Deep Learning para Business Analytics

Día 06 · Tema 2 — Redes recurrentes

Forecast de demanda, ventas y consumo eléctrico.

Eduardo C. Garrido-Merchán

Profesor Propio Adjunto, Métodos Cuantitativos, ICADE

ecgarrido@comillas.edu

Madrid, Día 06

INDITEX · MERCADONA · iberdrola · zalando · amazon · endesa

ejemplos del día

COMILLAS UNIVERSIDAD PONTIFICIA ICAI ICADE CIHS

Lo que vamos a cubrir hoy.

1. Datos secuenciales y casos de forecasting

2. La RNN básica

3. LSTM y GRU

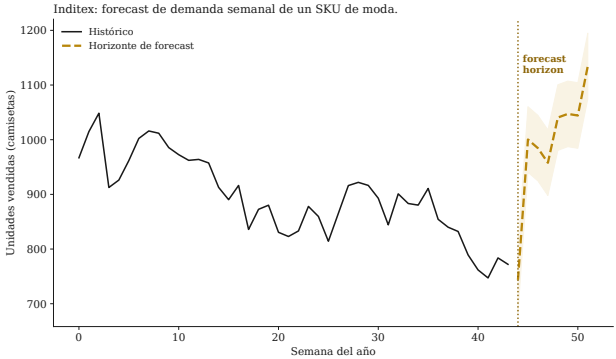
4. Caso de negocio: forecast de demanda

Al final del día: pasar tres pasos de 🧠 RNN a mano, justificar LSTM/GRU sobre RNN básica y montar un 📊 forecaster con PyTorch.

Parte 1 / 4

Datos secuenciales y casos de forecasting

Inditex necesita saber cuántas camisetas venderá la semana que viene.



El problema. 🛒 Producir demasiado = stock muerto. Producir poco = venta perdida. Cada punto de error en el forecast mueve **millones de euros** en logística y rebajas.

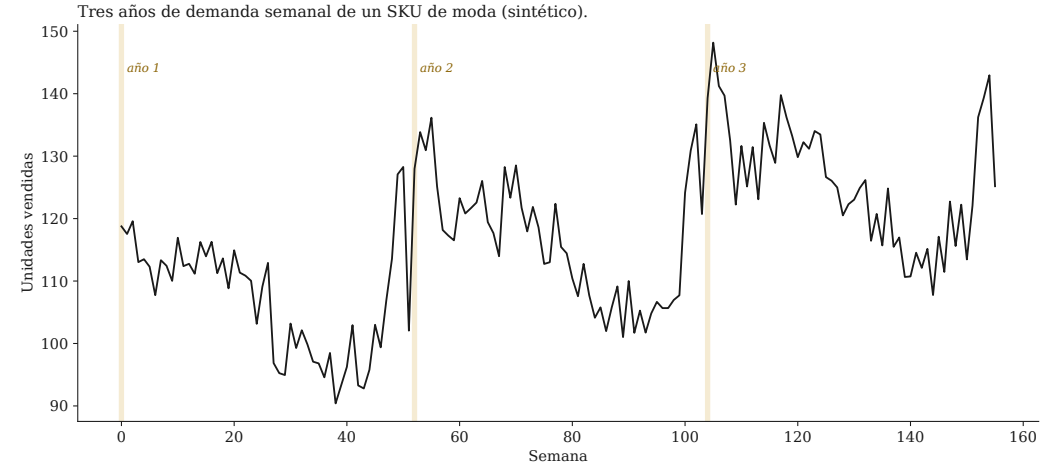
Los datos. 📊 Ventas semanales por SKU, 3+ años de histórico, covariables: precio, promoción, estacionalidad, festivos.

La decisión. El forecast alimenta producción (6 semanas vista), distribución (2 semanas) y rebajas (fin de temporada).

Error tolerable. MAPE < 10 % es el umbral competitivo. Un punto menos de error ≈ **8–12 M €/año** en Inditex.

Idea clave. El forecasting de demanda es el caso de negocio canónico para redes recurrentes.

⏱️ Una serie temporal típica de retail: demanda semanal.



📊 Concepto → aplicación: dónde usamos secuencias.

