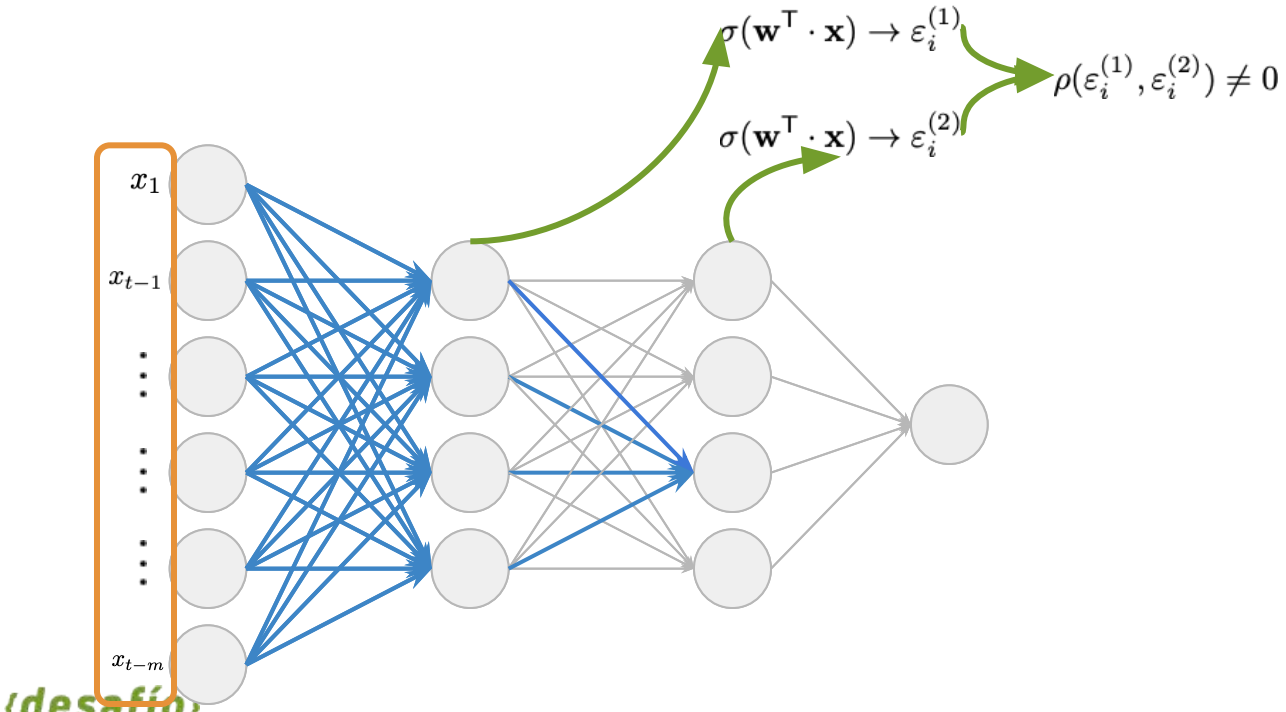
Esto no es problemático si estamos trabajando con datos "transversales" (una cantidad finita de observaciones en un momento específico.

Limitantes de feed forward



Resulta que si estamos trabajando con datos secuenciales, corremos el riesgo de sufrir correlación entre las observaciones.

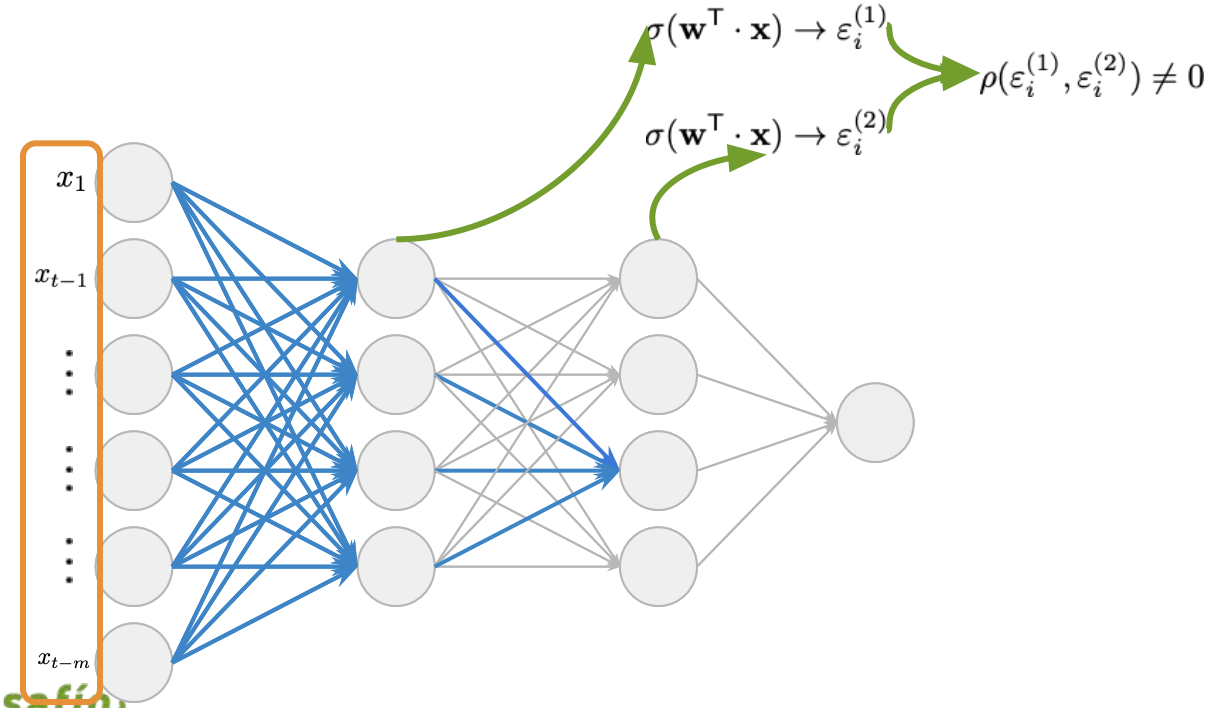
Limitantes de feed forward



Resulta que si estamos trabajando con datos secuenciales, corremos el riesgo de sufrir correlación entre las observaciones.

Esto es problemático si no se toma en consideración.

Limitantes de feed forward



Resulta que si estamos trabajando con datos secuenciales, corremos el riesgo de sufrir correlación entre las observaciones.

Esto es problemático si no se toma en consideración.

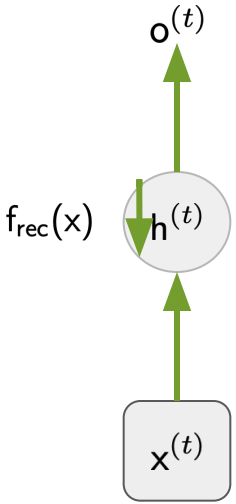
Las redes neuronales recurrentes ofrecen una solución a ésto mediante la incorporación de **contexto** y **memoria.**

Redes Neuronales Recurrentes

Definición

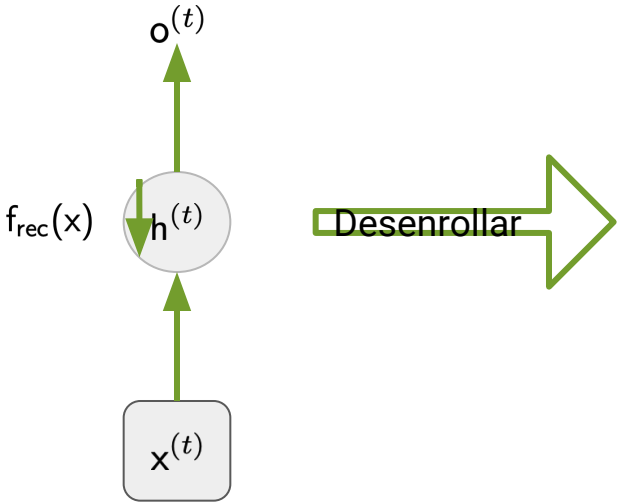
- Una red neuronal feed forward no puede contextualizar ni memorizar información.
- Todos los inputs son procesados en la capa oculta, olvidados una vez que el impulso procesado se propaga.
- Una red neuronal recurrente adopta el principio de procesar secuencias de manera iterativa y preserva aquella información relevante en un estado interno.

Simple Recurrent Neural Network



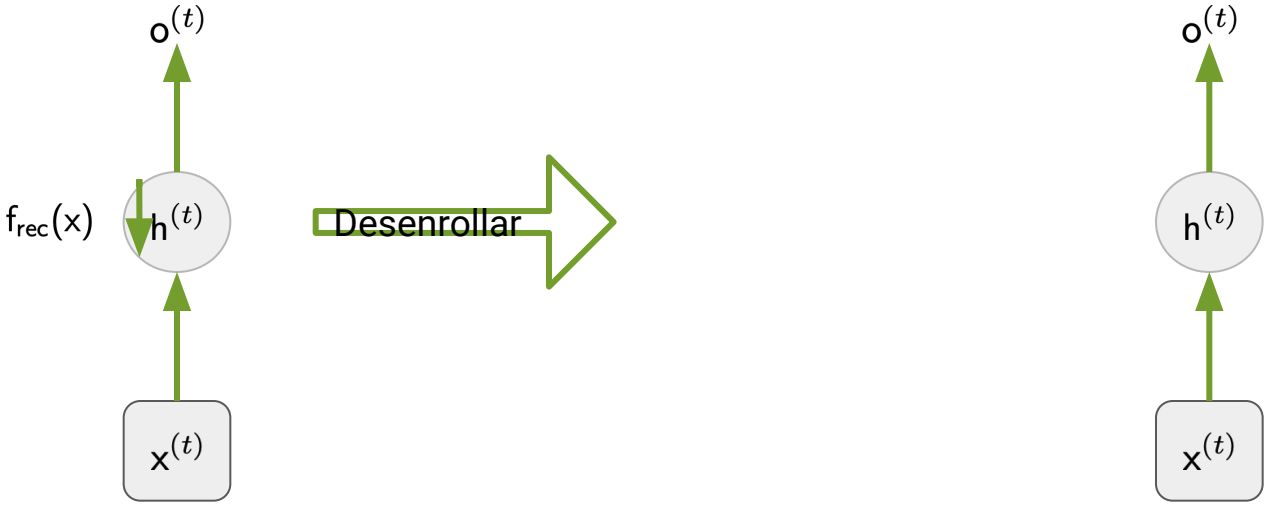
Una red neuronal recurrente simple se caracteriza por tener una función recursiva a lo largo de una serie finita de secuencias.

