




# Time Series con DNN

Oscar Rodriguez

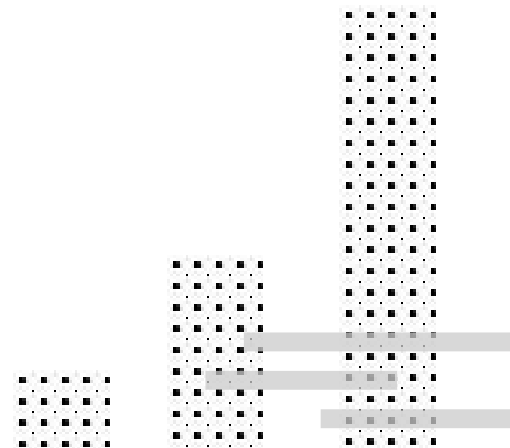
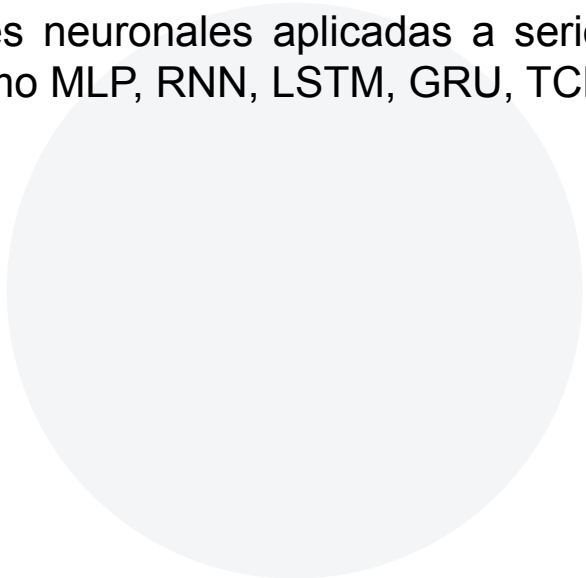
Advanced Time Series Analysis  
27 Agosto 2025



# Repaso



Revisamos algunas redes neuronales aplicadas a series de tiempo, desde arquitecturas simples como MLP, RNN, LSTM, GRU, TCN o Transformers.



## MLP

Red neuronal básica –*Multi-Layer Perceptron*– de capas totalmente conectadas. Captura relaciones no lineales, pero no tiene memoria temporal..

## RNN

Red neuronal –*Recurrent Neural Network*– que añade conexiones recurrentes (retroalimentación con estados pasados) que permiten mantener un "estado oculto" y procesar secuencias paso a paso, modelando dependencias temporales.

## LSTM

Red neuronal –*Long Short-Term Memory*– Variante de RNN que introduce celdas de memoria y compuertas (input, output, forget) para manejar dependencias de largo plazo en secuencias.



## GRU

–*Gated Recurrent Unit*– Variante más ligera de LSTM que combina compuertas de actualización y reinicio, con menos parámetros pero desempeño competitivo en muchas tareas secuenciales.

## TCN

–*Temporal Convolutional Network*– Red convolucional 1D diseñada para secuencias. Usa convoluciones (filtro deslizante) para capturar dependencias de corto y largo plazo de manera paralela y más eficiente que las RNN.

## Transformers

Tipo de red neuronal que usa mecanismos de atención en lugar de recurrencias o convoluciones, lo que les permite aprender relaciones entre todos los elementos de una secuencia de manera eficiente y precisa.

