Práctica 2. Aprendizaje por refuerzo profundo con aproximación de la función de valor de la acción Q

(Duración: 4 sesiones (4h))

1. Objetivos de la práctica

Los objetivos de esta práctica son los siguientes:

1. Comprender de forma teórica y práctica el funcionamiento de los métodos de Aprendizaje por Refuerzo Profundo basados en la aproximación de la función de valor de acción, con especial énfasis en DQN y sus extensiones.
2. Trabajar con entornos de control continuo realistas (MuJoCo) utilizando observaciones visuales (píxeles) en lugar de estados vectoriales.
3. Analizar el impacto de distintas mejoras algorítmicas sobre DQN (Rainbow) en términos de eficiencia muestral, estabilidad del aprendizaje y capacidad de exploración.
4. Comparar empíricamente DQN clásico frente a Rainbow DQN en tareas de locomoción y evitación de obstáculos.
5. Desarrollar criterio experimental y capacidad de análisis crítico sobre algoritmos de DRL en entornos complejos.

2. Introducción de la práctica

En esta práctica se aborda uno de los problemas clásicos del Aprendizaje por Refuerzo Profundo el control motor y la locomoción a partir de observaciones visuales.

A diferencia de ejemplos más sencillos, donde el estado del entorno se representa mediante variables físicas (posiciones, velocidades, ángulos), en esta práctica el agente solo observa imágenes RGB del entorno, lo que obliga al algoritmo a aprender simultáneamente una representación visual del estado, y una política de control basada en dicha representación.

El entorno virtual elegido es MuJoCo, un motor de simulación física ampliamente utilizado en investigación para estudiar locomoción, manipulación y control continuo. Sobre MuJoCo se trabajará a través de DeepMind Control Suite (dm_control) y la interfaz Gymnasium, que permite unificar el acceso a los entornos y reutilizar implementaciones estándar de algoritmos de DRL.

2.1. MuJoCo (Multi-Joint dynamics with Contact)

MuJoCo (https://github.com/google-deepmind/mujoco) es un motor de simulación física diseñado para modelar sistemas articulados complejos, simular contactos, colisiones y fricción, y estudiar el control y la locomoción de agentes como walkers, cuadrúpedos y humanoides.

MuJoCo se caracteriza por ofrecer una simulación precisa y estable, por su uso extensivo en benchmarks de DRL y por la definición de entornos con dinámica realista y no lineal.

2.2. DeepMind Control Suite (dm_control)

dm_control (https://github.com/google-deepmind/dm_control) es una librería desarrollada por DeepMind que proporciona un conjunto estandarizado de tareas de control en MuJoCo, incluye entornos de locomoción (walker, cheetah, quadruped, humanoid), permite trabajar tanto con estados vectoriales como con observaciones visuales e incorpora escenarios avanzados como corredores con obstáculos.

En esta práctica se utilizarán entornos de dm_control, incluyendo variantes en las que el agente debe avanzar y, posteriormente, evitar obstáculos mientras mantiene la locomoción.

2.3. Gymnasium

Gymnasium (https://gymnasium.farama.org/) es la evolución oficial de OpenAI Gym, que actúa como un framework estandarizado para definir entornos de RL lo cual permite integrar fácilmente algoritmos existentes y compararlos entre ellos.

En la práctica, se utilizarán los entornos de dm_control y los wrappers de Gymnasium para hacerlos compatibles y trabajar con observaciones en píxeles (RGB), redimensionadas y con frame stacking.

2.4. Deep Q-Network

Deep Q-Network (DQN) es un algoritmo de Aprendizaje por Refuerzo Profundo basado en valores cuyo objetivo es aproximar la función de valor de acción óptima:

$$Q^*(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, a_0 = a\right] \tag{1}$$

En DQN, la política es implícita y se define como la acción que maximiza el valor estimado:

$$a_t = \arg\max_a Q(s_t, a; \theta) \tag{2}$$

El aprendizaje se realiza mediante actualizaciones *off-policy* con *bootstrapping* temporal (TD).

En esta práctica, DQN se utilizará como baseline, sobre el cual posteriormente se introducirán mejoras mediante Rainbow DQN.

En esta práctica, la DQN seguirá una arquitectura estándar **CNN + MLP**, similar a la propuesta original de Mnih et al.:

- **Entrada**: imagen RGB (opcionalmente con *frame stacking*).
- **Encoder convolucional**: varias capas convolucionales 2D con activación ReLU.
- **Capas fully-connected**.
- **Salida**: valores Q para cada acción discreta.

