Artificial Intelligence

17. Markov Decision Processes

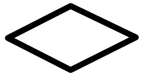
Shashi Prabh

School of Engineering and Applied Science
Ahmedabad University

Recap: Decision Networks

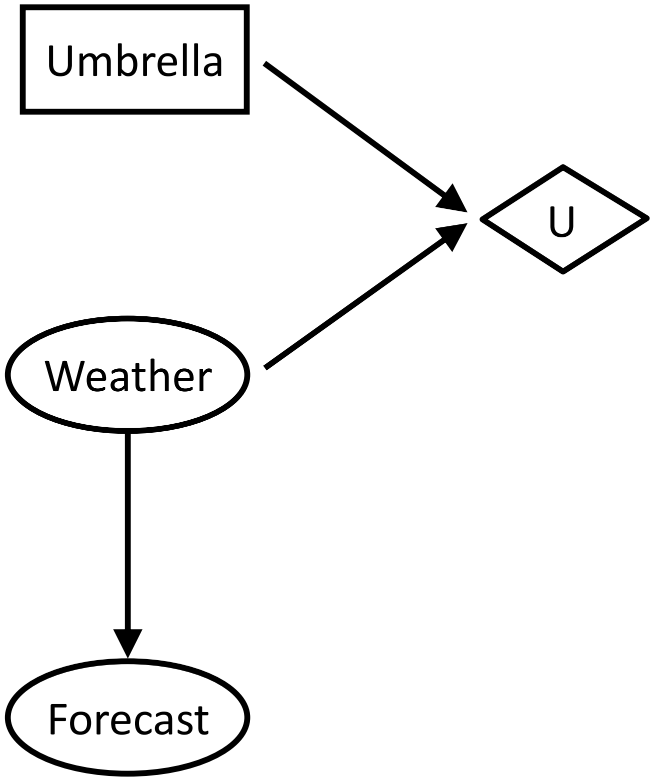
- Decision network = Bayes net + Actions + Utilities

- **Action nodes** (rectangles, cannot have parents, will have value fixed by algorithm)



- **Utility nodes** (diamond, depends on action and chance nodes)

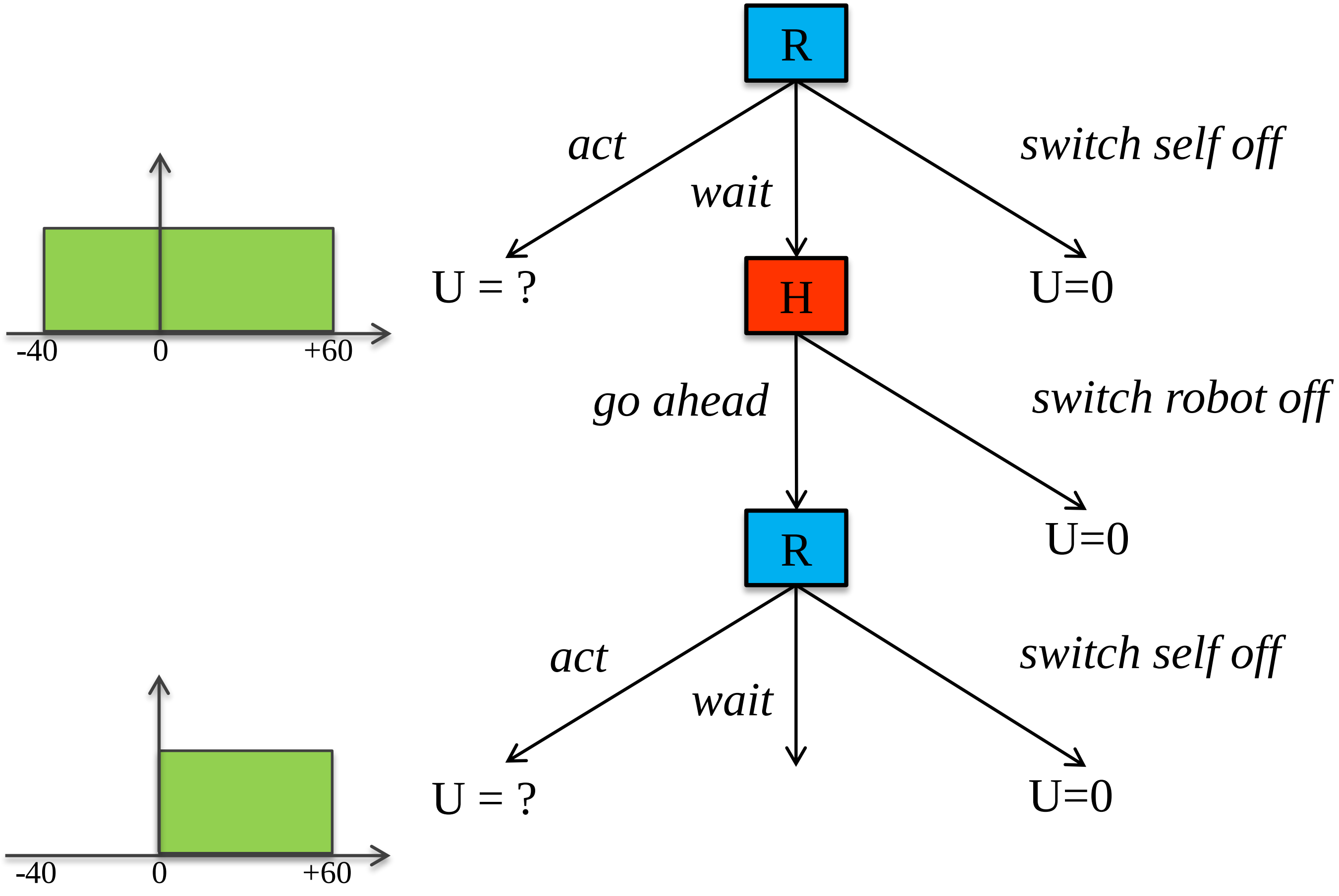
- Decision network represents a decision problem, containing all the information needed to for the agent to decide
  - What action to take given evidence ***e***
    - Decision algorithm
  - Value of information
    - $\mathrm{VPI}(E_i \mid e) = \left[\ \sum_{e_i} P(e_i \mid e) \max_a \mathrm{EU}(a \mid e_i, e)\ \right] - \max_a \mathrm{EU}(a \mid e)$



Decisions with unknown preferences

- In reality the assumption that we can write down our exact preferences for the machine to optimize is false
- A machine optimizing the wrong preferences causes problems
- A machine that is explicitly uncertain about the human's preferences will defer to the human (e.g., allow itself to be switched off)

Off-switch problem

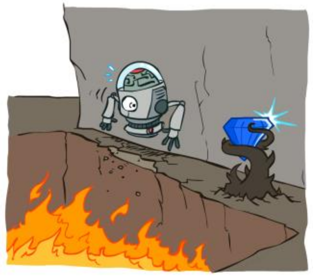


EU(act) = +10

EU(wait) = (0.4 * 0) + (0.6 * 30) = +18

Sequential decisions under uncertainty

- So far, decision problem was one-shot
    - Concerning only one action
- Sequential decision problem: agent's utility depends on a sequence of actions



Markov Decision Process (MDP)

Andrey Markov
(1856-1922)

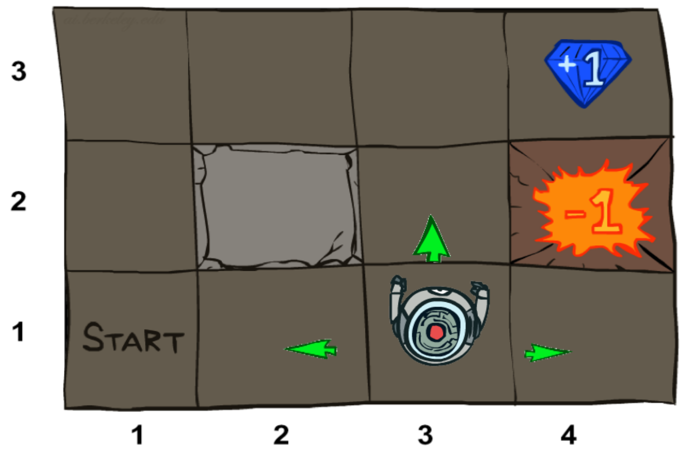
- Environment history: [s_0, a_0, s_1, a_1, …, s_t]
- **Markov** means that given the present state, the future and the past are independent : First Order Markov Chain
- For Markov decision processes, Markov means action outcomes depend only on the current state

$$P(S_{t+1} = s' | S_t = s_t, A_t = a_t, S_{t-1} = s_{t-1}, A_{t-1}, \ldots S_0 = s_0) = P(S_{t+1} = s' | S_t = s_t, A_t = a_t)$$

- This is just like search, where the successor function could only depend on the current state (not the history)

Markov Decision Process (MDP)

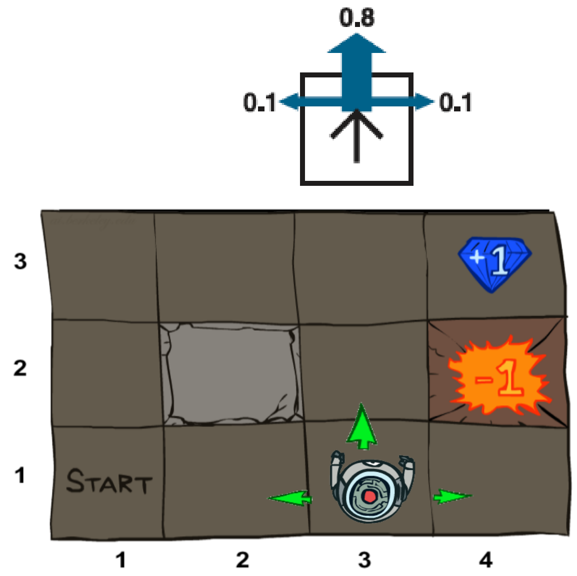
- An MDP is defined by:
  - A set of states $s \in S$
  - A set of actions $a \in A$
  - A transition model $T(s, a, s')$
    - Probability that a from s leads to s', i.e., $P(s' | s, a)$
  - A reward function $R(s, a, s')$ for each transition
  - A start state
  - Possibly a terminal state (or absorbing state)
  - Utility function which is additive (discounted) rewards



- MDPs are fully observable but probabilistic search problems

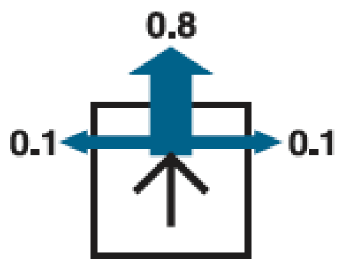
[Demo – gridworld]

Example: Grid World



- A maze-like problem
  - The agent lives in a grid
  - Walls block the agent's path
- Noisy movement: actions do not always go as planned
  - 80% of the time, the action North takes the agent North (if there is no wall there)
  - 10% of the time, North takes the agent West; 10% East
  - If there is a wall in the direction the agent would have been taken, the agent stays put

Example: Grid World



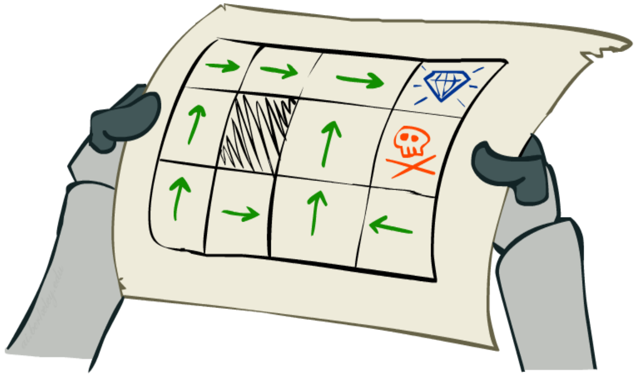
- The agent receives rewards each time step
  - Small “living” reward r each step (can be negative)
  - Big rewards come at the end (good or bad)
- Goal: maximize sum of rewards

Policies

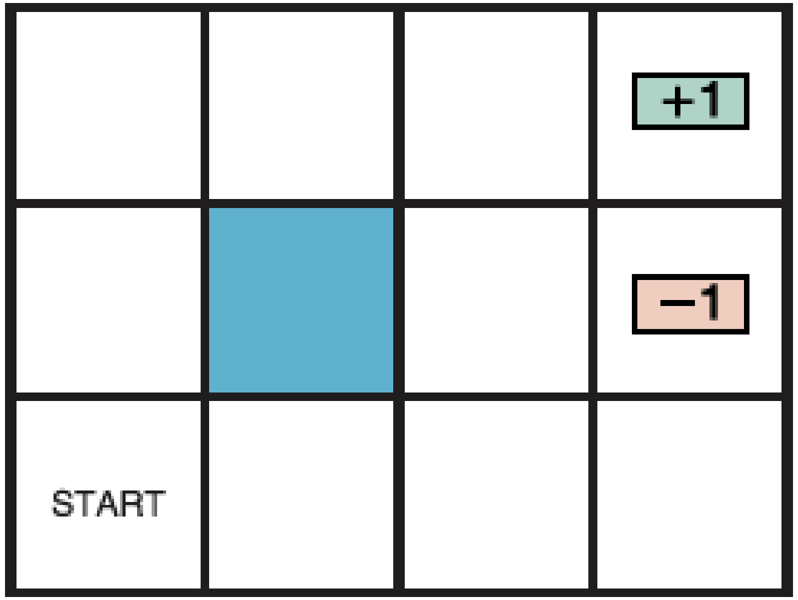
- A policy π gives an action for each state, $\pi: S \rightarrow A$

- In deterministic single-agent search problems, we wanted an optimal plan, or sequence of actions, from start to a goal

- For MDPs, we want an optimal policy $\pi^*: S \rightarrow A$
  - An optimal policy maximizes expected utility
  - An explicit policy defines a reflex agent

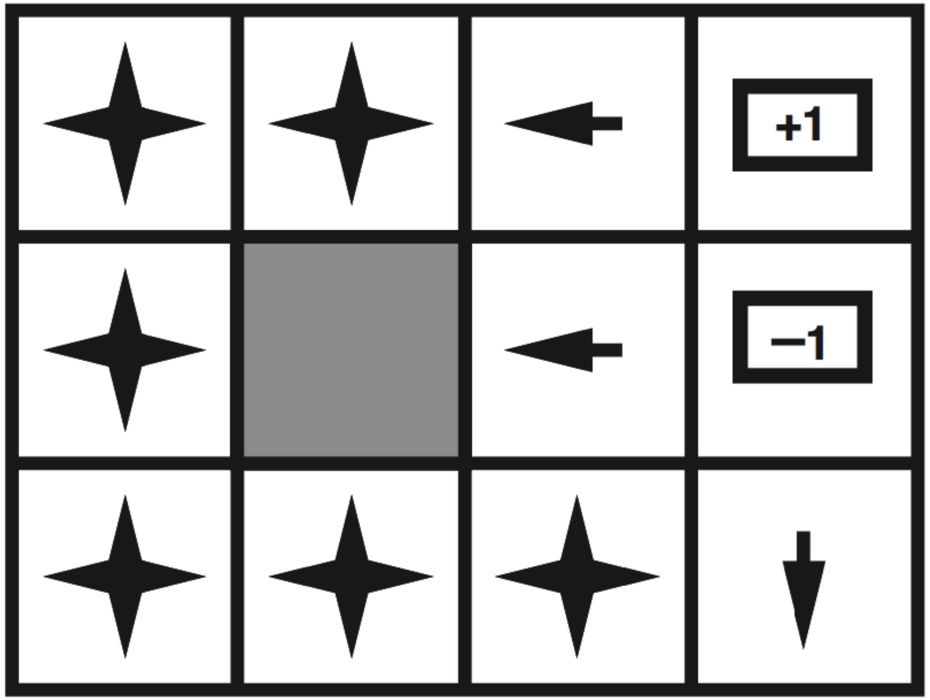


Optimal policy for r > 0



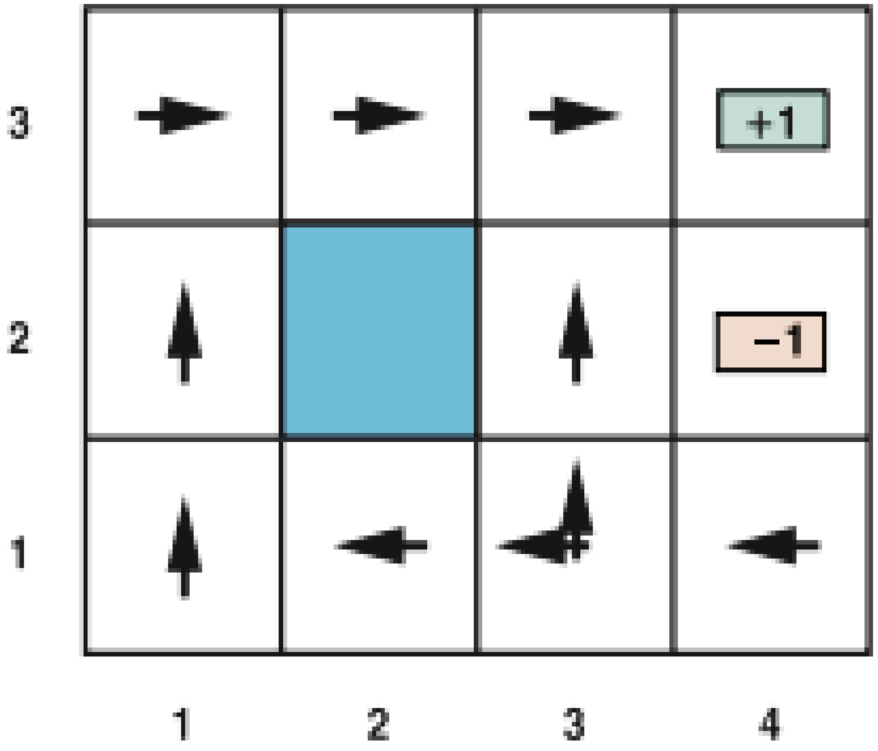
r > 0

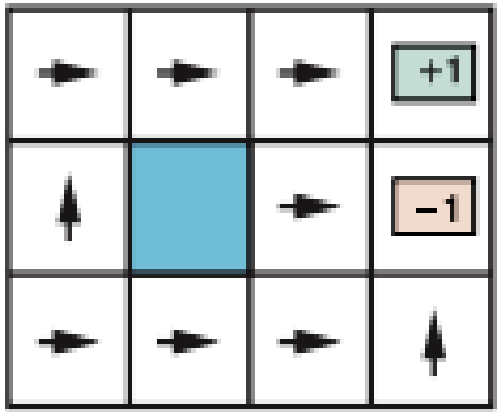
Optimal policy for r>0



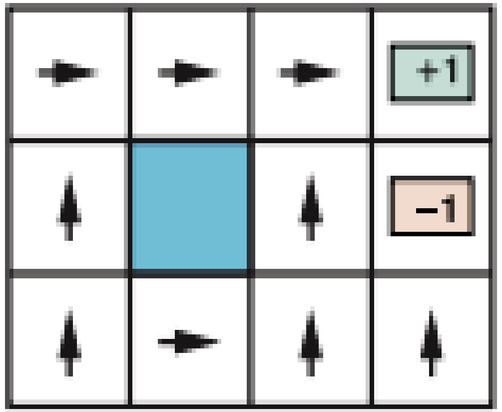
r > 0

Sample Optimal Policies

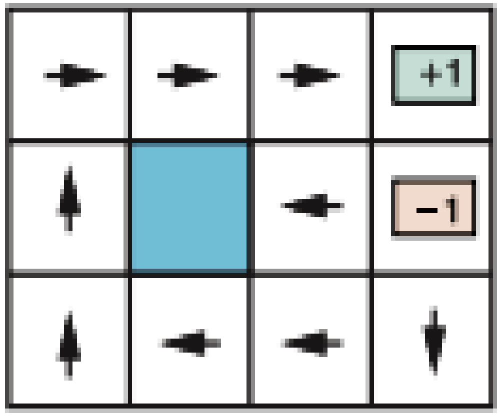




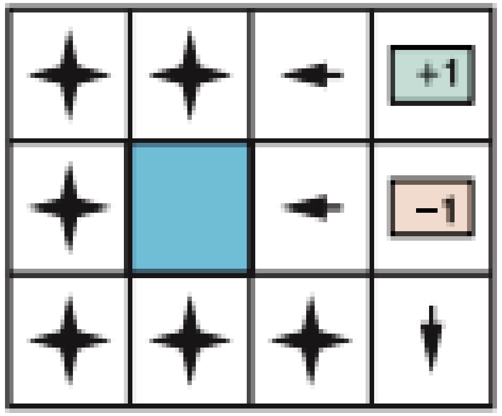
$r < -1.6497$



$-0.7311 < r < -0.4526$



$-0.0274 < r < 0$



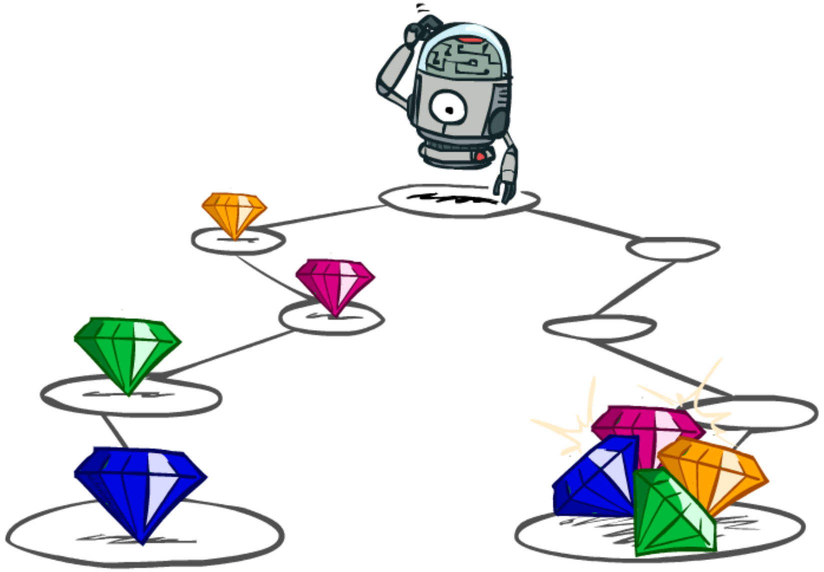
$r > 0$

Utilities of Sequences

- What preferences should an agent have over reward sequences?

- More or less? [1, 2, 2] or [2, 3, 4]

- Now or later? [0, 0, 1] or [1, 0, 0]



Discounting

- Discounting conveniently solves the problem of infinite reward streams!
  - Geometric series: $1 + \gamma + \gamma^2 + \ldots = 1/(1 - \gamma)$
  - Assume rewards bounded by $\pm\, R_{max}$
  - Then $r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots$ is bounded by $\pm\, R_{max}/(1 - \gamma)$

$$U_h([s_0, a_0, s_1, a_1, s_2, \ldots]) = R(s_0, a_0, s_1) + \gamma R(s_1, a_1, s_2) + \gamma^2 R(s_2, a_2, s_3) + \cdots$$

Worth r now

Worth γr next step

Worth $\gamma^2 r$ in two steps

Quiz: Discounting

- Given:

10				1
a	b	c	d	e

  - Actions: East, West, and Exit (only available in exit states a, e)
  - Transitions: deterministic

- Quiz 1: For $\gamma = 1$, what is the optimal policy?

