Aprendizaje por Refuerzo en Lunar Lander

Inteligencia Artificial – Ingeniería del Software
Curso 2024/2025

Propuesta de trabajo del profesor Miguel Bermudo Bayo

1. Introducción y objetivos

Este trabajo se centra en la resolución del entorno de aterrizaje del módulo lunar "LunarLander"dado por el paquete gymnasium de la fundación Farama mediante algoritmos de aprendizaje por refuerzo, donde la política de decisiones será una red neuronal Feed Forward simple.

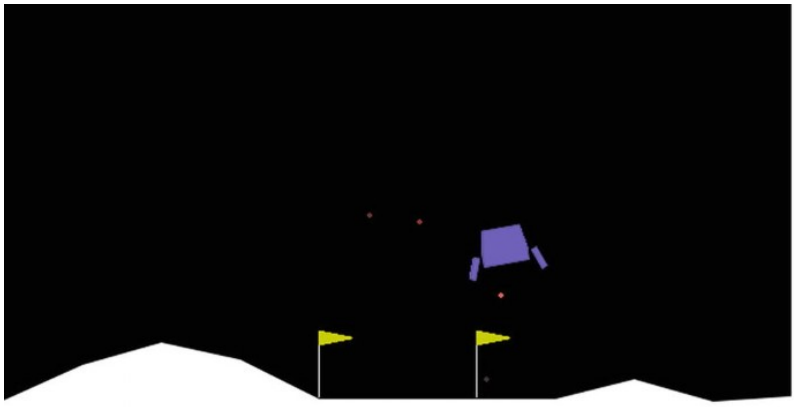
El **objetivo principal** es diseñar un algoritmo de aprendizaje que itere sobre los estados del entorno LunarLander, consiguiendo que el modelo resuelva el entorno en al menos un 30 % de los casos para considerarse como una solución correcta (a mejor porcentaje de resolución, mejor se considerará la puntuación conseguida en ese apartado).

Para ello será necesario alcanzar los siguientes **objetivos específicos:**

1. El correcto uso de las librerías sugeridas (Keras, Pytorch, Gymnasium).
2. La correcta resolución del entorno usando el algoritmo requerido
3. Documentar el trabajo realizado usando un formato de artículo científico.
4. Realizar una presentación (PDF, PowerPoint o similar).

2. Lunar Lander

El entorno de entrenamiento LunarLander consta de un módulo lunar que debe aterrizar en una superficie irregular y entre dos banderas a una velocidad lineal y angular adecuadas para no destruirse.



El modulo Lunar activando su cohete izquierdo y inferior.

El entorno de módulo lunar genera aleatoriamente un terreno y coloca las banderas de aterrizaje en la misma posición, acto seguido, lanza el modulo con una velocidad horizontal y angular aleatorias dentro de sus rangos posibles.

El estado del problema consiste por tanto en 8 elementos que lo describen, el vector que lo compone se desglosa en la siguiente tabla.

Posición	Descripción	Mínimo	Máximo
1	Posición en X	-2.5	2.5
2	Posición en Y	-2.5	2.5
3	Velocidad lineal en X	-10.0	10.0
4	Velocidad lineal en Y	-10.0	10.0
5	Ángulo	-2π	2π
6	Velocidad angular	-10.0	10.0
7	Contacto de la pierna izquierda (bool)	0.0	1.0
8	Contacto de la pierna derecha (bool)	0.0	1.0

Cuadro 1: Rangos de observación del módulo lunar

El entorno LunarLander tambien contiene 4 acciones discretas posibles que controlan el modulo lunar, se desglosan en la siguiente tabla.

Acción	Descripción
0	No hacer nada
1	Encender el motor de orientación izquierdo
2	Encender el motor principal
3	Encender el motor de orientación derecho

Cuadro 2: Acciones disponibles para el módulo lunar

Finalmente, el entorno también devuelve una recompensas para cada estado del problema, la recompensa es la suma de los siguientes elementos

- La recompensa aumenta o disminuye según qué tan cerca o lejos esté el módulo de la plataforma de aterrizaje.
- La recompensa aumenta si el módulo se mueve más lento, y disminuye si se mueve más rápido.
- La recompensa disminuye mientras más inclinado esté el módulo (es decir, si el ángulo se aleja de la horizontal).
- Se otorgan 10 puntos adicionales por cada pierna que esté en contacto con el suelo.
- Se resta 0.03 puntos por cada fotograma en que se utilice un motor lateral.
- Se resta 0.3 puntos por cada fotograma en que se utilice el motor principal.

En general el entorno se usara de la siguiente manera, para cada paso:

- Solicitar la información del estado (observation space) al entorno
- Aplicar la política actual, la cual para nosotros siempre sera una red neuronal
- Calcular la acción a realizar y dar un paso en el entorno
- Almacenar los resultados del episodio para la fase de actualización.

