Actumlogos
La palabra reflexionada puesta en acción

# Deep Reinforcement Learning

*Curso • 36 horas • Zoom*

Aprende a programar y entrenar agentes que resuelven tareas por sí mismos: sistemas que descubren la mejor forma de actuar a base de prueba, error y recompensa, sin que nadie les indique el camino. Partirás de los fundamentos —bandidos, procesos de decisión de Markov y métodos de diferencia temporal— y avanzarás hasta los algoritmos que hoy usan OpenAI y Google DeepMind: DQN, PPO, DDPG y SAC. Todo se implementa desde código y se pone a prueba entrenando a tus propios agentes para que jueguen videojuegos, controlen robots, aterricen una nave o estacionen un auto en simulación. Al terminar no te llevas solo la teoría, sino un repertorio de algoritmos de RL que sabrás construir, entrenar y depurar por tu cuenta.

Casos de Estudio:

- Resolver ambientes GridWorld (Acantilado, Lago Congelado) para aprender los fundamentos del aprendizaje por refuerzo.
- Controlar inventario de una tienda
- Maximizar ventas en un sitio web personalizando la publicada a cada cliente
- Resolver sistemas de control (Sistema carro-péndulo, péndulo)
- Jugar Super Mario Bros (de Super Nintendo) a partir de la imagen del videojuego.
- Jugar Atari.
- Controlar una nave espacial para que aterrice de forma perfecta.
- Estacionar un auto en un simulador.
- Controlar un manipulador para alcanzar un punto de interés.
- Controlar un manipulador para mover un objeto.
- Controlar la caminata de una hormiga humanoide.

Herramientas a aprender a usar: PyTorch, Gymnasium, Numpy, Matplotlib.

## Contenido

1. **Introducción al aprendizaje por refuerzo**
   Qué es el aprendizaje por refuerzo y en qué se distingue del supervisado: aquí no hay etiquetas, solo un agente que actúa y una señal de recompensa que lo guía. Dejarás el entorno listo y te familiarizarás con Gymnasium —la interfaz estándar para entrenar agentes— y con el vocabulario base: estado, acción, política y recompensa.
   a. Motivación del Aprendizaje por Refuerzo
   b. Instalación
   c. Introducción a Gym & Gymnasium
   d. Fundamentos de RL

2. **Algoritmos de Bandidos**
   El problema más simple del RL —sin estados, solo acciones y recompensas— para aislar el dilema central de todo agente: explorar opciones nuevas o explotar lo que ya funciona. Aquí se construye la intuición de exploración vs. explotación que reaparecerá en todos los algoritmos posteriores.
   a. Bandidos K armados
   b. Exploración y Explotación

3. **Procesos de Decisión de Markov, Monte Carlo, Programación Dinámica**
   Este es marco matemático que sostiene el curso: los procesos de decisión de Markov (MDP) formalizan qué significa tomar decisiones secuenciales, y las funciones de valor y Q cuantifican qué tan buena es una situación o una acción. Verás cómo resolver el problema de forma exacta cuando conoces el modelo del ambiente (programación dinámica: iteración de política y de valor) y por muestreo cuando no lo conoces (Monte Carlo).
   a. Terminología de Probabilidad Básica
   b. Procesos de Markov
   c. Procesos de Recompensa de Markov
   d. Procesos de decisión de Markov
   e. Procesos de decisión de Markov parcialmente observables
   f. Monte Carlo
   g. Función de valor de acción o función Q
   h. Estrategia Óptima
   i. Programación Dinámica
   j. Iteración de la Política
   k. Iteración de la función de valor

4. **Métodos de Diferencia Temporal**
   La idea que unifica Monte Carlo y programación dinámica: aprender en cada paso, sin esperar a que termine el episodio, corrigiendo las estimaciones sobre la marcha. Implementarás los caballos de batalla del RL tabular —Q-learning y SARSA— y verás cómo los rasgos de elegibilidad y TD(λ) equilibran aprender rápido contra aprender con precisión.
   a. SARSA
   b. n-step SARSA
   c. Q-learning
   d. Dyna Q
   e. Rasgos de Elegibilidad
   f. TD (λ)

