3.5 - Policies and Value Functions: Dec 25, 2021

Value Function: A state function that estimates how good it is for the agent to be in a certain state or perform a given action in a given state.

↳ Almost all RL algorithms involve estimating value functions.

Policy: A mapping from states to the probabilities of selecting each possible action.

e.g. Agent follows π at time t.

↳ $\pi(a|s)$ is probability that $A_t = a$ if $S_t = s$.

The value function of a state s under policy π is the expected return when starting in s and following π after.

✱ state-value function

↳ $$v_\pi(s) \equiv E_\pi[G_t | S_t = s] = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right]$$

for all $s \in S$

$v_\pi(s)$: state-value function for policy π

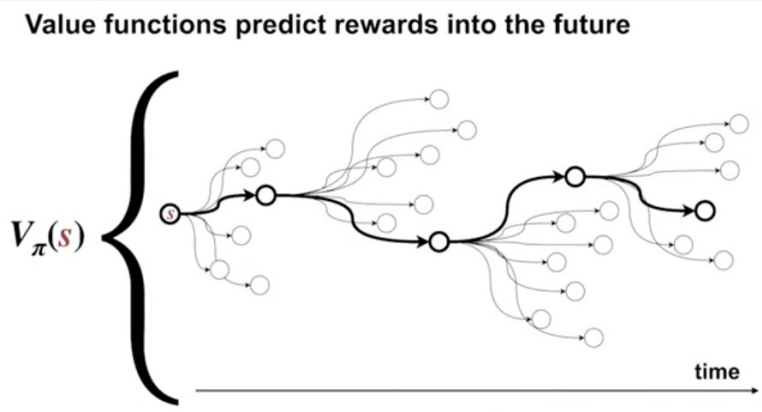
$E_\pi[G_t | S_t = s]$: expected value of G_t given $S_t = s$ and agent follows policy π

$$q_\pi(s,a) \equiv E_\pi[G_t \mid S_t = s, A_t = a] \quad \text{action-value function}$$

$$= E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s, A_t = a\right]$$

$q_\pi(s,a)$: the action-value function for policy π (i.e. the value of taking action a in state s under a policy π)

Value functions predict rewards into the future!

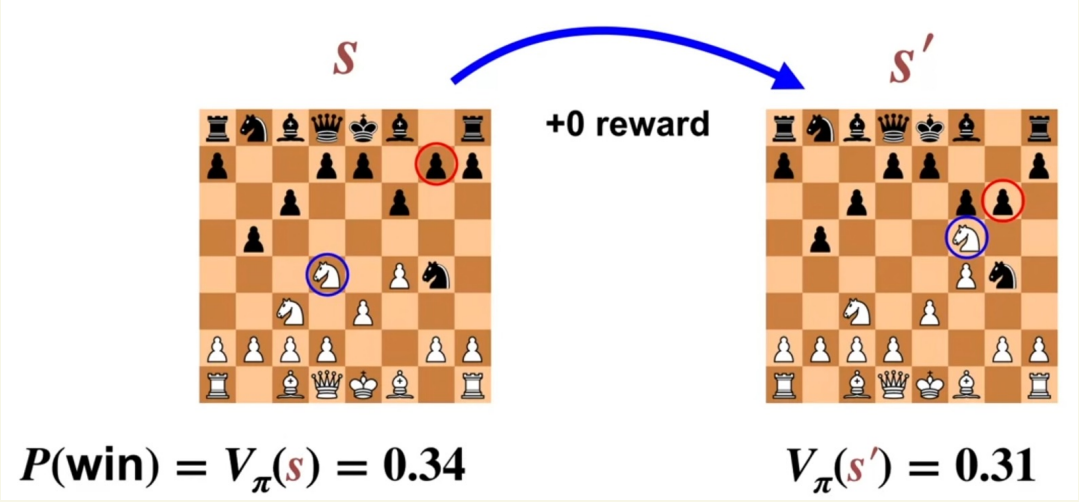


Agent doesn't have to wait to see the outcome.

→ The value function summarizes all possible futures by averaging over returns.

↳ This is why it's defined as $E_\pi[G_t]$, and why we covered the concept of gain.

e.g. Chess



Reward is +1 if win, +0 if lose.

↳ But reward doesn't tell agent how well it plays during the game.

↳ The value function tells us much more.

i.e. $V_\pi(s) = 0.34 \rightarrow V_\pi(s') = 0.31$

↳ An action-value function would tell us that we are less likely to win the game if we make this move.

※ In essence, the RL agent should always select the best action based on the action-value function.

※ How to compute $V_\pi(s)$ and $q_\pi(s,a)$ will be discussed later.

State-Value Bellman Equation:

The Bellman equation for the state-value function defines a relationship between the value of a state and the values of its possible successor states.

Derivation:

$$v_\pi(s) \equiv \mathbb{E}_\pi[G_t \mid S_t = s] \quad \text{※ state-value equation}$$

$$= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s]$$

$$= \underbrace{\sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)}\Big[r + \gamma \underbrace{\mathbb{E}_\pi[G_{t+1} \mid S_{t+1} = s']}_{v_\pi(s')}\Big]$$

1. Expand expected return as a sum over possible action choices made by the agent.
2. Expand over possible rewards and next states conditioned on state s and action a.

※ This is just a sum of possible outcomes weighted by the probability that they occur.

r: Each possible reward outcome
s': Each possible next state
a: Each possible current action

$$\therefore \boxed{v_\pi(s) \equiv \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)[r + \gamma v_\pi(s')]}$$

※ state-value Bellman equation

Action-Value Bellman Equation:

Defines the value of a current state-action pair in terms of possible successor state-action pairs.

↳ However, instead of only expressing it in terms of the next state, we want a recursive equation summing over all possible agent actions.

Derivation:

$$q_\pi(s,a) \equiv E_\pi[G_t | S_t = s, A_t = a]$$

$$= \sum_{s'} \sum_r p(s', r|s, a)\left[r + \gamma E_\pi[G_{t+1} | S_{t+1} = s']\right]$$

※ Does not begin w/ policy selecting an action, since action is fixed as part of the state-action pair.

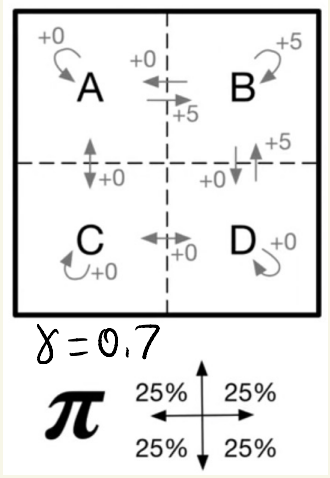
$$= \sum_{s'} \sum_r p(s', r|s, a)\left[r + \gamma \sum_{a'} \pi(a'|s') E_\pi[G_{t+1} | S_{t+1} = s', A_{t+1} = a']\right]$$

※ Change it s.t. the expected return depends on both the next state and the next action.

$$q_\pi(s,a) = \sum_{s'}\sum_{r} p(s',r|s,a)\left[r + \gamma \sum_{a'} \pi(a'|s') E_\pi\left[G_{t+1} \,\middle|\, \begin{matrix} S_{t+1}=s' \\ A_{t+1}=a' \end{matrix}\right]\right]$$

※ action-value Bellman equation

Gridworld Example:



$\gamma = 0.7$

π 25% 25% 25% 25%

$$V_\pi(A) \equiv E_\pi[G_t \mid S_t = A]$$
$$V_\pi(B) \equiv E_\pi[G_t \mid S_t = B]$$
$$V_\pi(C) \equiv E_\pi[G_t \mid S_t = C]$$
$$V_\pi(D) \equiv E_\pi[G_t \mid S_t = D]$$

※ There are infinitely many sequences an agent can choose!