10				1

- Quiz 2: For $\gamma = 0.1$, what is the optimal policy?

10				1

- Quiz 3: For which γ are West and East equally good when in state d?

The utility of a policy

- Executing a policy π from any state s_0 generates a sequence

$s_0, \pi(s_0), s_1, \pi(s_1), s_2, \ldots$

- This corresponds to a sequence of rewards

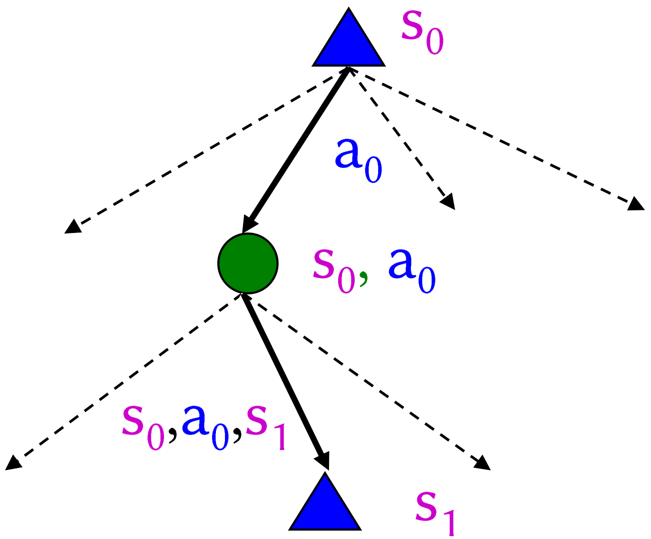
$R(s_0, \pi(s_0), s_1), R(s_1, \pi(s_1), s_2), \ldots$

- This reward sequence happens with probability

$P(s_1 \mid s_0, \pi(s_0)) \times P(s_2 \mid s_1, \pi(s_1)) \times \ldots$

- The value (expected utility) of π in s_0 is written $U^\pi(s_0)$
  - It's the sum over all possible state sequences of

    (discounted sum of rewards) x (probability of state sequence)



$$U^\pi(s_0) = E\left[\sum_{t=0}^{\infty} \gamma^t R(S_t, \pi(S_t), S_{t+1})\right]$$

Optimal Quantities

- The optimal policy:

  $\pi^*(s)$ = optimal action from state s

  Gives highest $U^{\pi}(s)$ for any π

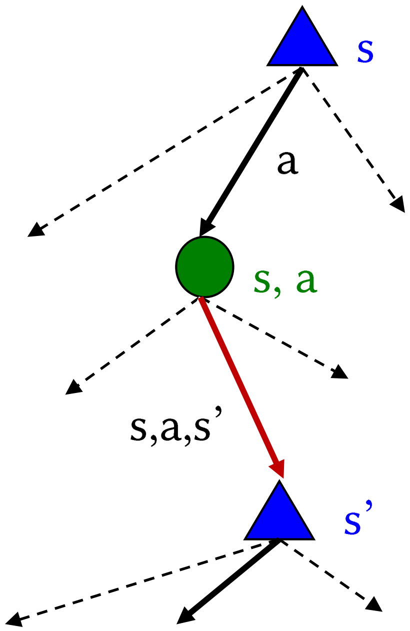
- The value (utility) of a state s:

  $U^*(s) = U^{\pi^*}(s)$ = expected utility starting in s and acting optimally

- The value (utility) of a q-state (s,a):

  $Q^*(s,a)$ = expected utility of taking action a in state s and (thereafter) acting optimally

  $U^*(s) = \max_a Q^*(s,a)$



s is a *state*

(s, a) is a *q-state*

(s,a,s') is a *transition*

Bellman equations (Shapley, 1953)

- The value/utility of a state is
    - The expected reward for the next transition plus the discounted value/utility of the next state, assuming the agent chooses the optimal action

- Hence we have a recursive definition of value (Bellman equation):

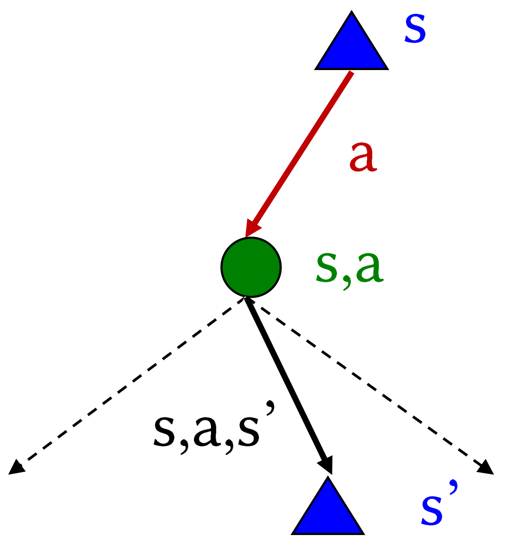
$$U(s) = \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a)[R(s, a, s') + \gamma U(s')]$$

- Similarly, Bellman equation for Q-functions

$$\begin{aligned} Q(s,a) &= \sum_{s'} P(s' \mid s, a)[R(s, a, s') + \gamma U(s')] \\ &= \sum_{s'} P(s' \mid s, a)[R(s, a, s') + \gamma \max_{a'} Q(s', a')] \end{aligned}$$

Value Iteration

- Start with (say) $U_0(s) = 0$ and some termination parameter ε
- Repeat until convergence (i.e., until all updates smaller than ε)
  - Do a **Bellman update** (essentially one ply of expectimax) from each state:
  - $U_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s' \mid a,s) \, [R(s,a,s') + \gamma U_k(s')]$



- Theorem: will converge to unique optimal values

Extracting policy

- How should the agent act given **U** (s)?
- Maximize expected utility! (as if **U** is correct)
- That is, do a mini-expectimax (greedy one-step):

