M2.883 Aprendizaje por refuerzo

Práctica:
Implementación de un agente para la robótica espacial

Contenidos

1. Presentación

A lo largo de las tres partes de la asignatura hemos entrado en contacto con diferentes clases de algoritmos de aprendizaje por refuerzo que permiten solucionar problemas de control en una gran variedad de entornos.

Esta práctica, que se va a extender a lo largo de un mes aproximadamente, da la posibilidad de enfrentarse al diseño de un agente para solucionar un caso específico de robótica.

Atacaremos el problema a partir de la exploración del entorno y sus observaciones. Luego, pasaremos a la selección del algoritmo más oportuno para solucionar el entorno en cuestión. Finalmente, pasaremos por el entrenamiento y la prueba del agente, hasta llegar al análisis de su rendimiento.

Para ello, se presentará antes el entorno de referencia, y luego, se pasará a la implementación de un agente Deep Q-Network (DQN) que lo solucione. Después de estas dos primeras fases de toma de contacto con el problema, se buscará otro agente que pueda mejorar el rendimiento del agente DQN anteriormente implementado.

2. Competencias

En esta actividad se trabajan las siguientes competencias:

- Capacidad para analizar un problema desde el punto de vista del aprendizaje por refuerzo.
- Capacidad para analizar un problema en el nivel de abstracción adecuado en cada situación y aplicar las habilidades y conocimientos adquiridos para resolverlos.

3. Objetivos

Los objetivos concretos de esta actividad son:

- Conocer y profundizar en el desarrollo de un entorno real que se pueda resolver mediante técnicas de aprendizaje por refuerzo.
- Aprender a aplicar y comparar diferentes métodos de aprendizaje por refuerzo para poder seleccionar el más adecuado a un entorno y problemática concretos.
- Saber implementar los diferentes métodos, basados en soluciones tabulares y soluciones aproximadas, para resolver un problema concreto.
- Extraer conclusiones a partir de los resultados obtenidos.

4. Entorno

Estamos trabajando sobre un problema de robótica espacial y en particular queremos solucionar el problema de aterrizaje propio, por ejemplo, de drones autónomos.

Para ello, se elige **lunar-lander** como entorno simplificado. El entorno se puede encontrar en el siguiente enlace:
https://github.com/openai/gym/blob/master/gym/envs/box2d/lunar_lander.py

Lunar Lander consiste en una nave espacial que debe aterrizar en un lugar determinado del campo de observación. El agente conduce la nave y su objetivo es conseguir aterrizar en la pista de aterrizaje, coordenadas (0,0), y llegar con velocidad 0.

La nave consta de tres motores (izquierda, derecha y el principal que tiene debajo) que le permiten ir corrigiendo su rumbo hasta llegar a destino.

Las acciones que puede realizar la nave (espacio de acciones) son discretas.

Las recompensas obtenidas a lo largo del proceso de aterrizaje dependen de las acciones que se toman y del resultado que se deriva de ellas.

- Desplazarse de arriba a abajo, hasta la zona de aterrizaje, puede resultar en [+100,+140] puntos
- Si se estrella al suelo, pierde -100 puntos
- Si consigue aterrizar en la zona de aterrizaje (velocidad 0), gana +100 puntos
- Si aterriza, pero no en la zona de aterrizaje (fuera de las banderas amarillas) se pierden puntos
- El contacto de una pata con el suelo recibe +10 puntos (si se pierde contacto después de aterrizar, se pierden puntos)
- Cada vez que enciende el motor principal pierde -0.3 puntos
- Cada vez que enciende uno de los motores de izquierda o derecha, pierde -0.03 puntos

La solución óptima es aquella en la que el agente, con un desplazamiento eficiente, consigue aterrizar en la zona de aterrizaje (0,0), tocando con las dos patas en el suelo y con velocidad nula. Se considera que el agente ha aprendido a realizar la tarea (i.e. el “juego” termina) cuando obtiene una media de al menos 200 puntos durante 100 episodios consecutivos.

Ejercicio 1.1 (0.5 puntos)

Se pide explorar el entorno y representar una ejecución aleatoria.

Ejercicio 1.2 (0.5 puntos)

Explicar los posibles espacios de observaciones y de acciones (informe escrito).

Nota**:** el entorno Lunar Lander requiere de la librería `box2d-py`.
Si se usa Google Colab[1], iniciar siempre notebook con:

```
!pip install box2d-py
```

[1] https://colab.research.google.com/

5. Agente de referencia

En la tercera parte de la asignatura hemos introducido el agente DQN con *replay buffer* y *target network*, que resulta ser un buen candidato para la solución del problema de robótica que estamos analizando, visto que permite controlar entornos con un número elevado de estados y acciones de forma eficiente.

Se pide resolver los 3 ejercicios siguientes.

Ejercicio 2.1 (1.5 puntos)

Implementar un agente DQN para el entorno **lunar-lander.**

Ejercicio 2.2 (1 punto)

Entrenar el agente DQN y buscar los valores de los hiperparámetros que obtengan un alto rendimiento del agente. Para ello, es necesario listar los hiperparámetros bajo estudio y presentar las gráficas de las métricas que describen el aprendizaje.

Ejercicio 2.3 (0.5 puntos)

Probar el agente entrenado en el entorno de prueba. Visualizar su comportamiento (a través de gráficas de las métricas más oportunas).

6. Propuesta de mejora

En esta parte se pide implementar otro agente, entre aquellos que hemos visto a lo largo de la asignatura, que pueda solucionar el problema de robótica espacial de forma más eficiente con respecto al agente DQN.

En particular, se pide solucionar los 3 puntos siguientes.

Ejercicio 3.1 (2 puntos)

Implementar el agente identificado en el entorno **lunar-lander.**

Justificar las razones que han llevado a probar este tipo de agente. Detallar qué tipos de problemas se espera se puedan solucionar con respecto a la implementación DQN anterior.

Ejercicio 3.2 (2 puntos)

Entrenar el agente identificado y buscar los valores de los hiperpárametros que obtengan el rendimiento “óptimo” del agente.

Ejercicio 3.3 (2 puntos)

Analizar el comportamiento del agente identificado entrenado en el entorno de prueba y compararlo con el agente implementado en el punto 2 (a través de gráficas de las métricas más oportunas).

7. Entrega

El entregable será un fichero comprimido en formato ZIP con los siguientes dos documentos:

1. **Informe en formato PDF** de entre 10 y 15 páginas de longitud, aproximadamente;
2. **Código** utilizado, ya sea en ficheros Jupyter notebook (.ipynb) o Python (.py)

Para el **informe** se puede usar la siguiente guía:

- Tamaño de fuente 11 o 12.
- Fuente: Arial o similar.
- Interlineado sencillo.
- Tres apartados definidos según el guión.
    - Especificar el ejercicio correspondiente como subapartado.
    - Por ejemplo, Apartado 2 Agente de referencia, apartado 2.1 Implementación agente de referencia, apartado 2.2 Entrenamiento agente de referencia, apartado 2.3 Prueba agente de referencia.
- Las capturas de pantalla (por ejemplo, las gráficas de rendimiento) o los fragmentos de código (si se consideran relevantes) deben estar pensados para ilustrar y no para ser protagonistas.

El **código fuente** empleado para todas las etapas de la práctica debe estar correctamente comentado para facilitar su comprensión. Pueden emplearse ficheros Python nativos (.py) o basados en Jupyter Notebook (en este caso se debe entregar la versión .ipynb y la exportación en formato .html)