Simple Recurrent Neural Network



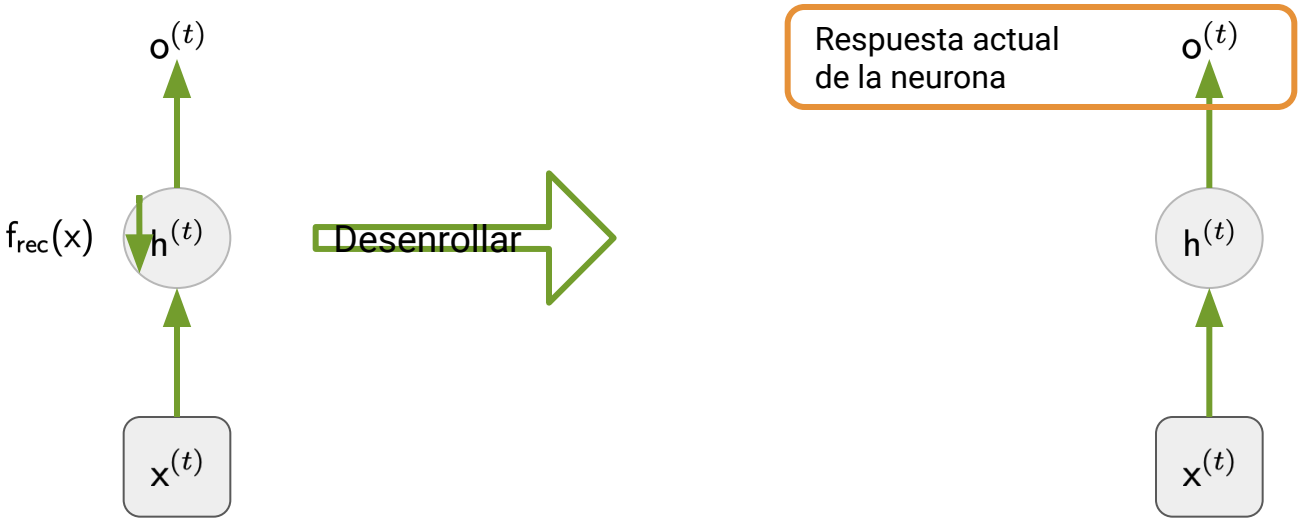
Por lo general, la notación presentada en la izquierda es la versión “enrollada” de la red recurrente.

Simple Recurrent Neural Network



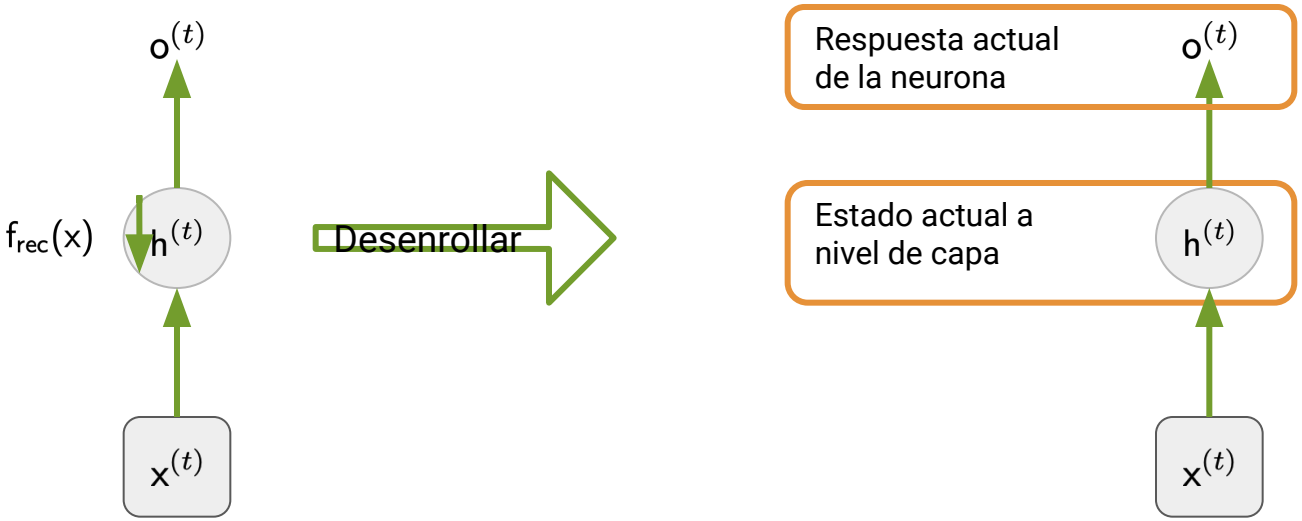
Partamos con la expresión desenrollada de la red. Cada neurona tendrá un dato de ingreso, una capa oculta y un output del estado asociado.

Simple Recurrent Neural Network



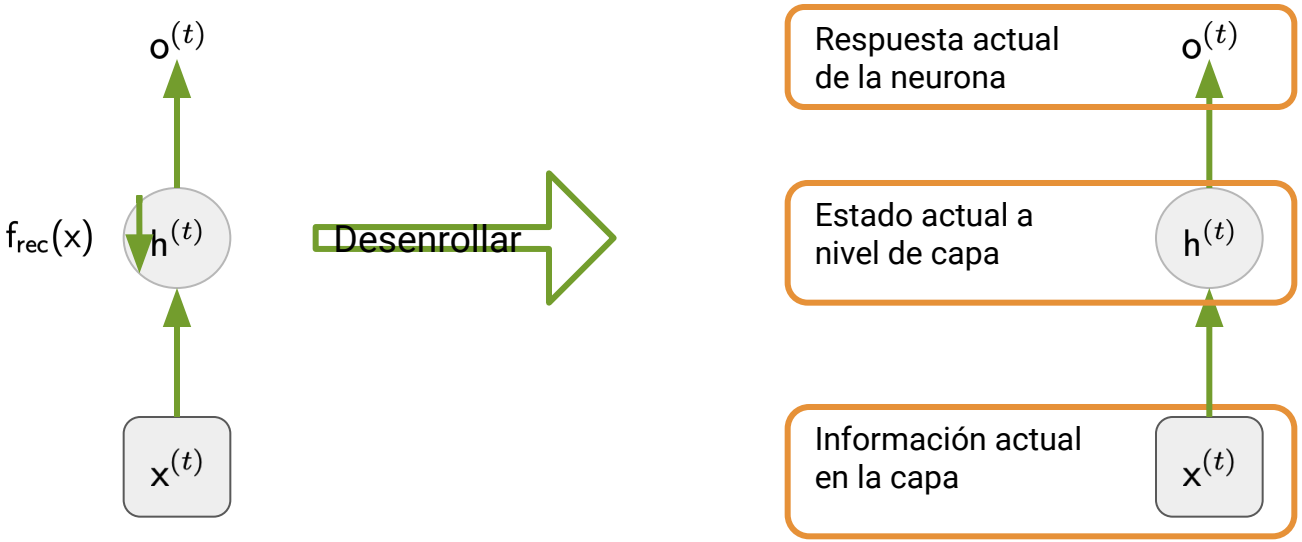
Partamos con la expresión desenrollada de la red. Cada neurona tendrá un dato de ingreso, una capa oculta y un output del estado asociado.

Simple Recurrent Neural Network



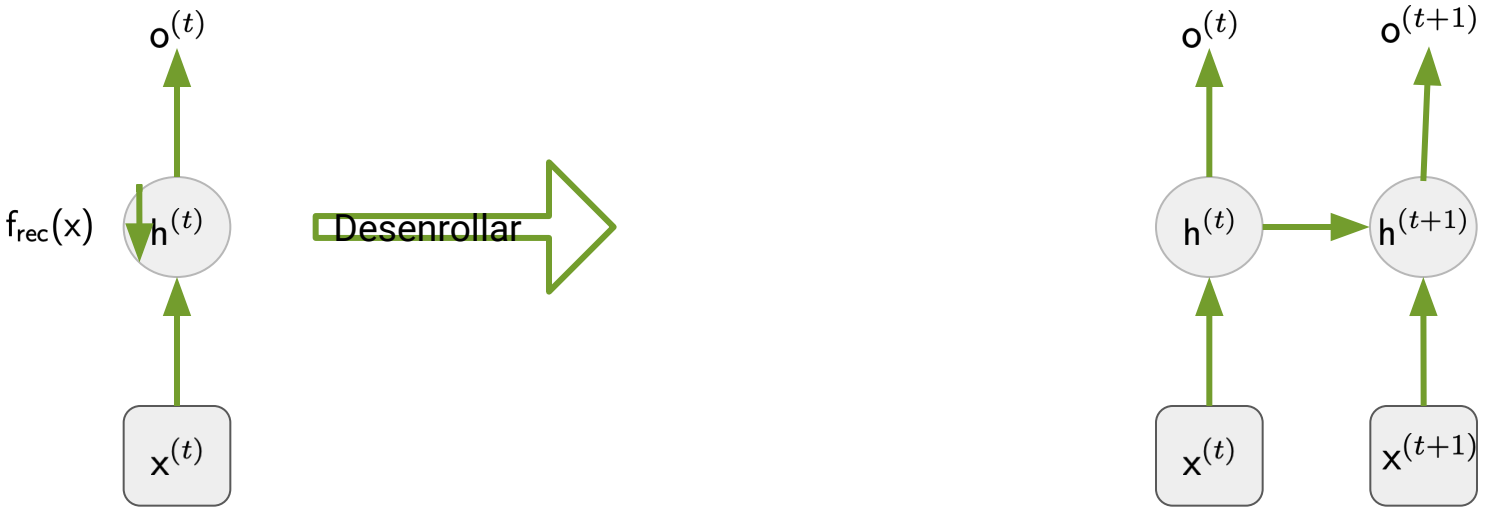
Partamos con la expresión desenrollada de la red. Cada neurona tendrá un dato de ingreso, una capa oculta y un output del estado asociado.

Simple Recurrent Neural Network



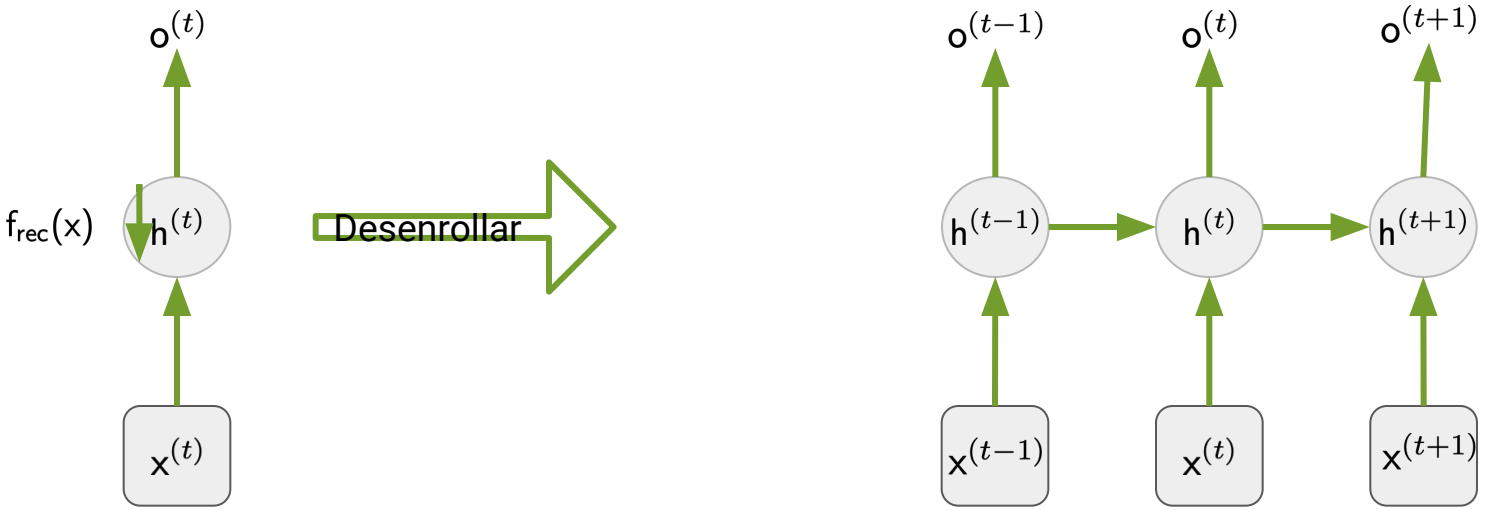
Partamos con la expresión desenrollada de la red. Cada neurona tendrá un dato de ingreso, una capa oculta y un output del estado asociado.

