Proyecto - Aprendizaje por refuerzo
Control con DQN sobre tanque de agua

Emmanuel Peto Gutiérrez

Junio 2023

1. El modelo matemático

Se modeló un sistema dinámico de un tanque de agua con una bomba de entrada y un agujero de salida, el cual se obtuvo de [1] (ver Figura 1). La ecuación diferencial que define cómo cambia la altura del nivel de agua respecto al tiempo es (página 97)

$$\frac{dH}{dt} = -(\frac{A_2}{A_1}\sqrt{2g})\sqrt{H} + \frac{1}{\rho A_1}Q_1$$

El significado y valor de cada variable se muestran en la siguiente tabla

Variable	**significado**	**Valor**
A_1	área del tanque	$\pi/4m^2$
A_2	área del agujero	$\pi/400m^2$
ρ	densidad del agua	$1000kg/m^3$
Q_1	flujo de entrada	
H	nivel del agua	

Se define H^* como la altura de equilibrio del tanque (o altura deseada). Para este caso, $H^* = 1$. El flujo de entrada para que se dé el punto de equilibrio es $Q_1^* = 34.77L/s$.

Dadas esas variables y constantes, se obtiene la función de transferencia del tanque (página 99)

$$G(s) = \frac{0.0013}{s + 0.0221}$$

El sistema se discretizó usando la función `c2d` de Matlab, con un periodo de muestreo de 0.2s y se obtuvo la siguiente función de transferencia

$$G(z) = \frac{2.5409 \times 10^{-4}}{z - 0.9956}$$

Dada esta ecuación, se obtuvo la función de la altura del tanque en el instante i, dada la señal de entrada u y dada la altura en el instante anterior $h^{(i-1)}$:

$$h^{(i)} = -den(1) * h^{(i-1)} + num(1) * u$$

Donde

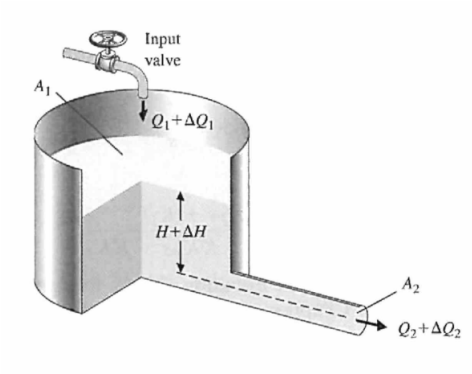


Figura 1: Imagen de tanque de agua tomada de *Modern Control Systems*.

- den(1) = -0.9956
- num(1) = 2.5409×10^{-4}

2. Implementación

Se reprodujeron resultados de [3]. Se utilizó el algoritmo DQN para encontrar la política. Se usa un buffer de repetición para obtener lotes y calcular la aproximación de $Q(s, a)$. La aproximación se realiza con una red neuronal. Para elegir la acción dado un estado se utiliza ϵ-greedy, donde la ϵ se conserva constante en 1 durante la mitad de los episodios (para priorizar la exploración) y luego empieza a decrementar de manera lineal hasta llegar a 0.01 (para priorizar la explotación). Se utiliza la biblioteca `PyTorch` para redes neuronales.

La idea del algoritmo DQN se obtuvo de [2].

3. Descripción del MDP

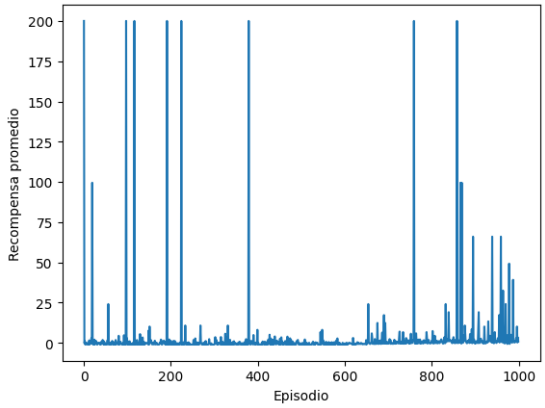
- **Estados**: $s \in [0, 2]$ (continuo)
- **Acciones**: $a \in [0, 50]$ (continuo)
- **Probabilidades**: $p(s', r|s, a) = 1$, donde $s' = 0.9956s + 2.5409 \times 10^{-4}a$.
- **Recompensa**: La recompensa es -1 por llegar a un estado no terminal y 200 por llegar a un estado terminal. Un estado terminal se alcanza cuando la altura actual del tanque se acerca a la altura objetivo con un margen de 0.01m.
- **Descuento**: $\gamma = 0.99$

Los hiperparámetros utilizados son:

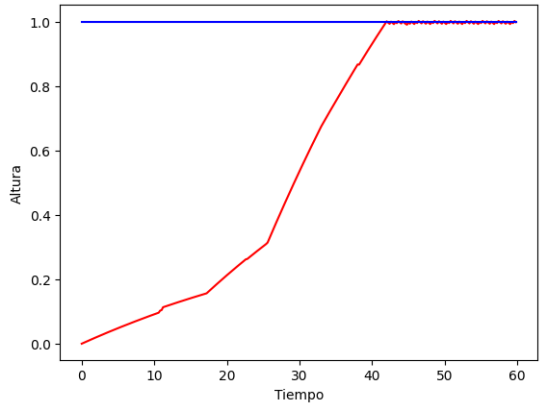
- α: 1×10^{-4}
- ϵ: valor decreciente de 1 a 0.01
- Tamaño de batch: 32
- Episodios: 1000
- Pasos (máximos) por episodio: 1000

4. Resultados

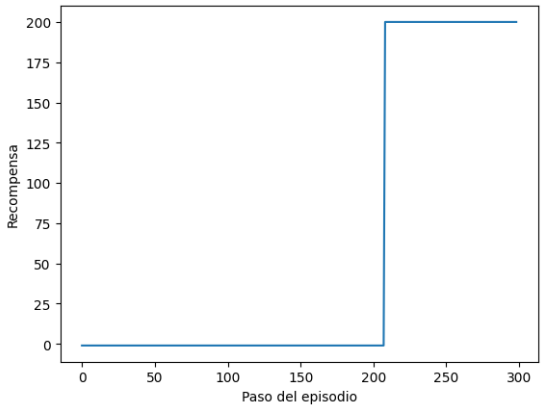
En la siguiente gráfica se muestra la recompensa media en función del número del episodio, en la fase de entrenamiento.



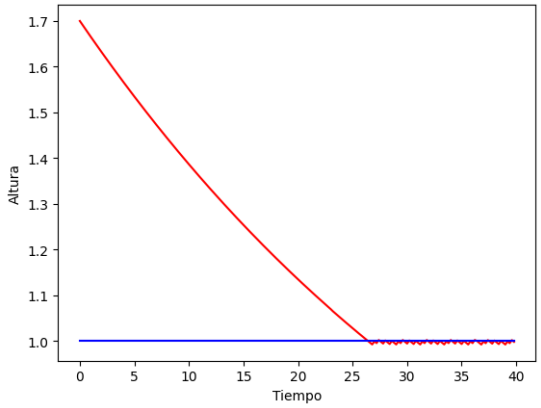
Una vez que se ha entrenado el modelo se puede aplicar control. En la siguiente gráfica se muestra el control sobre la altura del agua cuando la altura inicial es 0 y la altura objetivo es 1. Se puede observar que la altura objetivo se alcanza en aproximadamente 40s.



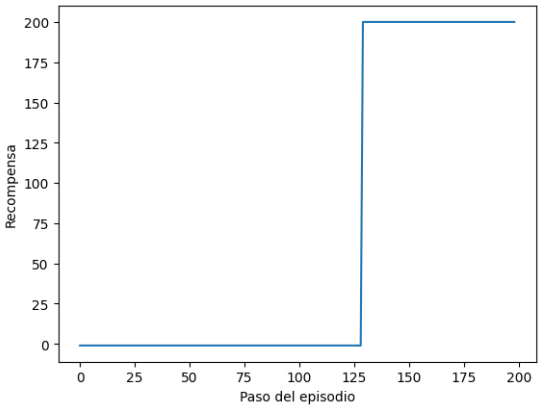
La recompensa por paso al seguir esta política se muestra en la siguiente gráfica.



El ejemplo anterior es cuando la altura objetivo (1) está por arriba de la altura inicial (0). En la siguiente gráfica se muestan los resultados cuando la altura inicial (1.7) está por arriba del objetivo (1). Se observa que se alcanza la altura objetivo en aproximadamente 27s.



La recompensa por paso al seguir esta política es la siguiente.



5. Ejecución

Los resultados de la ejecución se encuentran en el notebook `dqn_tanque.ipynb`. Se requiere el script `TanqueDisc.py` que contiene el ambiente. La fase de entrenamiento (función `qlearning()` del notebook) toma aproximadamente 40 minutos en ejecutarse.

Referencias

[1] Richard C. Dorf y Robert H. Bishop. *Modern Control Systems*. 12.ª ed. Prentice Hall, 2011.

[2] Maxim Lapan. *Deep reinforcement learning: Hands-on*. 2.ª ed. Packt, 2020.

[3] Markel Sanz. *Introducción al aprendizaje por refuerzo. Parte 3: Q-Learning con redes neuronales, algoritmo DQN*. 2020. URL: `https://markelsanz14.medium.com/introducci%5C%C3%5C%B3n-al-aprendizaje-por-refuerzo-parte-3-q-learning-con-redes-neuronales-algoritmo-dqn-bfe02b37017f#:~:text=%5C%5B1%5C%5D%5C%20inventaron%5C%20el%5C%20algoritmo%5C%20Deep,de%5C%20aproximar%5C%20funciones%5C%20no%5C%20lineales.`.