Reinforcement Learning (RL)

Intro Deep Q-Learning

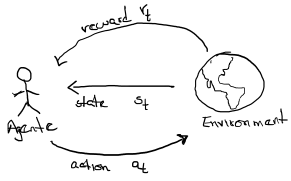
RL: Enfoque matemático para aprender toma de decisiones mediante ensayos

No pensamos en términos $f: x \to y$ ✗

Sí pensamos en términos $f: \max_{\pi} \pi(R|S \times A)$, donde π : policy

R: recompensa a largo plazo

r_t: recompensa inmediata en el tiempo t



Terminología:

- Markov decision process (MDP)
- Q learning
- Policy gradient
- Deep Q Network (DQN)
- Bellman Equation
- Exploration vs explotation
- Ambiente
- Agente a_t
- Estado s_t
- Acción
- Recompensa
- Episodio

$s_t \in S$

$a_t \in A$

$r_t \in \mathbb{R}$

$\pi(s_t, a_t) \to r_t$

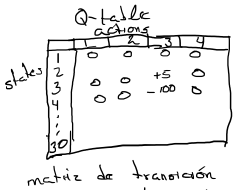
Encontrar $\pi^* = \arg\max_{\pi} \pi(R|S \times A)$

donde $R_t = \sum_{k=0}^{T} \gamma^k r_{t+k+1}$, $0 \le \gamma \le 1$

Q-Learning

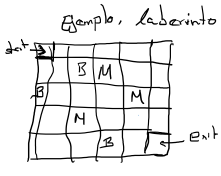
Uno de varios tipos de RL

- Construimos Q-table
- Q: quality



matriz de transición (probabilidad de realizar alguna acción dado un estado)

Ejemplo, laberinto



S

$s_0 = (1,1)$

$s_1 = (1,2)$

$s_2 = (1,3)$

$s_3 = (1,4)$

$\vdots$

$s_{29} = (5,6)$

A

$a_1 = \rightarrow$ 1

$a_2 = \leftarrow$ 2

$a_3 = \uparrow$ 3

$a_4 = \downarrow$ 4

función de calidad

$$q_\pi(s,a) = \mathbb{E}[R_t | s_t, a_t]$$

$$= \mathbb{E}\left[\sum_{k=0}^{T} \gamma^k r_{t+k+1} | s_t, a_t\right]$$

$$q^*(s,a) = \max_{\pi} q_\pi(s,a)$$

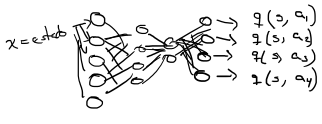
Bellman Ecuación

$$q^*(s,a) = \mathbb{E}\left[r_t + \gamma \max_{a'} q^*(s', a')\right]$$

s' = estado en $t+1$, despues de tomar acción a sugerida por q^*

a' = acción en $t+1$ despues de q^*

Reemplazar Tabla Q con una Red Neuronal



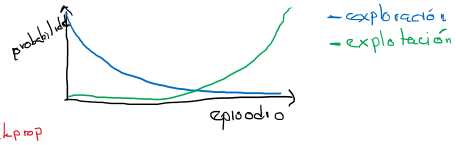
- Backprop usa recompensas para ajustar pesos
- Debemos encontrar la manera de considerar recompensas futuras (Bellman)

De manera iterativa actualizar tabla Q simulando estrategias "Programación lineal"

Trade-off exploración y explotación

- Exploración: tomar acciones at random y calcular recompensas inmediatas
- Explotación: tomar acciones usando conocimiento adquirido hasta ahora

$$a_t = f\left(\alpha \text{explor} + (1-\alpha)\text{explot}\right)$$



Deep Q Learning

$$\text{loss} = q^*(s,a) - q(s,a), \leftarrow \text{usado por backprop}$$

donde: $q(\ )$: valor de calidad (recompensa) inmediata

$q^*(\ )$: valor de calidad de la mejor acción despues del siguiente estado.

- Replay memory (experience replay)
  - Memoria (lista) con experiencias (tuplas): $e_t = (s_t, a_t, r_t, s_{t+1})$