



MDPs and Bellman Equations

Reinforcement learning | States, actions, rewards, transitions, return and discount, policies, V and Q functions, Bellman expectation and optimality equations worked by hand, gridworld value iteration preview

#RL #MDP #Bellman #ValueFunction

© The idea and the loop

Markov decision process

In one sentence
●○○

A board game with dice: where you are, what moves you can make, how likely each outcome is, and how many points each outcome gives.

Practitioner
●●○

A tuple (S, A, P, R, γ) and a policy π that the agent improves to maximise the expected return.

- S states, A actions
- $P(s'|s,a)$ transitions, R rewards
- γ discount: how much the future counts

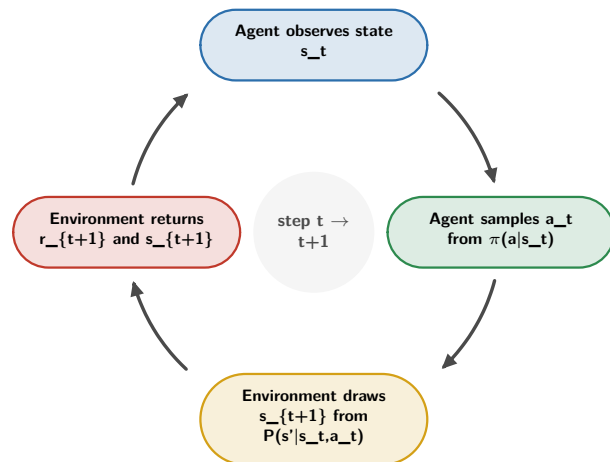
Expert
●●●

Values satisfy the Bellman equations, a set of $|S|$ linear equations for a fixed policy and a nonlinear (max) fixed point for the optimum; the Bellman operator is a γ -contraction, so iterating it converges.

- finite MDP $\rightarrow$ a deterministic optimal policy exists
- partial observability turns it into a POMDP

Analogy

A chess position is a state; the board alone (not the move history) tells you everything you need, which is exactly the Markov property.



The agent-environment loop, one turn per time step t . The reward r_{t+1} arrives after the action that caused it, together with the next state

Symbol	Meaning	FrozenLake-v1 (4×4)
S	set of states	16 cells, Discrete(16)
A	set of actions	left, down, right, up: Discrete(4)
$P(s' s,a)$	transition probabilities	slippery: intended move with prob 1/3, each perpendicular move 1/3
$R(s,a,s')$	reward for the transition	+1 on reaching the goal, 0 otherwise
γ	discount factor, $0 \leq \gamma \leq 1$	chosen by you, e.g. 0.99
$\pi(a s)$	policy: probability of a in s	what the agent learns

The five ingredients, with the FrozenLake 4×4 example from Gymnasium

Trajectory — The sequence $s_0, a_0, r_1, s_1, a_1, r_2, \dots$ produced by running a policy in the environment.

Deterministic vs stochastic policy — $a = \pi(s)$ always picks the same action; $\pi(a|s)$ samples from a distribution (needed for exploration and for policy gradients).

Reward hypothesis — Sutton and Barto's working assumption that any goal can be expressed as maximising the expected cumulative scalar reward.

⌚ Return, discount and horizon

Discounted return of a 4-step episode

FORMULA $G = r_1 + g \times r_2 + g^2 \times r_3 + g^3 \times r_4$

WHERE

- G = return from time 0: discounted sum of the rewards (reward units)
- r_1 = reward received after the first action (reward units)
- r_2 = reward after the second action (reward units)
- r_3 = reward after the third action (reward units)
- r_4 = reward after the fourth (last) action (reward units)
- g = discount factor γ (0-1)

WORKED

- $r_1 = 1$
- $r_2 = 0$
- $r_3 = 2$
- $r_4 = 5$

READ IT The 5 points at the end are worth 3.645 today: each step of delay multiplies a reward by $\gamma = 0.9$.

MISTAKES

- Rewards are indexed from $t+1$: the first reward is discounted by $\gamma^0 = 1$
- With $\gamma = 1$ a continuing task can have an infinite return
- Changing γ changes which policy is optimal, not just the numbers

Quantity	Formula	Note
Return	$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$	finite if $\gamma < 1$ and rewards bounded
Recursive form	$G_t = r_{t+1} + \gamma G_{t+1}$	the seed of every Bellman equation
Bound	$ G_t \leq \frac{r_{max}}{1-\gamma}$	geometric series
Effective horizon	$H \approx \frac{1}{1-\gamma}$	$\gamma = 0.9 \rightarrow 10$ steps; $0.99 \rightarrow 100$; $0.999 \rightarrow 1,000$

where G_t = return from time t (reward units); r_{t+k+1} = reward received k+1 steps after time t (reward units); γ = discount factor (0-1); r_{max} = largest absolute reward per step (reward units); H = number of steps that still matter (steps)

Return and horizon

EXAM TIP

Discounting has three justifications, and interviewers ask for them: it keeps infinite sums finite, it models uncertainty about the future (a $1-\gamma$ chance per step that the episode ends), and it prefers sooner rewards. It is also a variance-reduction knob: smaller γ , shorter credit assignment, more bias.