Esta arquitectura permite aprender simultáneamente una representación visual del estado y la función de valor de acción asociada.

2.5. Rainbow DQN

Rainbow integra las siguientes extensiones sobre DQN:

1. **Double DQN**: desacopla la selección y evaluación de la acción.
2. **Prioritized Experience Replay**: prioriza transiciones con mayor error TD.
3. **Dueling Architecture**: descompone la función Q como $Q(s,a) = V(s) + A(s,a)$.
4. **n-step returns**: utiliza retornos multi-step para acelerar el aprendizaje.
5. **Distributional RL**: instancia varios entornos y comparte pesos.
6. **Noisy Networks**: sustituyen la exploración ε-greedy por ruido paramétrico aprendido.

Rainbow utiliza una arquitectura **Dueling Distributional DQN con capas Noisy**:

- Encoder convolucional compartido.
- Stream de valor $V(s)$ que produce una distribución del retorno.
- Stream de ventaja $A(s,a)$ que produce una distribución por acción.
- Agregación dueling con normalización de la ventaja.
- Capas *Noisy Linear* en lugar de capas fully-connected estándar.

La salida del modelo es una distribución del retorno para cada acción, y el valor Q se obtiene como la esperanza matemática de dicha distribución.

3. Desarrollo de la práctica

3.1. Parte 0: Elección del entorno

Los estudiantes podrán elegir el entorno en el que trabajan, con distintos niveles de dificultad. Según el nivel elegido, se exigirán distintos experimentos.

- *Entornos fáciles (ES)*: **Walker2D, Hopper o HalfCheetah**. Son entornos de baja dimensionalidad de control (pocos actuadores), con una dinámica relativamente estable y en los que la discretización de las acciones es directa.
- *Entorno complejo (EC)*: **Humanoid**. Es un entorno de alta dimensionalidad de control, con una dinámica poco estable y en los que la discretización de las acciones no es trivial.

3.2. Parte 1: Locomoción con la DQN (ES-4ptos // EC-5ptos)

La primera parte de la práctica consiste en entrenar la DQN original (la arquitectura puede ser ligeramente modificada para facilitar o mejorar el entrenamiento) para que el agente aprenda a desplazarse hacia adelante. No hace falta que corra; simplemente que siga un desplazamiento lo más recto posible, lineal y uniforme. Los pasos que se deben seguir son:

1. Desarrollar el código de entrenamiento y evaluación en base al paper original o clonar un repositorio en el que ya esté hecha la implementación. Algunos ejemplos de repositorios que ofrecen implementaciones de la DQN y Rainbow son: https://github.com/vwxyzjn/cleanrl?utm_source=chatgpt.com, y https://github.com/Denys88/rl_games?utm_source=chatgpt.com.

El desarrollo original del código se valorará positivamente en la nota final de la práctica, pero como un extra, no es necesario hacerlo para obtener la máxima nota.

2. Preparar el entorno de Python y PyTorch con las dependencias necesarias.
3. Configurar el entorno elegido, esto es, definir el set de acciones discreto, la observación RGB y la recompensa.
4. Entrenar y evaluar la DQN.

La discretización del set de acciones puede realizarse como un discretización de cada dimensión en pocos niveles (-1, 0, +1), una definición de un conjunto finito de acciones prototipo, o la selección de un subconjunto de actuadores relevantes. La estrategia de discretización debe estar bien justificada y mantenerse constante en todos los experimentos comparativos.

El diseño de la recompensa podrá basarse en la literatura que haya disponible, pero debe quedar justificado y siempre teniendo en cuenta las premisas de que i) al agente hay que decirle qué hacer, no como, y ii) cuanto más sencilla, mejor.

La observación consistirá únicamente en la imagen RGB del entorno. Se pueden concatenar observaciones a la entrada de la arquitectura o usar alguna capa con redes LSTM para proporcionar memoria al agente. Excepcionalmente, y sólo de forma justificada si no se consigue entrenar de ningún modo un agente, se podría concatenar a las variables del espacio latente información proprioceptiva del elemento a controlar.

Como resultado de esta parte, se deberían tener: i) un *checkpoint* con los pesos del agente ya entrenado, y ii) las curvas de entrenamiento, que en este caso se corresponden a la gráfica de la evolución de las pérdidas y/o la gráfica de la evolución de las recompensas. Asimismo, se deben guardar los hiperparámetros empleados para poder realizar luego la discusión en el informe.

