RL

Reuben Brasher

December 6, 2022

Pictures of games

k-armed Bandits

Reinforcement Learning

k-armed Bandit

Figure: From https://en.wikipedia.org/wiki/Slot_machine

Markov Decision Process

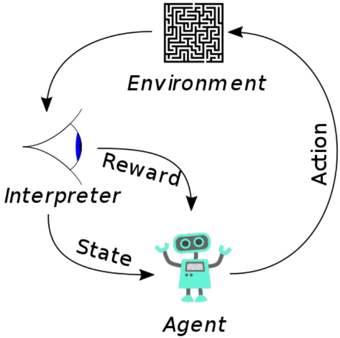

Figure: From https://en.wikipedia.org/wiki/Reinforcement_learning

Problem

Each round t an agent may choose an action from k possible. The agent receives a reward sampled from a distribution conditioned on the action.

$$P(R_t|A_t)$$

The objective of the game is to learn which action will give the highest expected reward.

q_*

- If only we knew the *value* of each action

$$q_*(a) = \mathbb{E}[R_t | A_t = a].$$

q_*

- If only we knew the *value* of each action

$$q_*(a) = \mathbb{E}[R_t | A_t = a].$$

- We do not. We know $Q_t(a)$.

Explore vs. Exploit

- Greedy strategy is always choose current best

$$\underset{a}{\operatorname{argmax}}\, Q_t(a)$$

Explore vs. Exploit

- Greedy strategy is always choose current best

$$\underset{a}{\operatorname{argmax}}\, Q_t(a)$$

- ε-Greedy strategy is to choose uniformly randomly probability ε, and to follow greedy strategy otherwise.

Explore vs. Exploit

- Greedy strategy is always choose current best

$$\underset{a}{\operatorname{argmax}}\, Q_t(a)$$

- ε-Greedy strategy is to choose uniformly randomly probability ε, and to follow greedy strategy otherwise.
- Upper confidence bound strategy is to choose

$$\underset{a}{\operatorname{argmax}} \left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right]$$

Finite Markov Decision Process

Agent and environment interact to produce a trajectory

$$S_0,\ A_0,\ R_1,\ S_1,\ A_1,\ R_2,\ S_2,\ A_2,\ R_3, \ldots$$

State and reward depend on previous state and agent action

$$p(s', r|s, a) = \Pr\left(S_t = s', R_t = r|S_{t-1} = s, A_{t-a} = a\right)$$

G_t

- *Return*

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T$$

G_t

- *Return*

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T$$

- *Discounted return*

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Policy, value and action value function

Agent responds to environment by sampling from *policy* $\pi(a|s)$.
Value of state s is

$$v_\pi(s) = \mathbb{E}[G_t | S_t = s]$$

Action value function is

$$q_\pi(s, a) = \mathbb{E}[G_t | S_t = s, A_t = a]$$

If only we knew...

The *optimal policy* π_* or
the *optimal value* of state s

$$v_*(s) = \max_\pi v_\pi(a)$$

or the *optimal action value function*

$$q_*(s,a) = \max_\pi q_\pi(s,a)$$

Note that

$$q_*(s,a) = \mathbb{E}\left[R_{t+1} + \gamma v_*\left(S_{t+1}\right) | S_t = s, A_t = a\right]$$

Bellman optimality equations

- Bellman equation is

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_\pi(s')\right]$$

Bellman optimality equations

- Bellman equation is

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_\pi(s')\right]$$

- Bellman optimality equation for value is

$$v_*(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')]$$

Bellman optimality equations

- Bellman equation is

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_\pi(s')\right]$$

- Bellman optimality equation for value is

$$v_*(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')]$$

- Bellman optimality equation for action value is

$$q_*(s, a) = \sum_{s',r} p(s', r|s, a)[r + \gamma \max_{a'} q_*(s', a')]$$

Finding optimal policy

- By policy iteration (pg 121)

$$\pi_0 \to v_{\pi_0} \to \pi_1 \to v_{\pi_1} \to \cdots \to \pi_* \to v_*$$

Finding optimal policy

- By policy iteration (pg 121)

$$\pi_0 \to v_{\pi_0} \to \pi_1 \to v_{\pi_1} \to \cdots \to \pi_* \to v_*$$

- By computing q_*

Finding optimal policy

- By policy iteration (pg 121)

$$\pi_0 \to v_{\pi_0} \to \pi_1 \to v_{\pi_1} \to \cdots \to \pi_* \to v_*$$

- By computing q_*
- By computing v_*

Temporal Difference

Iteratively update V to estimate v_π.
Update V by

$$V(S_t) \leftarrow V(S_t) + \alpha \left[G_t - V(S_t) \right]$$

Why wait though?

$$V(S_t) \leftarrow V(S_t) + \alpha \left[R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \right]$$

SARSA

Iteratively update Q to estimate q_*. (pg 151)
Update Q by

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t) \right]$$

Q-learning

Iteratively update Q to estimate q_*. (pg 153)
Update Q by

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right]$$

Policy Gradient

Parameterize policy π by θ, for example

$$\pi(a|s,\theta) = \frac{e^{h(s,a,\theta)}}{\sum_b e^{h(s,b,\theta)}}$$

where $h(s,a,\theta)$ is ANN or similar.
Then π has a gradient with respect to θ, and v_π can be improved by gradient ascent.

Actor critic

Parameterize policy π by θ, and estimate v by w

$$\pi(a|s,\theta) = \frac{e^{h(s,a,\theta)}}{\sum_b e^{h(s,b,\theta)}}$$

where $h(s,a,\theta)$ is ANN or and so is $v(s,w)$.
Then π has a gradient with respect to θ, and v with respect to w.
Improve v and π in turn by applying gradient updates.