



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Facultat d'Informàtica de Barcelona



Sistemas Inteligentes Distribuidos

Agente de aprendizaje por refuerzo para el entorno Cliff Walking

Lluc Martínez Busquets
Eric Medina León
Àlex Rodríguez Rodríguez

2024/2025 Q2

Resumen

Este trabajo presenta un estudio detallado sobre la implementación de cuatro algoritmos de aprendizaje por refuerzo en el entorno *CliffWalking-v0* de la librería de Python *Gymnasium*: *Value Iteration*, *Direct Estimation*, *Q-Learning* y *REINFORCE*. El entorno se configura con el modo *is_slippery* activado, lo cual introduce estocasticidad en las transiciones de estado. El objetivo es evaluar el rendimiento de cada algoritmo, así como qué parámetros e hiperparámetros son los óptimos para su funcionamiento.

Índice

1	Introducción	3
1.1	Caracterización del entorno	3
1.1.1	Espacio de acciones	4
1.1.2	Estados inicial y terminal	4
1.1.3	Función de recompensa	4
1.2	Entorno experimental	4
2	Value Iteration	5
2.1	Descripción del algoritmo	5
2.2	Experimentación	5
3	Direct Estimation	5
3.1	Descripción del algoritmo	5
3.2	Experimentación	5
4	Q-learning	6
4.1	Descripción del algoritmo	6
4.2	Experimentación	6
5	Reinforce	6
5.1	Descripción del algoritmo	6
5.2	Experimentación	6
6	Conclusiones	6

1. Introducción

El aprendizaje por refuerzo es un paradigma de aprendizaje automático en el que un agente aprende a tomar decisiones interactuando con un entorno y recibiendo recompensas o penalizaciones.

Este trabajo se centra en la implementación y evaluación experimental de cuatro algoritmos de aprendizaje por refuerzo en el entorno *CliffWalking-v0*, de la librería de Python *Gymnasium*, con el objetivo de comparar su rendimiento y eficiencia en dicho entorno. Los algoritmos implementados son: *Value Iteration*, *Direct Estimation*, *Q-Learning* y *REINFORCE*.

1.1. Caracterización del entorno

El entorno de *Cliff Walking*, propuesto originalmente por *Sutton & Barto*, es un entorno clásico para evaluar algoritmos de aprendizaje por refuerzo. En este entorno, el agente debe navegar por una cuadrícula de dimensiones 4x12 evitando caer en un acantilado, lo que representa una penalización significativa. El objetivo del agente es llegar a la esquina inferior derecha de la cuadrícula de la forma más eficiente posible, maximizando la recompensa acumulada a lo largo del tiempo y minimizando el número de pasos necesarios para alcanzar la meta.

A continuación se caracteriza de forma detallada el entorno:

- **Observabilidad:** Totalmente observable. El agente recibe en cada paso su posición exacta en la cuadrícula, sin ruido ni información oculta, por lo que tiene acceso total al estado relevante.
- **Número de agentes:** Un único agente.
- **Determinismo:** Estocástico, ya que está activado el modo `is_slippery=True`. En este caso, por cada acción que el agente toma, hay una probabilidad de aproximadamente el 66.7% de que el agente se resvale hacia una dirección perpendicular a la acción deseada.
- **Atomicidad:** Secuencial. Las decisiones del agente tienen consecuencias que dependen de toda la historia de acciones y percepciones, y los efectos futuros de las acciones importan para maximizar la recompensa acumulada.
- **Dinamicidad:** Estático. El estado del entorno sólo cambia cuando el agente toma una acción; no hay cambios “por sí mismos” mientras el agente razona.
- **Continuidad:** Discreto. Tanto el espacio de estados (posiciones en la cuadrícula) como el de acciones (arriba, abajo, izquierda, derecha) y el tiempo de decisión son discretos.
- **Conocimiento:** Conocido. Las reglas de transición (aunque estocásticas) y la función de recompensa están definidas de antemano y son accesibles al agente.

1.1.1. Espacio de acciones

El agente dispone de un conjunto finito de acciones

$$\mathcal{A} = \{\text{Arriba, Derecha, Abajo, Izquierda}\},$$

cada una de las cuales intenta desplazar al agente una celda en la dirección indicada.

1.1.2. Estados inicial y terminal

- **Estado inicial** $s_0 = (3, 0)$, correspondiente a la esquina inferior izquierda de la cuadrícula.
- **Estado terminal** $s_T = (3, 11)$, la meta en la esquina inferior derecha; al llegar aquí, el episodio termina.

Si durante el episodio el agente cae en el acantilado, este regresa al estado inicial s_0 y continua con el episodio.

1.1.3. Función de recompensa

La señal de recompensa $R(s, a, s')$ se define como:

$$R(s, a, s') = \begin{cases} -100, & \text{si } s' \text{ es una celda de acantilado (cliff),} \\ -1, & \text{en cada transición válida que no alcance la meta ni el cliff,} \\ 0, & \text{al alcanzar el estado terminal } s_T. \end{cases}$$

De este modo, el agente está incentivado a llegar cuanto antes a la meta evitando caer en el precipicio.

1.2. Entorno experimental

Componente	Descripción
Sistema operativo	Ubuntu 24.04.1 LTS
Kernel Linux	6.8.0-59-generic
CPU	Intel Core i7-10750H (6 núcleos, 12 hilos, hasta 5.0 GHz)
GPU discreta	NVIDIA GeForce GTX 1650 Mobile (4 GB GDDR6)
GPU integrada	Intel CometLake-H GT2 (UHD Graphics)
Memoria RAM	16 GB DDR4-2933 MHz
Intérprete de Python	Python 3.12.9

Cuadro 1: Entorno de hardware y software utilizado en los experimentos

2. Value Iteration

2.1. Descripción del algoritmo

2.2. Experimentación

Observación	
Planteamiento	
Hipótesis	
Método	▪

Cuadro 2: Experimento 1

3. Direct Estimation

3.1. Descripción del algoritmo

3.2. Experimentación

Observación	
Planteamiento	
Hipótesis	
Método	▪

Cuadro 3: Experimento 1

4. Q-learning

4.1. Descripción del algoritmo

4.2. Experimentación

Observación	
Planteamiento	
Hipótesis	
Método	▪

Cuadro 4: Experimento 1

5. Reinforce

5.1. Descripción del algoritmo

5.2. Experimentación

Observación	
Planteamiento	
Hipótesis	
Método	▪

Cuadro 5: Experimento 1

6. Conclusiones