Alumnos: Emiliano Martino, Juan Pablo Hagata y Lara Rosenberg
Repositorio GitHub: https://github.com/emimemos/TP_AR

TRABAJO PRACTICO APRENDIZAJE POR REFUERZO

Introducción

En este trabajo se presenta una implementación basada en la técnica de aprendizaje por refuerzo conocida como Q-learning, aplicada a un entorno simulado de jardinería automatizada.

El objetivo principal de este proyecto fue entrenar a un robot inteligente que sea capaz de gestionar de manera eficiente el riego de un jardín de plantas dispuestas en una cuadrícula de 5x5, maximizando su rendimiento mediante la toma de decisiones informadas. Para lograrlo, el agente (robot) aprende a interactuar con el entorno tomando acciones que afectarán el estado de las plantas, que se representan principalmente a través de su nivel de humedad.

A través de múltiples episodios, el robot explora y explota el entorno con el fin de identificar las mejores estrategias de riego y movimiento. Utilizando el algoritmo de Q-learning, el robot ajusta su comportamiento progresivamente para optimizar su política de acción, es decir, aprende cuándo y dónde moverse o regar, buscando maximizar la recompensa acumulada y mantener las plantas saludables sin agotar antes de tiempo su energía.

Este es entorno en el que se mueve el robot es dinámico: cambia con el tiempo ya que las plantas se secan aleatoriamente y con el robot cuando las riega, modifica también su humedad. Ademas, el robot tiene impuestas restricciones de energía, castigos por decisiones inadecuadas (como regar de más o no regar a tiempo), y recompensas por acciones beneficiosas (regar las plantas cuando tienen una humedad baja).

Desarrollo

1. Entorno

El entorno se modeló como una grilla de 5x5 que representa un jardín con 25 plantas. Cada celda de la grilla contiene una planta que tiene un nivel de humedad entre 0 y 10. El robot puede moverse por esta grilla y ejecutar acciones para interactuar con las plantas. Al comienzo de cada episodio, los niveles de humedad de todas las plantas se inicializan aleatoriamente con valores entre 2 y 9, para representar diferentes condiciones de riego.

A medida que el tiempo avanza (con cada paso que da el robot), un porcentaje de las plantas pierde una unidad de humedad, lo que simula el secado natural. Esta probabilidad se controla mediante un parámetro configurable (prob_secado_plantas), que para este ejercicio establecimos en 5%.

2. Acciones posibles

Alumnos: Emiliano Martino, Juan Pablo Hagata y Lara Rosenberg
Repositorio GitHub: https://github.com/emimemos/TP_AR

TRABAJO PRACTICO APRENDIZAJE POR REFUERZO

El robot tiene disponibles cinco acciones:

- Moverse hacia arriba
- Moverse hacia abajo
- Moverse hacia la izquierda
- Moverse hacia la derecha
- Regar la planta, lo cual implementa la humedad de la misma en un parámetro configurable (aumento_por_regar) que establecimos en 5 unidades.

3. Recompensas y penalizaciones

El sistema de recompensas guía al robot para que aprenda un comportamiento eficiente. Las recompensas las definimos de la siguiente forma:

- Recompensa positiva (+10) por regar una planta que lo necesita. Como el riego aumenta en 5 unidades la humedad de la planta, decidimos que esta recompensa positiva la obtenga únicamente cuando al regar, no supere la humedad máxima de la planta.
- Castigo por regar de más (cuando la humedad ya es alta). Lo definimos en -1.
- Castigo por moverse cuando la planta tiene una humedad baja (<=3) y no es regada. Este parámetro lo definimos en -10.
- Decidimos además, castigar al robot por moverse (-0.5).

Estas recompensas están diseñadas para reforzar conductas útiles y castigar conductas que generen que las plantas se sequen o que se ahoguen por demasiada agua.

4. Energía del robot

Cada acción del robot consume energía y el mismo cuenta con una energía máxima limitada (45 unidades). Una vez que se queda sin energía, el episodio termina. Sin embargo, agotar su energía no es lo único que puede hacer que un episodio termine, también puede finalizar si todas las plantas estan secas.

