

Reinforcement Learning por medio de Q learning

Aarón Sonabend

Department of Biostatistics, Harvard University

Agosto 28, 2019

Reunión Internacional de Inteligencia Artificial y sus Aplicaciones

El éxito de *Reinforcement Learning*

Reinforcement Learning ha tenido éxito en diferentes ámbitos

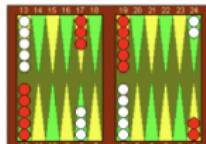
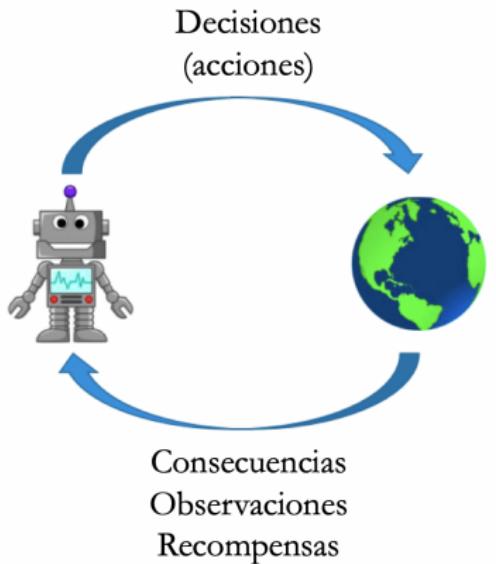
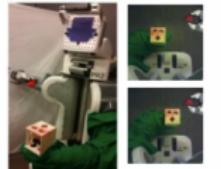
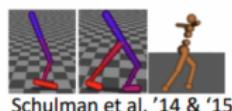
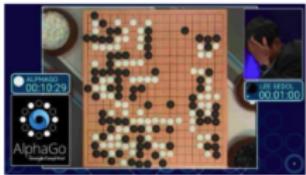
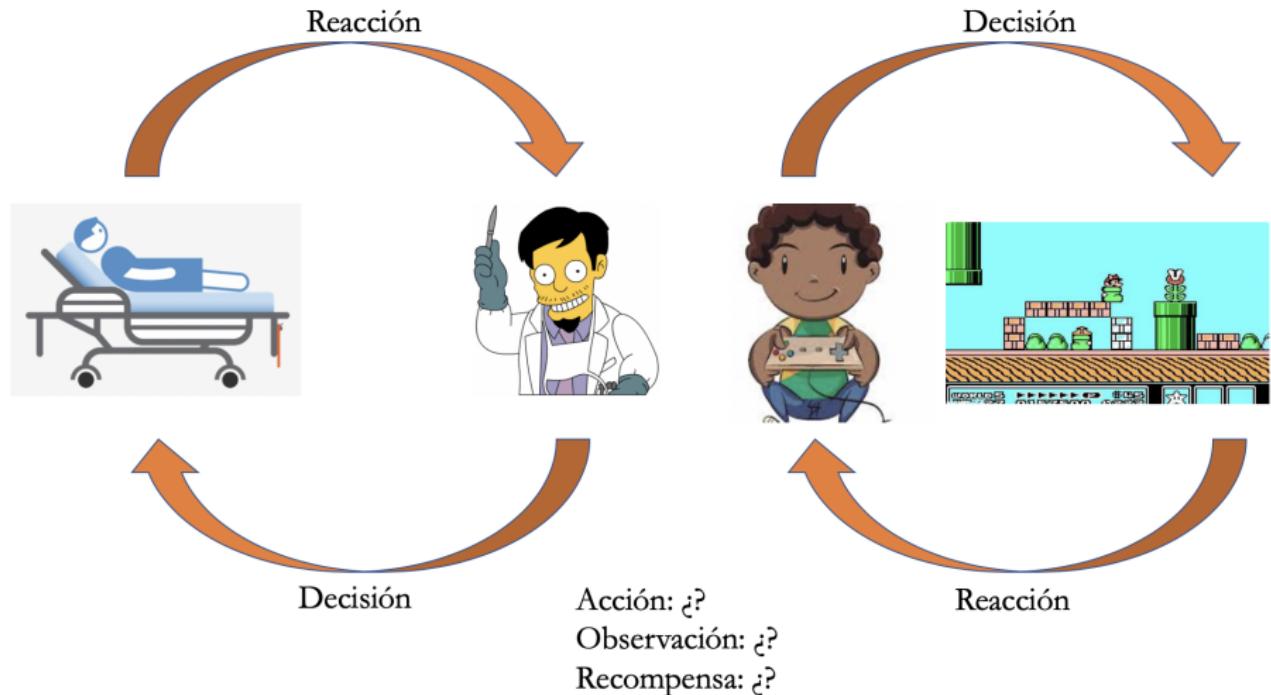


Figure 2. An illustration of the normal opening position in backgammon. TD-Gammon has sparked near-universal concern in the way experts play certain openings. For example, with an opening of 13-14, TD-Gammon's players have now switched from the traditional move of 13-9, 6-5, to TD-Gammon's preference, 13-9, 24-23. TD-Gammon's analysis is given in Table 2.



