Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación



IIC2613 - Inteligencia Artificial

Q-Learning

Hans Löbel
Dpto. Ingeniería de Transporte y Logística
Dpto. Ciencia de la Computación

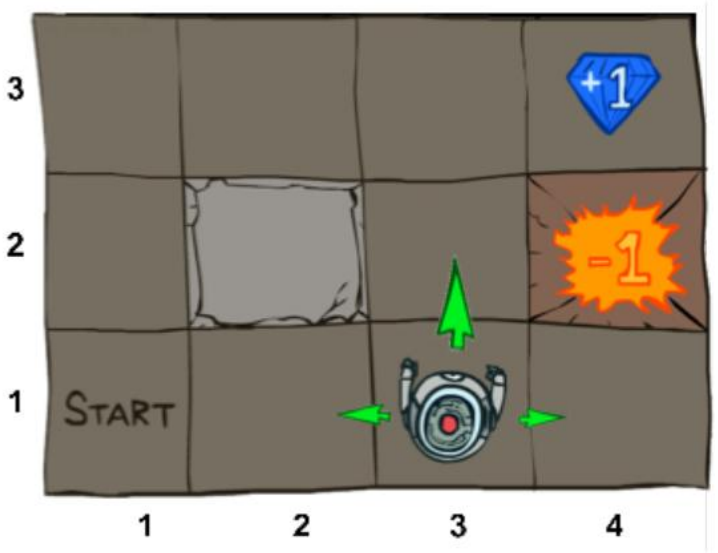
Una manera de enfrentar el problema de toma de decisiones es moverse siempre entre “buenos estados” del mundo



La idea de “buenos estados” se puede formalizar a través de la función de valor

$V^*(s)$= suma de las recompensas con descuento al empezar en el estado s y actuar óptimamente

$$V^*(s) = \max_{\pi} \mathbb{E}\left[\sum_{t=0}^{H} \gamma^t R(s_t, a_t, s_{t+1}) \mid \pi, s_0 = s\right]$$



Supongamos acciones con $P = 0{,}8$, $\gamma = 0{,}9$, $H = 100$

$V^*(4,3) =$

$V^*(3,3) =$

$V^*(2,3) =$

$V^*(1,1) =$

$V^*(4,2) =$

La función de valor puede estimarse mediante el algoritmo *Value Iteration*

Start with $V_0^*(s) = 0$ for all s.

For k = 1, … , H:

For all states s in S:

$$V_k^*(s) \leftarrow \max_a \sum_{s'} P(s'|s,a) \left(R(s,a,s') + \gamma V_{k-1}^*(s')\right)$$

$$\pi_k^*(s) \leftarrow \arg\max_a \sum_{s'} P(s'|s,a) \left(R(s,a,s') + \gamma V_{k-1}^*(s')\right)$$

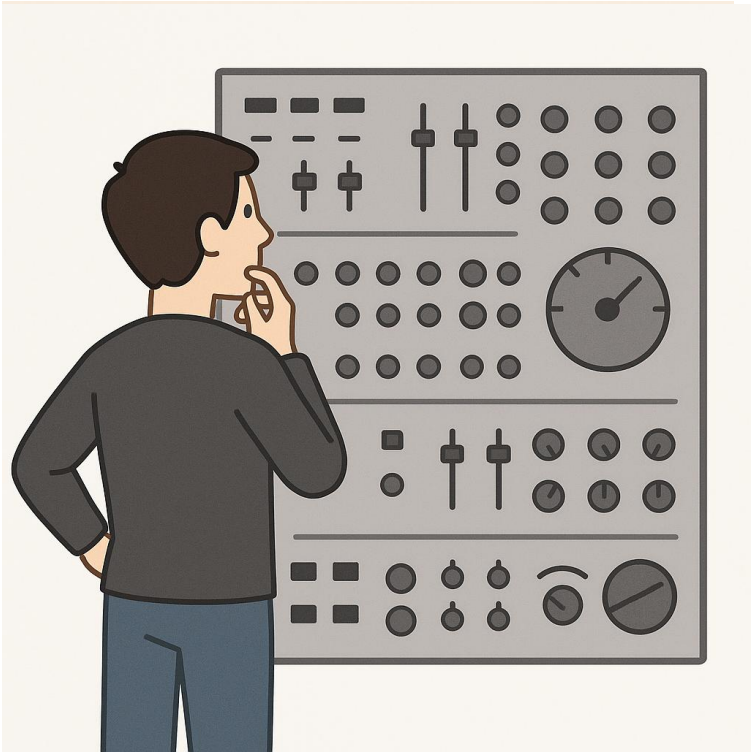
Es posible demostrar que este algoritmo converge a la función de valor óptima $V^*(s)$, la cual satisface las ecuaciones de Bellman:

$$\forall\, s \in S: \quad V^*(s) = \max_a \sum_{s'} P(s'|s,a)\left[R(s,a,s') + \gamma V^*(s')\right]$$