# MLP

Una red neuronal MLP es una composición de funciones lineales y no lineales. Matemáticamente, una MLP de  $n$  capas la podemos definir como:

$$F(x;\theta) = N^n \circ L^n \circ \cdots \circ N^1 \circ L^1(x)$$

Donde para todo  $i=1, \dots, n$  se tiene:

- 1)  $L^i(x) = Wx + b$  (Aplicación lineal afín)
- 2)  $N^i(x) = \sigma^i(x)$  (Aplicación no lineal)

# Ejemplo (MLP)

Supongamos que queremos un MLP de dos capas ocultas y tenemos que entrenar una función definida de  $\mathbb{R}^p$  en  $\mathbb{R}^q$ . Esto indica que el input (x) y el output (y) de la red son respectivamente de dimensión p y q. Ahora consideremos las capas ocultas con a y c número de nodos. Con ésta información procedemos a construir nuestra red neuronal de la siguiente forma:

$$\begin{aligned} 1) Z^1_{ax1} &= \sigma^1(W^1_{axp} x_{px1} + b^1_{ax1}) \\ 2) Z^2_{cx1} &= \sigma^2(W^2_{cxa} Z^1_{ax1} + b^2_{cx1}) \\ 3) \hat{y}_{qx1} &= \sigma^3(W^3_{qxc} Z^2_{cx1} + b^3_{qx1}) \end{aligned}$$

# RNN

Una RNN ([paper](#)) es una composición de funciones lineales y no lineales, pero con la diferencia de que incorpora dependencia temporal mediante un estado oculto que se actualiza recursivamente. Matemáticamente, una RNN con  $n$  pasos temporales se puede definir como:

$$F(x_T; \theta) = (h_1, h_2, \dots, h_T)$$

Donde para todo  $t=1, \dots, T$  se tiene:

- 1)  $h_i = Wx + W'h_{i-1} + b$  (Aplicación lineal afín)
- 2)  $N^i(x) = \sigma^i(h_i)$  (Aplicación no lineal)

# Ejemplo (RNN)

Una RNN no solo depende de espacio definido ( $\mathbb{R}^p$  en  $\mathbb{R}^q$ ). También depende de la cantidad de datos en el tiempo que tengamos ( $x_1, x_2, \dots, x_T$ ). Además una capa recurrente depende de la cantidad de estados o ocultos. Con esto, consideremos 2 capas con 2 y 3 estados en cada una. Por simplicidad, consideremos el mismo número  $m$  de nodos en todos los estados. Con ésta información procedemos a construir nuestra red neuronal de la siguiente forma:

Supuestos:

- 1) Secuencia ( $x_1, x_2, \dots, x_T$ ), donde  $x_i$  esta en  $\mathbb{R}^p$  para  $i=1, \dots, T$
- 2) Dos capas ocultas, una con 2 y otra con 3 estados
- 3) Cada estado  $h_{t,ij}^i$  es un vector de  $m$  nodos ( $j=1,2$ .  $i1=1,2$ .  $i2=1,2,3$ )

# Ejemplo (RNN)

1) Primera capa ( $i=1,2$ ):

$$h_{t,i}^{(1)} = \sigma^i(W_i^{(1)}x_t + \sum_{j=1}^2 U_{ij}^{(1)}h_{t-1,j}^{(1)} + b_i^{(1)})$$

Donde  $W_i^{(1)} \in \mathbb{R}^{m \times p}$ ,  $U_{ij}^{(1)} \in \mathbb{R}^{m \times m}$ ,  $b_i^{(1)} \in \mathbb{R}^m$  y  $h_{0,j}^{(1)} = 0$

2) Segunda capa ( $k=1,2,3$ ):

$$h_{t,k}^{(2)} = \sigma^k(\sum_{i=1}^3 W_{ki}^{(2)}h_{t,i}^{(1)} + \sum_{j=1}^3 U_{kj}^{(2)}h_{t-1,j}^{(2)} + b_k^{(2)})$$

Donde  $W_{ki}^{(2)} \in \mathbb{R}^{m \times m}$ ,  $U_{kj}^{(2)} \in \mathbb{R}^{m \times m}$ ,  $b_k^{(2)} \in \mathbb{R}^m$  y  $h_{0,k}^{(2)} = 0$

3) Salida

$$\hat{y}_t = \sigma^{(y)}(\sum_{k=1}^3 W_k^{(y)}h_{t,k}^{(2)} + b^{(y)})$$

Donde  $W_k^{(y)} \in \mathbb{R}^{q \times m}$  y  $b^{(y)} \in \mathbb{R}$



# Recurrent Neural Networks

