# Día 01 · Tema 0 — Marco formal de la asignatura: aprendizaje profundo, agentes y coste de inferencia

Eduardo C. Garrido-Merchán · ecgarrido@comillas.edu Universidad Pontificia Comillas · ICADE

## 1. Riesgo esperado y aprendizaje supervisado

Toda la asignatura es una variante del mismo problema. Dada una distribución desconocida $\mathcal{D}$ sobre pares $(x, y) \in \mathcal{X} \times \mathcal{Y}$, buscamos un modelo $f_\theta : \mathcal{X} \to \mathcal{Y}$, parametrizado por $\theta \in \Theta$, que minimice el riesgo esperado

$$\mathcal{R}(\theta) \;=\; \mathbb{E}_{(x,y)\sim\mathcal{D}}\big[\,\ell(f_\theta(x),\, y)\big], \tag{1}$$

donde $\ell$ es la *función de pérdida*. Como $\mathcal{D}$ no es accesible, sustituimos el riesgo por su versión empírica sobre $n$ ejemplos i.i.d.:

$$\hat{\mathcal{R}}_n(\theta) \;=\; \frac{1}{n}\sum_{i=1}^{n} \ell\big(\hat{y}_i,\, y_i\big), \qquad \hat{y}_i := f_\theta(x_i). \tag{2}$$

| | |
|---|---|
| $\mathcal{D}$ | distribución conjunta de datos y etiquetas |
| $(x_i, y_i)$ | i-ésimo par observado, con $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$ |
| $f_\theta$ | el modelo, una función parametrizada por $\theta$ |
| $\ell$ | pérdida: cuán mal predice $\hat{y}$ frente a $y$ |
| $n$ | tamaño del conjunto de entrenamiento |
| $\hat{\mathcal{R}}_n$ | riesgo empírico, lo que minimiza el optimizador |

**Idea clave.** Tanto un MLP como un Transformer como un agente *son*, en el fondo, instancias de (2) con distintas $f_\theta$ y distintas $\ell$.

## 2. La red profunda como composición

Una red neuronal profunda es una composición $L$-capa

$$f_\theta(x) \;=\; \big(\,h_L \circ h_{L-1} \circ \cdots \circ h_1\,\big)(x), \quad h_\ell(z) = \sigma_\ell\big(W_\ell z + b_\ell\big), \tag{3}$$

donde cada capa transforma su entrada con un mapeo afín seguido de una no-linealidad. Sin las $\sigma_\ell$ la composición colapsa a una función lineal, así que la no-linealidad es lo que da expresividad real.

| | |
|---|---|
| $W_\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ | matriz de pesos de la capa $\ell$ |
| $b_\ell \in \mathbb{R}^{d_\ell}$ | sesgo (bias) de la capa $\ell$ |
| $\sigma_\ell$ | activación (ReLU, GELU, SiLU, sigmoide, …) |
| $L$ | profundidad: cuántas capas componen el modelo |
| $\theta$ | conjunto de todos los $(W_\ell, b_\ell)$ apilados |

## 3. Familias de modelos vistas en el deck

La taxonomía de la diapositiva "Qué arquitectura para qué tipo de datos" se reduce a las familias siguientes, todas instancias de (3) con restricciones específicas en la estructura de $W_\ell$:

- **MLP / tabular**. Sin restricciones; $W_\ell$ es densa. Apropiada para vectores estructurados de tamaño moderado.
- **CNN**. $W_\ell$ es Toeplitz por bloques con *pesos compartidos* (la convolución reutiliza el mismo *kernel* a lo largo del espacio). Reduce parámetros y codifica equivarianza espacial.
- **RNN / LSTM / GRU**. Capa recurrente $h_t = \sigma(W h_{t-1} + U x_t + b)$, con el mismo $(W, U, b)$ en cada paso de tiempo. Coste lineal en la longitud $T$.
- **Transformer**. La operación dominante es la atención escalada

$$\mathrm{Att}(Q, K, V) \;=\; \mathrm{softmax}\!\left(\frac{QK^\top}{\sqrt{d_k}}\right) V, \tag{4}$$

  que es no local (cada token mira a todos) y, por construcción, tiene coste cuadrático $O(T^2)$ en la longitud de la secuencia.
- **Mixture of Experts**. Capa $h = \sum_{k=1}^{K} g_k(x)\, E_k(x)$ con *gating* $g_k$ esparso (*top-k*); la capacidad total $K$ es grande pero el coste por token depende sólo de los $k$ activados.
- **State Space (Mamba, SSM)**. Recurrencia continua discretizada $h_t = A_t h_{t-1} + B_t x_t$, $y_t = C_t h_t$, con matrices que dependen de la entrada. Coste lineal en $T$ y memoria fija.

### 4. Coste de inferencia y la decisión cloud vs. local

La diapositiva de coste compara modelos según dos magnitudes que se calculan de forma estándar. La *tasa de generación* en tokens por segundo es

$$\text{tps} \;=\; \frac{N_{\text{out}}}{t_{\text{eval}}}, \qquad \text{con } t_{\text{eval}} = \text{eval\_duration en s.} \tag{5}$$

$N_{\text{out}}$ tokens generados (eval_count que devuelve Ollama)
$t_{\text{eval}}$ tiempo de *decoding* en segundos

El coste monetario de una llamada cloud es

$$C \;=\; \frac{N_{\text{in}}}{10^6}\, p_{\text{in}} \;+\; \frac{N_{\text{out}}}{10^6}\, p_{\text{out}}, \tag{6}$$

con $p_{\text{in}}$ y $p_{\text{out}}$ los precios por millón de tokens (input/output). Para el pipeline local con `Ollama` el coste marginal $C \approx 0$ tras descarga, así que la métrica relevante pasa a ser la latencia, dominada por la VRAM disponible y los *tokens/seg* de la GPU o CPU.

### 5. El agente como bucle de toma de decisiones

Lo que el deck llama *context engineering* corresponde a un bucle formal donde, en cada paso $t$, el agente $f_\theta$ ve un contexto $c_t$, produce una acción $a_t$ (puede ser texto, llamada a una herramienta o escritura en memoria) y recibe una observación $o_t$ del entorno:

$$\big(a_t,\; c_{t+1}\big) \;=\; f_\theta\big(c_t\big), \qquad c_{t+1} \;=\; c_t \;\cup\; \{a_t, o_t\}. \tag{7}$$

Lejos de ser un chatbot, esto es un proceso de Markov: la *política* $\pi(a_t \mid c_t)$ es lo que el LLM implementa, y el contexto $c_t$ —que va creciendo— se aproxima a una memoria episódica. Una parte del curso (Bloque II) trata exactamente esta construcción.

### 6. Política de uso de IA, formalmente

Si el alumno entrega un artefacto $A$ producido en colaboración con una herramienta $T$, llamamos $\text{tr}(A)$ a la *traza*: el conjunto ordenado de prompts y respuestas usados. La regla de la asignatura es sencilla:

$$\text{nota}(A) \;=\; \begin{cases} \text{calificada} & \text{si } \text{tr}(A) \neq \varnothing \text{ y se defiende,} \\ 0 & \text{si } \text{tr}(A) \neq \varnothing \text{ pero no se defiende,} \\ \text{calificada} & \text{si } \text{tr}(A) = \varnothing \text{ (trabajo propio).} \end{cases} \tag{8}$$

La trazabilidad y la capacidad de defender son condiciones necesarias para que la nota exista: una entrega sin trazabilidad o sin defensa oral sólida es 0, independientemente de cuán brillante parezca.