Policies and value functions

Equation	Formula
V and Q link	$V^\pi(s) = \sum_a \pi(a s) Q^\pi(s, a)$
Expectation, V	$V^\pi(s) = \sum_a \pi(a s) \sum_{s'} P(s' s, a) [R(s, a, s') + \gamma V^\pi(s')]$
Expectation, Q	$Q^\pi(s, a) = \sum_{s'} P(s' s, a) [R(s, a, s') + \gamma \sum_{a'} \pi(a' s') Q^\pi(s', a')]$
Optimality, V	$V^*(s) = \max_a \sum_{s'} P(s' s, a) [R(s, a, s') + \gamma V^*(s')]$
Optimality, Q	$Q^*(s, a) = \sum_{s'} P(s' s, a) [R(s, a, s') + \gamma \max_{a'} Q^*(s', a')]$
Optimal policy	$\pi^*(s) = \arg \max_a Q^*(s, a)$

where s, s' = current state, next state; a, a' = current action, next action; $\pi(a|s)$ = probability that the policy picks a in s (0-1); $P(s'|s, a)$ = transition probability (0-1); $R(s, a, s')$ = expected reward of that transition (reward units); γ = discount factor (0-1); V, Q = state value, action value (expected return) (reward units)

The four Bellman equations (finite MDP, sums over all next states s' and actions a')

Bellman expectation and optimality backups by hand

PROBLEM In state s the policy picks left or right with probability 0.5 each. Compute $Q(s, \text{left})$, $Q(s, \text{right})$, $V^\pi(s)$ and the optimal backup $V^*(s)$.

GIVEN

- $\gamma = 0.9$ (discount factor)
- left: reward 0, always goes to s1 with $V(s1) = 4$
- right: reward 1, then s2 with prob 0.8 ($V(s2) = 10$) or s3 with prob 0.2 ($V(s3) = 0$)
- $Q(s, a) = \sum P(s'|s, a) [R + \gamma V(s')]$; $V^\pi(s) = \sum \pi(a|s) Q(s, a)$; $V^*(s) = \max_a Q(s, a)$

1 Back up left

$Q(s, \text{left}) = 1.0 \times (0 + 0.9 \times 4) = 3.6$

Why: one next state, probability 1

2 Back up right

$Q(s, \text{right}) = 0.8 \times (1 + 0.9 \times 10) + 0.2 \times (1 + 0.9 \times 0) = 8.0 + 0.2 = 8.2$

Why: average over the two possible next states

3 Expectation backup

$V^\pi(s) = 0.5 \times 3.6 + 0.5 \times 8.2 = 5.9$

Why: average over the policy's action probabilities

4 Optimality backup

$V^*(s) = \max(3.6, 8.2) = 8.2$, greedy action = right

Why: replace the policy average by a max

ANSWER **Q = 3.6 (left), 8.2 (right); $V^\pi(s) = 5.9$; optimal backup 8.2 by going right**

CHECK  $V^\pi \leq V^*$ always: an average of the Q values can never exceed their maximum ($5.9 \leq 8.2$)

V(s)

value of being in s

vs

Q(s,a)

value of doing a in s

Tell: Q lets you act greedily without a model: argmax over a; V needs P to look one step ahead

Bellman expectation

average over π

vs

Bellman optimality

max over a

Tell: Expectation is linear (solve exactly); optimality has a max (iterate)

Episodic

ends at a terminal state

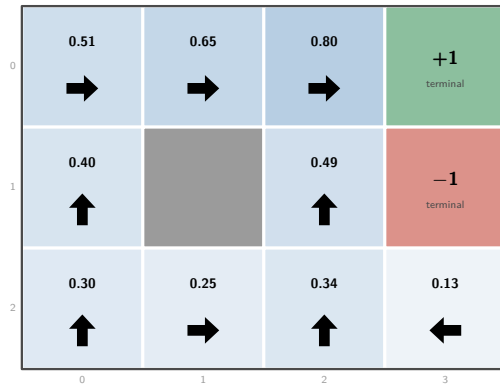
vs

Continuing

runs forever

Tell: Continuing tasks need $\gamma < 1$ (or average reward)

Gridworld: values and greedy policy



$\gamma = 0.9$ · step reward -0.04 · noise 0.2 (0.1 to each side) · converged after 21 sweeps · arrows: greedy policy



$$V_{k+1}(s) = R(s) + \gamma \cdot \max_a \sum_{s'} P(s' | s, a) \cdot V_k(s')$$

$$\pi(s) = \operatorname{argmax}_a \sum_{s'} P(s' | s, a) \cdot V(s') \text{ (greedy policy: the arrows)}$$

where
 $V_k(s)$ value of cell s after k sweeps (expected discounted return)
 $R(s)$ reward for being in s : step reward (+ cell reward); terminals keep V = reward
 γ discount factor, $0 < \gamma \leq 1$
 a action: up, right, down, left; s' = next cell
 $P(s' | s, a)$ move as intended 0.8, slip to each side 0.1; walls and edges: stay
 $\pi(s)$ action chosen in s