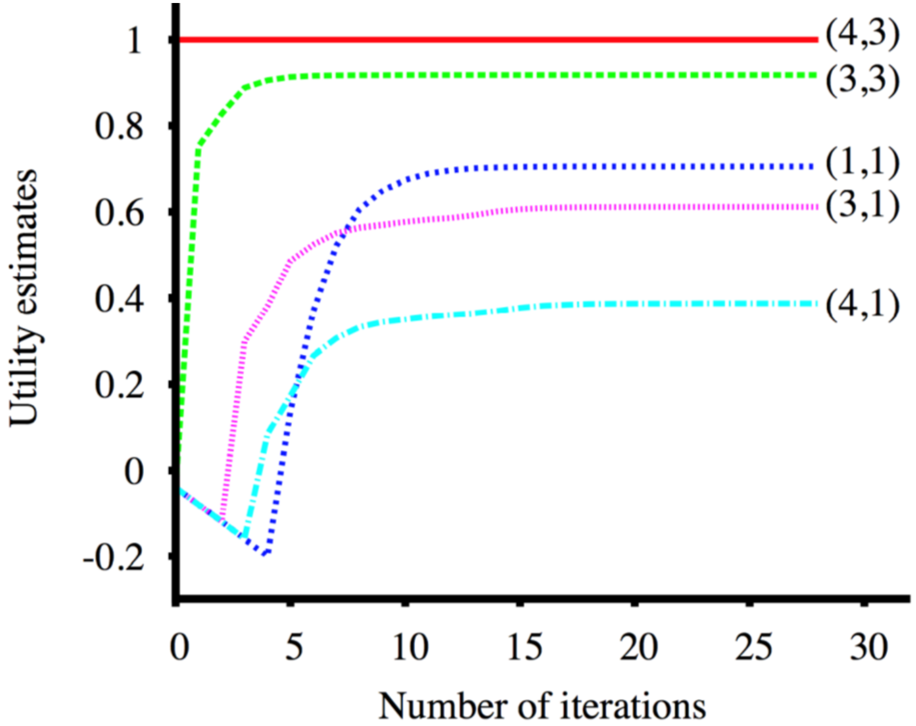
$$\pi_U(s) = \text{argmax}_a \sum_{s'} P(s' \mid a,s) [R(s,a,s') + \gamma \mathbf{U}(s')]$$

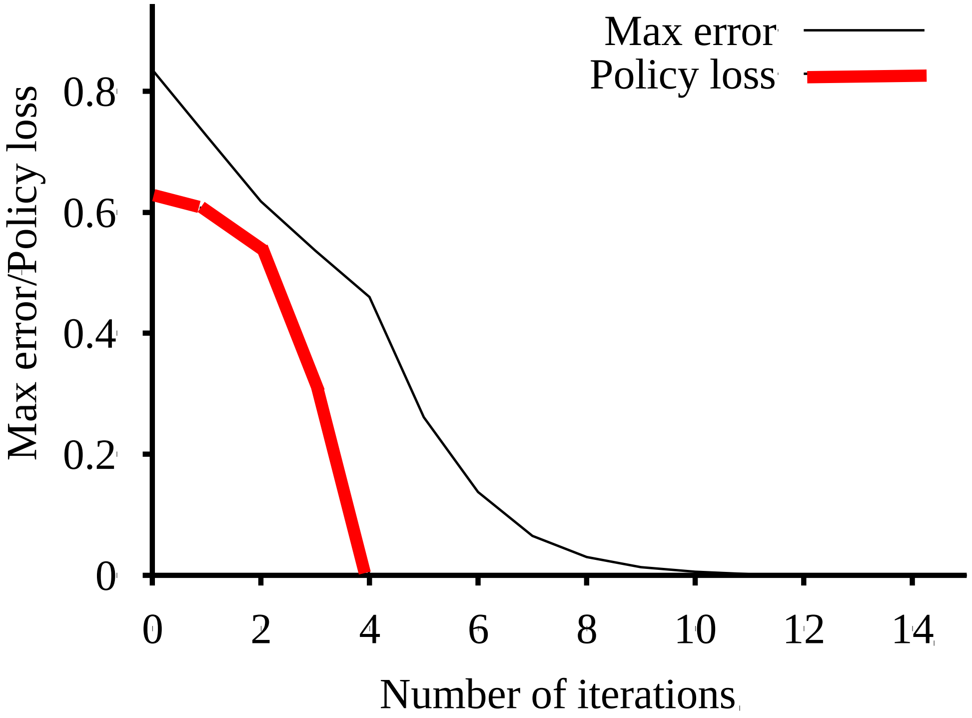
- This is called **policy extraction**, since it finds the policy π_U implied by the values **U**

How good is the policy extracted from VI?

- The quality of a policy π is measured by the policy loss $\| U^{\pi} - U^{*} \|$
- Let $\pi_k = \pi_{U_k}$ i.e. the implied policy at step k
  - When $\| U_k - U^* \| \leq \varepsilon$, policy loss is bounded:

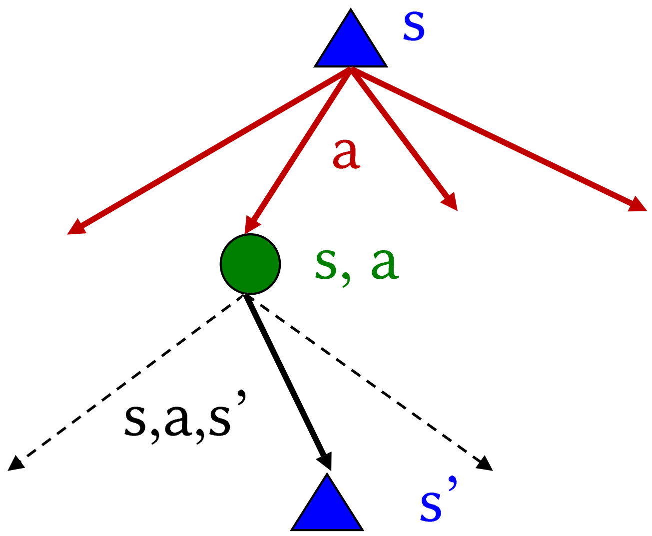
$$\| U^{\pi_k} - U^* \| \leq 2\varepsilon\gamma/(1-\gamma)$$



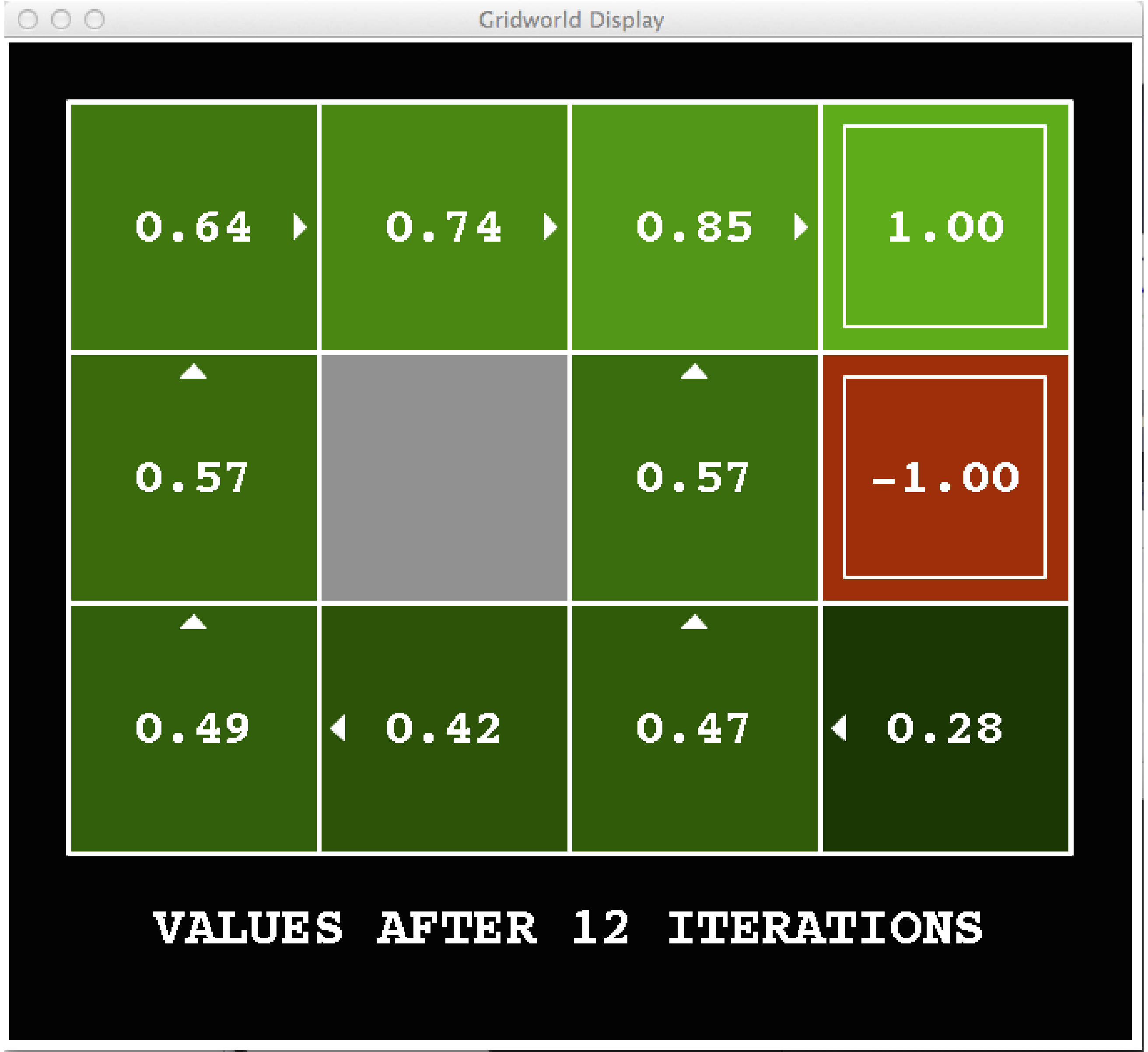


Problems with Value Iteration

- Value iteration repeats the Bellman updates:
  - $U_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s' \mid a,s)\,[R(s,a,s') + \gamma U_k(s')]$
- Problem 1: It's slow – $O(S^2A)$ per iteration
- Problem 2: The "max" at each state rarely changes
- Problem 3: The policy often converges long before the values