¿Qué limitaciones tiene un esquema basado en la estimación de la función de valor?

$$V_k^*(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)\left(R(s,a,s') + \gamma V_{k-1}^*(s')\right)$$

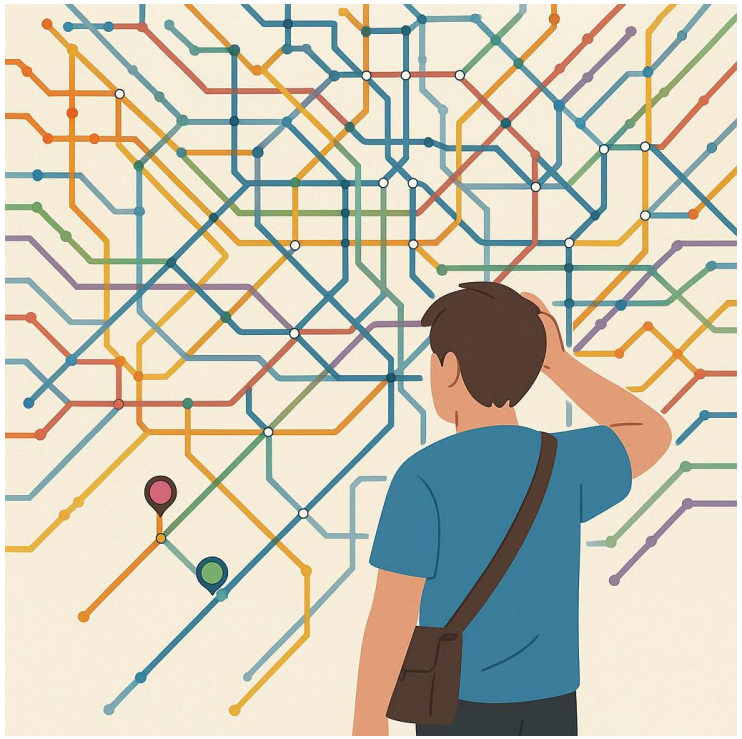
- El algoritmo necesita conocer las probabilidades de transición $P(s'|s,a)$.
- Esto significa que asume que es posible construir un modelo completo del mundo.
- Un efecto de esto es que no puede ser usado en situaciones donde se debe aprender explorando.



¿Qué pasa con entornos de gran tamaño?

$$V_k^*(s) \leftarrow \max_a \sum_{s'} P(s'|s,a) \left(R(s,a,s') + \gamma V_{k-1}^*(s')\right)$$

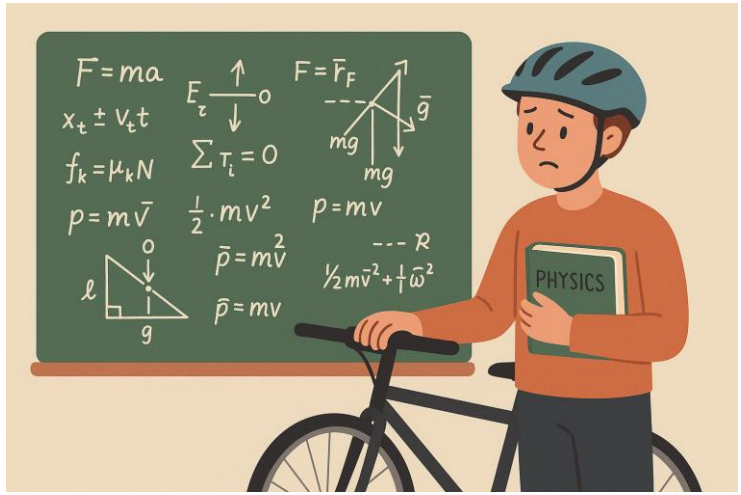
- Value Iteration calcula y actualiza valores para todos los estados, independiente de si es posible o no alcanzarlos.
- Costo computacional $O(|S|^2 \cdot |A|)$, que se vuelve prohibitivo.
- Requiere simplificación del entorno o discretización agresiva.



