Facsímil 10

Aprendizaje por refuerzo

Capítulo 08

Recapitulación de aprendizaje por refuerzo

686f6c61 · 2026

Capítulo 08: Recapitulación de aprendizaje por refuerzo

Un facsímil sobre aprendizaje por refuerzo no debería terminar con una frase bonita sobre agentes que aprenden. Debería terminar con una carpeta que puedas ejecutar, una política que puedas comparar, una recompensa que puedas discutir y una decisión que puedas defender.

Ese es el objetivo de este cierre. No vamos a entrenar un modelo grande. Vamos a hacer algo más controlado y mucho más útil para aprender: construir un circuito pequeño donde se vean las piezas esenciales de RL aplicado a sistemas de IA.

La pregunta final del facsímil no es:

> ¿Qué algoritmo parece más sofisticado?

La pregunta final es:

> ¿Publicarías esta política, con esta recompensa, sobre estos datos, bajo estos límites?

Si no puedes contestar eso, todavía no tienes un sistema de refuerzo. Tienes una simulación suelta.

Qué deberías poder hacer al terminar

Este facsímil no pretende que salgas entrenando un modelo de razonamiento desde cero. Pretende algo más práctico: que puedas mirar cualquier sistema que optimiza comportamiento y preguntar qué política ejecuta, qué recompensa persigue, qué futuro ignora y qué evidencia deja.

RESULTADO DE APRENDIZAJE	EVIDENCIA DE QUE LO SABES HACER
Formalizar una decisión secuencial.	Escribes S, A, P, R, γ, identificas estado, acción, transición, recompensa y horizonte.
Leer una política como contrato operativo.	Distingues la política diseñada, la política versionada, la política servida y la política de reserva.
Calcular retorno y valor.	Separas recompensa inmediata, retorno descontado, valor esperado y coste de oportunidad.
Validar exploración.	Mides regret, coste, cobertura de acciones y límites de tráfico.
Diseñar recompensas con cuidado.	Detectas términos que premian señales cómodas pero equivocadas.
Entender post-training moderno.	Separas SFT, DPO, RLHF, RLAIF, RFT y RLVR por la señal que usan.
Evaluar sin desplegar a ciegas.	Usas trazas, propensión, replay, casos de prueba y gates.
Preparar una publicación limitada.	Defines shadow, piloto, rollback, drift, SLI, SLO y runbook.

La idea de cierre:

> RL es ingeniería de consecuencias. No basta con que una acción parezca buena: hay que medir qué futuro abre y qué coste deja detrás.

Lo que hemos construido

CAPÍTULO	PREGUNTA	ARTEFACTO TÉCNICO
01	¿Qué significa decidir por interacción?	MDP, política, retorno, valor y Bellman.
02	¿Qué datos hacen posible aprender de interacción?	Eventos, trayectorias, linaje, propensión y replay buffer.
03	¿Cómo aprende una política sin dejar de actuar?	Bandits, exploración, regret, UCB y límites de exposición.
04	¿Cómo evaluaríamos sin desplegar?	Evaluación offline, contrafactual, OPE y sesgo de logging.
05	¿Cómo se conectan preferencias y post-training?	SFT, pares de preferencia, recompensa, DPO, RLHF, RLVR.
06	¿Cómo diseñamos recompensas defendibles?	Reward cards, verificadores, casos negativos y gates.
07	¿Cómo vive una política en producción?	Serving, monitorización, drift, trazas y rollout.
08	¿Cómo lo practico sin entrenar un LLM?	Simulador, auditoría, entrega reproducible y decisión de publicación.

El patrón completo es este:

1. Definir el problema como decisión.
2. Registrar datos de interacción.
3. Comparar políticas.
4. Diseñar o auditar la recompensa.
5. Validar antes de publicar.
6. Publicar con límites.
7. Medir si el mundo cambió.

Fórmulas mínimas que debes llevarte

Un cierre de RL sin fórmulas se queda blando. No hace falta convertir cada práctica en un paper, pero sí conviene tener una chuleta mental. Estas son las piezas que aparecen una y otra vez.