5. **Deep Q-Networks**
   El salto de las tablas a las redes neuronales: cuando el espacio de estados es enorme —por ejemplo, los píxeles de un videojuego— ya no puedes guardar un valor por estado y necesitas aproximarlo con una red. DQN combina Q-learning con deep learning (el algoritmo que aprendió a jugar Atari desde la imagen); aquí verás los trucos que lo hacen estable —replay buffer y red objetivo— y sus mejoras Double y Dueling.

a. DQN
b. Double DQN (DDQN)
c. Dueling DQN
d. Mejorando los métodos de DQN

6. **Métodos Policy Gradient**
   Un cambio de filosofía: en lugar de aprender valores y deducir de ahí qué hacer, optimizas directamente la política subiendo por el gradiente de la recompensa esperada. Es la puerta a los espacios de acción continuos y estocásticos; empezarás con REINFORCE y verás cómo una línea base reduce su varianza para que el entrenamiento sea viable.
   a. REINFORCE
   b. REINFORCE con baseline

7. **Métodos Actor Critic**
   Lo mejor de los dos mundos: un actor que decide (policy gradient) guiado por un crítico que evalúa (función de valor). Aquí llegas al estado del arte —A2C, TRPO, PPO, DDPG, SAC— los algoritmos que hoy controlan robots reales, resuelven control continuo y son la base del RLHF con el que se afinan los modelos de lenguaje; también verás HER, que aprende incluso de los intentos fallidos.
   a. Algoritmos Actor Crítico
   b. Advantage Actor Critic (A2C)
   c. Aprendizaje por Refuerzo Avanzado
   d. Trust Region Policy Optimisation (TRPO)
   e. Proximal Policy Optimisation (PPO)
   f. Deep Deterministic Policy Gradient (DDPG)
   g. Hindsight Experience Replay (HER)
   h. Soft Actor Critic (SAC)



## Al terminar tendrás:

- Un repertorio de agentes de RL implementados desde cero en PyTorch, desde bandidos y Q-learning tabular hasta DQN, PPO, DDPG y SAC.
- El criterio para elegir el algoritmo correcto según el problema: espacio de acciones discreto o continuo, on-policy vs. off-policy, eficiencia de muestras.

- Experiencia práctica entrenando agentes en Gymnasium: de GridWorld a Atari, control continuo y robótica simulada.
- La habilidad de diagnosticar y estabilizar entrenamientos de RL — que es justo donde la mayoría se atora.
- Constancia de participación verificable en línea.

## ¿Para quién es este taller?

- **Ingenieros de ML y científicos de datos** que ya trabajan con deep learning supervisado y quieren dar el salto al aprendizaje por refuerzo: la familia de métodos detrás de los agentes que juegan, controlan y deciden.
- **Personas que trabajan en robótica, control o simulación** y necesitan que un sistema aprenda su política a partir de la interacción con el entorno, en lugar de programar cada regla a mano.
- **Profesionales técnicos** que han oído hablar de DQN, PPO o AlphaGo y quieren pasar de la intuición a implementar, entrenar y depurar estos algoritmos por su cuenta.

### Este taller NO es para ti si:

- Buscas una introducción general a la IA o quieres usar herramientas sin programar. Aquí se escribe y entrena código de RL.
- No tienes bases de Python ni de redes neuronales.
- Esperas montar entrenamientos de escala industrial (semanas de cómputo distribuido en cientos de GPUs). Aquí construyes el fundamento completo y los algoritmos de principio a fin, no la infraestructura de despliegue masivo.

**Clases**

9 sesiones de 4 horas (36 hrs en total)
Sábados del 29 de agosto al 24 de octubre
10:00–14:00, hora de la Ciudad de México
Las clases son impartidas vía Zoom y tendrás acceso a las grabaciones por 1 año después de terminado el curso.

**Instructor**

Dr. Adolfo Perrusquía — LinkedIn

**Testimonios**

Hemos capacitado a más de 2,000 personas en IA — visita sitio anterior

**Medios de pago**

Tarjetas de crédito/débito, transferencias, Link, PayPal, Apple Pay, Google Pay.
Ver checkout

## Constancia de participación con folio único, verificable en línea

Al terminar recibirás una constancia de participación firmada por el instructor y el director de Actumlogos — agente capacitador registrado ante la Secretaría del Trabajo y Previsión Social. Cada constancia incluye un folio único con el que cualquier persona puede verificar su autenticidad en línea, y podrás compartirla directamente en LinkedIn.

*Registro STPS: ZAGE-810930-FW2-0005*

Este es un ejemplo para ilustrar que esperar: