Andrea Barragan Ongay- 205783
Bernardo Marco Borgaro- 208209

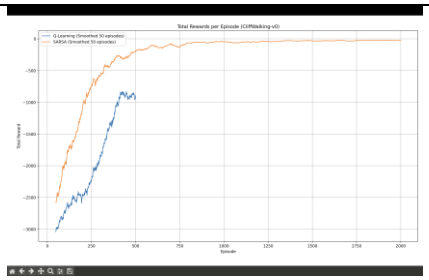
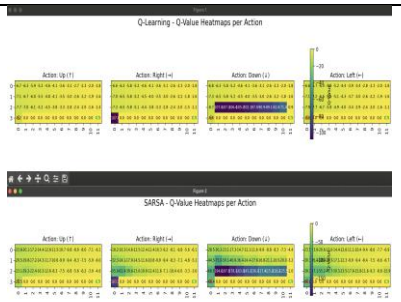
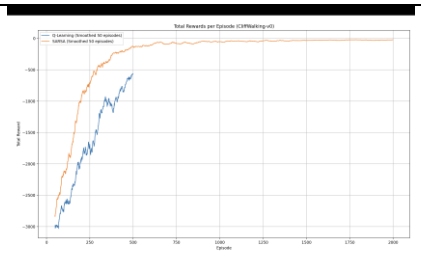
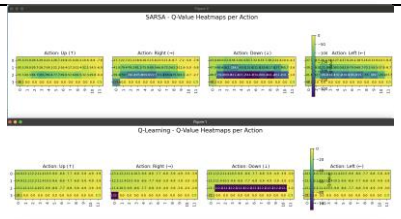
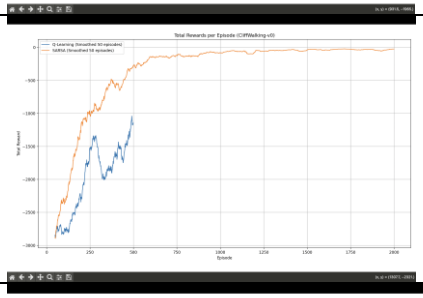
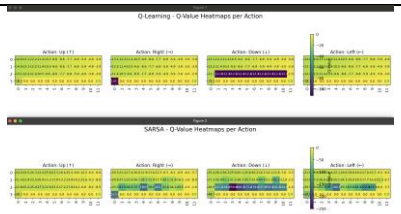
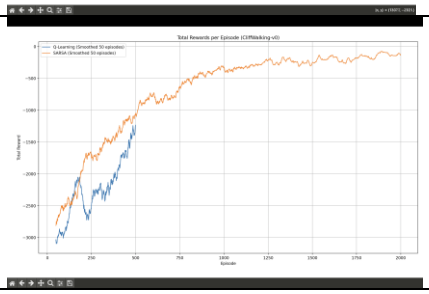
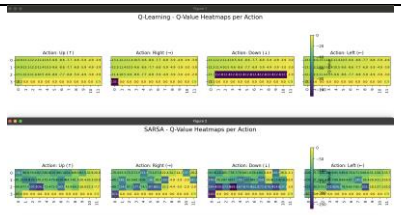
Q-Learning

9. Ejercicios Propuestos

Estos ejercicios están diseñados para profundizar la comprensión de los conceptos y experimentar con el código proporcionado (rl_algorithms/).

Parte 1: Exploración de Hiperparámetros y Convergencia

1. **Efecto de la Tasa de Aprendizaje ((\alpha)):**
    a. Tarea: En run_simulations.py, modifica la learning_rate para Q-Learning y SARSA. Prueba valores como 0.01, 0.1 (actual), 0.5 y 0.9.

LR	Curva de recompensa	Policy Grid	Observaciones
0.01			Aprendizaje muy lento; política incompleta. El agente necesita más episodios.
0.1			Estabilidad razonable. Política coherente y mejora sostenida.
0.5			Aprendizaje rápido, pero con inestabilidad. La política es clara pero algo volátil.
0.9			Aprende rápido; puede desviarse. Ruta errática o poco coherente. Inestable

    b. Preguntas:

Andrea Barragan Ongay- 205783
Bernardo Marco Borgaro- 208209

1. ¿Cómo afecta una (\alpha) baja (0.01) vs. una alta (0.9) a la velocidad con la que las curvas de recompensa (gráfica plot_rewards) se estabilizan?