5. Q-learning y estados

Se utilizó el algoritmo Q-learning para permitir que el robot aprenda de la experiencia. Este algoritmo utiliza una tabla Q (Q) que guarda los valores estimados para cada combinación posible de estado y acción.

El estado está representado por una tupla: (fila, columna, nivel de humedad), donde el nivel de humedad puede ser alto (mayor o igual a 7), medio (menor o igual a 4) y o bajo (menor o igual a 3), según la humedad de la planta en la posición actual del robot.

Alumnos: Emiliano Martino, Juan Pablo Hagata y Lara Rosenberg
Repositorio GitHub: https://github.com/emimemos/TP_AR

TRABAJO PRACTICO APRENDIZAJE POR REFUERZO

6. Selección de acciones

El robot selecciona acciones usando una estrategia epsilon-greedy. Con una pequeña probabilidad ε (por ejemplo, 0.01), elige una acción aleatoria para explorar. El resto del tiempo, elige la acción con mayor valor Q (explotación). Esto ayuda a que el robot aprenda un balance entre explorar nuevas posibilidades y usar lo que ya sabe.

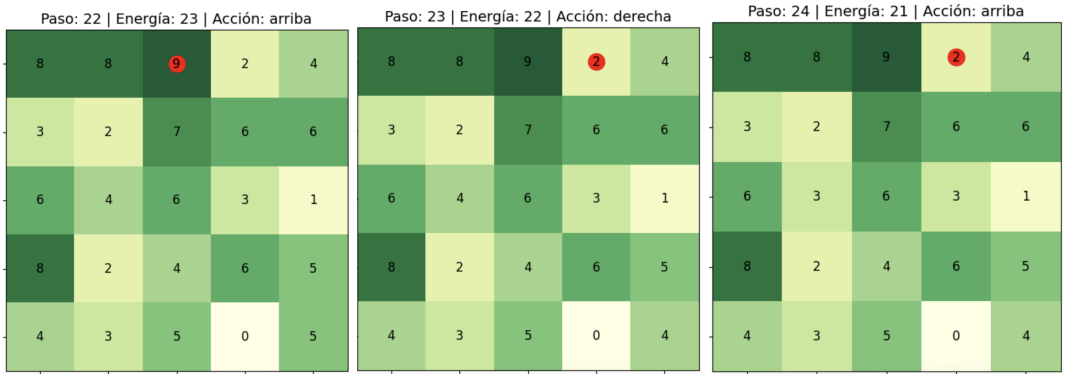
7. Entrenamiento y visualización

El entrenamiento del robot se realiza a lo largo de varios episodios (en este caso particular definimos 1000). Al final de cada episodio, guardamos la recompensa acumulada para poder analizar la convergencia del aprendizaje y graficamos los resultados con una curva de recompensa por episodio y una media móvil para observar tendencias a lo largo del tiempo.

También se desarrolló una visualización del entorno paso a paso, para poder observar cómo se mueve el robot, qué decisiones toma, y cómo evoluciona la humedad de las plantas.

Resultados y conclusiones

El primer paso fue visualizar como se veía la evolución de un episodio, para validar que todo el sistema de recompensas estuviese funcionando correctamente. A modo de ejemplo:

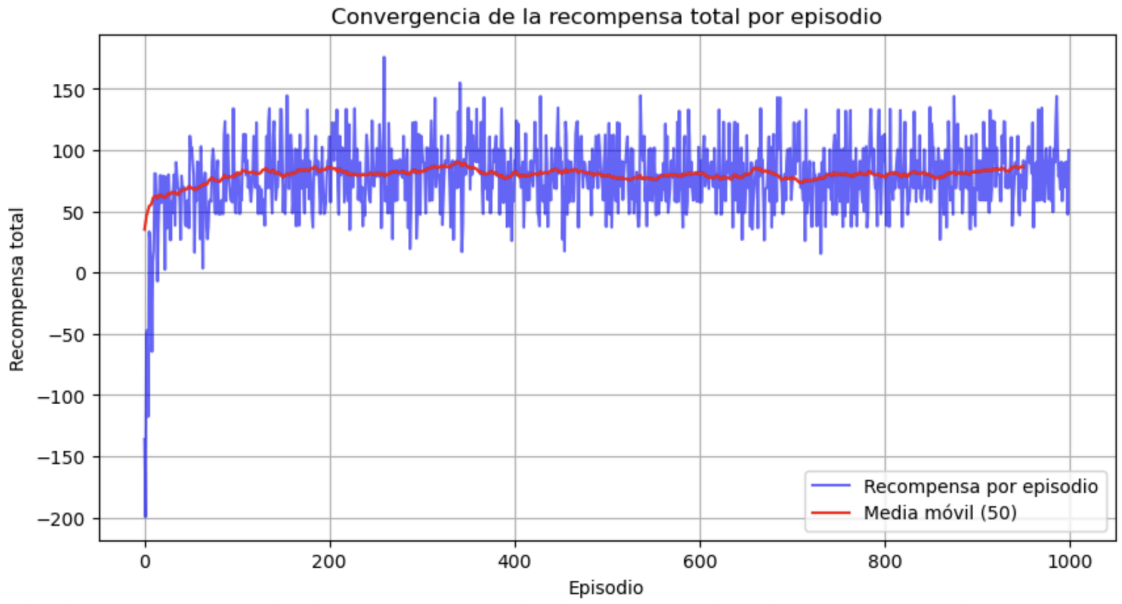


Como podemos ver, en el paso 22 el robot optó por ir hacia arriba quedando en una planta con humedad alta y partiendo de una planta con humedad alta, obteniendo una recompensa de -0.5. En el paso 23, el robot elige ir a la derecha, con lo cual también obtiene una recompensa de -0.5 ya que la planta anterior tenía una humedad de 9. En el paso 24, el robot opta por ir hacia arriba pero esa planta ya tenía una humedad muy baja, con lo cual recibe una recompensa de -10.
Sumado a esto, podemos observar que en el cuadrante inferior izquierdo, entre el paso 22 y el 23, la humedad de la planta disminuyó de 5 a 4.

Alumnos: Emiliano Martino, Juan Pablo Hagata y Lara Rosenberg
Repositorio GitHub: https://github.com/emimemos/TP_AR

TRABAJO PRACTICO APRENDIZAJE POR REFUERZO

Como segundo paso, lo que hicimos fue entrenar al robot en 1000 episodios, obteniendo el siguiente grafico de convergencia:



Lo que notamos es que en los primeros episodios el robot obtiene recompensas negativas pero luego, pasa a obtener recompensas positivas. Sin embargo, notamos que se estabiliza muy rápido la recompensa del episodio, lo que nos lleva a considerar que no son necesarias tantas iteraciones.
A su vez, revisamos la tabla Q final obtenida para ver si los resultados tienen sentido:

Fila	Columna	Humedad	Acciones (Q-values)
0	0	alta	arriba (-0.11) abajo (-0.11) izquierda (-0.11) **derecha (0.23)** regar (-0.12)
0	0	baja	arriba (-0.19) abajo (-0.10) izquierda (-0.10) derecha (-0.10) **regar (2.39)**
0	0	media	arriba (-0.01) abajo (-0.01) izquierda (-0.01) derecha (-0.01) **regar (1.57)**

Por ejemplo, para la primer planta, si la humedad de la misma es alta, su mejor acción es ir hacia la derecha. En cambio, si la humedad es baja, su mejor acción está siendo regar la planta, lo cual tiene sentido.