¿Se puede aprender y actuar al mismo tiempo?

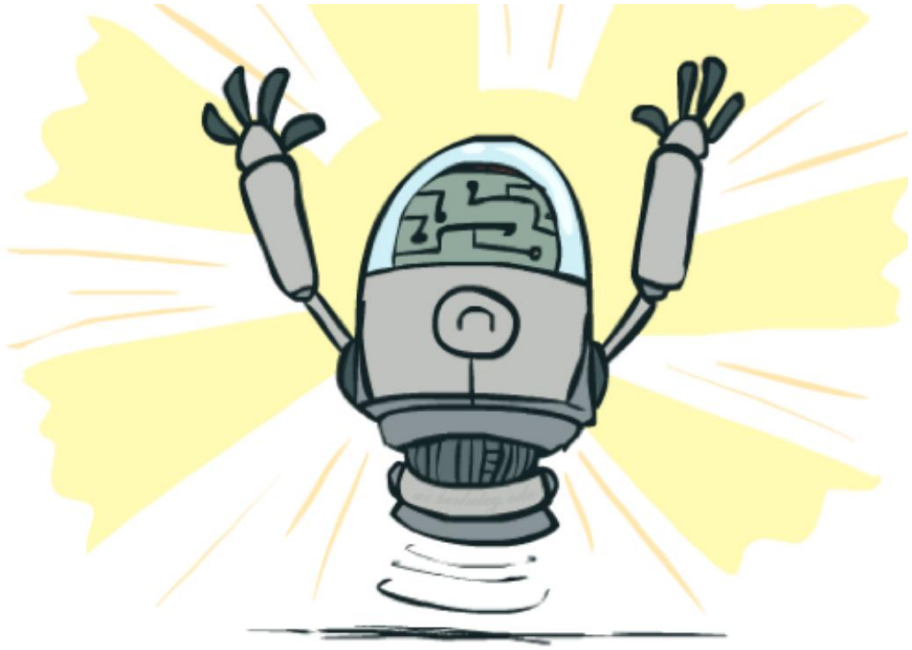
$$V_k^*(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)\left(R(s,a,s') + \gamma V_{k-1}^*(s')\right)$$

- Value Iteration necesita converger antes de usarse (es un algoritmo offline).
- Esto implica que no permite “aprender haciendo”.
- No se adapta a entornos cambiantes o desconocidos.



No todo está perdido con Value Iteration

- Nuestro siguiente esquema buscará superar estas limitaciones, moviéndonos a un esquema online.
- Aprenderemos la política interactuando con el entorno, sin necesidad de conocerlo previamente.
- Lo mejor de todo, es que podemos partir del mismo *good old* Value Iteration.



Podemos desagregar la función de valor con respecto a las acciones y ahora estimar la función Q

Bellman Equation:

$$Q^*(s,a) = \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma \max_{a'} Q^*(s',a'))$$

Q-Value Iteration:

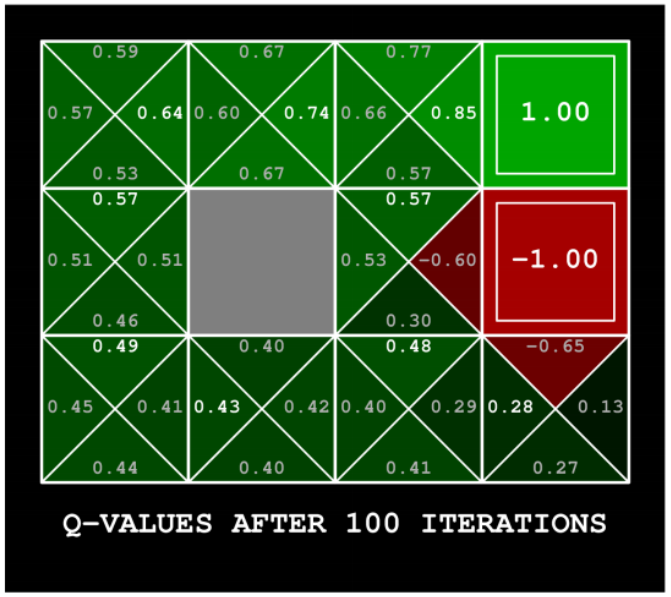
$$Q_{k+1}(s,a) \leftarrow \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma \max_{a'} Q_k(s',a'))$$