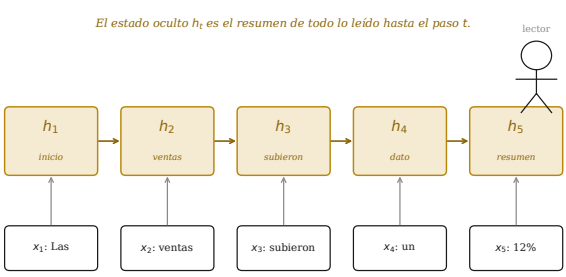
Caso	Variable predicha	Entradas	KPI
🛒 Demanda retail (Inditex)	Unidades / SKU / semana.	Hist. 156 sem., promo, precio comp.	MAPE, días sin stock.
💡 Consumo eléctrico (Iberdrola)	MWh / hora / región.	Hist. horario, temperatura, festivo.	RMSE €/MWh.
🕸️ Tráfico datos (Telefónica)	Tráfico GB / celda / 5 min.	Hist. 5-min, evento, calendario.	Pico no servido.
💰 Precio energía (Endesa)	€/MWh hora siguiente.	Hist. precio, oferta, demanda.	€ de cobertura.
💬 Voz sintética (at. cliente)	Audio token siguiente.	Texto + tono.	MOS, % humanos.

Idea clave. El mismo tipo de modelo cambia de sector con solo cambiar features de entrada.

Parte 2 / 4

La RNN básica

Una RNN es alguien que lee palabra a palabra y va tomando notas.



La analogía. 🧠 Imagina que lees un informe de ventas palabra a palabra. Después de cada palabra, actualizas tus “notas mentales”.

Estado oculto h_t. 📖 Es el resumen de todo lo leído hasta el paso t. Se actualiza en cada paso con la entrada nueva x_t y las notas anteriores h_{t-1}.

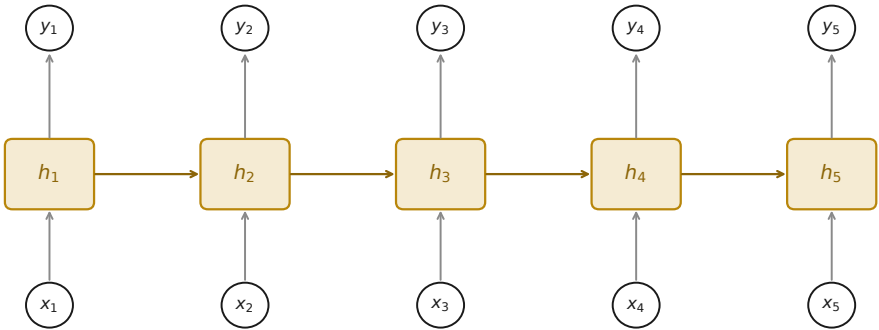
Pesos compartidos. La regla de actualización es la misma en cada paso: los mismos W_x, W_h, b.

Límite. Después de muchos pasos, las notas “se desvanecen”: la RNN olvida lo antiguo. Por eso necesitamos LSTM.

Idea clave. h_t es la “memoria de trabajo” de la red: comprime toda la historia en un vector.

🧠 RNN desenrollada: la memoria que viaja en el tiempo.

RNN "desenrollada": el estado oculto h_t es la memoria que viaja en el tiempo.



Ecuación de la RNN, en una línea.

$$h_t = \tanh(W_x x_t + W_h h_{t-1} + b), \qquad y_t = w_y h_t.$$

Lectura. El estado h_t depende de la entrada actual x_t y del estado anterior h_{t-1}. Los pesos W_x, W_h, w_y, b son los mismos en todos los pasos.

Limitación. ⚠️ Al hacer backprop a través del tiempo, los gradientes pueden *desaparecer* o *explotar*. La RNN básica olvida lo de hace 20-30 pasos.

Idea clave. LSTM y GRU son la solución a esa limitación.

Ejemplo a mano: tres pasos de RNN sobre demanda.

Tres pasos a mano de una RNN sobre demanda escalada (Wx=0,01; Wh=0,5; b=-0,2; wy=1,5).

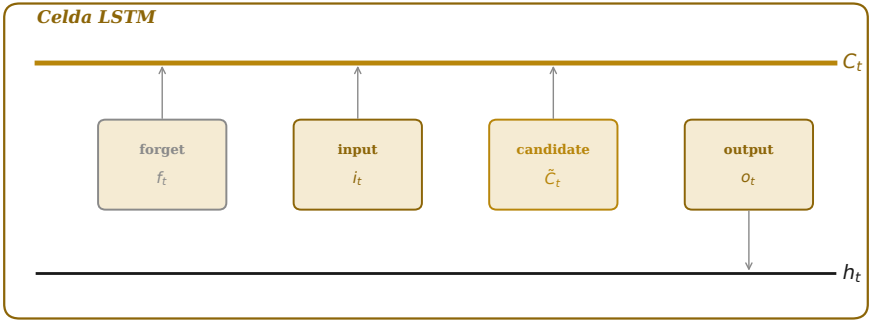
Paso t	x_t **(venta)**	h_{t-1}	$W_x x_t + W_h h_{t-1} + b$	$h_t = \tanh(\cdot)$	$y_t = w_y h_t$
1	100	0,00	$0.01(100) + 0.5(0) - 0.2 = 0.80$	$\tanh(0.80) = 0.66$	$1.5(0.66) =$ **0.99**
2	108	0,66	$0.01(108) + 0.5(0.66) - 0.2 = 1.21$	$\tanh(1.21) = 0.84$	$1.5(0.84) =$ **1.26**
3	112	0,84	$0.01(112) + 0.5(0.84) - 0.2 = 1.34$	$\tanh(1.34) = 0.87$	$1.5(0.87) =$ **1.30**

Parte 3 / 4

LSTM y GRU

⚙️ LSTM: tres puertas controlan memoria y salida.