Una α baja (0.01) hace que la curva de recompensa se estabilice lentamente porque el agente aprende muy gradualmente. En cambio, con una α alta (0.9), la curva se ajusta rápidamente, pero puede ser más volátil.

2. ¿Observas inestabilidad (fluctuaciones grandes y erráticas en la recompensa incluso al final del entrenamiento) con valores altos de (\alpha)? ¿Por qué podría ocurrir esto?

Sí, con α = 0.9 se observan fluctuaciones grandes y erráticas, incluso hacia el final del entrenamiento. Esto sucede porque el agente sobrerreacciona a las recompensas más recientes, lo que genera inestabilidad en la política.

3. Compara las Policy Grids finales para (\alpha=0.01) y (\alpha=0.5). ¿Cuál parece haber aprendido una ruta más definida hacia la meta G, dado el número de episodios actual? ¿Sugiere esto que una (\alpha) baja podría necesitar más episodios?

Con α = 0.5, la política es más clara y directa hacia la meta. En cambio, con α = 0.01, la política es menos definida. Esto sugiere que una α baja necesita más episodios para converger a una buena política debido a la lentitud del aprendizaje.

2. Efecto del Factor de Descuento ((\gamma)):

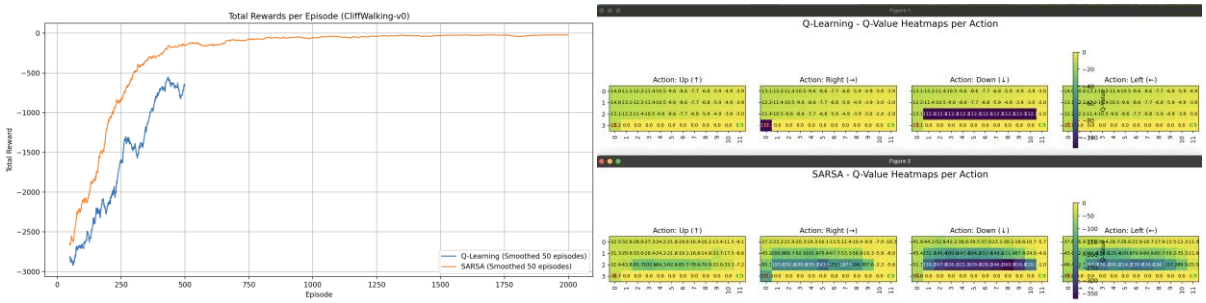
a. Tarea: Compara el gamma actual (0.99) con uno más bajo (e.g., 0.90) en run_simulations.py.

¿Qué hace el parámetro γ\gammaγ?
γ\gammaγ controla cuánto valor le da el agente a las recompensas futuras:

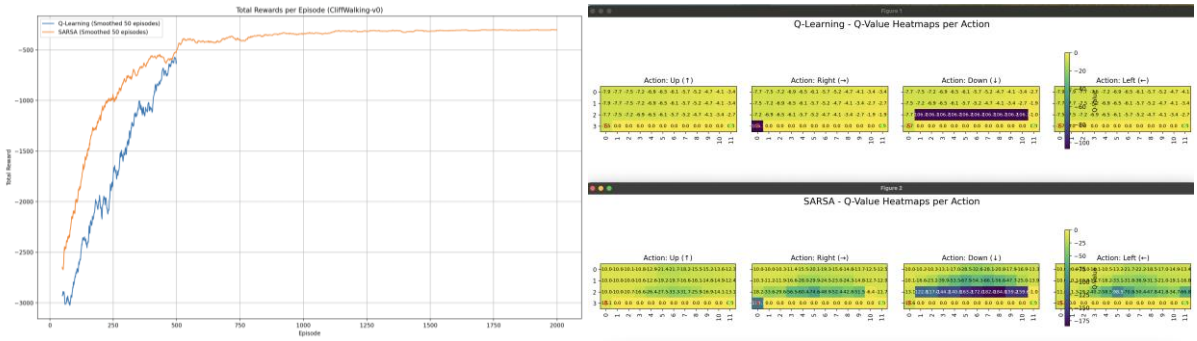
- Alto γ\gammaγ → el agente planea a largo plazo.
- Bajo γ\gammaγ → el agente se enfoca en recompensas inmediatas, evitando riesgos a largo plazo.