3.3. Parte 2: Locomoción con Rainbow DQN (ES-2.5ptos // EC-3ptos)

La segunda parte de la práctica consiste en realizar ahora el entrenamiento con Rainbow DQN y comparar los resultados con los obtenidos en el entrenamiento con la DQN estándar de la sección anterior. La comparativa debe realizarse tanto a nivel del proceso de entrenamiento como en la evaluación.

En principio, las consideraciones de diseño del entorno realizadas en la parte 1 deberían ser las mismas que en esta parte. No es necesario modificar ni el set de acciones ni la recompensa.

Como resultado de esta parte, se deberían tener: i) un *checkpoint* con los pesos del agente ya entrenado, ii) las curvas de entrenamiento, que en este caso se corresponden a la gráfica de la evolución de las pérdidas y/o la gráfica de la evolución de las recompensas, y iii) las gráficas comparativas entre el agente DQN y el agente Rainbow. Asimismo, se deben guardar los hiperparámetros empleados para poder realizar luego la discusión en el informe.

3.4. Parte 3: Estudio de ablación en Rainbow DQN (ES-2ptos)

SÓLO PARA LOS ENTORNOS SENCILLOS

A continuación, se llevará a cabo un estudio de ablación de las mejoras propuestas por Rainbow DQN sobre el algoritmo original para evaluar su impacto en la velocidad de aprendizaje, la estabilidad,

la exploración y el rendimiento final del agente. De las mejoras que introduce Rainbow DQN **SÓLO** se deberá analizar el efecto de **DOS** de ellas.

Como resultado de esta parte, se deberían tener: i) dos *checkpoints* con los pesos de los agentes ya entrenados, ii) las curvas de entrenamiento, que en este caso se corresponden a la gráfica de la evolución de las pérdidas y/o la gráfica de la evolución de las recompensas, y iii) las gráficas comparativas entre los agentes modificados y el agente Rainbow. Asimismo, se deben guardar los hiperparámetros empleados para poder realizar luego la discusión en el informe.

3.5. Parte 4: Locomoción con obstáculos (ES-1,5ptos // EC-2ptos)

Como último apartado de la práctica, se propone entrenar al agente usando la DQN o Rainbow DQN en el entorno de dm_control que tiene obstáculos, por lo tanto, el agente ya no sólo deberá avanzar hacia adelante, sino que además tendrá que aprender a esquivar los obstáculos.

Para este apartado, en la observación del estado se puede incluir información propioceptiva además de la imagen RGB.

Si un grupo consigue un rendimiento muy bueno en este entorno y el diseño del agente y la arquitectura son los adecuados (bajo el criterio de la profesora), se les otorgarán **+0.15 puntos extra en la nota final de la asignatura**. Los entrenamientos deben ser reproducibles y la propuesta no puede basarse completamente en trabajo ya realizado por otros estudiantes/investigadores.

4. Evaluación de la práctica

La práctica se evaluará a través de un informe en el que se presentarán y discutirán convenientemente el procedimiento, las decisiones de diseño tomadas y los resultados obtenidos en cada una de las partes.

El informe debe presentarse en formato artículo científico a doble columna de no más de 8 páginas, o a una sola columna no más de 12 páginas. El informe debe tener las siguientes secciones:

1. **Introducción**: breve introducción al método y algoritmo.
2. **Parte 1**: metodología, resultados, discusión (resultados y discusión pueden ir en el mismo subapartado) y conclusiones de la primera parte de la práctica.
3. **Parte 2**: metodología, resultados, discusión (resultados y discusión pueden ir en el mismo subapartado) y conclusiones de la segunda parte de la práctica.
4. **Parte 3**: metodología, resultados, discusión (resultados y discusión pueden ir en el mismo subapartado) y conclusiones de la tercera parte de la práctica.
5. **Parte 4**: metodología, resultados, discusión (resultados y discusión pueden ir en el mismo subapartado) y conclusiones de la cuarta parte de la práctica.
6. **Conclusiones generales** de la práctica.

En el informe deben justificarse todas las decisiones de diseño realizadas, comentarse todas las figuras que se incluyan, y realizar una discusión que no sea la mera descripción de lo que se ve en las gráficas. Todas las figuras y tablas deben llevar su correspondiente pie de figura/tabla y referenciarse correctamente. Las gráficas deben presentarse de forma que sean legibles y con los ejes bien rotulados. Incumplir alguno de estos puntos conlleva penalizaciones en la nota de la práctica.