VOLCANOLAKE: SISTEMA DE APRENDIZAJE POR REFUERZO

Documento Técnico

Asignatura: Inteligencia Artificial
Curso: 3º de Ingeniería Informática
Grupo: 3
Autores: Eduardo Estefanía Ovejero, Álvaro Martín García

Control de Documento

Versión	Fecha	Autores	Descripción / Cambios
0.1	22/11/2025	Eduardo Estefanía, Álvaro Martín	Esqueleto de las versiones 1, 2 y 3
0.5	22/11/2025	Eduardo Estefanía, Álvaro Martín	Explicaciones de las versiones 1, 2 y 3. Creación del anexo
1.0	23/11/2025	Eduardo Estefanía, Álvaro Martín	Versión final (Imágenes insertadas de los resultados del entrenamiento junto a las descripciones)

Índice

1. Introducción

El proyecto **VolcanoLake** consiste en la implementación de un sistema de Aprendizaje por Refuerzo (*Reinforcement Learning*) tabular, diseñado para resolver problemas de navegación en entornos de cuadrícula (*grid-worlds*) estocásticos y dinámicos.

El núcleo del sistema se basa en el algoritmo **Q-Learning** (Off-Policy TD Control) aplicado sobre un entorno personalizado compatible con la API estándar de **Gymnasium**. El desarrollo sigue una metodología incremental dividida en tres versiones, evolucionando desde un entorno estándar hasta una simulación compleja con físicas personalizadas y wrappers de modificación de comportamiento.

2. Metodología y Flujo de Trabajo

El desarrollo del proyecto se ha regido por una metodología iterativa e incremental, dividida en tres fases principales para garantizar la estabilidad del código y la correcta implementación de los conceptos teóricos.

2.1. Estrategia de Desarrollo Incremental

En lugar de abordar el problema final directamente, se dividió el desarrollo en tres hitos funcionales (versiones), asegurando que cada fase validara un componente crítico del sistema antes de añadir complejidad:

1. **Fase 1 (Validación del Algoritmo):** Se implementó el agente Q-Learning sobre un entorno estándar (`FrozenLake-v1`).
    - *Objetivo:* Verificar que la implementación de la Ecuación de Bellman y la política Epsilon-Greedy funcionaban correctamente en un entorno controlado y conocido.
2. **Fase 2 (Modularidad y Wrappers):** Se introdujeron modificaciones al entorno mediante el patrón de diseño *Decorator* (Wrappers de Gymnasium).
    - *Objetivo:* Probar la capacidad de adaptación del agente ante cambios dinámicos (lava creciente) y restricciones sensoriales (visión limitada) sin alterar el código base del entorno.
3. **Fase 3 (Entorno Personalizado):** Se desarrolló la clase `VolcanoLakeEnv` desde cero.
    - *Objetivo:* Implementar físicas personalizadas (resbalones de agua, consumo de tesoros) y escalar el problema a mapas grandes (25x25), integrando todo lo aprendido en las fases anteriores.

2.2. Flujo de Trabajo de Experimentación (Pipeline)

Para cada versión, se siguió un ciclo de experimentación constante:

1. **Diseño del Escenario:** Definición de mapas y reglas (ej. posición de tesoros).
2. **Entrenamiento:** Ejecución de bucles largos (100k episodios) monitorizados con `tqdm`.

3. **Evaluación de Métricas:** Análisis de las curvas de recompensa y error TD generadas.
    - *Feedback:* Si el agente no aprendía (ej. ignoraba tesoros), se ajustaba la función de recompensa (*Reward Shaping*) y se reiniciaba el entrenamiento.
4. **Validación Visual:** Generación de Heatmaps y vídeos de los episodios finales para confirmar que la política aprendida era lógica y segura.

3. Stack Tecnológico

El sistema ha sido desarrollado utilizando **Python 3.13** y las siguientes librerías científicas:

- **Gymnasium (v1.2.1):** Framework base para la estandarización del entorno (`Env`) y el patrón de diseño Decorator (`Wrapper`).
- **NumPy (v2.3.4):** Gestión eficiente de matrices para la Q-Table y manipulación de arrays del mapa.
- **Pygame (v2.6.1):** Motor de renderizado gráfico 2D para la visualización en tiempo real (`human`) y generación de frames para video (`rgb_array`).
- **Matplotlib (v3.10.7):** Generación de analíticas visuales (curvas de aprendizaje, heatmaps).
- **TQDM:** Monitorización del progreso del bucle de entrenamiento.

4. Arquitectura del Software

El proyecto sigue una estructura modular que separa la lógica del agente, la física del entorno y las herramientas de análisis.

4.1. Estructura de Directorios

La solución final (v3) se organiza de la siguiente manera:

- `agent/`: Contiene la clase `VolcanoLakeAgent`, encapsulando la lógica de selección de acciones y actualización de valores Q.
- `envs/`: Contiene la clase `VolcanoLakeEnv`, que hereda de `gym.Env`.
- `wrappers/`: Módulos de modificación del entorno (Decoradores).
- `training/`: Lógica del bucle principal de entrenamiento.
- `utils/`: Herramientas de persistencia de datos y visualización (`plotting.py`).
- `maps/`: Archivos CSV que definen la topología de los escenarios.

4.2. Diseño del Entorno (VolcanoLakeEnv)

La clase principal `VolcanoLakeEnv` implementa la interfaz `gym.Env`.

- **Espacio de Estados (S):** Discreto. Se representa como un entero único calculado mediante $s = row \times ncols + col$
- **Espacio de Acciones (A):** Discreto(8). Permite 8 grados de libertad (Cardinal + Diagonal).
- **Dinámica de Transición (P):**
    - **Determinista:** En casillas de tierra (`.`), $P(s'|s,a) = 1$.
    - **Estocástica:** En casillas de agua (`W`), la transición sigue una distribución de probabilidad:

- 0.8: Movimiento deseado.
- 0.1: Desvío ortogonal izquierdo.
- 0.1: Desvío ortogonal derecho.

4.3 Diagrama de Clases del Sistema

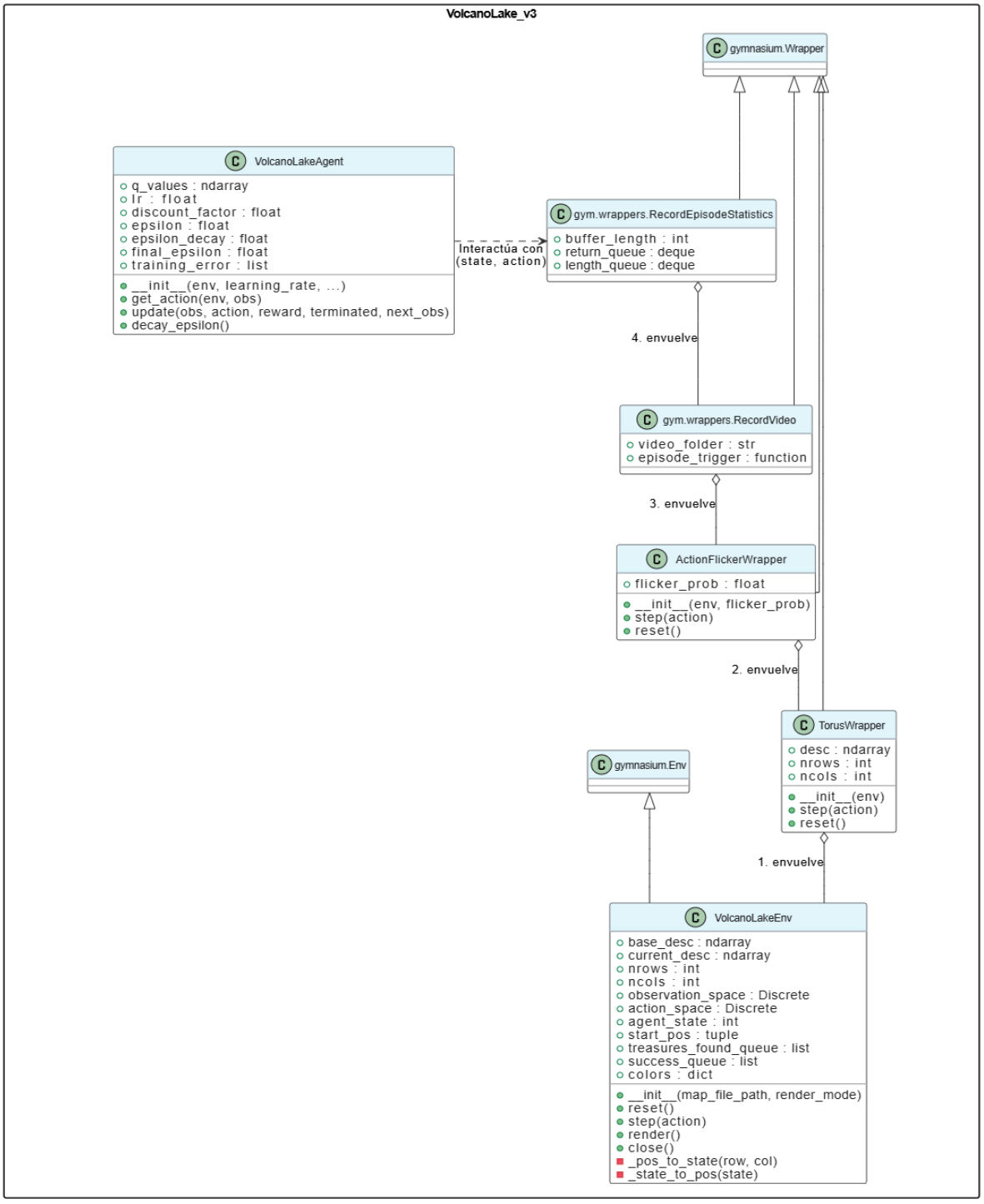


Figura 4.1: Diagrama de Clases UML (Versión 3) *El diagrama ilustra la arquitectura orientada a objetos del sistema. La clase `VolcanoLakeEnv` implementa la interfaz estándar `gym.Env`, gestionando la lógica de física y estado. La funcionalidad modular se inyecta mediante el patrón de diseño Decorator, representado por las clases que heredan de*

`gym.Wrapper` (`TorusWrapper`, etc.), permitiendo extender el comportamiento del entorno dinámicamente sin alterar su código base. El `VolcanoLakeAgent` interactúa con esta abstracción del entorno de forma agnóstica.

5. Algoritmo de Aprendizaje (Q-Learning)

El agente implementa **Q-Learning**, un algoritmo de diferencia temporal (TD) que busca aprender la función de valor óptima $Q^*(s, a)$.

5.1. Ecuación de Bellman

La actualización de la Q-Table se realiza en cada paso del episodio siguiendo la ecuación:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma * max_{a'} Q(s', a') - Q(s, a)]$$

Donde:

- s, a: Estado y acción actuales.
- r: Recompensa inmediata recibida (incluyendo penalizaciones de paso y bonus de tesoro).
- s': Nuevo estado resultante.
- α (Alpha): Tasa de aprendizaje (0.01).
- γ (Gamma): Factor de descuento (0.99).

5.2. Política de Exploración (Epsilon-Greedy)

Para equilibrar la exploración y explotación, se utiliza una política ε-greedy con decaimiento lineal:

1. Se genera un número aleatorio $p \in [0, 1]$.
2. Si $p < \varepsilon$: Selecciona $a \in A$ aleatoriamente (Exploración).
3. Si $p \geq \varepsilon$: Selecciona $a = argmax_a Q(s, a)$ (Explotación).

El valor de ε decae linealmente desde `start_epsilon=1.0` hasta `final_epsilon=0.1` durante la primera mitad del entrenamiento.

6. Implementación de Modificadores (Wrappers)

Se ha utilizado el patrón de diseño **Decorator** nativo de Gymnasium para alterar la funcionalidad del entorno sin modificar su código fuente.

6.1. `PersistentIncreasingLavaHoles` (v3)

- **Lógica Técnica:** Mantiene un conjunto (`set`) persistente de coordenadas `(row, col)`.
- **Intervención:** En cada `reset()`, inyecta las coordenadas almacenadas en la matriz `current_desc` del entorno base, asegurando que la degradación del terreno se mantenga entre episodios.

6.2. `LimitedVision`

- **Lógica Técnica:** Calcula vectores de visión basados en la última acción tomada (a_{t-1}).
- **Intervención:** Intercepta `step()`. Calcula la coordenada (x', y') proyectada. Si (x', y') contiene un obstáculo (`L`, `W`) o sale de los límites (`clip`), inyecta una penalización negativa adicional en la recompensa escalar r.