3. Descripción del trabajo

A continuación se describe con más detalle cómo debe llevarse a cabo el trabajo. Este tendrá una parte común para todas las convocatorias y una parte específica para la convocatoria de junio y julio, en la convocatoria de octubre el alumno es libre de escoger una de las partes específicas anteriores.

3.1. Metodología

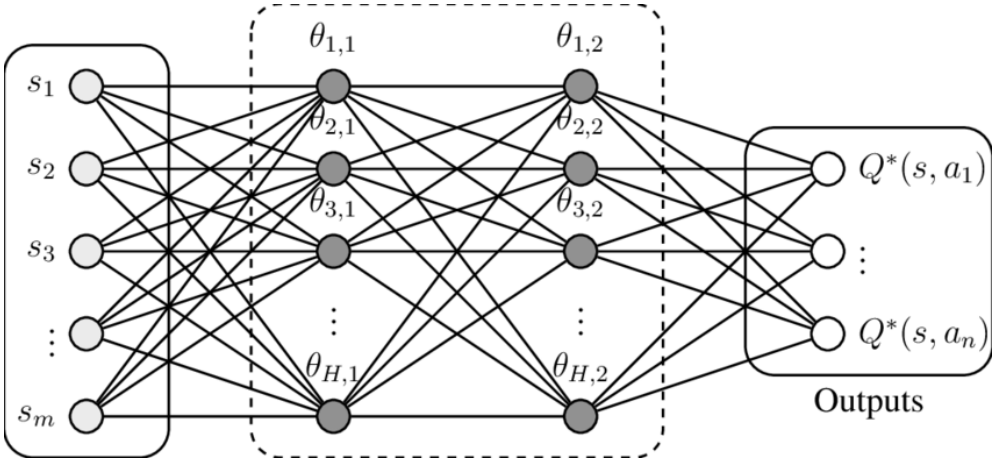
3.1.1. Parte común

Independientemente del algoritmo elegido el alumno debe construirse una política basada en una red neuronal Feed Forward compatible con el entorno de entrenamiento presentado. Los módulos creados también deben de ser capaces de guardar y cargar modelos entrenados para la posterior prueba de los mismos y deben recibir valores de variable para modificar los hiperparámetros necesarios para cada algoritmo.

3.1.2. Convocatoria de junio

Los alumnos que se presenten a esta convocatoria deberán completar el Modulo asociado a Deep Q-Network (DQN) suministrado. En este problema las redes neuronales profundas reemplazan al modelo Q-learning clásico, donde para cada combinación de estado-acción posibles se tiene un valor Q (Q-tabla), la cual se usa para tomar decidir la mejor acción posible.

El modelo de DQN reemplaza la Q-tabla por una red neuronal profunda la cual sirve una doble función, habilitar a los agentes de aprendizaje por refuerzo aprender en un entorno continuo así como entrenar estados similares simultáneamente.



Fases de una iteración de MCTS

El modelo DQN tiene 3 componentes principales, la **red neuronal que debe estimar el Q-valor** para el conjunto de estado-acción (normalmente llamada Q-network), un **buffer de experiencias** donde se almacenaran las recompensas, acciones, estados previos y actuales y por ultimo una red neuronal estática **(red objetivo)** que copiara a la Q-network, es usada para mantener la estabilidad en las fases de actualización.

A diferencia de otros modelos vistos previamente, la función de pérdida en DQN es dependiente de la diferencia entre los Q-valores obtenidos por la Q-network y su red objetivo (target network), se observa el progreso de la red con estos deltas pequeños que la guían hacia un pico de gradiente.

Dada la funcion Q-valor $Q(s, a; \theta)$, los parámetros (pesos) de la red principal que la implementa, se representan por θ. los parametros de la *red objetivo* (target network), se representan por θ^-.
Las experiencias pasadas (s, a, r, s') se almacenan en un *replay buffer*, y se extraen por lotes (minibatches) para actualizar la red.

A diferencia de otros modelos vistos previamente, **la función de pérdida** en DQN es dependiente de la diferencia entre los Q-valores obtenidos por la Q-network y su red objetivo (target network):

$$L(\theta) = \mathbb{E}\left[\left(r + \gamma \,\text{máx}_{a'}\, Q(s', a'; \theta^-) - Q(s, a; \theta)\right)^2\right]$$

donde:

- $Q(s, a; \theta)$: valor Q estimado por la red principal.
- $Q(s', a'; \theta^-)$: valor Q estimado por la red objetivo.
- r: recompensa inmediata.
- γ: factor de descuento, que controla la importancia de las recompensas futuras.

Los hiperparámetros más relevantes en este contexto son:

- γ: Factor de descuento $(0 < \gamma < 1)$.
- Tamaño del *replay buffer*.
- Frecuencia de actualización de la red objetivo $(\theta^- \leftarrow \theta)$.
- Tasa de aprendizaje (α).
- Tamaño del minibatch usado en cada actualización.

3.1.3. Convocatoria de julio

Los alumnos que se presenten a esta convocatoria deberán completar el modelo asociado al algoritmo **REINFORCE**, un enfoque dentro de los métodos de aprendizaje por refuerzo basados en políticas. A diferencia de los métodos basados en valores como DQN, que utilizan una función $Q(s, a)$ para decidir acciones, REINFORCE aprende directamente una **política estocástica** $\pi(a|s; \theta)$ que devuelve una distribución de probabilidad sobre las acciones posibles, dada una observación del entorno.

Esta política está parametrizada mediante una red neuronal, cuyos parámetros θ se optimizan para maximizar la recompensa total esperada. Para ello, se utiliza el método del **gradiente de política**, que calcula una dirección en el espacio de parámetros que incrementa la probabilidad de las acciones que conducen a buenas recompensas.

El flujo general del algoritmo REINFORCE es el siguiente: el agente interactúa con el entorno durante un episodio completo, seleccionando acciones de forma estocástica según la política actual. Una vez finalizado el episodio, se calcula el retorno total acumulado R para cada paso. A continuación, se aplica **retropropagación** para actualizar los pesos de la red, utilizando el siguiente objetivo:

$$L(\theta) = -\mathbb{E}\left[\log \pi(a|s; \theta) \cdot R\right]$$

Esta expresión indica que se aumentará la probabilidad de aquellas acciones que hayan generado altos retornos, y se disminuirá la de aquellas que hayan sido poco beneficiosas.

Este procedimiento es ilustrado en la Figura 1, donde se observa el ciclo completo: observación del entorno, generación de acción vía la política estocástica, recepción de recompensa y almacenamiento de trayectorias para su posterior uso en la actualización.

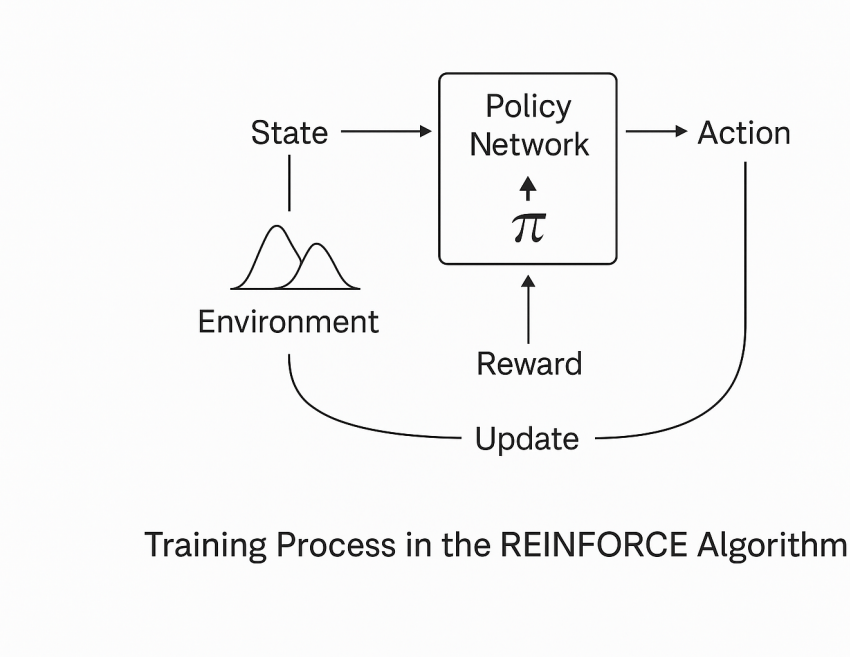


Figura 1: Flujo de entrenamiento en el algoritmo REINFORCE

Una vez recogidas las trayectorias del episodio, se calcula la función de pérdida para cada paso del agente y se aplica descenso de gradiente estocástico (SGD o alguno de sus derivados como Adam) sobre los parámetros de la red. Esto permite ajustar la política directamente, reforzando las acciones que condujeron a buenos resultados.

Los hiperparámetros más relevantes en este modelo son:

- Tasa de aprendizaje (α).
- Factor de descuento (γ).
- Número de episodios por actualización.