MDP

$$\mathcal{M} = (S, A, P, R, \gamma)$$

SÍMBOLO	LECTURA
S	Conjunto de estados observables o parcialmente observables.
A	Conjunto de acciones posibles.
$P(s' \mid s, a)$	Dinámica: probabilidad de pasar al siguiente estado.
$R(s, a, s')$	Recompensa asociada a la transición.
γ	Factor de descuento entre presente y futuro.

La pregunta de ingeniería es: ¿de verdad tengo estado, acción, consecuencia y horizonte? Si solo tengo una clasificación aislada, quizá no necesito RL. Si tengo decisiones repetidas con consecuencias acumuladas, entonces sí empieza a oler a problema secuencial.

Política

$$a_t \sim \pi_{\theta,v}(a \mid s_t, c_t)$$

La política no es una idea abstracta. En producción debería tener versión, contexto, parámetros, trazas y política de reserva. Cuando alguien dice “el sistema decidió”, un ingeniero debería poder preguntar: ¿qué versión de la política?, ¿con qué entrada?, ¿con qué probabilidad?, ¿contra qué alternativa?, ¿con qué gate?

Retorno

$$G_t = \sum_{k=0}^{T-t} \gamma^k r_{t+k}$$

La recompensa inmediata puede ser buena y el retorno malo. Un sistema puede ahorrar tokens hoy y crear más trabajo mañana. Puede resolver una pregunta de forma rápida y aumentar reclamaciones. Por eso el retorno fuerza a pensar en consecuencias, no solo en el marcador del instante.

Regret

$$\text{Regret}_T = T\mu^{*} - \sum_{t=1}^{T} r_t$$

SÍMBOLO	LECTURA
T	Número de rondas.
μ^{*}	Recompensa media de la mejor acción conocida en el escenario.
r_t	Recompensa obtenida en la ronda t.

El regret traduce una intuición muy práctica: cuánto te ha costado aprender. Una política puede estar aprendiendo, pero si el coste de exploración es demasiado alto para el dominio, no debería exponerse igual que en una demo.

Recompensa compuesta

$$R(x, y) = w_c C(x, y) + w_e E(x, y) + w_a A(x, y) + w_f F(x, y) - w_k K(x, y)$$

TÉRMINO	LECTURA EN LA PRÁCTICA
C	Corrección de la respuesta.
E	Evidencia o cita válida.
A	Abstención cuando falta fuente suficiente.
F	Formato validable.
K	Coste de tokens y herramientas.