To represent this, use Bellman equations:

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

$$V_\pi(A) = \sum_a \pi(a|A)(r + 0.7 V_\pi(s'))$$

↳ 4 discrete possibilities w/ $\frac{1}{4}$ chance

$$\hookrightarrow V_\pi(A) = \tfrac{1}{4}(5 + 0.7 V_\pi(B)) + \tfrac{1}{4} 0.7 V_\pi(C) + \tfrac{1}{2} 0.7 V_\pi(A)$$

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

$$V_\pi(A) = \frac{1}{4}(5 + 0.7V_\pi(B)) + \frac{1}{4}0.7V_\pi(C) + \frac{1}{2}0.7V_\pi(A)$$

$$V_\pi(B) = \frac{1}{2}(5 + 0.7V_\pi(B)) + \frac{1}{4}0.7V_\pi(A) + \frac{1}{4}0.7V_\pi(D)$$

$$V_\pi(C) = \frac{1}{4}0.7V_\pi(A) + \frac{1}{4}0.7V_\pi(D) + \frac{1}{2}0.7V_\pi(C)$$

$$V_\pi(D) = \frac{1}{4}(5 + 0.7V_\pi(B)) + \frac{1}{4}0.7V_\pi(C) + \frac{1}{2}0.7V_\pi(D)$$

} 4 equations, 4 unknowns

Solve!

$V_\pi(A) = 4.2$

$V_\pi(B) = 6.1$ → makes sense cuz $\frac{1}{2}$ the options gives +5

$V_\pi(C) = 2.2$ → makes sense cuz furthest from B

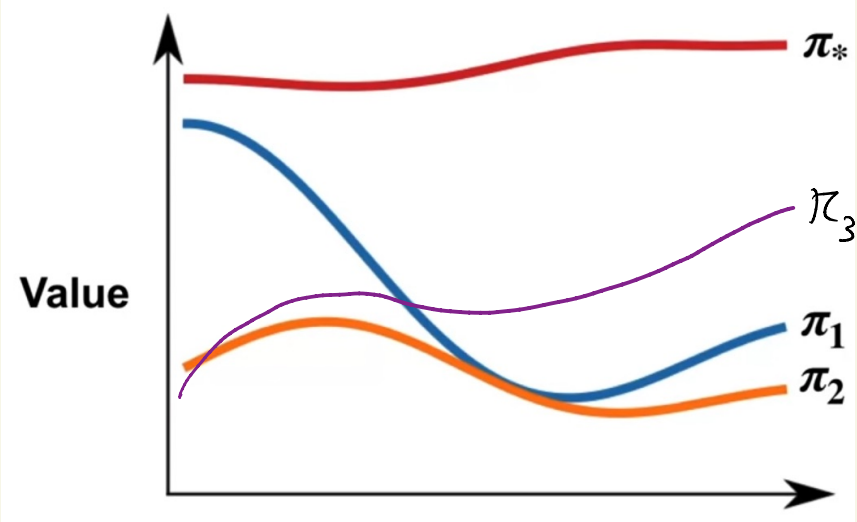
$V_\pi(D) = 4.2$

✱ Bellman equations reduce an infinite sum of possible futures to a simple linear equation.

✱ But Bellman equations can only be used to find the value functions (V_π) of simple MDP's.

↳ Must be scaled up for games like chess, w/ 10^{44} Bellman equations instead of 4.

Optimal Policies:

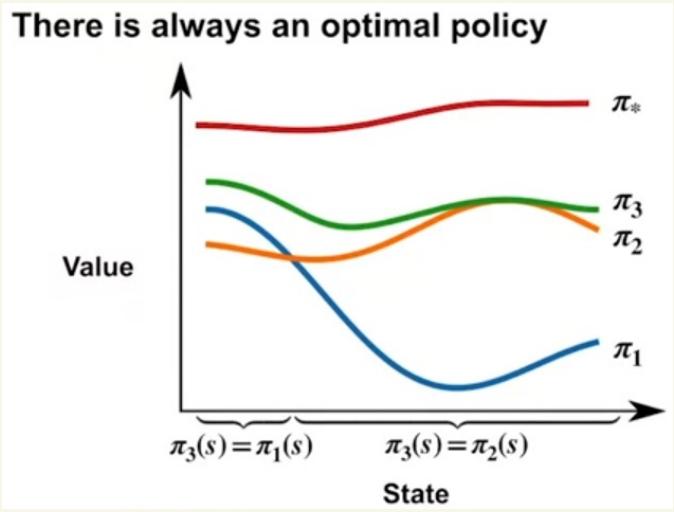


A policy π_1 is as good or better than another policy π_2 iff $\pi_1 \geq \pi_2$ for every state.

↳ $\pi_1 \geq \pi_2$ but not π_3

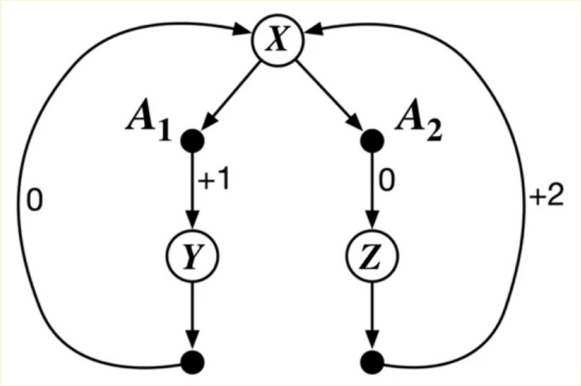
Optimal Policy: A policy that will have the highest possible value in every state.

✡ There is <u>always</u> an optimal policy, because you can combine policies.



Why is $\pi_3 > \pi_2, \pi_1$? (I have no idea.)

Simple MDP Example!



Is π_1 or π_2 better?
↳ Depends on discount factor!

$\gamma = 0$!
$V_{\pi_1}(X) = 1$
$V_{\pi_2}(X) = 0$
$\therefore \pi_1$ is optimal

$\pi_1(X) = A_1$
$\pi_2(X) = A_2$

$\gamma = 0.9$!

$$V_{\pi_1}(X) = \sum_{k=0}^{\infty} (0.9)^{2k} = \frac{1}{1-0.9^2} \doteq 5.3$$

$$V_{\pi_2}(X) = \sum_{k=0}^{\infty} (0.9)^{2k+1} = \frac{0.9}{1-0.9^2} \cdot 2 \doteq 9.5$$

$\therefore \pi_2$ is optimal

But how do we find the optimal policy for harder examples? In total there are A^S possible policies ($\#\text{actions}^{\#\text{states}}$), and calculating the value function for each one is inefficient.

∴ Use Bellman Optimality equations to find optimal policy.

Optimal Value Functions:

$\pi_1 \geq \pi_2$ iff $v_{\pi_1}(s) \geq v_{\pi_2}(s)$ for all $s \in S$

For an Optimal Policy:

$$v_{\pi*}(s) = E_{\pi*}[G_t \mid S_t = s] = \max_{\pi} v_{\pi}(s) \text{ for all } s \in S$$

$$q_{\pi*}(s,a) = \max_{\pi} q_{\pi}(s,a) \text{ for all } s \in S,\ a \in A$$

※ State-value functions for optimal policy are denoted w/ v_*, and for action-value functions q_*

Bellman Optimality Equations:

$$V_*(s) = \sum_a \pi_*(a|s) \sum_{s'} \sum_r p(s', r|s, a)[r + \gamma V_*(s')]$$

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

※ Optimal policy will always select the optimal action.

↳ Assigns prob = 100% to action w/ highest value, 0 for others.

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

※ Bellman Optimality Equation for V_*

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

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

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

※ Bellman Optimality Equation for q_*

Can we use a linear system solver to find the optimal state-value function?

