Environment

27 March 2023 19:49

Elements common to all **control tasks**

Environment

state (s_t)

action (a_t)

Reward (r_t)

agent

$\pi(s)$

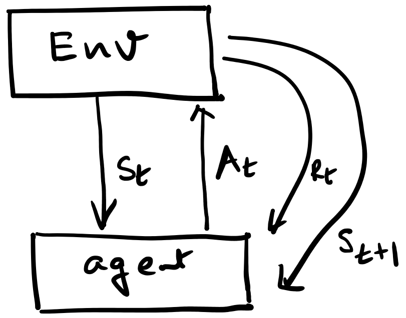
aim

Examples of Environment

28 March 2023 11:50

MDP

29 March 2023 14:19



Discuss St, at, rt, stp1 in the context of standard environment like chess, pacman, robot, etc

MDP : $\boxed{(S, A, R, P)}$

MDP has no memory

$$P(S_{t+1} \mid S_t = s) = P(S_{t+1} \mid S_t = s_t, S_{t-1} = s_{t-1}, \dots)$$

Episodic Vs Continuing MDPs] examples ??

Reward : R_t (instant Gratification)

Return : $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots$
(cumulative long term) gain

γ : discount factor

Q. Why bring in the discount factor?
[incentive to work faster]

- what happens if $\gamma = 0$?
- what happens if $\gamma = 1$?

Policy

30 March 2023 10:49

Central Problem in RL projects
learn to get policy π

$\pi(s) \rightarrow$ take a specipic action at state s

$\pi(a|s) \rightarrow$ probability of taking action a given state s

Aim: find optimum policy π_*

State and Action Values

30 March 2023 10:57

State Value

$$v_\pi(s) = E\left[G_t \mid S_t = s\right]$$

expected return starting from a particular state and following a policy π

$$= E\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \mid S_t = s\right]$$

q-value of a state-action pair:

$$q_\pi(s,a) = E\left[G_t \mid S_t = s,\ A_t = a\right]$$

= expected return starting from state s, taking action a and following policy π from there on.

Bellman Equations

30 March 2023 11:04

$$v_\pi(s) = E\left[G_t \mid S_t = s\right]$$

$$= E\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s\right]$$

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

Discuss Bellman Equation for v

$$q_\pi(s,a) = E\left[G_t \mid S_t = s , A_t = a\right]$$

$$= E\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s , A_t = a\right]$$

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

Discuss Bellman Equation for q

Solving a MDP

30 March 2023 11:19

Optimal Policy π_* depends on knowing optimal value

$$\pi_*(s) = \underset{a}{\arg\max} \sum_{r,s'} p(s',r \mid s,a) \left[r + \gamma v_*(s) \right]$$

$$= \underset{a}{\arg\max} \; q_*(s,a)$$

Optimal values depends on optimal policy

$$v_*(s) = E_{\pi_*}\left[G_t \mid S_t = s \right]$$

$$q_*(s,a) = E_{\pi_*}\left[G_t \mid S_t = s \,,\; A_t = a \right]$$

DILEMMA

chicken & egg

Bellman Optimality Equation

30 March 2023 11:30

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

$$v_*(s) = \max_a \sum_{s',r} p(s',r|s,a) \left[r + \gamma v_*(s')\right]$$] equation for optimal v

$$q_\pi(s,a) = \sum_{s',r} p(s',r|s,a) \left[r + \gamma \sum_{a'} \pi(a'|s') q_\pi(s',a')\right]$$ } eqn for q

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

Simple Grid

30 March 2023 11:42

Environment : 10 positions 

→ agent starts at a random position.

→ at each position, agent can move left or right.

→ one location is a target (fixed but not told to agent)

→ **reward** = 1 if next state is the target,
-1 else if next state is outside
0 all else

Exercise: Code this environment

- constructor function
- reset function
- step function
- view function

Aim

30 March 2023 15:55

Optimal Policy = ??

Recall the optimality equations

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

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

These Equations are used as update rules :-

start with random $V(s)$ $\forall s$

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

This is very similar to the Jacobi & Gauss Siedel methods of solving equation

Limitation : $p(s',r|s,a)$ needs to be known

Coding exercise: Program this

Value Iteration

30 March 2023

Value **Iteration**

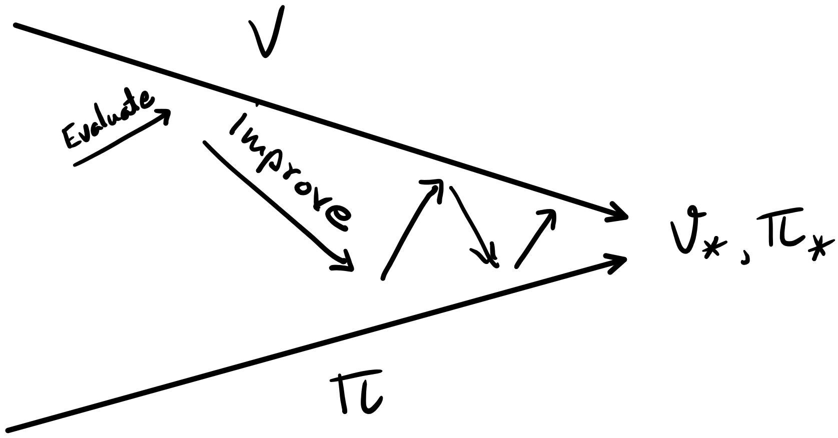
Algorithm 2 Value Iteration

```
Input: θ > 0 tolerance parameter, γ discount factor
Initialize V(s) arbitrarily, with V(terminal) = 0
repeat
    Δ ← 0
    for s ∈ S do
        v ← V(s)
        V(s) ← max_{a∈A(s)} Σ_{s',r} p(s',r|s,a)[r + γV(s')]
        Δ ← max(Δ, |v − V(s)|)
    end for
until Δ > θ
Output: π: greedy policy w.r.t. V(s)
```

Policy Iteration

31 March 2023 08:54

A process that alternatively improves the policy



Algorithm 2 Policy Iteration

```
Input: θ > 0 tolerance parameter, γ discount factor
Initialize V(s) and π(a|s) arbitrarily
while policy-stable = false do

    Policy Evaluation:
    while Δ > θ do
        Δ ← 0
        for s ∈ S do
            v ← V(s)
            V(s) ← Σ_a π(a|s) Σ_{s',r} p(s',r|s,a)[r + γV(s')]
            Δ ← max(Δ, |v − V(s)|)
        end for
    end while

    Policy Improvement:
    policy-stable = true
    for s ∈ S do
        old-action ← π(s)
        π(s) ← arg max_{a∈A(s)} Σ_{s',r} p(s',r|s,a)[r + γV(s')]
        if old-action ≠ π(s) then
            policy-stable ← false
        end if
    end for

end while
Output: Optimal policy π(a|s) and state values V(s)
```