De Berkeley's CS 294: Deep Reinforcement Learning
por John Schulman & Pieter Abbeel

Reinforcement Learning: Contexto



Reinforcement Learning: Contexto

Reinforcement Learning consiste en encontrar la mejor tayectoria de decisiones para maximizar la recompensa

- No hay datos de entrenamiento con etiquetas: las decisiones se basan en la experiencia previa
- Con base en prueba y error, el algoritmo experimenta con diferentes acciones para aprender como maximizar la recompensa

Optimizando la trayectoria de decisiones

Considren el siguiente escenario para un niño:

- Cada hora: ¿Quieres un dulce, dos, o nada?
- Un dulce: las ofertas se mantienen constantes la siguiente hora
- Dos dulces: una hora después no se ofrece nada
- Nada: una hora después las ofertas se duplican (dos dulces, 4 dulces o nada)
- Escoger el máximo número de dulces en la quinta hora le quita todo

Este juego se repite cada día y se juega durante 5 horas. El niño no sabe las reglas, solo escucha las preguntas ¿Cuál es la mejor estrategia?



Exploración vs. aprovechamiento

- Escoger dos cada vez que es posible es una mala idea ¿Con cuántos acaba el juego?
- Un poco de exploración en las opciones lleva al niño a darse cuenta que tantos dulces puede conseguir ¿Qué tanta exploración?
- Hay un dilema de que tanto explorar vs. cuando empezar a aprovechar el conocimiento ya obtenido
- Una estrategia común es aprovechar la mayoría de las veces, pero de vez en cuando explorar un poco,
- Ésta se llama ϵ -greedy:
 - Con probabilidad ϵ se escoge una acción aleatoria
 - Con probabilidad $1 - \epsilon$ se escoge la mejor acción posible
- ¿Qué problema tiene ésta estrategia?

Formalizando el contexto

- Consideren un proceso de decisión de Markov caracterizado por lo siguiente:

$$\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T} \rangle$$

- Un conjunto de estados \mathcal{S} (finito o infinito)
- Un conjunto de acciones \mathcal{A} (finito)
- Una función de transición $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$
- Una función de recompensa $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{R}$
- Propiedad de falta de memoria: conociendo el estado presente, el pasado es irrelevante para predecir el futuro

¿Programación dinámica?

¿Por qué no simplemente resolvemos el problema usando programación dinámica?

- No conocemos las funciones de recompensa ni de transición

$$\mathcal{R}(s, a) \rightarrow r \text{ or } \mathcal{T}(s, a) \rightarrow s' , (\mathcal{T} \in \mathbb{R}^{|S| \times |S| \times |\mathcal{A}|})$$

- Generalmente involucran relaciones de causalidad compleja (i.e. en medicina)

Solo observamos datos de la forma (s_t, a_t, r_t, s_{t+1})

Función de estrategia π

¿Cuál es la mejor decisión en cada estado? → La que obtenga la mayor cantidad de azúcar!

- La meta es encontrar la estrategia de acciones que genere la recompensa más alta a lo largo del trayecto,

$$R = \sum_{t=0}^{\infty} \gamma^t r_t$$

con $r_t = \mathcal{R}(S_t, A_t)$

- Esto se logra estimando la función de estrategia $\pi : \mathcal{S} \rightarrow \mathcal{A}$ que nos dice que acción tomar en cada posible estado



Estimando π : Q-learning

Q -learning usa la experiencia para encontrar la mejor estrategia:

- Estrategia basada en la función de acción-valor:

$$Q : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{R}$$

- Define el valor de un estado s_t como

$$V^\pi(s) = \mathbb{E}_{\pi, \mathcal{T}} \left[\sum_{t'=t}^{\infty} \gamma^{t'} r(s_{t'}, a_{t'}) | s_t = s \right]$$

- ① ¿Qué acción se toma en $t' = t + 2$?
- ② ¿Por qué hay un promedio involucrado? ($\mathbb{E}_{\pi, \mathcal{T}}$)

Función óptima de estrategia π^*

- Queremos la función de estrategia π^* tal que:

$$\pi^* = \arg \max_{\pi} V^{\pi}(s)$$

- Supongamos que ya conocemos π^* , empezando por s , el máximo valor posible es

$$V^*(s) = \max_{\pi} V^{\pi}(s)$$

Función de acción-valor

La función de acción-valor se define usando la ecuación de Bellman:

$$Q(s, a) = \mathbb{E}_{s^{t+1} \sim \mathcal{T}(s^{t+1} | s_t, a_t)} [r(s_t, a_t) + \gamma V^*(s^{t+1}) | s_t = s, a_t = a]$$

Nos encontramos en el estado s y ejecutamos la acción a , procedemos óptimamente a partir del siguiente estado

- ¿Para un solo estado s cuántos valores Q tenemos?
- ¿Sobre qué estamos promediando? ¿Sobre qué estamos condicionando?
- ¿Qué obtenemos con $Q(s, \pi^*(s))$?
- ¿Qué obtenemos con $\max_a Q(s, a)$?

Encontrando Q : con base en tablas

$Q(s_t, a_t)$

Estado\Acción	Nada	1 dulce	2 dulces
1	1	1	1
2	1	1	1
...			
24	1	1	1

$$Q(s_t, a_t)^{(i+1)} \leftarrow (1 - \alpha)Q(s_t, a_t)^{(i)} + \alpha \left(r_t + \gamma \max_a Q(s_{t+1}, a)^{(i)} \right)$$

$Q(s_t, a_t)$

Estado\Acción	Nada	1 dulce	2 dulces
1	20	10	.5
2	30	13	3
...			
24	15	8	20

Deep Q learning

¿Qué pasa en contextos más complejos como estados continuos?

Deep Q learning

¿Qué pasa en contextos más complejos como estados continuos?

- Podemos parametrizar la función Q_θ con ayuda de deep learning y estimar θ :

- 1 Tomamos una acción a_i y observamos (s_i, a_i, s'_i, r_i)
- 2 Definimos $y_i = r_i + \gamma \max_a Q_\theta(s'_i, a)$
- 3 Actualizamos los parámetros:

$$\theta \leftarrow \theta - \alpha \sum_j \frac{\partial Q_\theta(s_j, a_j)}{\partial \theta} \left(Q_\theta(s_j, a_j) - y_j \right)$$

¿Por qué va a fallar nuestra estrategia?

Deep Q learning

¿Qué pasa en contextos más complejos como estados continuos?

- Podemos parametrizar la función Q_θ con ayuda de deep learning y estimar θ :

- 1 Tomamos una acción a_i y observamos (s_i, a_i, s'_i, r_i)
- 2 Definimos $y_i = r_i + \gamma \max_a Q_\theta(s'_i, a)$
- 3 Actualizamos los parámetros:

$$\theta \leftarrow \theta - \alpha \sum_j \frac{\partial Q_\theta(s_j, a_j)}{\partial \theta} \left(Q_\theta(s_j, a_j) - y_j \right)$$

¿Por qué va a fallar nuestra estrategia?

$$\theta \leftarrow \theta - \alpha \sum_j \frac{\partial Q_\theta(s_j, a_j)}{\partial \theta} \left(Q_\theta(s_j, a_j) - (r_i + \gamma \max_a Q_\theta(s'_i, a)) \right)$$

- La secuencia de estados está fuertemente correlacionada
- El valor y_j va a cambiar en cada iteración

Encontrando Q : Classic Deep Q learning

Reducción de la pérdida: Descenso de gradiente estocástico

- 0 Empezar con un repositorio \mathcal{B} de experiencia previa, inicializar dos redes una de estrategia: Q_θ y una de etiqueta: $Q_{\theta'}$
- 1 Tomar acción a_i y observar (s_i, a_i, s'_i, r_i) , añadir a \mathcal{B}
- 2 Tomar una muestra de observaciones $\{(s_j, a_j, s'_j, r_j)\}$ de \mathcal{B}
- 3 Definir la "etiquetas" $y_j = r_j + \gamma \max_a Q_{\theta'}(s'_j, a)$
- 4 Actualizar los parámetros:

$$\theta \rightarrow \theta - \alpha \sum_j \frac{\partial Q_\theta(s_j, a_j)}{\partial \theta} \left(Q_\theta(s_j, a_j) - y_j \right)$$

- 5 Actualizar $\theta' \leftarrow \theta$ cada N pasos

Material para seguir aprendiendo

- ① El libro de Richard Sutton, Reinforcement Learning: An Introduction
https://people.inf.elte.hu/lorincz/Files/RL_2006/SuttonBook.pdf
- ② Clase de la universidad de Berkeley CS 294, Deep Reinforcement Learning:
<http://rail.eecs.berkeley.edu/deeprlcourse/>
- ③ Reinforcement Learning course de UCL:
<http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html>