



PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC3675 — Aprendizaje Reforzado — 1' 2025

Tarea 4 – Respuesta Pregunta a)

El resultado del experimento con **SARSA** y **Q-Learning** es:

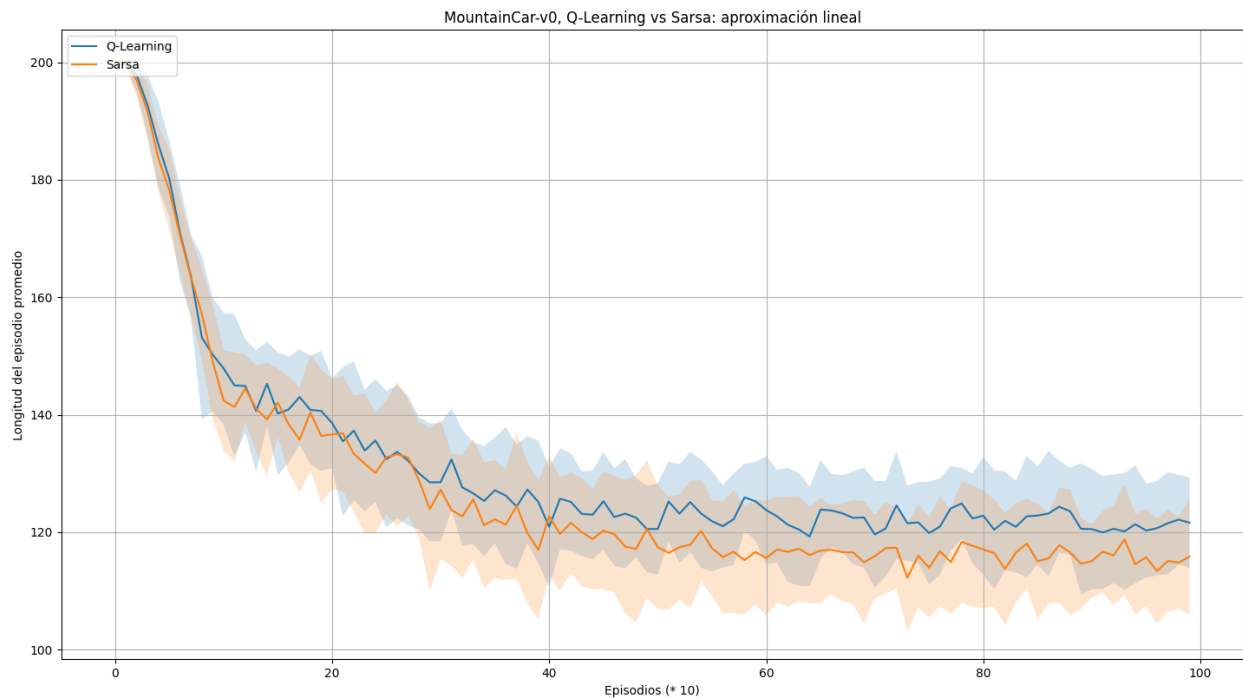


Figure 1: Largo promedio de los episodios en MountainCar-V0, comparación de SARSA con Q-Learning (reportado cada 10 episodios)

En base al gráfico, se observa que SARSA presenta un desempeño ligeramente superior al de Q-Learning a lo largo del entrenamiento. Aunque ambos algoritmos mejoran progresivamente, SARSA alcanza una menor longitud de episodio promedio y muestra una menor varianza, especialmente en los últimos episodios.

Esto indica que SARSA es más estable y consistente al aprender una política efectiva en este entorno, posiblemente debido a su naturaleza on-policy, que le permite adaptarse mejor bajo una política determinista ($\epsilon = 0.1$). Por tanto, en este experimento, SARSA puede considerarse ligeramente mejor que Q-Learning.



PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC3675 — Aprendizaje Reforzado — 1' 2025

Tarea 4 – Respuesta Pregunta b)

El resultado del experimento con **DQN** es:

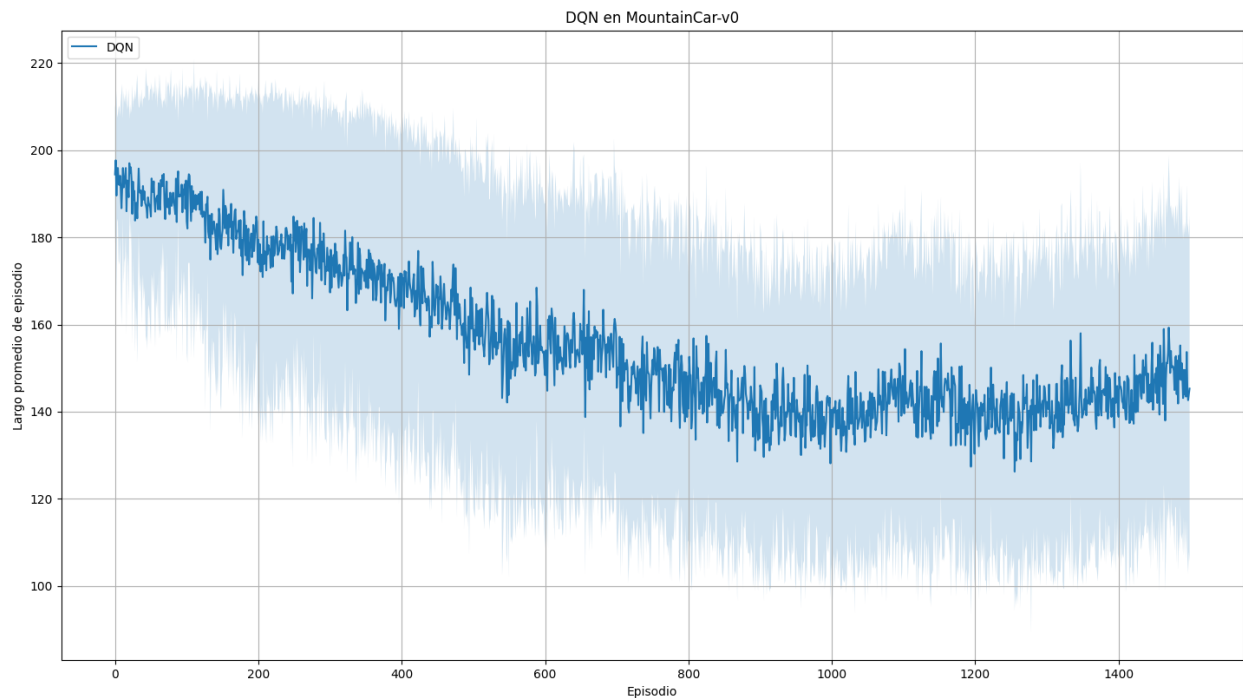


Figure 2: Largo promedio de los episodios en MountainCar-V0 para DQN (se muestra el promedio de los últimos 1500 episodios por cada *run*)

- **¿Es 'MountainCar-V0' un dominio fácil o difícil para DQN?:** Se puede considerar un dominio bastante difícil, ya que al principio del entrenamiento, se queda por muchos episodios fijo en 200 y nunca consigue superar notablemente a SARSA o Q-Learning con aproximaciones lineales.
- **¿Qué dificultades encontraste buscando hiperparámetros?:** Lo principal fue comprender qué debía modificarse. Posteriormente se intentó con *grid search*, sin embargo, no dio resultados. Finalmente lo más sencillo fue buscar implementaciones existentes en Hugging Face
- **Hiperparámetros más importantes:** Los principales fueron `target_update_interval`, `learning_rate` y `net_arch`. Esto tiene sentido ya que es importante que la **target network** no esté fluctuando todo el tiempo y que tampoco se quede fija durante casi todo el entrenamiento. Por otro lado, el **learning**

rate siempre es un parámetro sensible, y finalmente la **arquitectura de la red** es fundamental que sea suficientemente compleja como para aprender, pero no tanto que precisamente complica el proceso.

- **Configuración final de hiperparámetros:**

- `policy = 'MlpPolicy'`
 - `learning_rate = 0.004`
 - `buffer_size = 10000`
 - `learning_starts = 1000`
 - `batch_size = 128`
 - `gamma = 0.98`
 - `exploration_fraction = 0.2`
 - `exploration_final_eps = 0.07`
 - `train_freq = 16`
 - `gradient_steps = 8`
 - `target_update_interval = 600`
 - `policy_kwargs = {net_arch = [256, 256]}`

- **DQN vs SARSA y Q-Learning:** En comparación con SARSA y Q-Learning con aproximación lineal, DQN muestra un aprendizaje más lento y ruidoso. Aunque logra mejorar su desempeño con el tiempo, el gráfico evidencia una alta varianza entre episodios y una tendencia que se estabiliza en torno a los 140–150 pasos, sin lograr superar claramente a los métodos lineales. Este peor desempeño probablemente se deba a que DQN depende de redes profundas y necesita muchos datos; en problemas simples como **MountainCar-v0**, los métodos lineales con *tile coding* aprovechan mejor la estructura del estado y aprenden más rápido y estable.



PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC3675 — Aprendizaje Reforzado — 1' 2025

Tarea 4 – Respuesta Pregunta c)

Resultados del experimento con **Actor-Critic**:

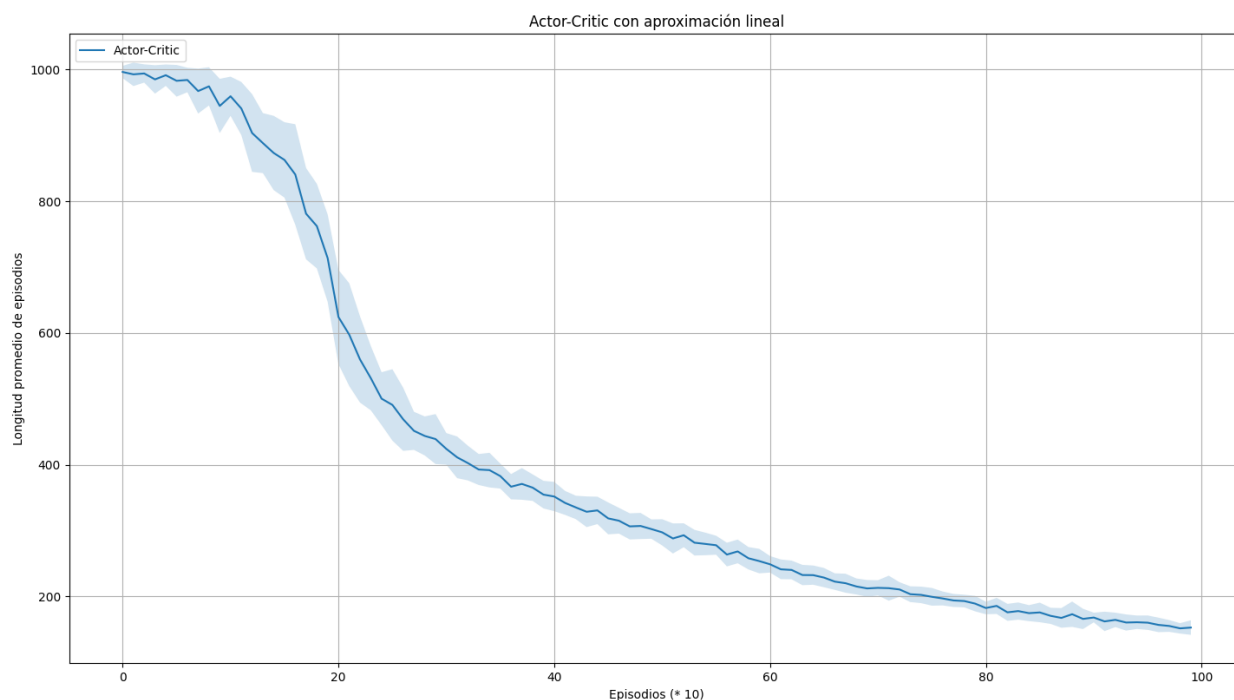


Figure 3: Largo promedio de los episodios en MountainCarContinuous-V0 para Actor-Critic (reportado cada 10 episodios)

Actor-Critic con aproximación lineal muestra una mejora rápida y sostenida, partiendo desde el máximo de 1000 pasos y bajando a menos de 200. Comparado con los métodos anteriores, aprende más lentamente que SARSA/Q-Learning al inicio, pero alcanza un rendimiento final más bajo en proporción al punto de inicio.



PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC3675 — Aprendizaje Reforzado — 1' 2025

Tarea 4 – Respuesta Pregunta d)

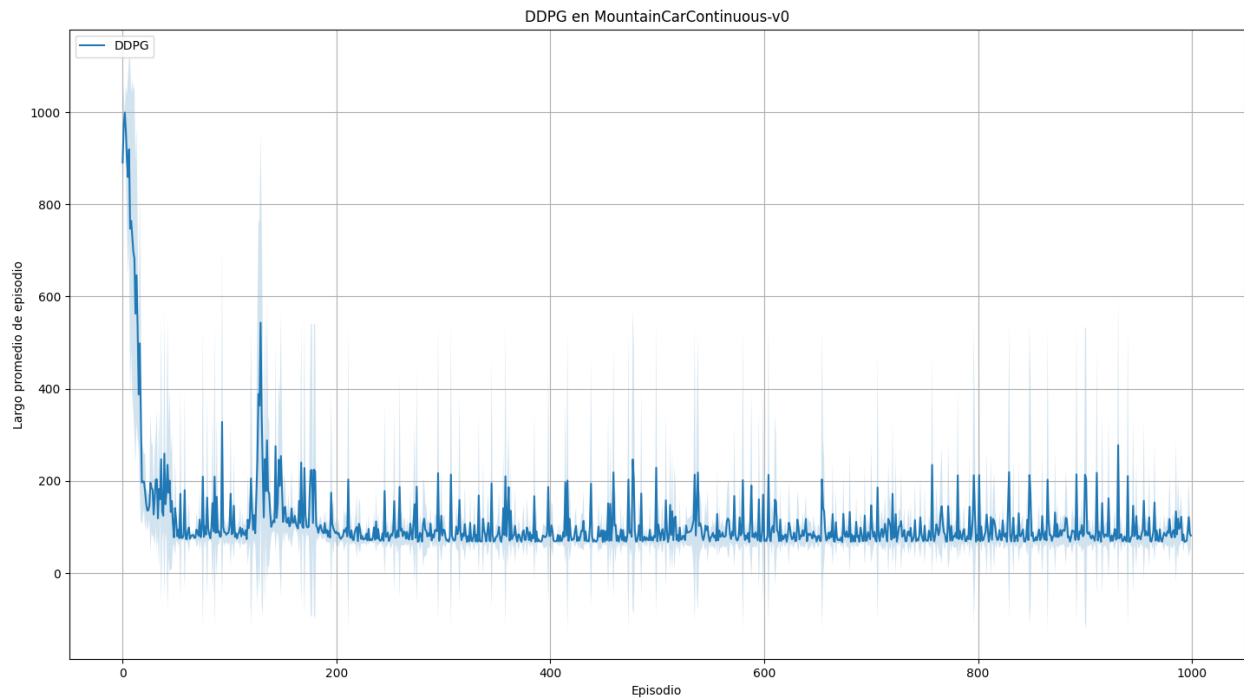


Figure 4: Largo promedio de los episodios en MountainCarContinuous-V0 para DDPG (se muestra el promedio de los primeros 1000 episodios por cada *run*)

- **¿Es un problema fácil o difícil para DDPG?:** Parece ser un problema bastante fácil para este algoritmo, ya que consigue llegar a largos de episodios considerablemente cortos bastante rápido. Además, observando con la visualización de **gym**, da la impresión que el agente realiza sus movimientos de la forma más eficiente posible.
- **¿Qué dificultades encontraste buscando hiperparámetros?:** Al igual que el caso anterior, fue difícil saber con que hiperparámetros comenzar la búsqueda. Primero encontré en **Hugging Face** un conjunto de hiperparams., pero el agente no aprendía nada. Finalmente encontré un repositorio con un listado de hiperparámetros que me funcionaron.
- **Configuración final de hiperparámetros:**

```

- policy = 'MlpPolicy'
- learning_rate = 1e-3
- batch_size = 256
- gradient_steps = 1
- train_freq = 1
- action_noise = OrnsteinUhlenbeckActionNoise(mean=0, sigma=0.5)
- policy_kwargs = {net_arch = [400, 300]}

```

- **¿Es mejor DDPG que Actor-Critic?:** DDPG llega a valores considerablemente más bajos en el largo de los episodios (alrededor de 90 vs los 130-150 de Actor-Critic), sin embargo, presenta mucha mayor variabilidad (también puede deberse a un menor número de runs). Actor-Critic con *tile coding* y una política estocástica, aprende de forma más estable, sin embargo, la complejidad de las redes neuronales de DDPG le da la capacidad de llegar a resultados mejores, pero con mayor inestabilidad.