6.3. `TorusWrapper`

- **Lógica Técnica:** Modifica la topología del espacio euclídeo.
- **Intervención:** Detecta si $s_{t+1} == s_t$ (colisión con muro). Si es verdadero, calcula la nueva posición aplicando aritmética modular sobre las dimensiones del mapa: $pos_{new} = (pos_{old} + vector)\ (mod\ dimensiones)$.

7. Metodología de Experimentación y Análisis

El entrenamiento se configuró con **100,000 episodios** en un mapa de 25x25 con tesoros y peligros dinámicos.

7.1. Análisis de Métricas de Convergencia

Las gráficas resultantes (ver Anexo I) muestran tres fases claras:

1. **Exploración (0 - 40k episodios):** El error TD es alto y la recompensa baja. El agente está mapeando el espacio de estados.
2. **Convergencia ("El Salto"):** Alrededor del episodio 40k, ε desciende lo suficiente para que la política greedy domine. La tasa de éxito se dispara del 10% al 90%.
3. **Estabilización (40k - 100k episodios):** La tasa de éxito se estabiliza en ~90%. El 10% de fallo residual es atribuible técnicamente a:
    - Exploración forzada ($\varepsilon = 0.1$).
    - Estocasticidad irreductible del terreno 'Agua' (20% de deslizamiento).

7.2. Validación de Comportamiento Emergente

El análisis de la métrica "Tesoros Recogidos" muestra una correlación directa con la tasa de éxito. El agente aprendió que la recompensa acumulada de la ruta $S \rightarrow T \rightarrow G(+\ 15)$ compensa el coste de pasos adicionales frente a la ruta directa $S \rightarrow G(+\ 10)$, validando el diseño de la función de recompensa (*Reward Shaping*).

8. Generación de Artefactos

El sistema produce automáticamente artefactos de validación post-entrenamiento:

- **Matriz de Valores (Heatmap):** Visualización directa de $max_a Q(s, a)$, permitiendo depurar zonas muertas (valor 0) y gradientes de recompensa.
- **Mapa de Política (Quiver Plot):** Representación vectorial de $argmax_a Q(s, a)$, permitiendo verificar visualmente la optimalidad de la ruta.

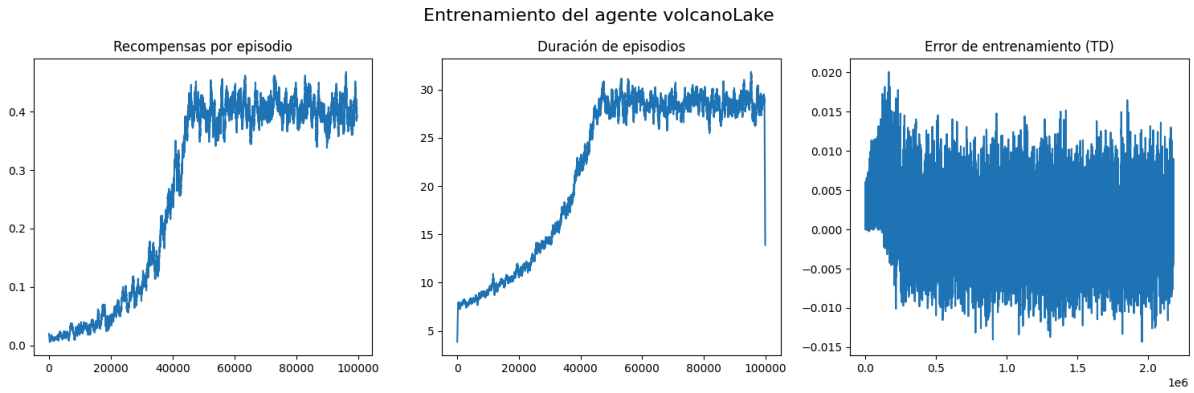
Anexo I: Gráficas y Resultados Visuales

En este anexo se recogen los artefactos visuales generados automáticamente por el sistema tras el entrenamiento de 100.000 episodios. Estas gráficas permiten validar la convergencia del algoritmo Q-Learning y analizar la estrategia final adquirida por el agente.