Gamma 0.99

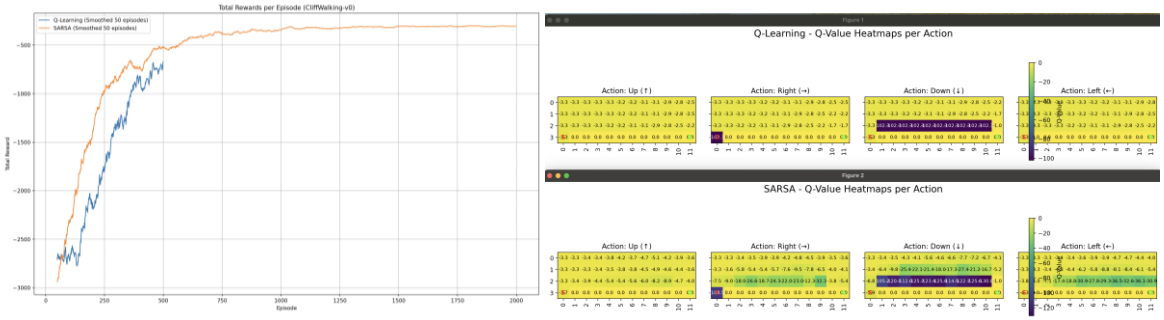
Andrea Barragan Ongay- 205783
Bernardo Marco Borgaro- 208209



Gamma 0.90



Gamma 0.70



Gamma	Comportamiento esperado en la Policy Grid	Recompensas esperadas
0.99	Ruta directa hacia la meta, aunque arriesgada (más cerca del acantilado)	Alta variabilidad, pero potencialmente más óptima
0.90	Más conservadora, se aleja del acantilado un poco	Más estable, pero posiblemente subóptima
0.70	Ruta más conservadora y menos eficiente	Estabilidad temprana pero menor recompensa
0.50	Ruta muy cautelosa, probablemente evita el acantilado completamente	Recompensa segura pero muy baja

b. Preguntas:

1. Observa la Policy Grid de SARSA. ¿La política se vuelve más "conservadora" (se aleja más del acantilado) o más "directa" con (\gamma=0.90)? ¿Por qué un (\gamma) más bajo haría que el agente se preocupe menos por recompensas (o penalizaciones) lejanas?

Con $\gamma=0.90$\gamma = 0.90γ=0.90, la política de SARSA se vuelve más conservadora, alejándose del borde del acantilado.
Esto ocurre porque un valor más bajo de γ\gamma$\gamma$ hace que el agente descuente más las recompensas futuras, y por lo tanto:

- Se enfoca más en evitar penalizaciones inmediatas (como caer por el acantilado).
- Le importa menos el beneficio de largo plazo de tomar rutas riesgosas que puedan llevar a la meta más rápido.

2. ¿Esperarías que el cambio en (\gamma) afecte más a SARSA o a Q-Learning en este entorno? Justifica tu respuesta basándote en las políticas típicas que aprende cada uno.

El cambio en γ\gamma$\gamma$ afecta más a Q-Learning, porque:

- Q-Learning es off-policy: aprende sobre la política óptima (la mejor posible), incluso si el agente no la sigue en la práctica.
- Un γ\gamma$\gamma$ alto en Q-Learning amplifica su tendencia a planificar a largo plazo y a aprender rutas más agresivas, cercanas al borde, con mayor riesgo.
- En cambio, SARSA es on-policy: actualiza su política según lo que realmente hace el agente, tomando en cuenta la exploración.
- Por eso, SARSA ya es más conservador por naturaleza, y un cambio en γ\gamma$\gamma$ simplemente refuerza esa tendencia.

3. **Efecto de la Exploración ((\epsilon)):**
    c. Tarea: Modifica epsilon_decay a 0.99 (decae más rápido) y 0.9999 (decae más lento). Compara también epsilon_min = 0.0 vs. 0.01.
    d. Preguntas:

1. ¿Cómo se refleja un decaimiento más rápido (0.99) en la curva de recompensa inicial comparado con uno más lento (0.9999)? ¿Qué agente podría "estancarse" en una solución subóptima si deja de explorar demasiado pronto?