Simple Recurrent Neural Network



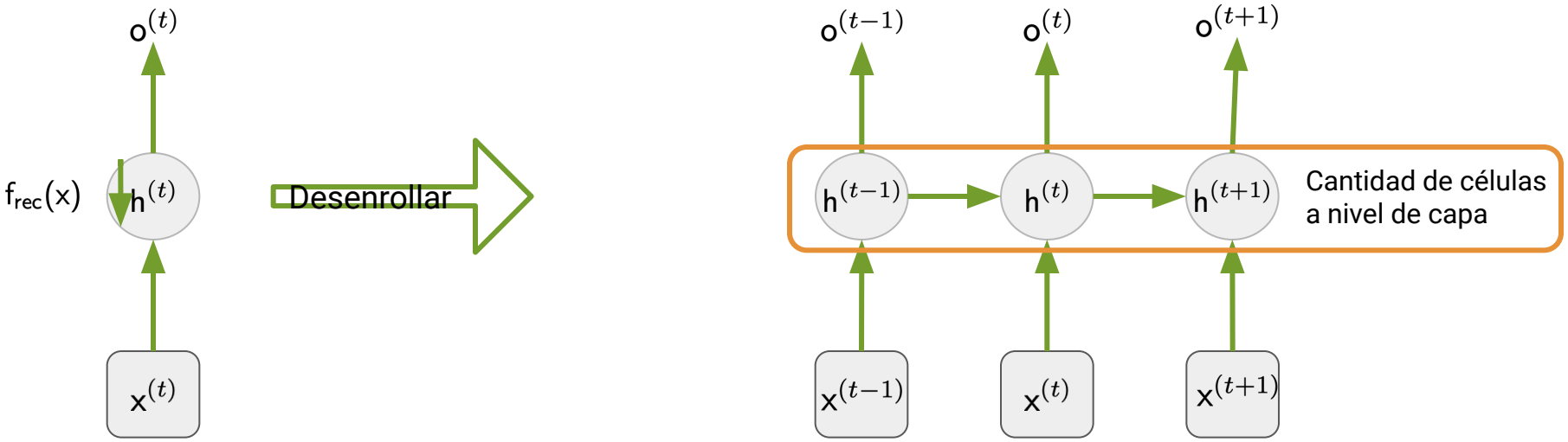
La virtud de trabajar con estructuras recurrentes es la factibilidad de implementar secuencialidad en los datos.

Simple Recurrent Neural Network



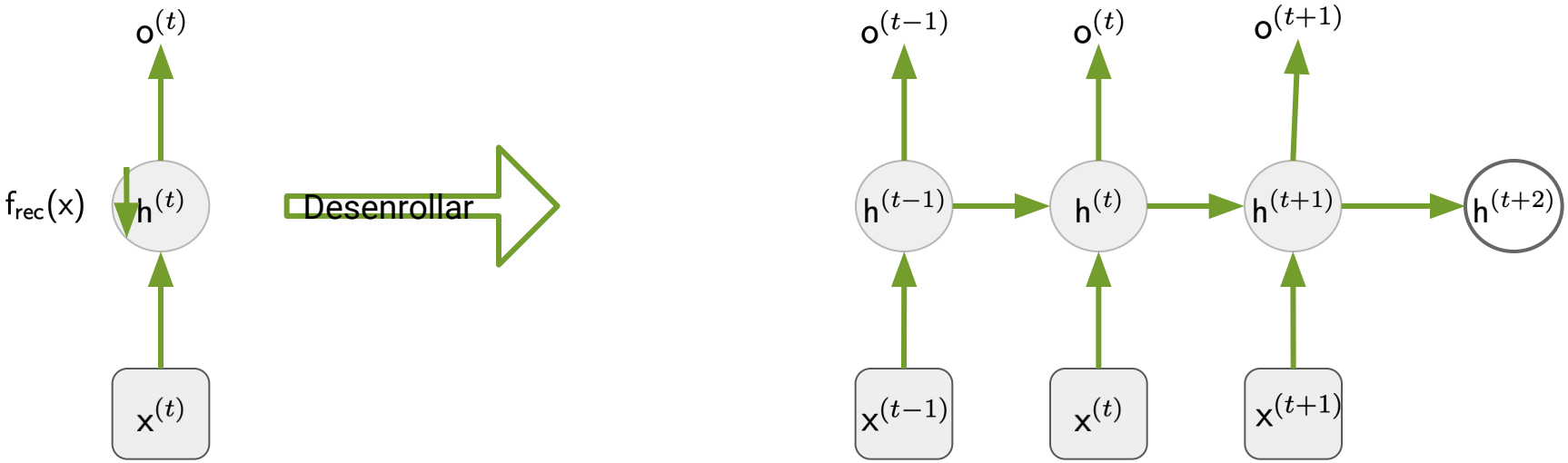
La virtud de trabajar con estructuras recurrentes es la factibilidad de implementar secuencialidad en los datos.

Simple Recurrent Neural Network



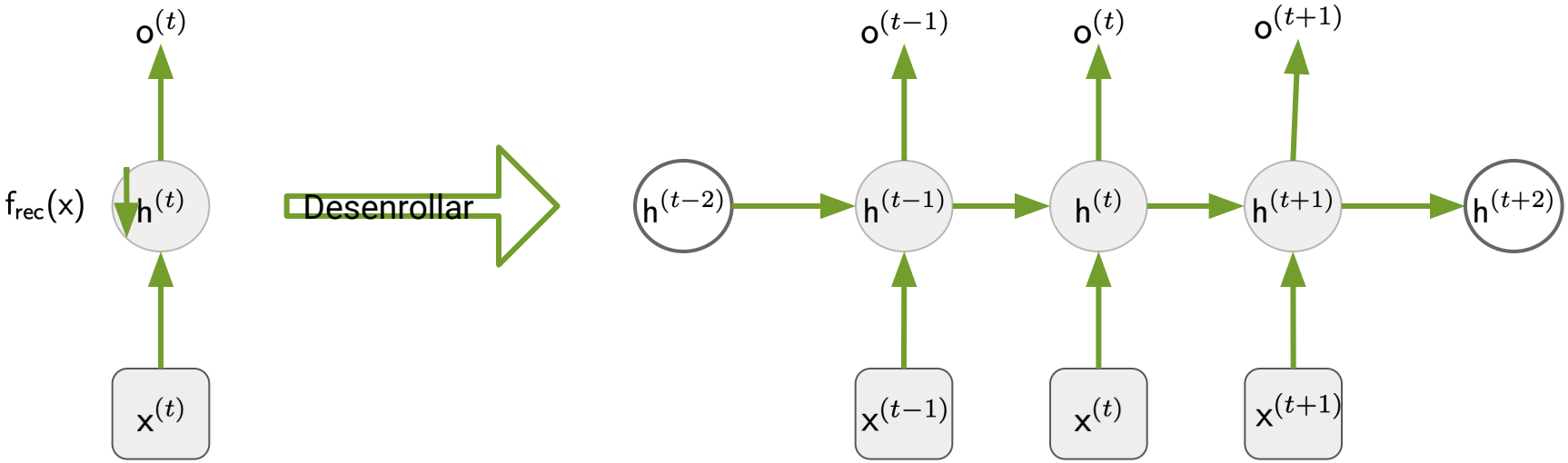
La virtud de trabajar con estructuras recurrentes es la factibilidad de implementar secuencialidad en los datos.

Simple Recurrent Neural Network



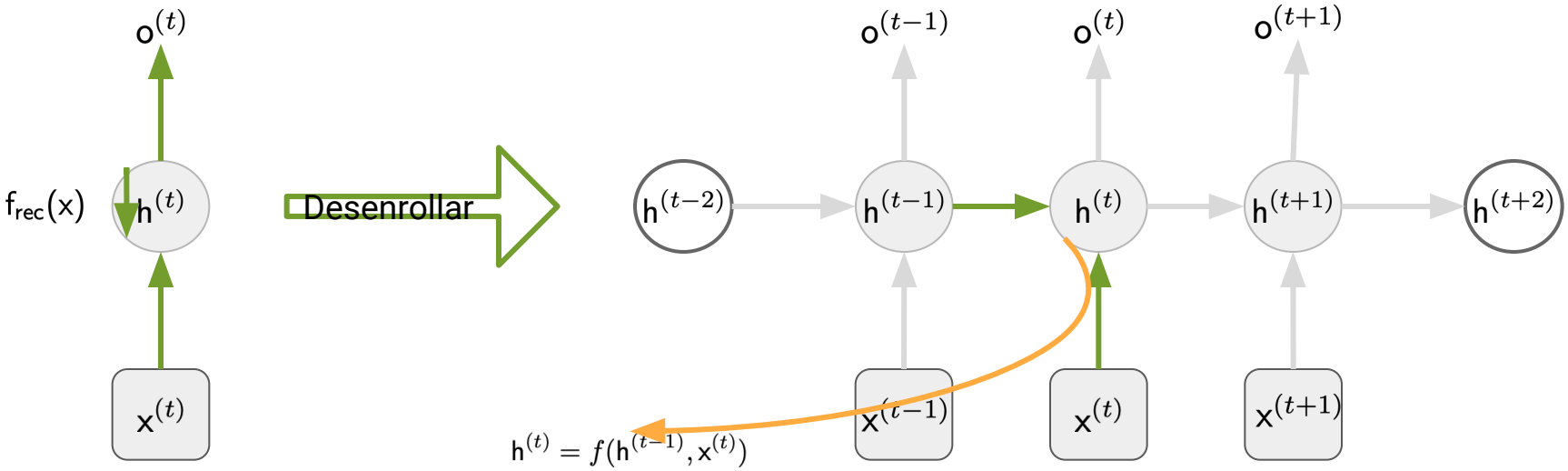
Parte de las virtudes es la capacidad de implementar **proyecciones** en base a la información retenida por las capas previas.