Tres puertas controlan qué olvidar, qué guardar y qué emitir.



⚙️ LSTM en cuatro ecuaciones esenciales.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \qquad \text{(puerta de olvido)}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \qquad \text{(puerta de entrada)}$$

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

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \qquad \text{(memoria)}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o);\ \ h_t = o_t \odot \tanh(C_t) \qquad \text{(salida)}$$

Idea clave. La memoria C_t es un canal aparte que conserva información a largo plazo.

LSTM paso a paso: dos instantes con números reales.

	$t=1$	$t=2$
Entrada x_t	1.0	0.5
$f_t = \sigma(z_f)$	$\sigma(0.60) = 0.646$	$\sigma(0.48) = 0.618$
$i_t = \sigma(z_i)$	$\sigma(0.70) = 0.668$	$\sigma(0.51) = 0.625$
$\tilde{C}_t = \tanh(z_c)$	$\tanh(0.70) = 0.604$	$\tanh(0.54) = 0.489$
$o_t = \sigma(z_o)$	$\sigma(0.80) = 0.690$	$\sigma(0.66) = 0.659$
$C_t = f_t C_{t-1} + i_t \tilde{C}_t$	0.404	0.556
$h_t = o_t \tanh(C_t)$	0.264	0.333

Pesos: W_f=0.5, W_i=0.8, W_c=0.7, W_o=0.6; sesgos: b_f=0.1, b_i=−0.1, b_c=0, b_o=0.2

Setup. $h_0=0$, $C_0=0$; $x_1=1{,}0$, $x_2=0{,}5$. Pesos: $W_f=0{,}5$, $W_i=0{,}8$, $W_c=0{,}7$, $W_o=0{,}6$. Sesgos: $b_f=0{,}1$, $b_i=-0{,}1$, $b_c=0$, $b_o=0{,}2$.

$t=1$: **la puerta de olvido.**
$f_1 = \sigma(0{,}5 \cdot 0 + 0{,}5 \cdot 1 + 0{,}1) = \sigma(0{,}60) = \mathbf{0.646}$.

$t=1$: **memoria.** $C_1 = 0{,}646 \cdot 0 + 0{,}668 \cdot 0{,}604 = \mathbf{0.403}$.

$t=1$: **salida.** $h_1 = 0{,}690 \cdot \tanh(0{,}403) = \mathbf{0.264}$.

Tabla de referencia. $\sigma(0{,}6)=0{,}646$, $\sigma(0{,}7)=0{,}668$, $\sigma(0{,}8)=0{,}690$, $\tanh(0{,}7)=0{,}604$, $\tanh(0{,}4)=0{,}380$.

Idea clave. La puerta de olvido $f_t \approx 0{,}65$ deja pasar ∼65 % de la memoria anterior.

⚙️ GRU: versión simplificada, dos puertas en vez de tres.

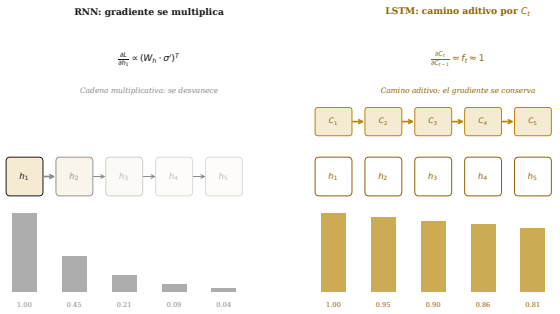
Diferencia. 📚 GRU fusiona las puertas de olvido e input en una sola "puerta de actualización" z_t, y elimina la celda C separada.

Resultado. 25 % menos parámetros que LSTM, métricas similares en la mayoría de tareas, entrenamiento ligeramente más rápido.

Cuándo preferir cada uno. LSTM cuando la dependencia temporal es muy larga; GRU cuando el dataset es pequeño y se quiere generalizar. En la práctica, probad ambas.

Idea clave. Si dudas entre LSTM y GRU, prueba GRU primero: menos parámetros, menos riesgo.

Por qué el gradiente se desvanece en RNN pero no en LSTM.



RNN. ⚠️ El gradiente de L respecto a h_1 pasa por una cadena multiplicativa: $\prod_t (W_h^\top \cdot \mathrm{diag}(\sigma'))$. Con $\|W_h \sigma'\| < 1$, el producto tiende a **cero** exponencialmente.