The classic 3×4 gridworld (Russell and Norvig): +1 and -1 exits, one wall, a small cost per step, and 20 % of moves slipping sideways. Value iteration computes each cell's V^* ; arrows show the greedy policy, which steers away from the -1 cell even if the path gets longer

TIP

Read the gridworld like a Bellman equation (Russell and Norvig form): each cell's value = step reward + $\gamma \times$ the best, over the four moves, of $0.8 \times$ value of the cell straight ahead + $0.1 \times$ value of each side cell (bumping into a wall or edge leaves you in place). The mini-grids show the values spreading one cell per sweep from the exits. Raise the step cost to -2 and the agent dives into the -1 exit: the reward defines the behaviour.

One Bellman backup on FrozenLake with the true model (Gymnasium toy-text exposes P)

```
import gymnasium as gym
import numpy as np

env = gym.make("FrozenLake-v1", is_slippery=True)
P = env.unwrapped.P # P[s][a] = [(prob, s2, r, terminated)]
gamma, V = 0.99, np.zeros(16)
V[14] = 0.5 # pretend we already value cell 14

def q(s, a):
    return sum(p * (r + gamma * V[s2] * (not done))
              for p, s2, r, done in P[s][a])

s = 10
print([round(q(s, a), 3) for a in range(4)])
# V*(10) backup = max of the four Q values
```

✗ Bootstrapping through a terminal state

At a terminal transition the target is just r , not $r + \gamma V(s')$: the episode is over and there is no future. Forgetting the (not done) factor makes values leak out of goal and trap cells; time-limit truncation is the opposite case and must keep the $\gamma V(s')$ term.

⚠ Traps and recap

MYTH

✗ The reward should tell the agent how to solve the task

✗ The optimal policy must be random to explore

✗ A higher γ is always better

✗ Observations are states

FACT

✓ The reward says what to achieve, not how
 Rewarding sub-steps (e.g. touching the ball) invites agents to farm the sub-step; shape only with potential-based terms $\gamma \Phi(s') - \Phi(s)$, which keep the optimal policy.

✓ In a finite MDP a deterministic optimal policy always exists
 Randomness is needed while learning (exploration) and in games against adversaries or under partial observability, not for the optimal MDP solution.

✓ γ trades horizon against variance and changes the optimum
 $\gamma = 0.999$ looks 1,000 steps ahead but makes value estimates noisier and learning slower; many tasks train best with 0.95-0.99.

✓ Only if they satisfy the Markov property
 One Atari frame hides the ball's velocity; DQN stacks 4 frames to restore an (approximate) Markov state.

Tick a box only when you could explain the point to someone else

☐ MDP = (S, A, P, R, γ)
 next state depends only on current state and action

☐ V averages Q over the policy
 $V^* \pi(s) = \sum \pi(a|s) Q^* \pi(s, a)$

☐ Optimality equation finds the best policy
 max over actions, then act greedily on Q^*

☐ Return $G_t = r_{t+1} + \gamma G_{t+1}$
 horizon $\approx 1/(1-\gamma)$

☐ Expectation equation evaluates a policy
 linear in V

☐ Terminal $\neq$ truncated
 stop bootstrapping only at true terminals

Q1 Write the MDP tuple and name each element.

A (S, A, P, R, γ): states, actions, transition probabilities $P(s'|s,a)$, rewards $R(s,a,s')$, discount factor.

Q2 Return of rewards 1, 0, 2, 5 with $\gamma = 0.9$?

A $1 + 0 + 0.81 \times 2 + 0.729 \times 5 = 6.265$.

Q3 Effective horizon for $\gamma = 0.99$?

A About $1/(1 - 0.99) = 100$ steps.

Q4 How do V^π and Q^π relate?

A $V^\pi(s) = \sum_a \pi(a|s) Q^\pi(s,a)$: V is the policy-weighted average of Q .

Q5 What changes between the Bellman expectation and optimality equations?

A The average over $\pi(a|s)$ is replaced by a max over actions.

Q6 $Q(s,\text{left}) = 3.6$, $Q(s,\text{right}) = 8.2$, $\pi = 50/50$. $V^\pi(s)$ and $V^*(s)$ backup?

A 5.9 and 8.2 (greedy action right).

Q7 Why does Q make model-free control easy?

A Acting greedily needs only $\arg\max_a Q(s,a)$; with V you need the model P to look one step ahead.

Q8 Why does DQN stack four Atari frames?

A One frame is not Markov (no velocity); four frames approximate a Markov state.

Q9 Which reward shaping keeps the optimal policy unchanged?

A Potential-based shaping $F = \gamma\Phi(s') - \Phi(s)$ (Ng, Harada and Russell, 1999).