Simple Recurrent Neural Network



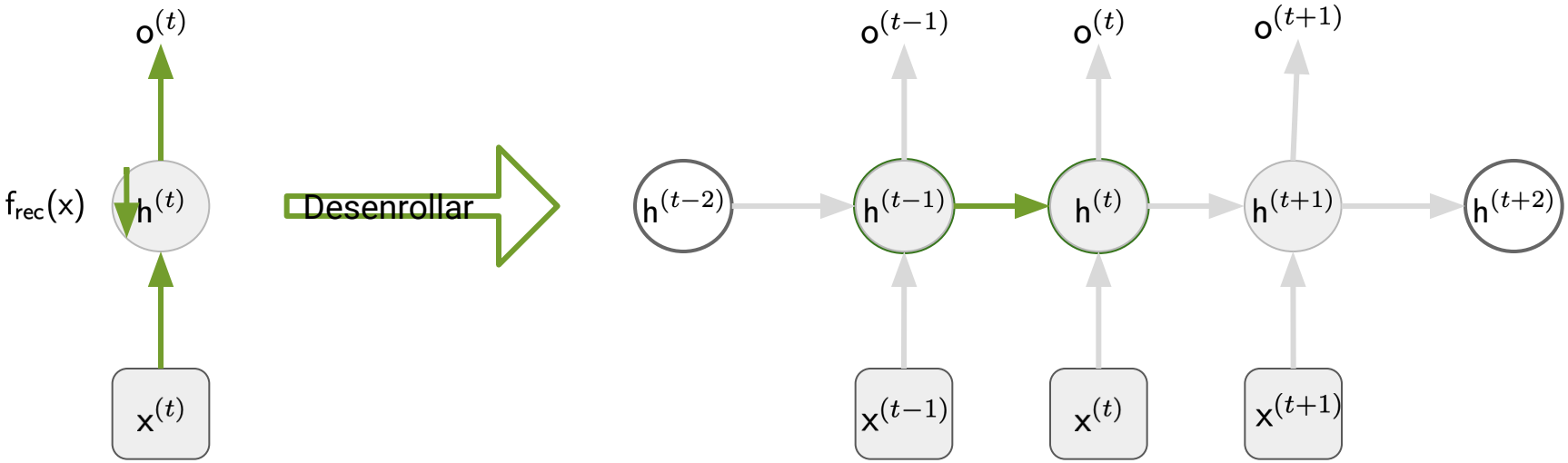
Parte de las virtudes es la capacidad de implementar **regresiones** en base a la información retenida por las capas previas.

Simple Recurrent Neural Network



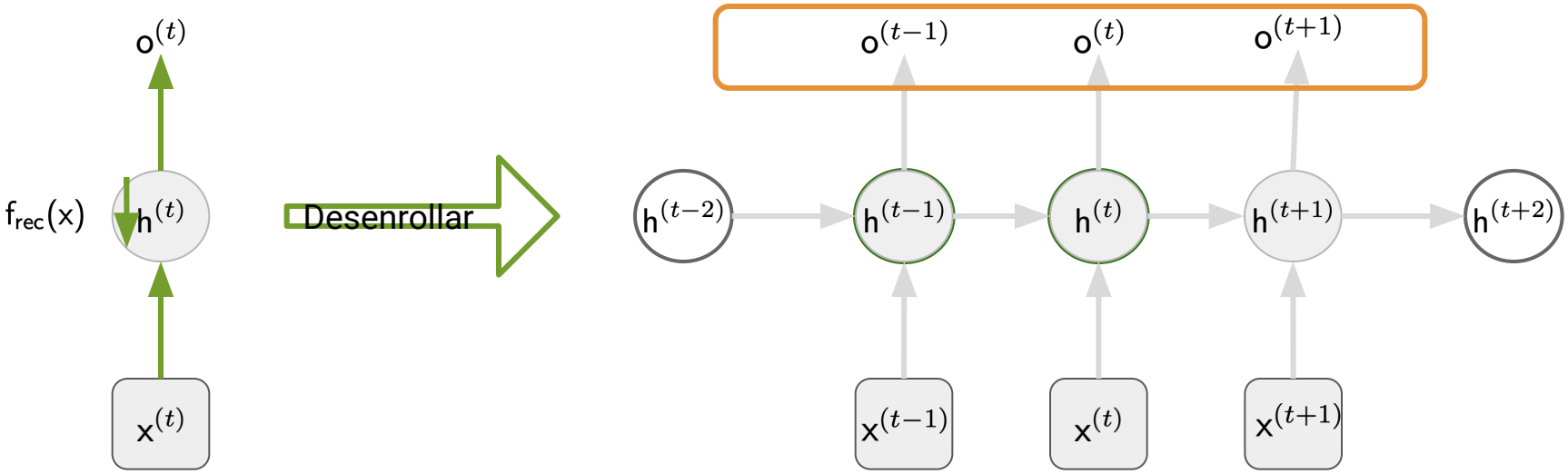
Impulsos: Cada neurona recurrente recibe dos impulsos:
Un impulso previo correspondiente a la neurona en el estado anterior (t-1).
Un impulso correspondiente a los datos en tiempo (t)

Simple Recurrent Neural Network



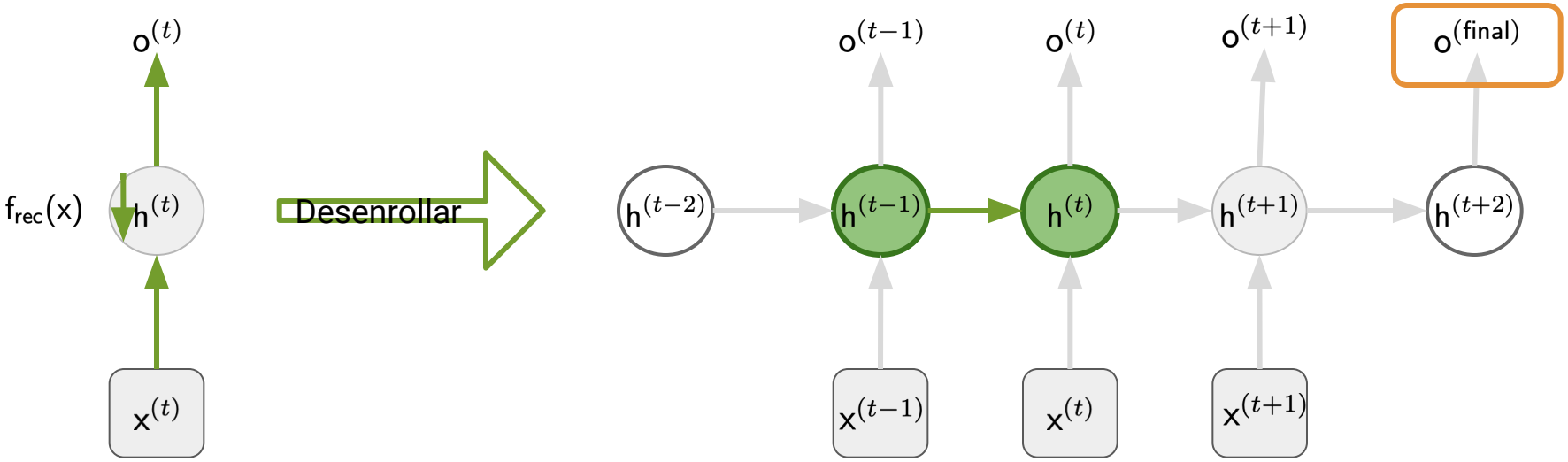
La emisión de impulsos entre capas permite generar contextualidad en los egresos de una neurona específica.
De esta manera, la neurona en el paso actual puede manejar de manera más eficiente cómo se manejan los inputs.

Variantes RNN



Modelo **Many To Many**: permite rescatar la predicción de cada paso secuencial.

Variantes RNN



Modelo **Many To One:** Entrega sólo la predicción final en la red recurrente.

Long Short Term Memory

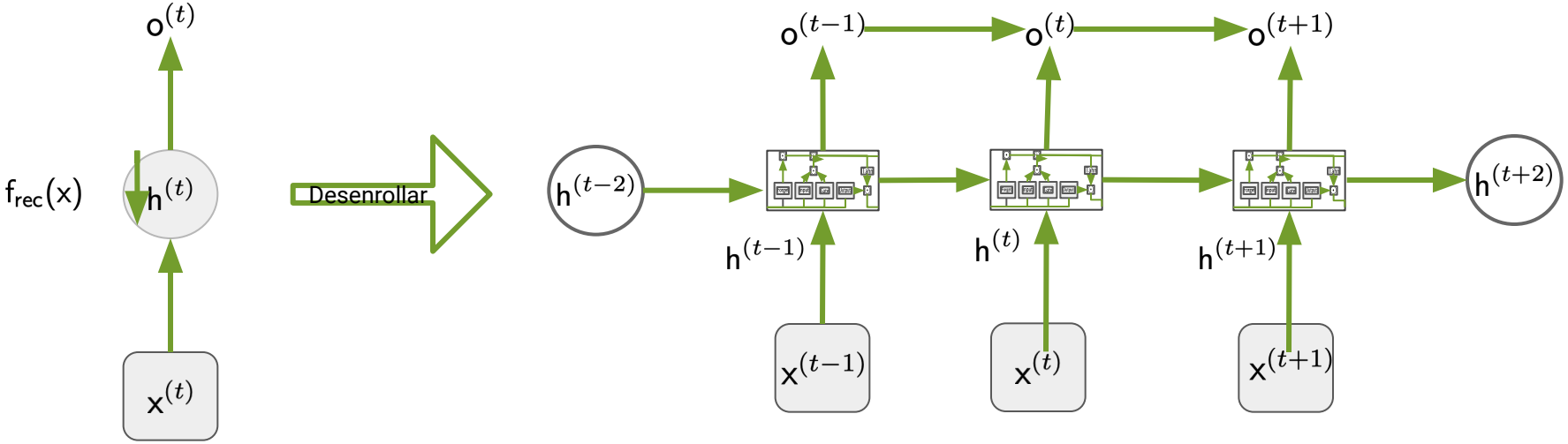
Long Short Term Memory

- **Problema de la Red Neuronal Recurrente Simple:** Por defecto guardará toda la información sobre los pasos previos en la secuencia.
- Esto es problemático dado que genera una inestabilidad en el poder representacional del modelo entrenado.
- También genera inestabilidad en el proceso de optimización de la función de pérdida dado la existencia de gradientes desvanecientes.
- Para superar este contratiempo, se propone la existencia de estructuras LSTM (Long Short Term Memory).