LSTM. ⚙️ La celda $C_t = f_t \odot C_{t-1} + \ldots$ introduce un camino *aditivo*. El gradiente $\frac{\partial C_t}{\partial C_{t-1}} = f_t$.

El truco. Si $f_t \approx 1$ (puerta de olvido casi abierta), el gradiente viaja intacto por la "autopista de la celda".

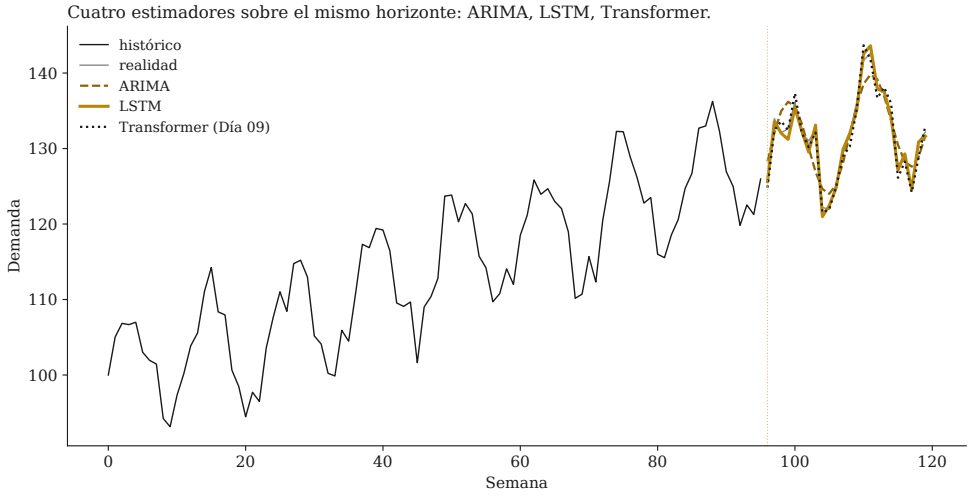
Números. Tras 5 pasos, el gradiente RNN cae a ~0.04. En LSTM se mantiene en ~0.77.

Idea clave. La celda C_t es un canal de gradiente: por eso LSTM aprende dependencias largas.

Parte 4 / 4

Caso de negocio: forecast de demanda

Cuatro estimadores sobre el mismo horizonte.



🔧 Receta exprés del primer LSTM de forecast.

```
import torch.nn as nn
class Forecaster(nn.Module):
    def __init__(self, n_features, hidden=64, n_layers=2):
        super().__init__()
        self.lstm = nn.LSTM(n_features, hidden, n_layers,
                dropout=0.2, batch_first=True)
        self.head = nn.Linear(hidden, 1)
    def forward(self, x): # x: (batch, T, n_features)
        out, _ = self.lstm(x)
        return self.head(out[:, -1]) # último paso

model = Forecaster(n_features=5)
opt = torch.optim.AdamW(model.parameters(), lr=1e-3)
```

Idea clave. Una LSTM de dos capas y dropout es el "primer prototipo correcto"para series temporales tabulares.

🎓 Lo que el alumno se lleva a casa de este día.

Concepto 1. 🧠 Una RNN reusa los mismos pesos en cada paso temporal y propaga un estado oculto.

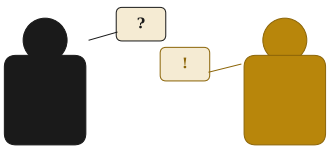
Concepto 2. ⚙️ LSTM añade memoria explícita (C_t) y tres puertas; GRU es la versión simplificada con dos.

Concepto 3. 📊 En 2026 hay siempre que comparar contra ARIMA/ETS y contra un Transformer (Día 09): la red recurrente gana cuando hay no-linealidades y covariables externas.

Para la práctica. 🔧 Forecast con LSTM sobre demanda multiproducto del dataset Online Retail II.

Próximo día. 🎯 Día 07 (Tema 2.4): conexión a la atención y al Transformer.

Pausa para debatir: ¿Por qué casi nadie usa LSTM en 2026?



La pregunta. 🧠 Los benchmarks aún defienden a LSTM en muchos forecasting, pero la industria casi solo despliega ⚡ Transformer. ¿Por qué?

Posición A. 🔥 Por moda y por capacidad de contratar talento, no por mérito técnico.

Posición B. 🚀 Por mérito: paralelismo, contexto largo y mejor calidad agregada.

Reglas. 5 minutos, dos turnos de palabra por bando, móviles boca abajo. Yo modero, no participo.

Hasta el Día 07.

De la recurrencia a la atención.

Eduardo C. Garrido-Merchán

ecgarrido@comillas.edu