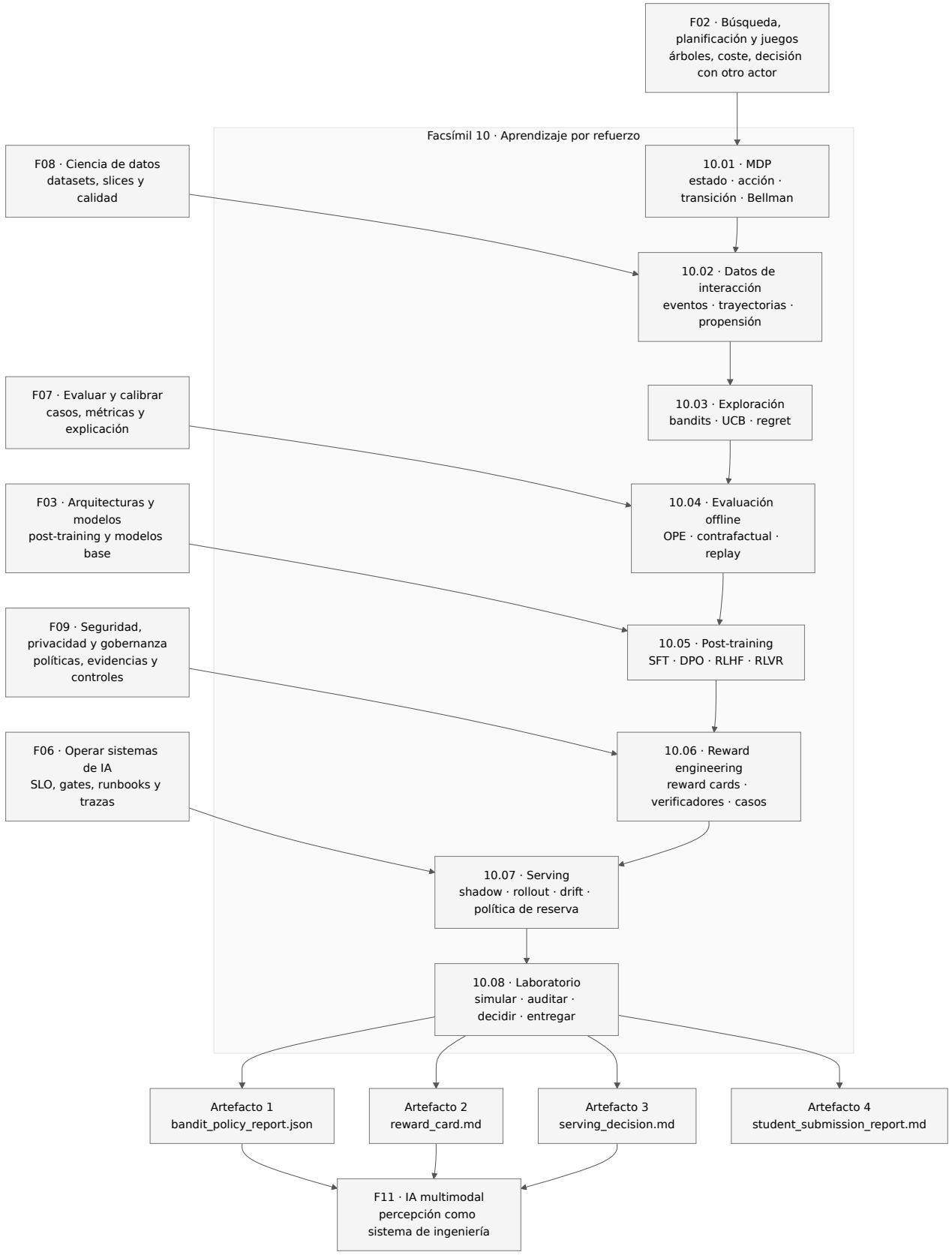
La recompensa compuesta obliga a poner números donde antes había intención. Eso no la vuelve perfecta. La vuelve revisable.

Gate de publicación

$$\text{gate}(\pi_v) = \mathbf{1}[R_{acum} \geq \tau_R \wedge \text{Regret} \leq \tau_{regret} \wedge \text{trazas} \geq \tau_T \wedge \text{drift} \leq \tau_D]$$

Un gate no sustituye criterio. Lo documenta. Si una política no deja trazas, no mide regret, no tiene política de reserva o no sabe detectar drift, no está lista aunque el promedio parezca atractivo.

Cómo encaja todo



Cuadernos para practicar

Has operado políticas a lo largo del facsímil; estos cuadernos te dejan ver nacer la decisión desde las recompensas. Son *notebooks* que se abren en Google Colab —gratis, en el navegador— o te puedes descargar. Cada uno, explicado paso a paso, con salidas reales, y anclado al capítulo del que sale.

MDP, Bellman y un mundo de casillas

Qué practicas: resolver un mundo de casillas con *value iteration* y ver emerger la política óptima. **Dónde encaja:** capítulo 1 (MDP, políticas, retorno y Bellman). **Qué necesitas:** un navegador. Corre en CPU; sin claves.

Sueltas un robot en un mundo de 3×4 con una salida (+1), una trampa (−1) y un muro. No le enseñas el camino: solo le das las recompensas (y un coste de −0,04 por paso) y dejas que la ecuación de Bellman calcule, iterando, lo que vale estar en cada casilla. En **15 iteraciones** los valores convergen (van de +0,51 cerca del inicio a +1,00 en la salida) y la política sale sola: las flechas rodean el muro y se alejan de la trampa, porque su valor negativo se propaga a los vecinos. Nadie programó la ruta; emergió del valor.

Decidir bajo incertidumbre con consecuencias que se encadenan es lo que hacen un robot, un recomendador o un agente que planea. *Value iteration* es la forma más limpia de ver cómo «el valor de aquí» depende del «valor de lo que viene».