- La función $Q^*(s,a)$ estima la utilidad esperada partiendo desde s, tomando la acción a y luego actuando óptimamente
- No es evidente en este momento por qué esto sirve de algo

Podemos refinar la función de valor y estimar la función Q

$$Q^*_{k+1}(s,a) \leftarrow \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma \max_{a'} Q^*_k(s',a'))$$

k = 100



Noise = 0.2
Discount = 0.9

Q-Learning al rescate

- La función Q nos entrega la solución a los problemas anteriores.
- Al desacoplar la optimalidad del estado y la acción, es posible utilizar un enfoque de muestreo, cambiando las transiciones por un valor esperado:

$$Q_{k+1}(s,a) \leftarrow \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma \max_{a'} Q_k(s',a'))$$



$$Q_{k+1} \leftarrow \mathbb{E}_{s' \sim P(s'|s,a)} \left[R(s,a,s') + \gamma \max_{a'} Q_k(s',a') \right]$$

Q-Learning al rescate

Start with $Q_0(s, a)$ for all s, a.

Get initial state s

For k = 1, 2, … till convergence

Sample action a, get next state s'

If s' is terminal:

$$\text{target} = R(s, a, s')$$

Sample new initial state s'

else:

$$\text{target} = R(s, a, s') + \gamma \max_{a'} Q_k(s', a')$$

$$Q_{k+1}(s, a) \leftarrow (1 - \alpha) Q_k(s, a) + \alpha \, [\text{target}]$$

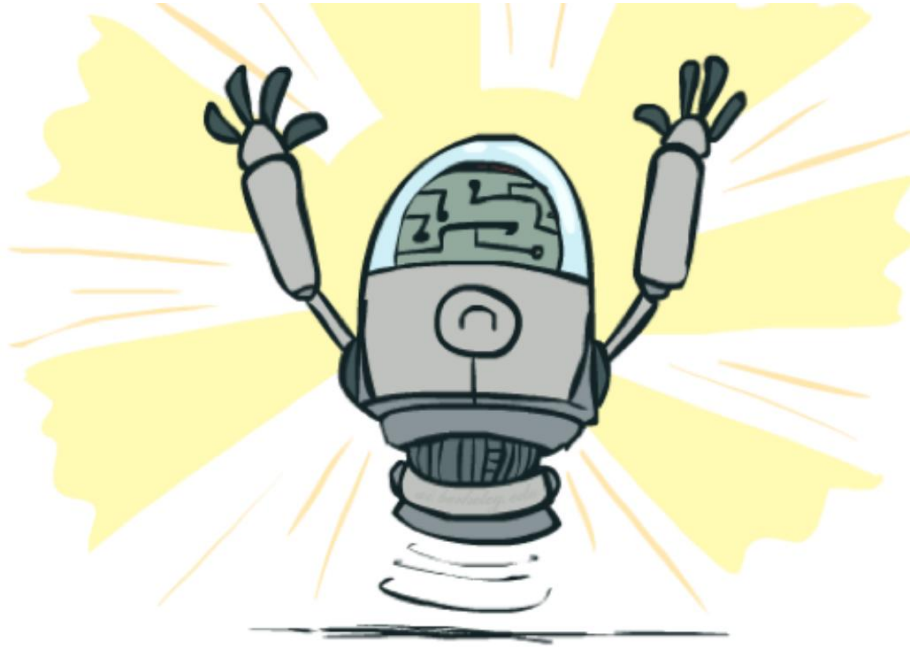
$$s \leftarrow s'$$

La pregunta es ahora cómo muestrear

- Esta es la madre de todas las batallas: exploration vs exploitation
- En otras palabras, ¿elijo la acción optima de acuerdo a $Q_k(s,a)$, o busco nuevas posibilidades?
- En la práctica se utiliza una técnica mixta, ϵ-Greedy: acción al azar con probabilidad ϵ, en otro caso, acción que maximiza $Q_k(s,a)$.

Propiedades de Q-learning

- El principal resultado de Q-Learning es que converge a una política óptima, a pesar de actuar de manera subóptima.
- En otras palabras, desacopla exploración de optimización: off-policy learning.
- Requiere mucha exploración y ajustes cuidadosos del learning rate, pero funciona.





https://youtu.be/AMnW-OsOcl8

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación



IIC2613 - Inteligencia Artificial

Q-Learning

Hans Löbel
Dpto. Ingeniería de Transporte y Logística
Dpto. Ciencia de la Computación