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

π, p, γ → Linear System Solver → v_π

& Also can't replace π here w/ π_* cuz we don't know π_*

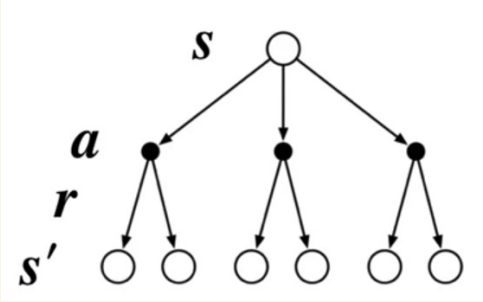
Not linear

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

p, γ → ~~Linear System Solver~~ → v_* No.

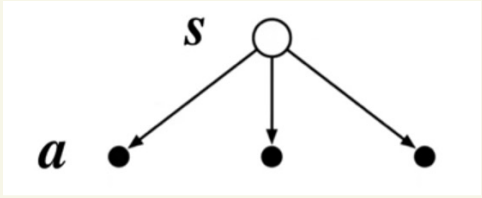
∴ We have to use other techniques to compute value functions and find the optimal policy using the Bellman Optimality Equations we derived.

Determining an Optimal Policy:



※Using optimal state-value function

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



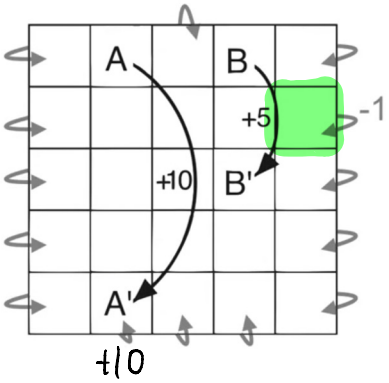
※Using optimal action-value function

$$\pi_*(s) = \arg\max_a q_*(s, a)$$

Optimal Action Example:

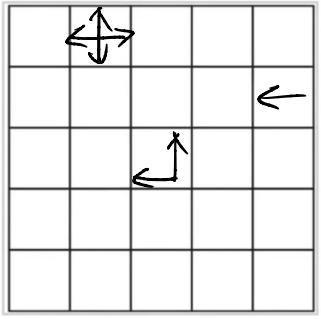
$$\pi_*(s) = \underset{a}{\text{argmax}} \sum_{s'} \sum_{r} p(s', r \mid s, a)\left[r + \gamma v_*(s')\right]$$

$\gamma = 0.9$



22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

v_*



π_*

Up: $0 + 0.9 \times 17.5 = 15.0$

Right: $-1 + 0.9 \times 16.0 = 13.4$

Down: $0 + 0.9 \times 14.4 = 13.0$

Left: $0 + 0.9 \times 17.8 = 16.0$ ✡Largest

∴ Optimal action is move left on that square.

✡ You can also see on v_* that the optimal next state value is 16.0.

✡ For a case w/ equally optimal values, policy is free to pick either w/ some probability.

Summary:

Policies tell an agent how to behave in their environment.

↳ Deterministic policies map each state to an action.

↳ Stochastic policies map each state to a distribution over all actions.

A policy depends only on the current state.

Value functions estimate future return under a specific policy.

↳ State-value function gives expected return from the current state under a policy.

$$v_\pi(s) \equiv E_\pi[G_t \mid S_t = s]$$

↳ Action-value function gives expected return from the state s if agent selects action a and follows π afterwards.

$$q_\pi(s, a) \equiv E_\pi[G_t \mid S_t = s, A_t = a]$$

Bellman equations define a relationship b/t the value of a state/state-action pair and its successors.

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

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

⛔ Do not find optimal policy.

An **optimal policy** achieves the highest value possible in every state.

↳ There is always at least one.

Optimal Value Functions:

$$v_*: v_{\pi*}(s) = \max_\pi v_\pi(s) \text{ for all } s \in S$$

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

$$q_*: q_{\pi*}(s,a) = \max_\pi q_\pi(s,a) \text{ for all } s \in S, a \in A$$

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

Then Use Optimal Value Functions to Obtain the Optimal Policy:

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

$$\pi_*(s) = \arg\max_a q_*(s, a)$$