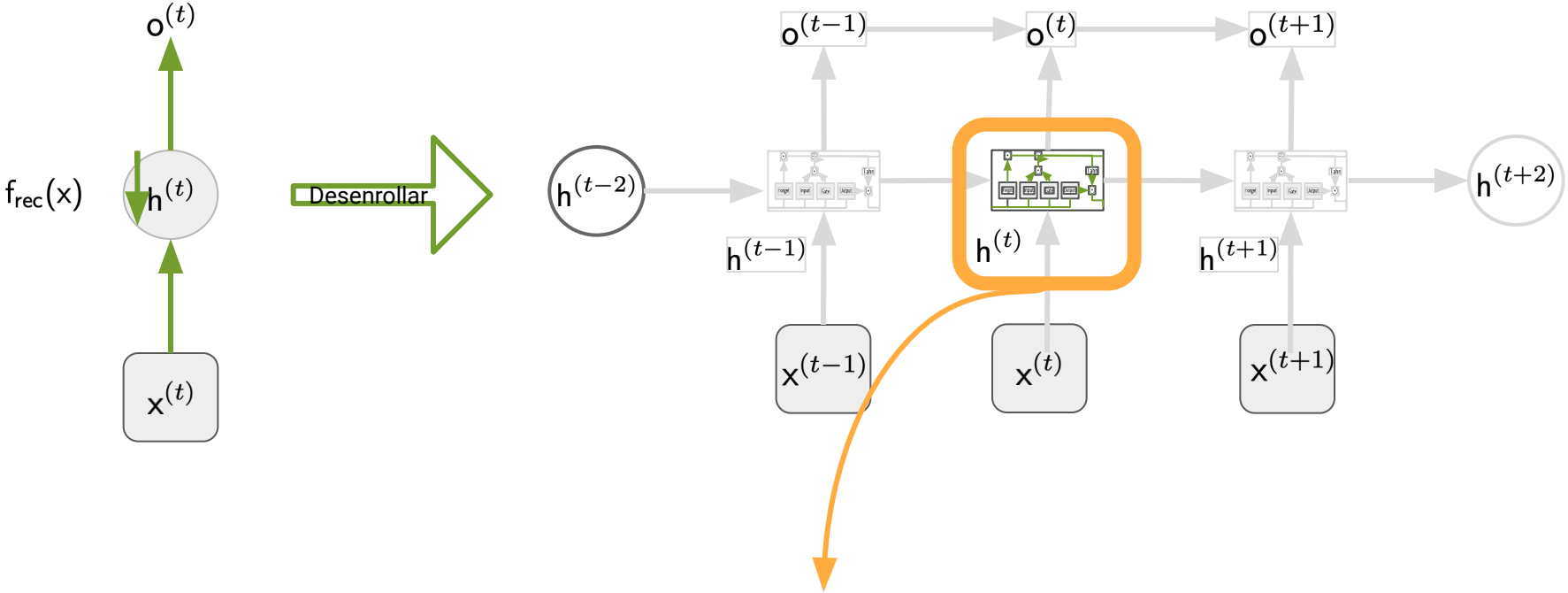
Long Short Term Memory

- **Intuición detrás de LSTM**: No necesitamos absolutamente toda la información previa.
- Buscamos separar el comportamiento de la capa oculta en tres compuertas:
  - **Forget**
  - **Input**
  - **Output**
- Mediante la desagregación de la función recurrente modificamos constantemente el estado de la memoria a preservar.

Long Short Term Memory

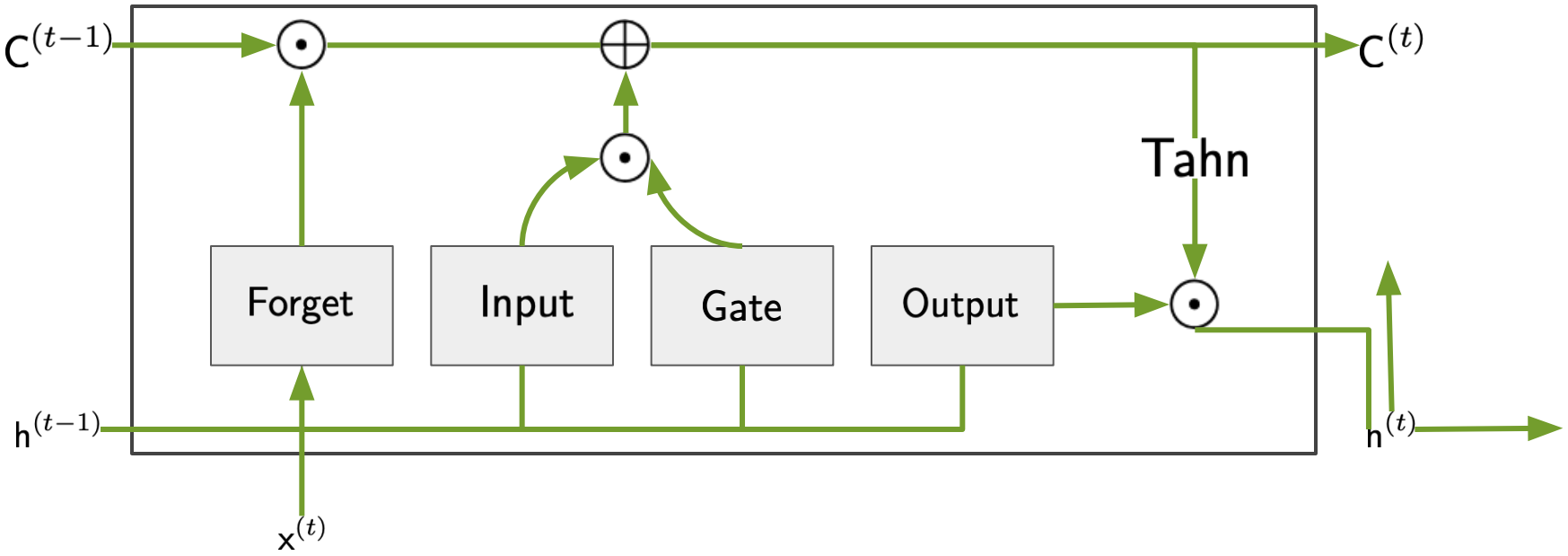


Long Short Term Memory

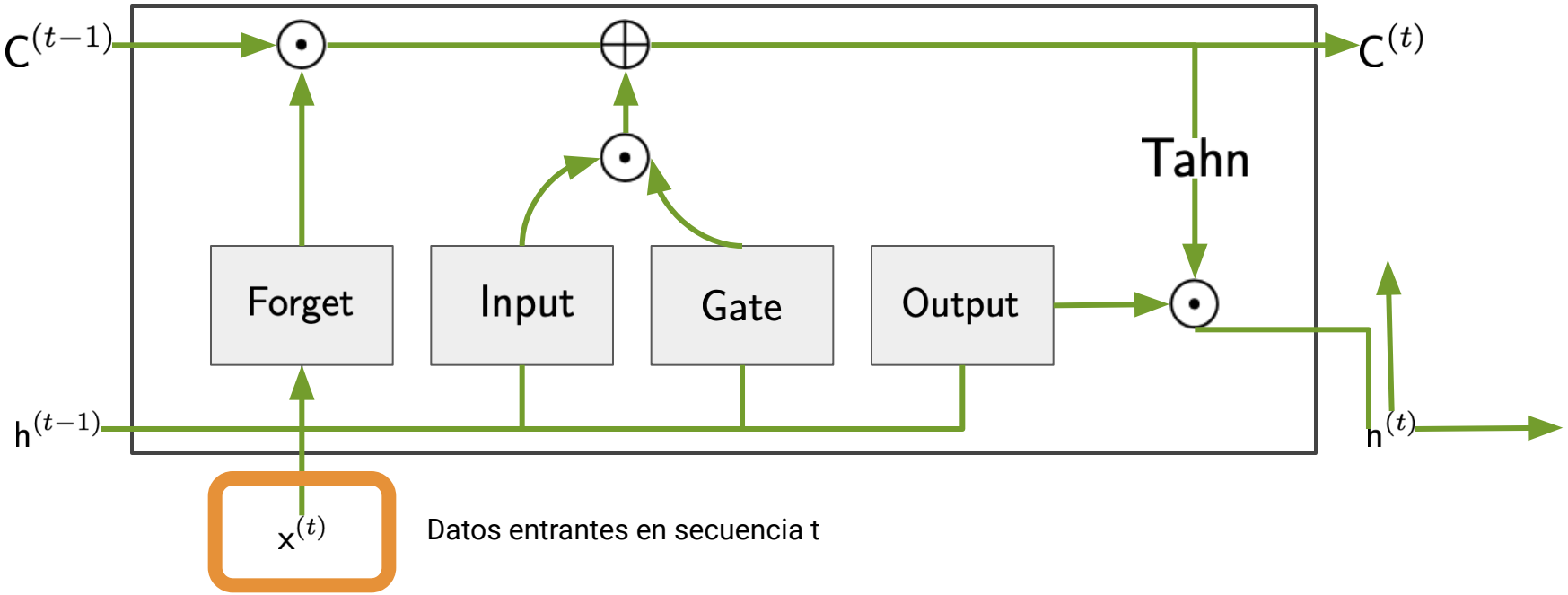


Cuando implementamos LSTM, la estructura entre neuronas se mantiene igual.
La principal diferencia se produce dentro de la neurona, donde se implementa una serie de compuertas para manejar los impulsos.

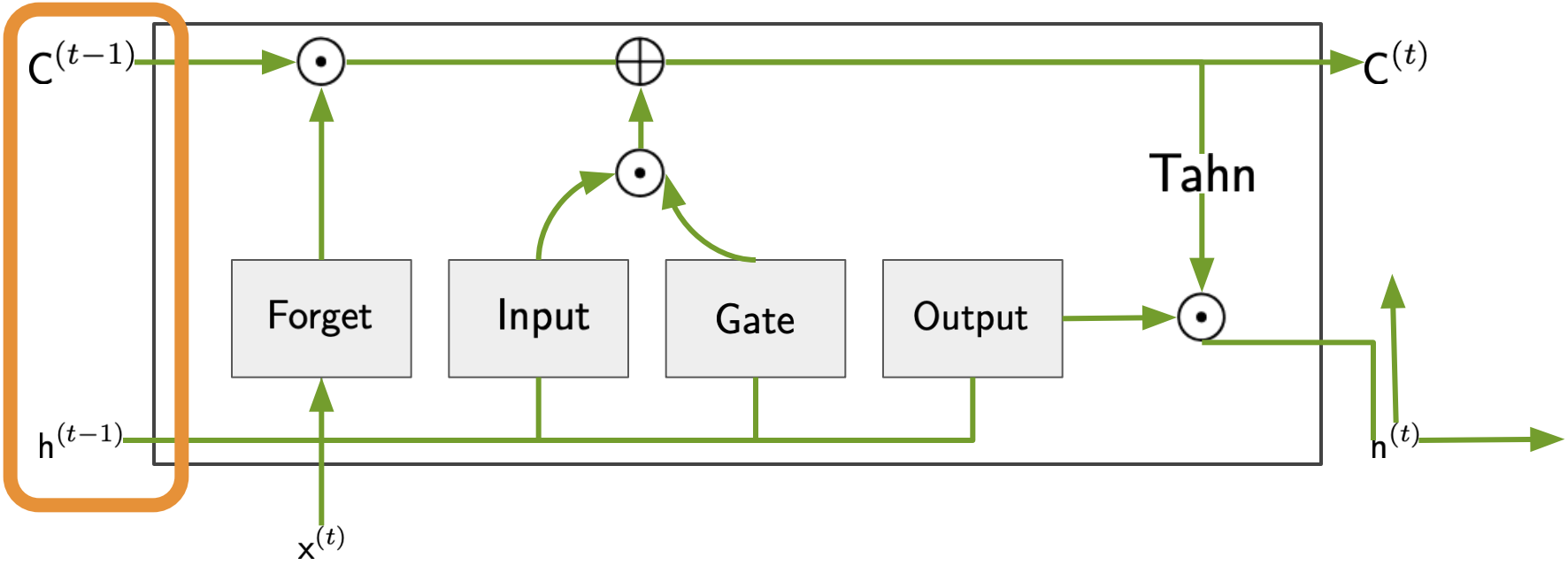
Anatomía de LSTM



Anatomía de LSTM

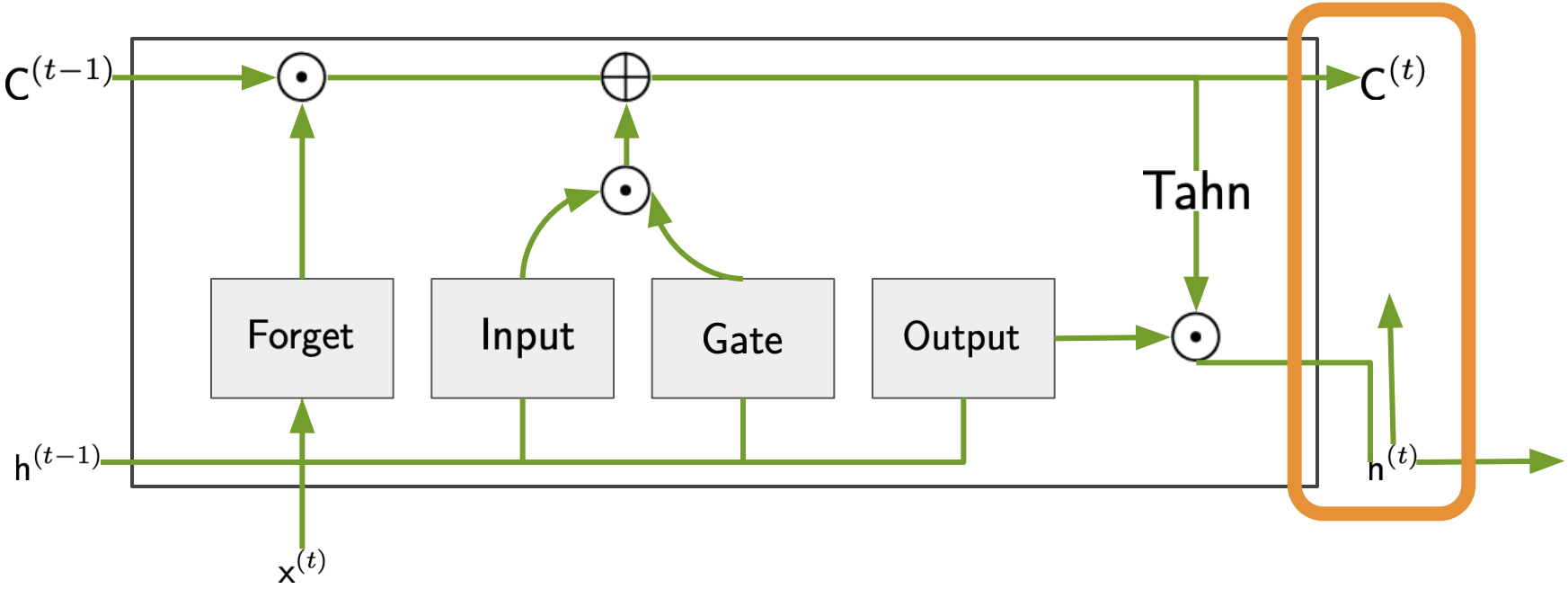


Anatomía de LSTM



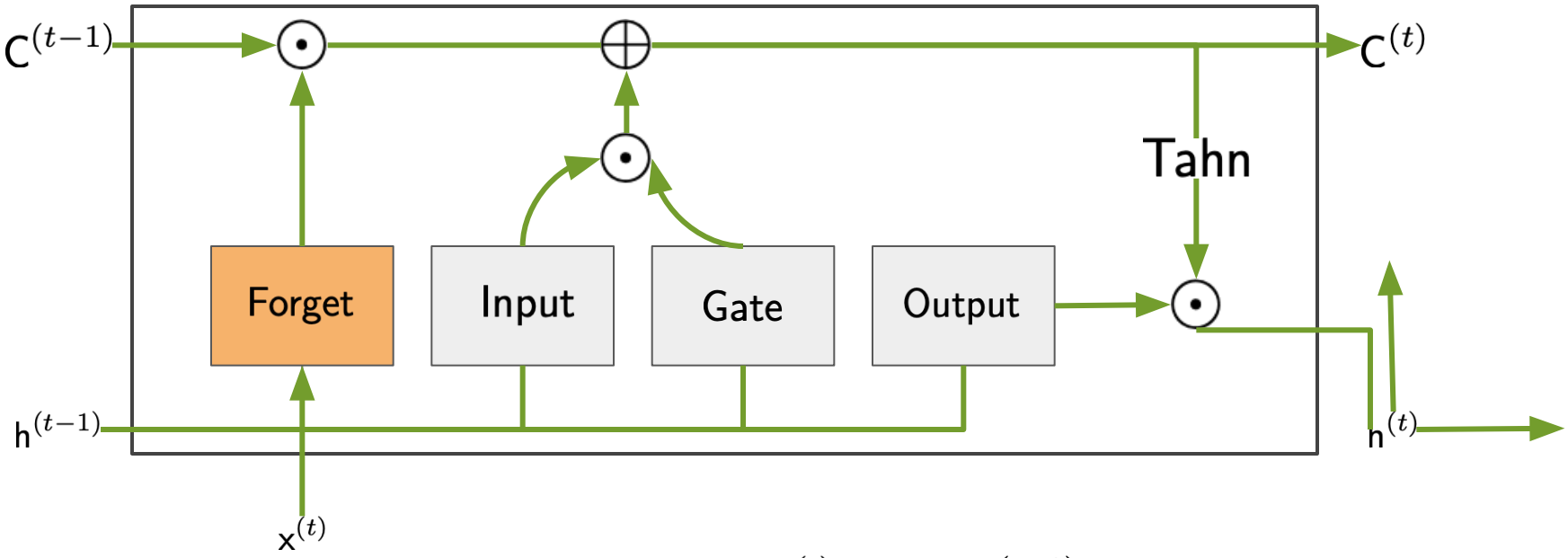
Pesos (h) y estado de celda en la secuencia anterior t-1

Anatomía de LSTM



Pesos (h) y estado de celda resultante propagado a la secuencia t+1

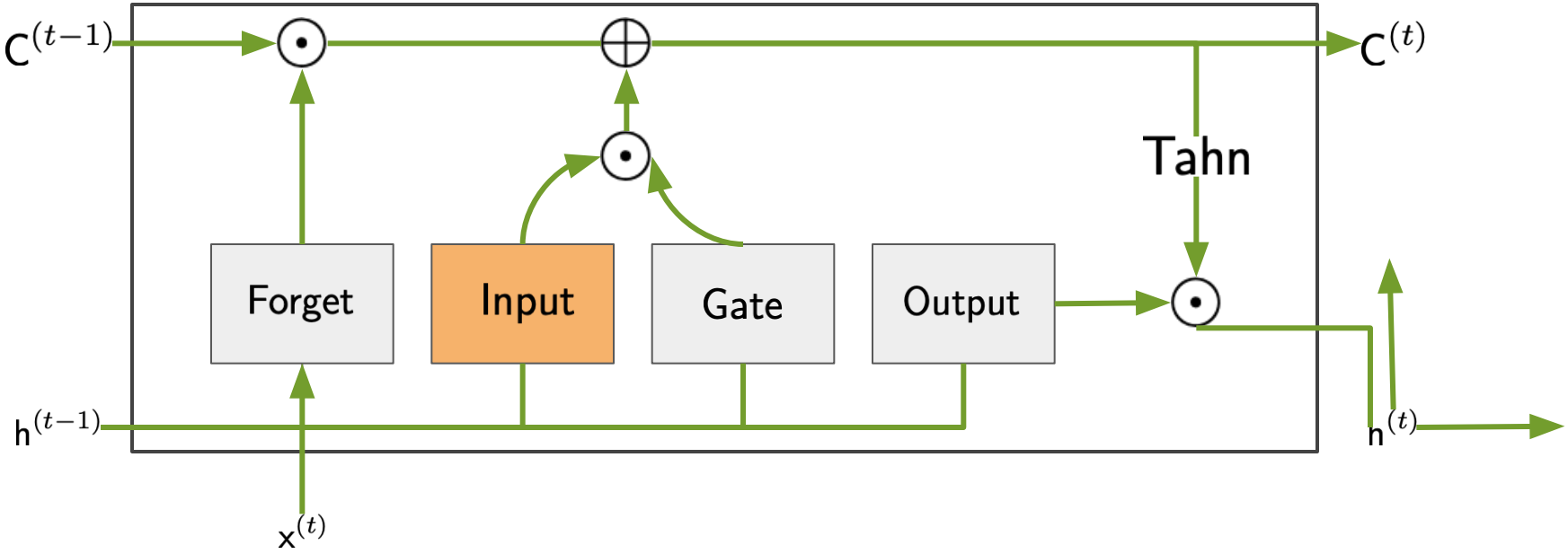
Anatomía de LSTM: Compuerta Forget



$$\text{Forget} = \sigma(\mathbf{W}_{fx}\mathbf{x}^{(t)} + \mathbf{W}_{fx}\mathbf{h}^{(t-1)} + \mathbf{b}_f)$$

Forget Gate: Permite controlar qué información se recuerda y qué información se olvida