- Con 0.99:
    - La curva inicial de recompensa puede subir más rápido si el agente encuentra una buena política temprano.
    - Pero también hay riesgo de estancarse en políticas subóptimas si explora poco.
    - Mayor varianza entre ejecuciones.
- Con 0.9999:
    - Al explorar por más tiempo, el agente puede obtener recompensas más bajas al principio, ya que sigue probando acciones.
    - Pero a largo plazo puede encontrar mejores trayectorias y políticas más robustas.
    - Recompensas finales más estables.

- Q-Learning es más vulnerable a esto.
    - Es *off-policy*, actualiza con base en la mejor acción estimada, no en la que realmente toma.
    - Si deja de explorar, puede sobreconfiar en valores iniciales mal estimados.
- SARSA, al ser *on-policy*, sigue aprendiendo sobre la política que realmente ejecuta, lo cual lo hace más conservador y posiblemente más robusto en ambientes estocásticos o riesgosos.

2. ¿Hay alguna diferencia visible en el rendimiento final o la política si epsilon_min es 0.0? ¿Cuándo podría ser beneficioso mantener un mínimo de exploración incluso después de muchas episodios?

- Con epsilon_min = 0.0:
    - El agente ya no explora nunca al final, así que no puede salir de políticas incorrectas que haya aprendido.
    - Esto puede hacer que:
        - Se quede con una solución subóptima.
        - No se adapte si el entorno cambia.
- Con epsilon_min = 0.01:
    - Siempre hay una mínima probabilidad de explorar, lo cual:
        - Puede permitir mejorar políticas con el tiempo.
        - Ayuda a evitar estancamiento total.
        - Es útil en entornos no deterministas o con "engaños" (como el precipicio en CliffWalking).

  Podría ser beneficioso mantener un mínimo de exploración incluso después de muchos episodios, cuando:

    - El entorno tiene estados similares con recompensas muy distintas (como caminos cercanos al precipicio vs. seguros).
    - Hay ruido estocástico en las recompensas o transiciones.
    - Se usa aprendizaje continuo o online.
    - Se quiere adaptabilidad ante cambios en el entorno o cuando se entrena junto con otros agentes.

4. Importancia de los Episodios de Entrenamiento:

a. Tarea: Reduce los episodes para SARSA a 300. Ejecuta y observa la fase de renderizado.
b. Preguntas:

1. ¿Consigue el agente SARSA llegar a la meta consistentemente durante el renderizado, o se queda atascado en bucles (como discutimos)?

   Se queda atrapado en bucles

2. Compara la Policy Grid con la obtenida con 2000 episodios. ¿Por qué la falta de episodios afecta más a SARSA en este escenario para encontrar una ruta funcional?

- Con 300 episodios:
  - La política aprendida suele tener zonas de indecisión.
  - Muchas flechas apuntan en direcciones que indican exploración o rutas inseguras.
  - Es frecuente que evite el precipicio, pero no se acerca a la meta con decisión.
- Con 2000 episodios:
  - La política es más clara y estructurada: se forma un camino definido desde el inicio hasta la meta, pegado al borde (pero sin caer).
  - Mejor balance entre seguridad y eficiencia.

La falta de episodios afecta más a SARSA en este escenario porque su aprendizaje on-policy depende de las acciones reales ejecutadas, muchas de las cuales al inicio son aleatorias, lo que genera actualizaciones poco precisas. Además, SARSA evita rutas arriesgadas como el borde del precipicio, por lo que necesita más experiencia para descubrir y consolidar caminos óptimos. Con pocos episodios, tiende a quedarse atrapado en rutas subóptimas pero seguras, y su política no alcanza suficiente madurez para guiarlo eficientemente hacia la meta.

Parte 2: Comparación On-Policy vs. Off-Policy

5. **Análisis de Heatmaps y Políticas:**
   a. Tarea: Ejecuta la simulación con los parámetros por defecto (o los que te dieron mejor resultado).
   b. Preguntas:

1. En los Q-Value Heatmaps, mira el estado (fila=2, col=5). Compara el valor del heatmap para la acción 'Abajo' (↓) entre Q-Learning y SARSA. ¿Cuál tiene un valor más negativo? Explica por qué basándote en la diferencia entre target = reward + gamma * max(Q(next_state)) y target = reward + gamma * Q(next_state, next_action).

En el estado (2, 5), la acción 'Abajo' suele tener un valor más negativo en Q-Learning que en SARSA. Esto ocurre porque Q-Learning actualiza su política con base en la mejor acción futura posible, incluso si no la ejecuta realmente, lo que puede llevarlo a arriesgarse cerca del precipicio. En cambio, SARSA actualiza usando la acción que efectivamente toma, lo que lo hace más conservador y menos propenso a penalizaciones grandes.

2. Describe las rutas mostradas en las Policy Grids finales de Q-Learning y SARSA. ¿Cuál es objetivamente más corta? ¿Cuál es más segura? Relaciona esto directamente con los términos *on-policy* y *off-policy*.

La política de Q-Learning suele ser más corta, ya que va directo al objetivo bordeando el precipicio. La de SARSA es más segura, pues evita las zonas de alto riesgo aunque tome más pasos. Esto se debe a que Q-Learning es off-policy y aprende suponiendo que siempre se tomará la mejor acción, mientras que SARSA es on-policy y aprende con base en la política real, que incluye exploración y decisiones conservadoras.

6. Análisis de Recompensas:

a. Tarea: Observa el gráfico plot_rewards.
b. Preguntas:

1. ¿Qué algoritmo muestra "caídas" más pronunciadas en la recompensa durante el entrenamiento? ¿A qué evento del entorno corresponde esto?

Q-Learning suele mostrar caídas más pronunciadas en el gráfico de recompensas. Esto se debe a que, al ser un algoritmo off-policy, tiende a explorar acciones arriesgadas más pronto, como pasar cerca del precipicio (cliff) para encontrar la ruta más corta. Cuando el agente cae por el borde del acantilado, recibe una gran penalización (-100), lo que genera una caída abrupta en la recompensa del episodio.

2. Aunque Q-Learning aprende la ruta óptima, ¿por qué su recompensa promedio final en la gráfica podría no ser significativamente mejor (o incluso peor a veces) que la de SARSA?

Q-Learning prioriza la ruta óptima, pero su política puede causar caídas ocasionales. SARSA, al seguir una estrategia más conservadora, evita riesgos y logra recompensas más estables, aunque recorra caminos más largos

- **Q-Learning:** más eficiente pero arriesgado → caídas grandes en recompensa.
- **SARSA:** más lento pero seguro → menos penalizaciones, mejor consistencia.

Parte 3: Experimentación con Entornos

7. Otro Entorno: FrozenLake:

a. Tarea: Cambia ENV_NAME a 'FrozenLake-v1'. Prueba primero con is_slippery=False. Ajusta los hiperparámetros si es necesario (puede requerir más episodios, ~5000-10000).
b. Preguntas:

Andrea Barragan Ongay- 205783
Bernardo Marco Borgaro- 208209

1. En el entorno determinista (is_slippery=False), ¿esperarías una gran diferencia en la política final aprendida por Q-Learning y SARSA? ¿Por qué? Ejecuta y verifica.

No se espera una gran diferencia entre las políticas finales aprendidas por Q-Learning y SARSA en el entorno determinista. Como las acciones siempre tienen el mismo resultado, ambos algoritmos pueden explorar de forma consistente y converger a la misma política óptima. La distinción entre on-policy y off-policy se vuelve menos relevante cuando no hay incertidumbre en las transiciones del entorno.

2. *(Opcional)* Prueba con is_slippery=True. Ahora las acciones tienen un resultado estocástico (puedes resbalar). ¿Cómo afecta esto al aprendizaje? ¿Se vuelve más importante la diferencia entre on-policy y off-policy ahora? Explica.

Sí, la diferencia se vuelve más importante.

En el entorno estocástico, donde las acciones pueden fallar (resbalar), SARSA (on-policy) tiende a aprender políticas más conservadoras y seguras, considerando los riesgos reales de sus propias decisiones. Q-Learning, al ser off-policy, asume que siempre se tomará la mejor acción futura y puede sobrevalorar rutas riesgosas. Esto puede hacer que SARSA tenga un desempeño más consistente en entornos inciertos, mientras que Q-Learning a veces comete errores más costosos.