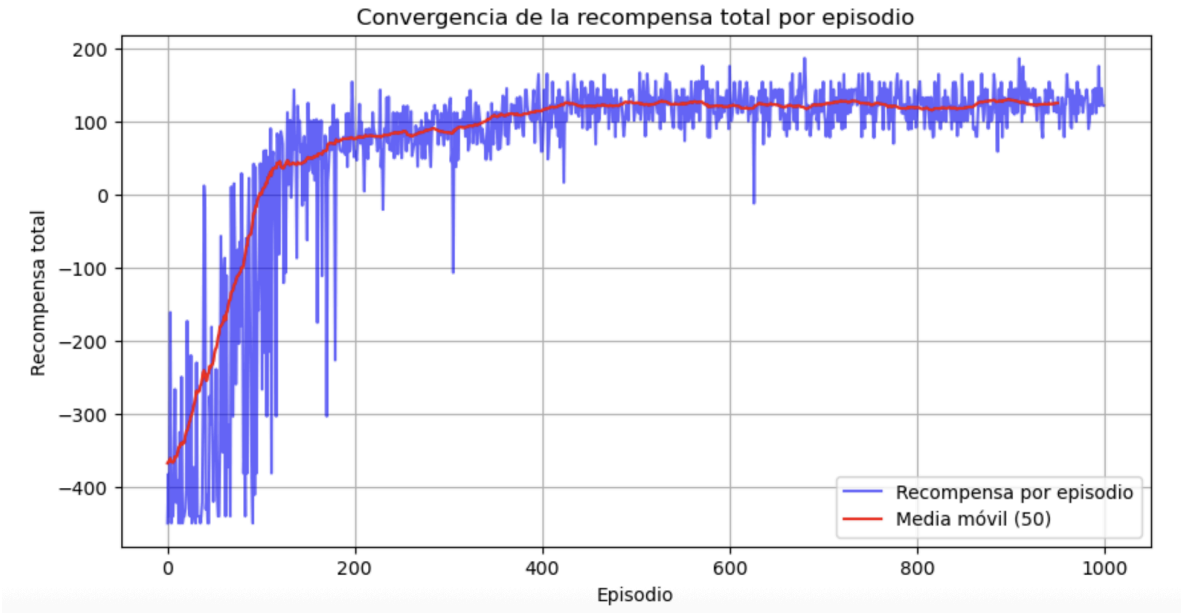
Alumnos: Emiliano Martino, Juan Pablo Hagata y Lara Rosenberg
Repositorio GitHub: https://github.com/emimemos/TP_AR

TRABAJO PRACTICO APRENDIZAJE POR REFUERZO

Como siguiente paso, realizamos la misma visualización que teníamos al comienzo de un solo episodio, pero utilizando la tabla Q que generamos y pudimos notar que el comportamiento del robot mejoraba sustancialmente respecto a la primera iteración (donde la tabla Q estaba vacía).

Posteriormente, realizamos pruebas modificando el parámetro Alpha y modificando el épsilon de forma tal que al comienzo explore más y sobre el final explote más. Sin embargo, no observamos diferencias sustanciales respecto a la primera versión.

Lo que observamos que sucedía en todos los casos, es que el robot parecía moverse solo por ciertos espacios de la grilla, a la cual no terminaba recorriendo de forma completa y dejaba que plantas que se encontraban lejos del sector de recorrido, se murieran. Esto consideramos que sucede porque el robot no tenía la visión de que hay plantas que se están secando a su alrededor, solo tenía en cuenta su posición y el nivel de humedad de la planta actual. Esta información no le daba suficiente contexto para saber si conviene moverse, ni hacia dónde.
Para atacar este problema, propusimos lo siguiente:
- Extender el estado incluyendo la cantidad de plantas secas en las celdas vecinas.
- Agregar una bonificación temporal en la elección de acción para guiar al robot hacia zonas más secas.
Lo que buscamos con esto es priorizar las acciones que lo acerquen a celdas vecinas con plantas secas **(**humedad baja).
Como resultado de esta implementación, observamos un mejor recorrido del robot, pero no ideal.



Adicionalmente, realizamos pruebas forzando el riego cuando la humedad de la planta era baja y penalizando si el robot se salía de la grilla, pero no obtuvimos resultados sustancialmente mejores.