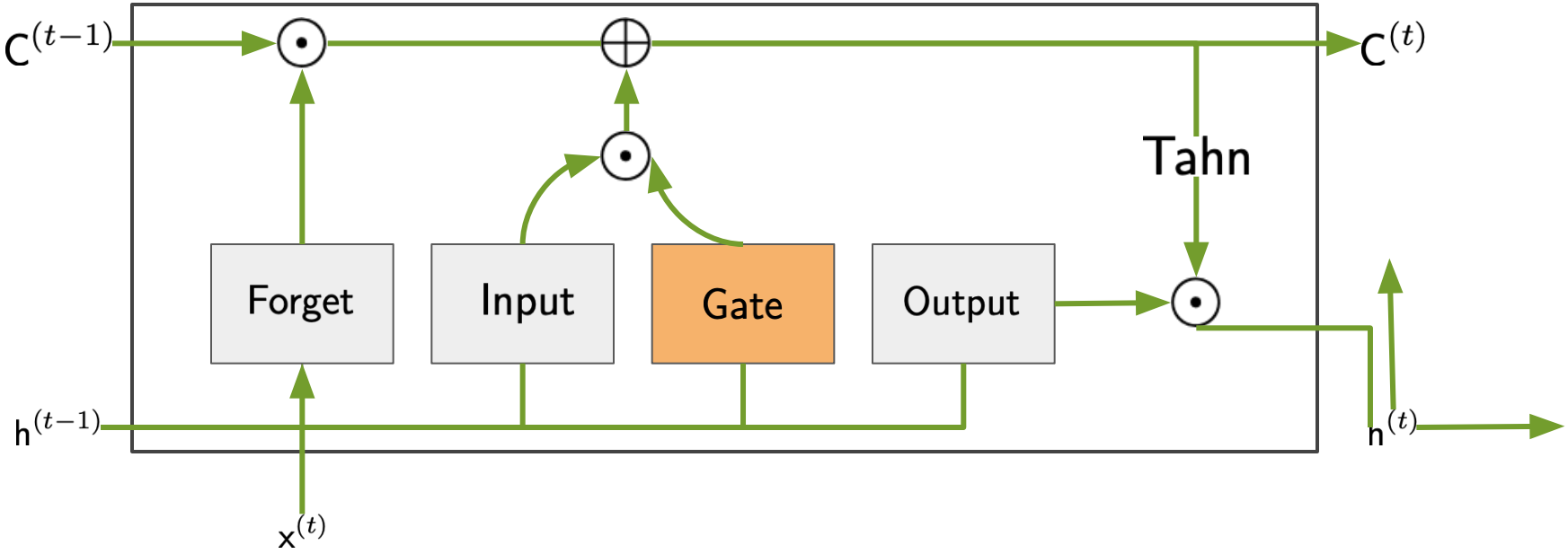
Anatomía de LSTM: Compuerta Input



$$\text{Input} = \sigma(\mathbf{W}_{ix}\mathbf{x}^{(t)} + \mathbf{W}_{ih}\mathbf{h}^{(t-1)} + \mathbf{b}_i)$$

Input: Evalúa qué información es relevante para la celda LSTM.

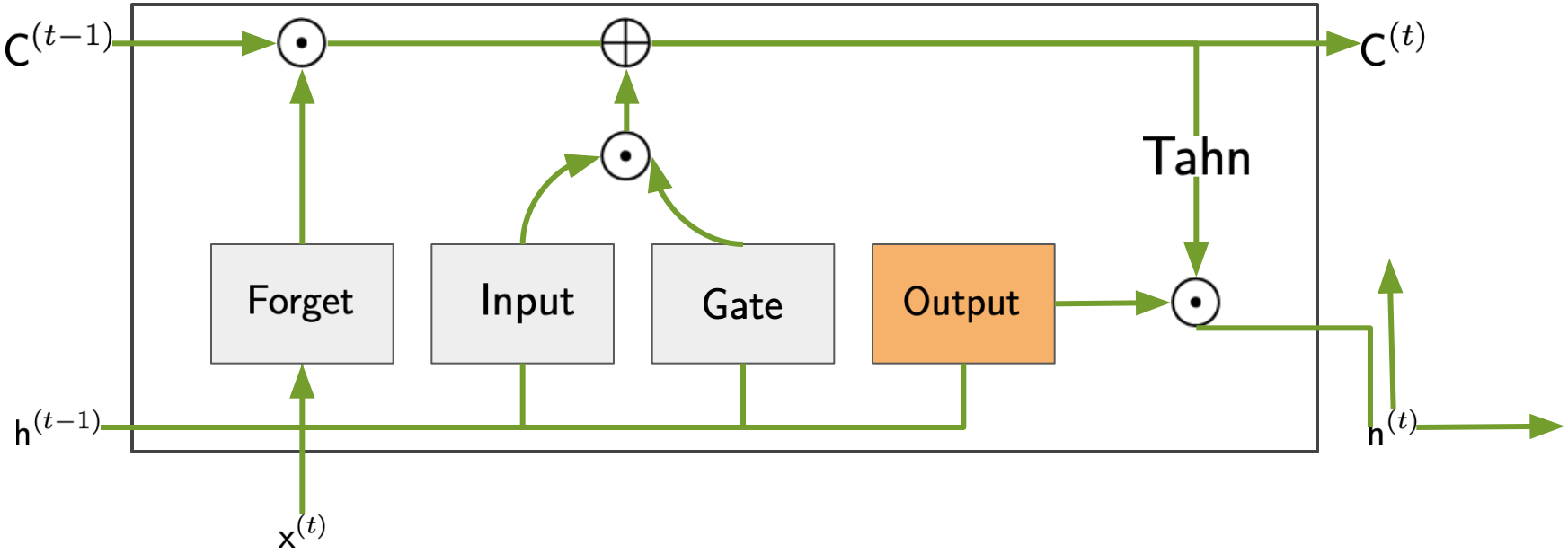
Anatomía de LSTM: Compuerta Gate



$$\text{Gate} = \tanh(\mathbf{W}_{gx}\mathbf{x}^{(t)} + \mathbf{W}_{gh}\mathbf{h}^{(t-1)} + \mathbf{b}_g)$$

Gate: Convierte los valores mediante el arco tangente para facilitar la regularización

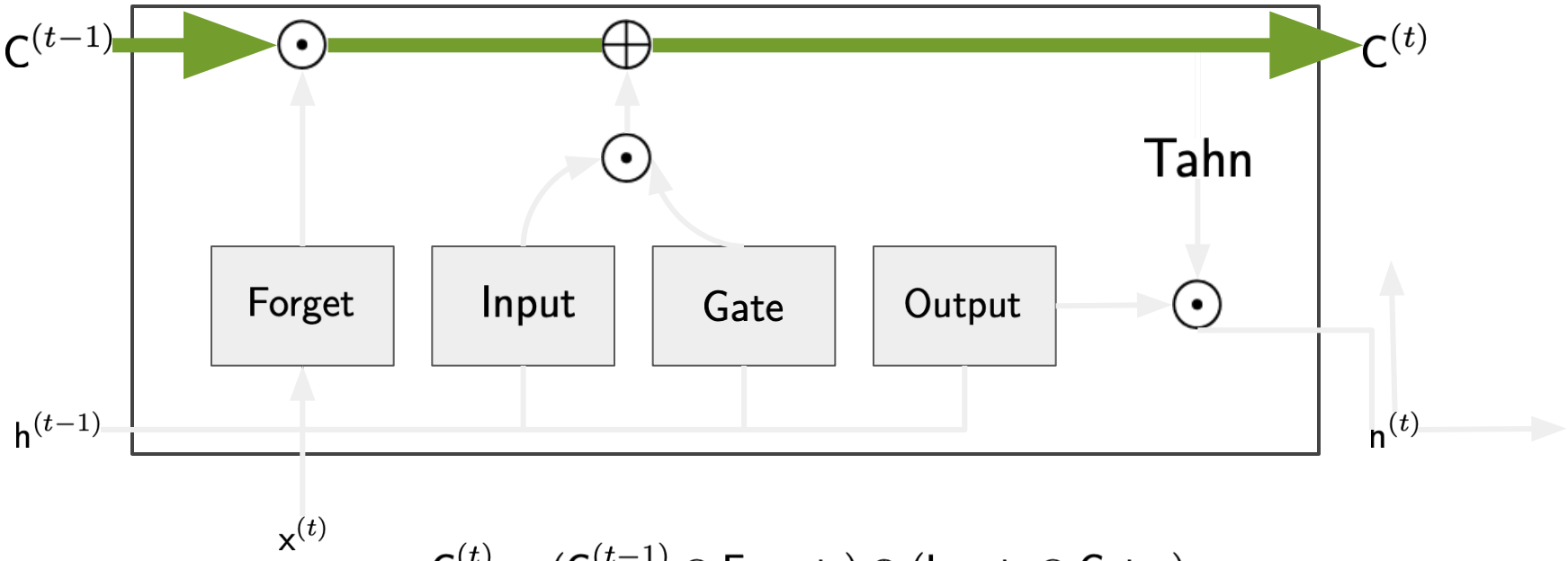
Anatomía de LSTM: Compuerta Output



$$\text{Output Gate}_t = \sigma(\mathbf{W}_{ox}\mathbf{x}^{(t)} + \mathbf{W}_{oh}\mathbf{h}^{(t-1)} + \mathbf{b}_o)$$

Output: Actualizamos los valores en la capa oculta de nuestra red neuronal para posterior uso.