A.1. Resultados de la Versión 1 (Línea base)

Entorno: FrozenLake Estándar (4x4) | Acciones: 4

Figura A.1: Métricas de Entrenamiento (v1)

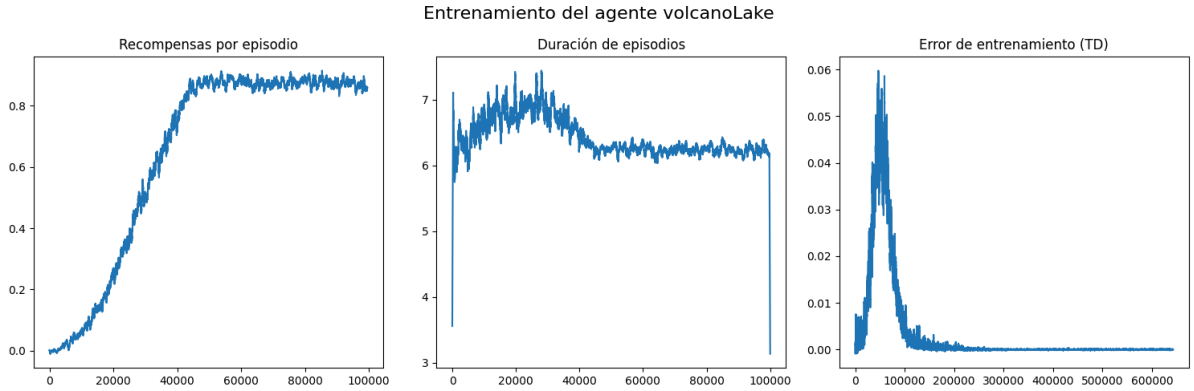


Descripción: Se observa una convergencia inestable y lenta debido a la falta de mecanismos de corrección avanzados. La recompensa media oscila sin estabilizarse completamente. La repentina creciente que se observa en 40000-50000 se correlaciona directamente con la fase final del decaimiento de ε, momento en el que agente pasa de un comportamiento exploratorio a uno mayormente explotador.

A.2. Resultados de la Versión 2

Figura A.2: Métricas de Entrenamiento (v2)

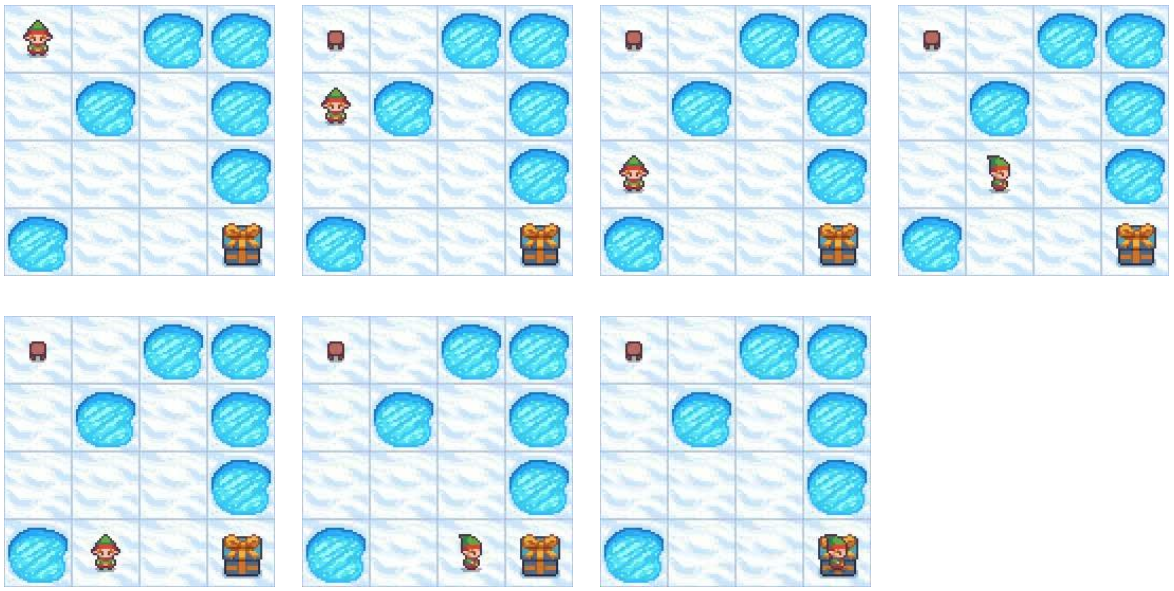
Entorno: FrozenLake Modificado (4x4) | Acciones: 4



*Descripción: Se observa una convergencia robusta a partir del episodio 40.000. La rapidez de este aprendizaje se atribuye principalmente al **Reward Shaping** (moldeado de recompensas) introducido por el wrapper `LimitedVision`, que guía al agente evitando zonas peligrosas antes de caer.Además, la estabilidad final es notable considerando la dificultad añadida por el wrapper `IncreasingHoles`, que convierte el entorno en **no-estacionario**, obligando al agente a readaptar su política ante la aparición de nuevos obstáculos. El `TimeLimit` actuó como mecanismo de seguridad para evitar bucles de exploración improductivos."*

Figura A.3: Frames del Episodio Final (v2)

Entorno: FrozenLake Modificado (4x4) | Acciones: 4

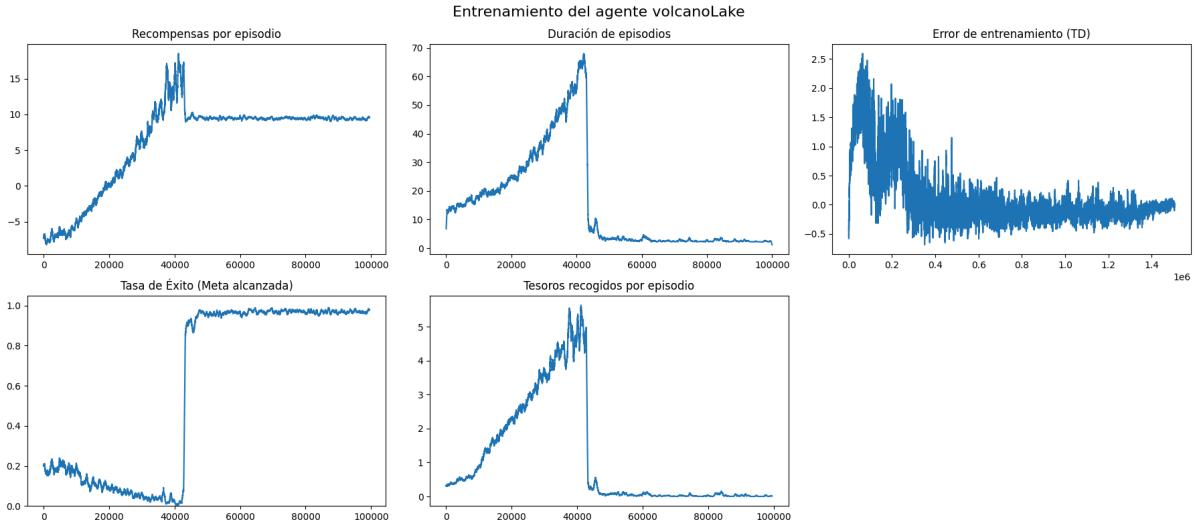


Descripción: Se observa como el agente sigue la única ruta óptima evitando los hoyos, llegando en 6 pasos.

A.3. Resultados de la Versión 3

Figura A.4: Métricas de Entrenamiento (v3)

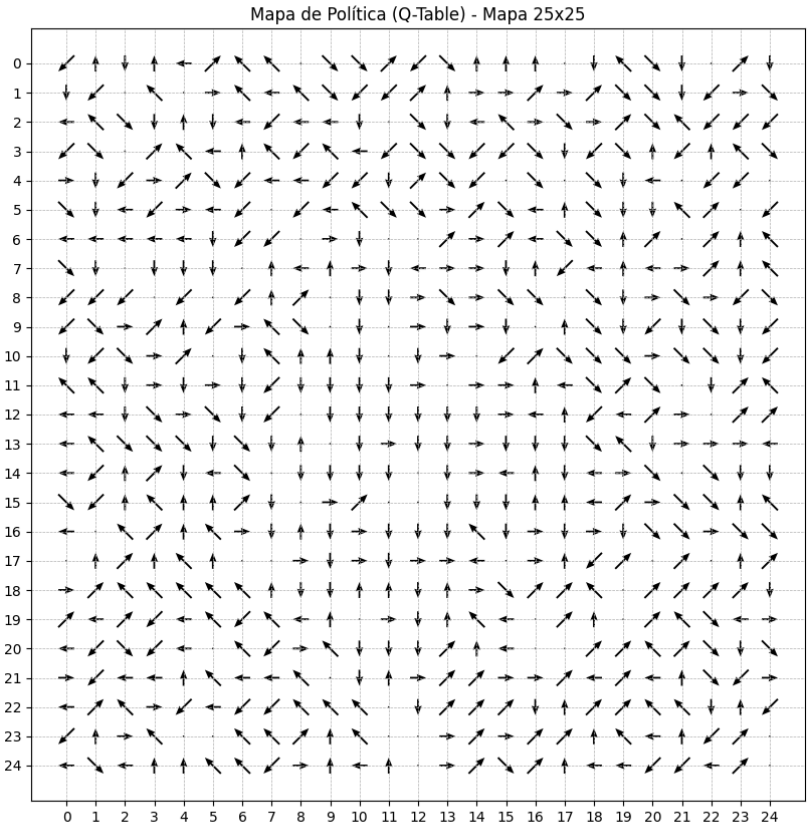
Entorno: VolcanoLake (25x25) | Acciones: 8



Descripción: La gráfica de Recompensas por Episodio (izquierda) muestra una clara transición de fase. Durante los primeros 40.000 episodios, el crecimiento es lineal pero ruidoso, correspondiendo a la fase de alta exploración ($\varepsilon > 0.5$).Se observa un punto de inflexión crítico ("codo") en el episodio 40.000, momento en el que la tasa de exploración desciende por debajo de un umbral significativo, permitiendo que la política explotadora domine. A partir de este punto, la recompensa media se estabiliza en torno a +10, validando que el agente solo llega a la meta ignorando el objetivo secundario de recoger los tesoros. Por último, la Tasa de Éxito (abajo-izq) se mantiene en un 90% y no en el 100%, lo cual es consistente con la estocasticidad del terreno 'Agua' (20% de deslizamiento) y la exploración residual forzada ($\varepsilon_{final} = 0.1$), demostrando que el agente ha alcanzado el límite teórico de rendimiento del entorno.

Figura A.5: Mapa de política (v3)

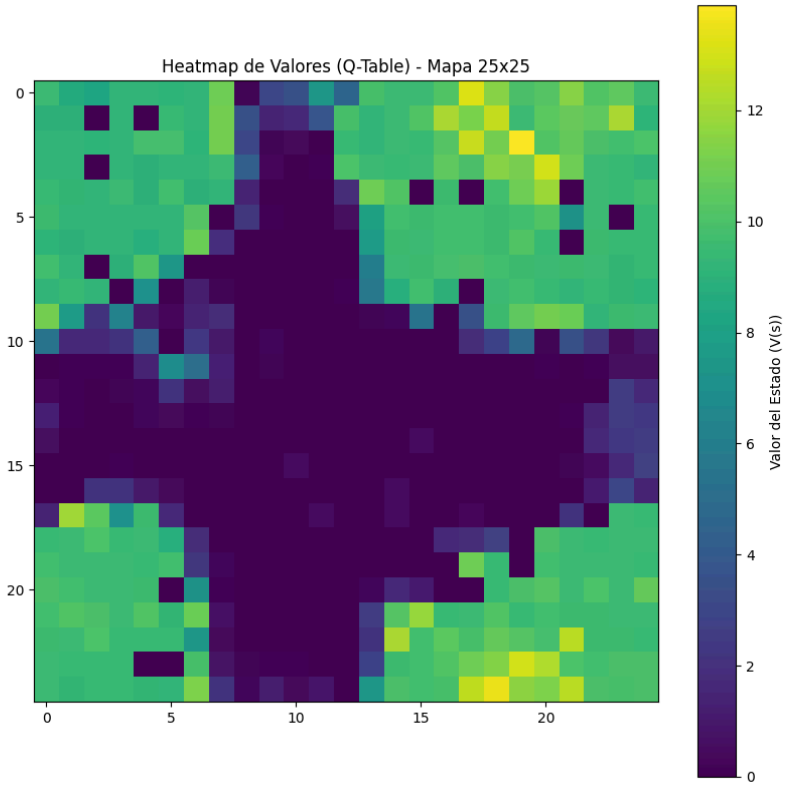
Entorno: VolcanoLake (25x25) | Acciones: 8



Descripción: La prevalencia de vectores diagonales confirma que el agente aprovecha eficazmente los 8 grados de libertad introducidos en la V3 para optimizar la distancia Manhattan, reduciendo el coste total por pasos. La visualización revela un "campo de flujo" convergente hacia la meta. Esto indica una política robusta: incluso si la estocasticidad del terreno ('Agua') desplaza al agente a una casilla adyacente no deseada, la política local de esa nueva casilla redirige al agente hacia la trayectoria óptima (recuperación de errores).

Figura A.6: Mapa de calor (v3)

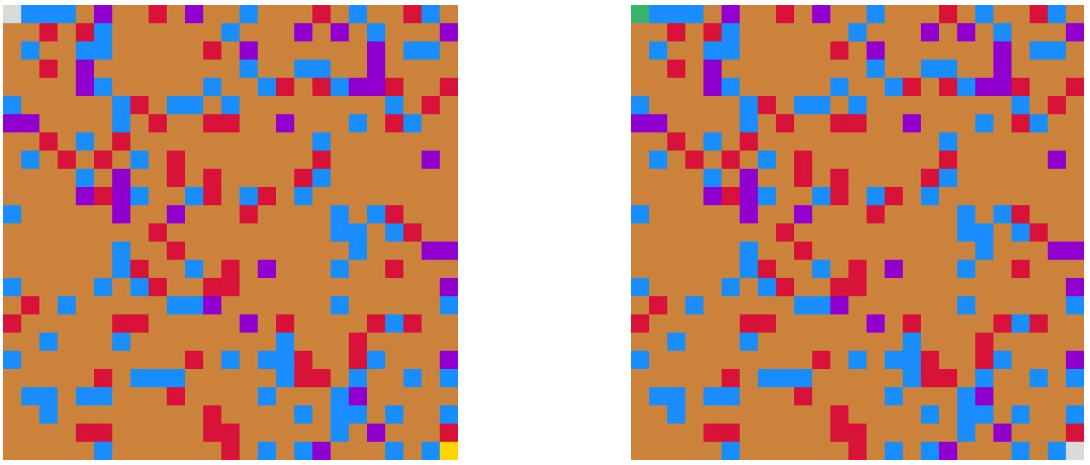
Entorno: VolcanoLake (25x25) | Acciones: 8



Descripción: $V(s) = max_a Q(s,a)$ *tras 100.000 episodios. La escala de colores indica la recompensa futura esperada desde cada celda (Amarillo = Alta, Morado = Baja/Nula). Se observa un fenómeno muy característico de las topologías toroidales. Las zonas de alto valor (colores claros/amarillos) se concentran en la periferia y las esquinas del mapa, mientras que el centro geométrico presenta un valor cercano a 0 (zona oscura en forma de cruz). Debido a la conexión de los bordes (wrap-around), la distancia geodésica entre el Inicio (0,0) y la Meta (24,24) es mínima a través de los bordes. El agente ha aprendido correctamente que atravesar el centro del mapa es una estrategia subóptima, ya que incrementa innecesariamente el coste por pasos acumulado. La zona central oscura indica que el agente ha descartado efectivamente esa gran área del mapa. Aunque es probable que la haya explorado en fases tempranas (por* ε*-greedy), los valores Q nunca convergieron a cifras altas porque cualquier ruta que pase por el centro es más costosa que la ruta perimetral/toroidal. El mapa de calor valida que el agente no solo ha aprendido la ubicación de la meta, sino que ha interiorizado la física no euclídea del entorno (toro), optimizando su política para ignorar más del 50% del mapa (el centro) que resulta irrelevante para la solución óptima.*

Figura A.7: Frames del Episodio Final (v3)

Entorno: VolcanoLake (25x25) | Acciones: 8



Leyenda para Fig. A.7

Color	*Significado*
Verde	*Casilla de Salida*
Marrón	*Casilla de Tierra*
Azul	*Casilla de Agua*
Rojo	*Casilla de Lava*
Dorado	*Casilla de Meta*

Descripción: La secuencia muestra la ejecución de la política óptima. Se observa cómo el agente solamente da un único paso. El agente ejecuta la acción Arriba-Izquierda (Diagonal). Debido a la conectividad del TorusWrapper, este movimiento desde el origen (0,0) provoca un desbordamiento negativo en ambos ejes, transportando al agente directamente a la coordenada (N-1, N-1), correspondiente a la Meta.