▸ Abrir en Google Colab **↓ Descargar el cuaderno (.ipynb)**

Explorar o explotar: el dilema de los bandidos

Qué practicas: medir, en *regret*, tres estrategias para aprender cuál de varias opciones es la mejor. **Dónde encaja:** capítulo 3 (exploración, *bandits* y validación de políticas). **Qué necesitas:** un navegador. Corre en CPU; sin claves.

Tienes seis máquinas con probabilidades de premio desconocidas y 2000 tiradas. ¿Sigues con la que parece buena (explotar) o pruebas otras por si lo son más (explorar)? Comparas cuatro formas de decidir por su *regret* —lo que pierdes frente a haber jugado siempre la mejor—. El azar acumula **608** de *regret* (no aprende nada); ε-greedy, UCB y Thompson lo dejan muy atrás (**63, 114 y 45**) porque descubren la buena y se quedan con ella. La diferencia entre ganar y malgastar está en explorar lo justo.

Este dilema vive en cualquier sistema que aprende mientras actúa: tests A/B, recomendaciones, anuncios, dosis. Explotar de más te ancla en lo mediocre; explorar de más malgasta.

► **Abrir en Google Colab** ↓ **Descargar el cuaderno (.ipynb)**

Dónde solía tropezar yo

TROPIEZO	POR QUÉ OCURRE	ANTÍDOTO
Pensar que RL exige entrenar un modelo grande	La literatura impresiona.	Empezar con simulaciones pequeñas y decisiones reproducibles.
Confundir recompensa con métrica de dashboard	Ambas son números.	Escribir qué comportamiento induce la recompensa.
No medir regret	Solo miramos recompensa final.	Comparar contra la mejor acción conocida o una referencia estable.
Usar verificadores incompletos	Pasan rápido y dan sensación de seguridad.	Crear casos negativos, slices y cobertura mínima.
Olvidar la política de reserva	La política cambia con datos.	Mantener versión estable y condición de retorno.
Publicar por promedio	El promedio tapa slices dañados o drift.	Medir por ventana, slice y acción.

Vocabulario aprendido

TÉRMINO	DEFINICIÓN
Laboratorio de refuerzo	Práctica reproducible para simular una política, auditar una recompensa y defender una decisión.
Bandit	Problema donde se elige repetidamente entre acciones y se observa recompensa de la acción elegida.
Regret	Coste acumulado de no haber elegido siempre la mejor acción disponible bajo el escenario de referencia.
Reward card	Documento técnico que explica objetivo, términos, pesos, casos, límites y gates de una recompensa.
Gate de política	Regla ejecutable que decide si una política puede avanzar, bloquearse o quedarse en revisión.
Modo sombra	Ejecución sin afectar la decisión real para comparar comportamiento y trazas antes del piloto.
Política de reserva	Versión estable a la que se vuelve si la candidata degrada calidad, coste, trazas o seguridad.
Entrega reproducible	Carpeta con comandos, datos, salidas y decisión que otra persona puede ejecutar y revisar.

Antes de pasar página

- ¿Qué significa que RL sea ingeniería de consecuencias?
- ¿Por qué un bandit puede ser más adecuado que un MDP completo?
- ¿Qué diferencia hay entre recompensa acumulada, retorno y regret?
- ¿Qué debe contener una reward card?
- ¿Por qué RLVR depende tanto de la calidad del verificador?
- ¿Qué evidencia pedirías antes de publicar una política candidata?
- ¿Qué conexión hay entre este facsímil y producto/UX?

Recursos para seguir: leer, construir y experimentar

El aprendizaje por refuerzo cambia la pregunta: ya no es «¿la salida es correcta?», sino «¿qué pasa después de actuar?». Has visto MDP, políticas, bandits, RLHF y reward engineering. Aquí tienes por dónde seguir.

Para experimentar sin código. En la caja «Pruébalo en 5 minutos» del primer capítulo lo viste: la demo GridWorld de Andrej Karpathy (`cs.stanford.edu/people/karpathy/reinforcejs/gridworld_dp.html`) deja ver la ecuación de Bellman trabajando, cómo los valores se extienden desde las recompensas y cómo emerge la política óptima a base de iterar. Es una de las formas más claras de entender qué significa «valor» en refuerzo.