4. Lenguaje de programación y bibliotecas recomendadas

Para la realización del trabajo debe utilizarse el lenguaje de programación Python y se podrá hacer uso de cualquier librería que se desee. Para este trabajo en especifico ademas son requeridas las siguientes librerías

- keras/tensorflow La librería mas común para entrenamiento de redes neuronales, su API de alto nivel permite generar modelos de manera sencilla.
- pytorch Otra librería para creación de modelos de redes neuronales a bajo nivel. Permite controlar los pasos internos para el aprendizaje de manera mas cercana que Keras.
- gymNasium contiene el entorno del modulo lunar accesible por una API estandarizada, es el modelo de referencia para aprendizaje por refuerzo.

5. Documentación y entrega

El trabajo deberá documentarse siguiendo un formato de artículo científico, con una **extensión mínima de 6 páginas**. En la página web de la asignatura se pueden encontrar plantillas donde se sugiere una estructura general. Estas plantillas siguen el formato de los IEEE conference proceedings, en cuyo sitio web la guía para autores ofrece información más detallada. El documento entregado deberá estar en formato PDF. Se valorará el uso del sistema LaTeX.

En el caso concreto de este trabajo, la memoria deberá al menos incluir: introducción; descripción de los algoritmos implementados, explicando las dificultades encontradas y las decisiones de diseño adoptadas para abordarlas; experimentación y descripción de los resultados alcanzados (la cual debe contener graficas de preogreso de los modelos entrenados); conclusiones; bibliografía. En ningún caso debe incluirse código en la memoria.

La entrega del trabajo consistirá de un **único fichero comprimido zip** conteniendo la memoria del trabajo, el modelo guardado en formato .keras o .h5 segun la libreria escogida, el modulo creado para la resolución del problema (DQN.py o REINFORCE.py) que se ejecutarán junto a los archivos LunarRL.ipynb y lunar.py **LOS CUALES DEBEN QUEDAR SIN MODIFICAR**.

6. Presentación y defensa

Como parte de la evaluación del trabajo se deberá realizar una defensa del mismo, para lo que se citará a los alumnos de manera conveniente.

Al inicio de la defensa se deberá realizar una pequeña presentación (PDF, PowerPoint o similar) de diez minutos en la que participarán activamente todos los miembros del grupo que han desarrollado el trabajo. Esta presentación deberá seguir a grandes rasgos la misma estructura que la memoria del trabajo, haciendo especial mención a los resultados obtenidos y al análisis de los mismos.

En los siguientes diez minutos de la defensa el profesor procederá a realizar preguntas sobre el trabajo, que podrán ser tanto de la memoria como del código fuente.

7. Evaluación del trabajo

Para la evaluación del trabajo se tendrán en cuenta los siguientes criterios, considerando una nota total máxima de 4 puntos:

- *Memoria del trabajo* (hasta 1 punto): se valorará la claridad de las explicaciones, el razonamiento de las decisiones, el análisis y presentación de resultados y el correcto uso del lenguaje. La elaboración de la memoria debe ser original, por lo que no se evaluará el trabajo si se detecta cualquier copia del contenido.
- *Búsqueda de la solución* (hasta 2 puntos): Se valorará la implementación de la red neuronal según el algoritmo así como las dificultades especificas presentes para cada uno. También se valorará positivamente la mejora de los algoritmos si procede, pero esto no es necesario.
- *Código fuente* (hasta 1 punto): se valorará la claridad y buen estilo de programación, corrección y eficiencia de la implementación y calidad de los comentarios. El código debe ser original, por lo que no se evaluará el trabajo si se detecta código copiado o descargado de internet.

- *Presentación y defensa*: se valorará la claridad de la presentación y la buena explicación de los contenidos del trabajo así como, especialmente, las respuestas a las preguntas realizadas por el profesor, lo que dará lugar para cada alumno por separado a un factor multiplicativo en el intervalo [0, 1] de la nota total obtenida a partir de los apartados anteriores.

IMPORTANTE: cualquier plagio, compartición de código o uso de material que no sea original y del que no se cite convenientemente la fuente, significará automáticamente la calificación de cero en la asignatura para todos los alumnos involucrados. Por tanto, a estos alumnos no se les conserva, ni para la actual ni para futuras convocatorias, ninguna nota que hubiesen obtenido hasta el momento. Todo ello sin perjuicio de las correspondientes medidas disciplinarias que se pudieran tomar.

Uso de inteligencia artificial generativa

El uso de sistemas de inteligencia artificial generativa está permitido con las siguientes condiciones:

- Debe explicarse para qué se han utilizado esos sistemas, así como describir las entradas (*prompts*) proporcionadas a los mismos.
- En la defensa del trabajo **ambos alumnos** deben demostrar conocimiento y entendimiento de la **totalidad del trabajo**, lo que incluye las respuestas obtenidas de esos sistemas.