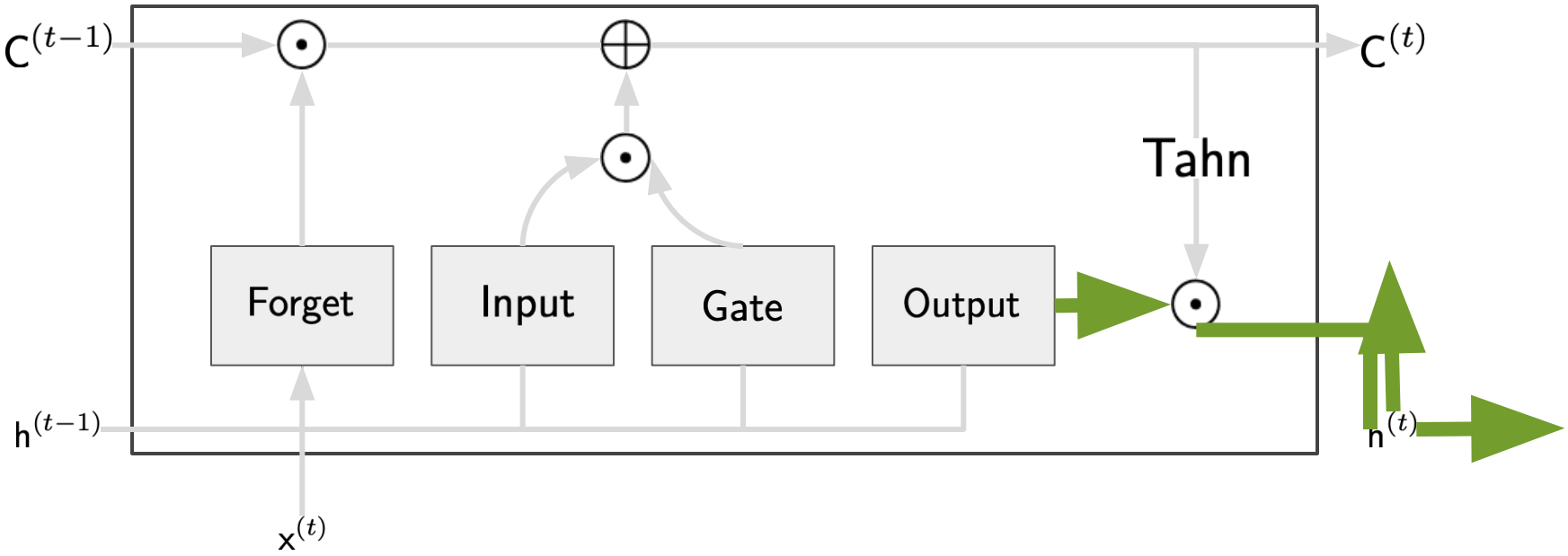
Anatomía de LSTM: Estado de Celda



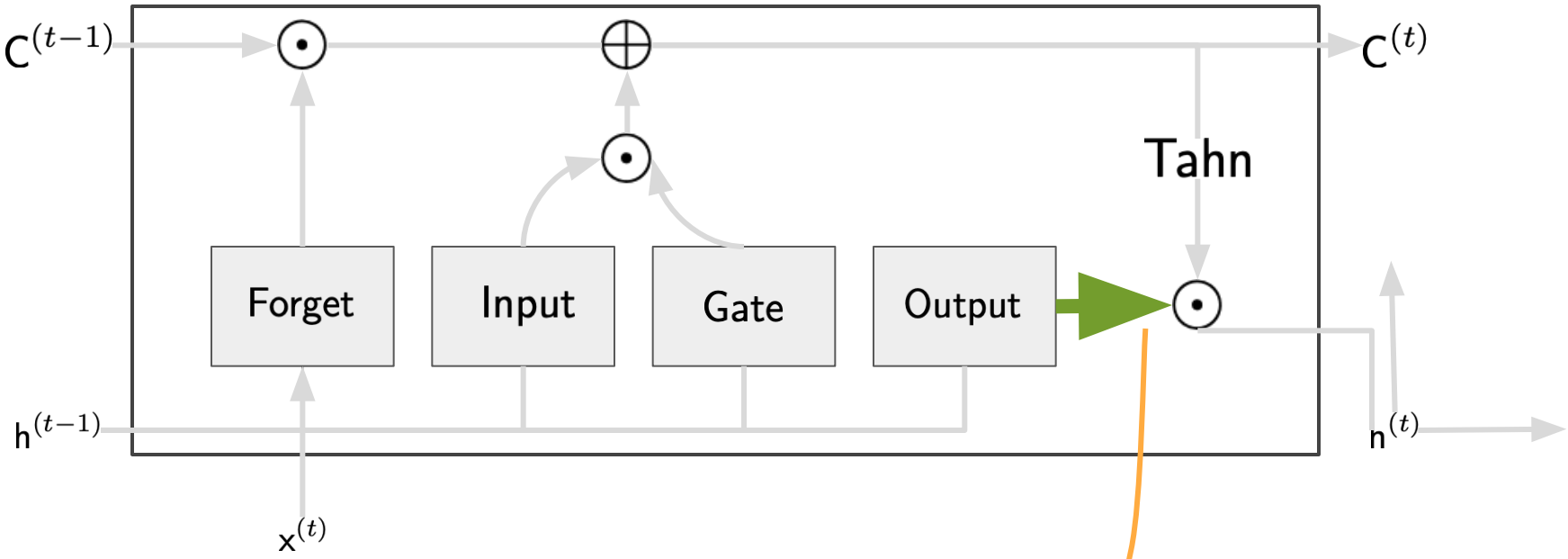
$$C^{(t)} = (C^{(t-1)} \odot \text{Forget}_t) \oplus (\text{Input}_t \odot \text{Gate}_t)$$

Estado de Celda: Representa la memoria contextual de la celda y permite actualizar el output con un valor representativo del estado actual

Anatomía de LSTM: Salida de Capas

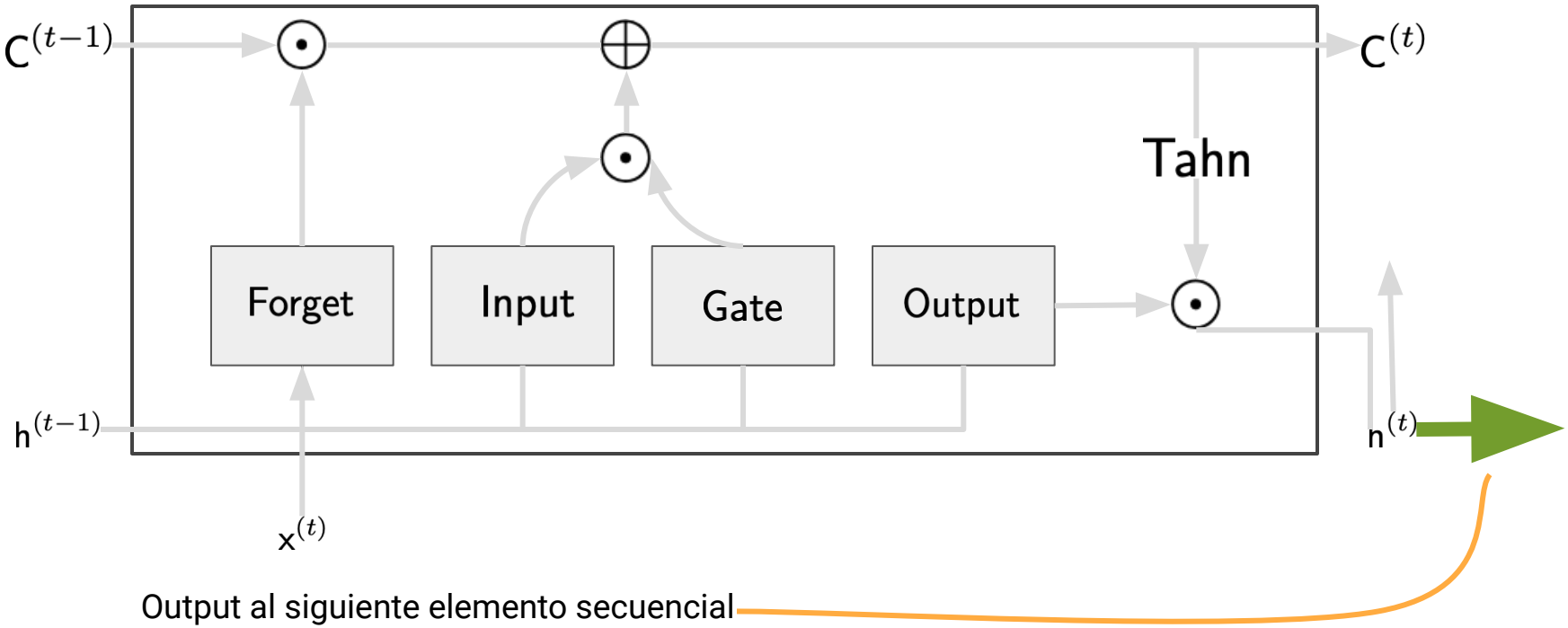


Anatomía de LSTM: Salida de Capas

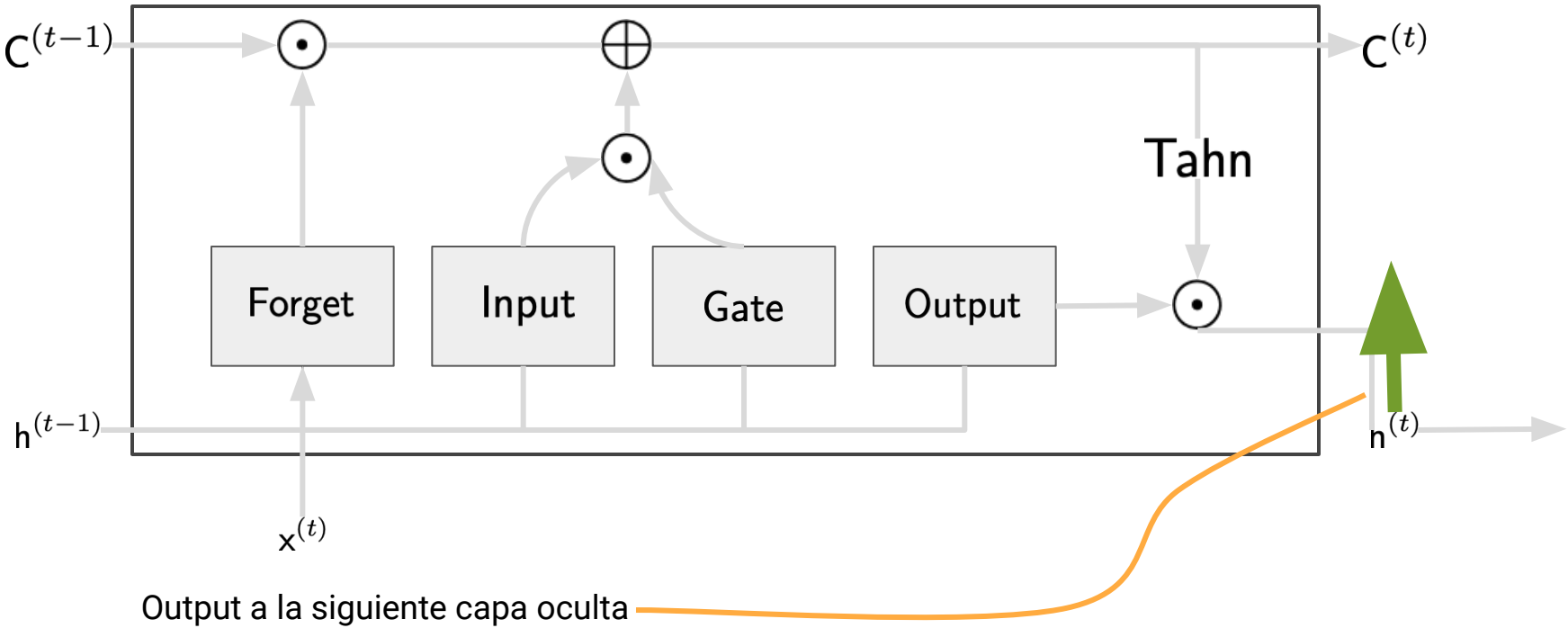


Output a procesar (considerando el estado actual de la celda y la capa previa).

Anatomía de LSTM: Salida de Capas



Anatomía de LSTM: Salida de Capas



Gated Recurrent Units

- **Problema de las LSTM**: La cantidad de compuertas (Input, Forget, Output) aumenta la complejidad del entrenamiento de nuestra red neuronal.
- **Gated Recurrent Units (GRU)** reexpresa los pasos en 2 compuertas:
  - **Reset:** Compuerta que determina cómo mezclar la información previa en t-1 y el input actual en t.
  - **Update:** Compuerta que determina la cantidad de datos en t-1 a preservar.
- De esta forma GRU permite exponer todo el contenido en memoria de la celda.

{desafío} latam_

Academia de talentos digitales

www.desafiolatam.com