Para construir. El entorno estándar para experimentar con RL es Gymnasium (de la Farama Foundation, sucesor de OpenAI Gym), que ofrece entornos listos para entrenar agentes; y librerías como Stable-Baselines3 o CleanRL para los algoritmos. Para el post-training de modelos de lenguaje (RLHF, DPO), las librerías de la comunidad de Hugging Face son el punto de partida.

Para leer. La referencia canónica e insustituible es *Reinforcement Learning: An Introduction*, de Sutton y Barto, disponible gratis en la web de los autores. Cada capítulo enlaza sus fuentes en «Para saber más». Conectar la teoría con la operación de una política es lo que separa la ecuación de Bellman de un sistema que decide con consecuencias.

Para saber más

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd ed.). MIT Press. https://incompleteideas.net/book/the-book-2nd.html

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). *Proximal Policy Optimization Algorithms*. arXiv:1707.06347. https://arxiv.org/abs/1707.06347

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., & Lowe, R. (2022). *Training Language Models to Follow Instructions with Human Feedback*. arXiv:2203.02155. https://arxiv.org/abs/2203.02155

Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., & Finn, C. (2023). *Direct Preference Optimization: Your Language Model is Secretly a Reward Model*. arXiv:2305.18290. https://arxiv.org/abs/2305.18290

Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., Chen, A., Goldie, A., Mirhoseini, A., McKinnon, C., Chen, C., Olsson, C., Olah, C., Hernandez, D., Drain, D., Ganguli, D., Li, D., Tran-Johnson, E., Perez, E., … Kaplan, J. (2022). *Constitutional AI: Harmlessness from AI Feedback*. arXiv:2212.08073. https://arxiv.org/abs/2212.08073

DeepSeek-AI, Guo, D., Yang, D., Zhang, H., Song, J., Wang, P., Zhu, Q., Xu, R., Zhang, R., Ma, S., Bi, X., Zhang, X., Yu, X., Wu, Y., Wu, Z. F., Gou, Z., Shao, Z., Li, Z., Gao, Z., Liu, A., … Liang, W. (2025). *DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning*. arXiv:2501.12948. https://arxiv.org/abs/2501.12948

Breck, E., Cai, S., Nielsen, E., Salib, M., & Sculley, D. (2017). The ML Test Score: A rubric for ML production readiness and technical debt reduction. *2017 IEEE International Conference on Big Data*, 1123-1132. https://doi.org/10.1109/BigData.2017.8258038

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys, 46*(4), 1-37. https://doi.org/10.1145/2523813

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J.-F., & Dennison, D. (2015). Hidden Technical Debt in Machine Learning Systems. *Advances in Neural Information Processing Systems, 28*. https://papers.nips.cc/paper_files/paper/2015/hash/86df7dcfd896fcaf2674f757a2463eba-Abstract.html

1

2

3

En resumen

IDEA	QUÉ DEBES RECORDAR
MDP	Si no puedes definir estado, acción, transición, recompensa y descuento, quizá no tienes un problema RL formal.
Política	Es la regla que realmente se ejecuta, no la que aparece en una presentación.
Retorno	El futuro cuenta; ignorarlo crea decisiones miopes.
Bandits	Son útiles cuando eliges repetidamente entre opciones y puedes medir recompensa.
Regret	Aprender tiene coste; medirlo evita decisiones ingenuas.
Reward design	La recompensa es una especificación de producto e ingeniería.
Post-training	El método depende de la señal: ejemplos, preferencias, recompensa o verificador.
Serving	Una política no se publica: se expone gradualmente, se observa y se puede detener.
Entrega	La entrega buena no solo obtiene resultados; deja evidencia para que otra persona los revise.

Notas

1. Sutton y Barto (2018) es la referencia de base para separar MDP, política, valor, retorno y exploración sin mezclarlo con marketing de agentes. ↩

2. Ouyang y otros (2022), Rafailov y otros (2023) y Bai y otros (2022) ayudan a conectar el cierre con post-training moderno: feedback humano, preferencias directas y feedback asistido por IA. ↩

3. Breck y otros (2017), Gama y otros (2014) y Sculley y otros (2015) sostienen la parte de operación: readiness, drift y deuda técnica en sistemas ML. ↩