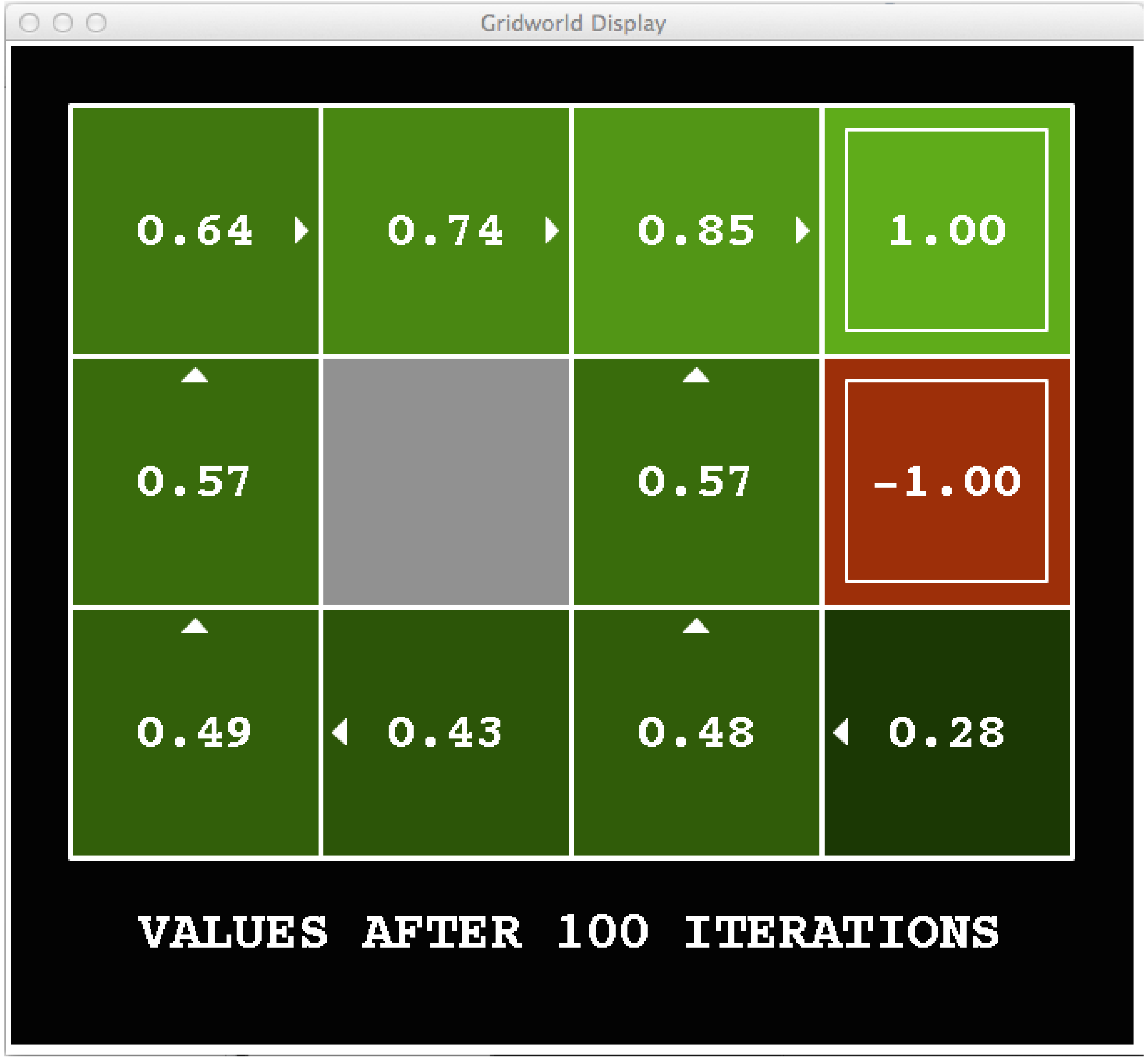


Policy Iteration



Noise = 0.2
Discount = 0.9
Living reward = 0

Policy Iteration



Noise = 0.2
Discount = 0.9
Living reward = 0

Policy Iteration

- Basic idea: make the implied policy in U explicit, compute its long-term implications for value
- Repeat until no change in policy:
  - Step 1: Policy evaluation: calculate value U^{π_k} for current policy π_k
  - Step 2: Policy improvement: extract the new implied policy π_{k+1} from U^{π_k}

- It's still optimal!
- Can converge (much) faster under some conditions

Quiz

- Which of the following correctly defines a Markov Decision Process (MDP)?

  A. A deterministic search problem with a single terminal state
  B. A probabilistic model defined by states, actions, rewards, and transition probabilities
  C. A logical model defined by inference rules and operators
  D. A supervised learning task with labeled examples

- What does the Markov property state in an MDP?

  A. The future is independent of the past, given the present state.
  B. The future depends on all previous actions equally.
  C. Each action has deterministic outcomes.
  D. Rewards depend only on the initial state.

Quiz

- Which of the following correctly defines a Markov Decision Process (MDP)?

  A. A deterministic search problem with a single terminal state
  B. A probabilistic model defined by states, actions, rewards, and transition probabilities
  C. A logical model defined by inference rules and operators
  D. A supervised learning task with labeled examples

- What does the Markov property state in an MDP?

  A. The future is independent of the past, given the present state.
  B. The future depends on all previous actions equally.
  C. Each action has deterministic outcomes.
  D. Rewards depend only on the initial state.

Quiz

- Which of the following expresses the Bellman optimality condition?
  A. $V(s)=R(s)$
  B. $V_*(s)=\max_a \Sigma_{s'} P(s'|s,a)[R(s,a,s')+\gamma V_*(s')]$
  C. $Q(s,a)=R(s,a)$
  D. $V_*(s)=\Sigma_a P(a|s)R(s,a)$
- How do value iteration and policy iteration differ?
  A. Value iteration alternates policy evaluation and improvement; policy iteration updates all values simultaneously.
  B. Policy iteration alternates between policy evaluation and improvement; value iteration merges them into a single update loop.
  C. Policy iteration uses exploration; value iteration does not.
  D. Value iteration is stochastic; policy iteration is deterministic.

Quiz

- Which of the following expresses the Bellman optimality condition?
  A. $V(s)=R(s)$
  B. $V*(s)=\max_a \Sigma_{s'} P(s'|s,a)[R(s,a,s')+\gamma V*(s')]$
  C. $Q(s,a)=R(s,a)$
  D. $V*(s)=\Sigma_a P(a|s)R(s,a)$
- How do value iteration and policy iteration differ?
  A. Value iteration alternates policy evaluation and improvement; policy iteration updates all values simultaneously.
  B. Policy iteration alternates between policy evaluation and improvement; value iteration merges them into a single update loop.
  C. Policy iteration uses exploration; value iteration does not.
  D. Value iteration is stochastic; policy iteration is deterministic.

Quiz

- Why are MDPs central to modern AI?
  A. They describe one-shot classification problems.
  B. They model reasoning under certainty.
  C. They formalize sequential decision-making under uncertainty — the foundation of reinforcement learning.
  D. They only apply to deterministic planning tasks.

Quiz

- Why are MDPs central to modern AI?

  A. They describe one-shot classification problems.

  B. They model reasoning under certainty.

  C. They formalize sequential decision-making under uncertainty — the foundation of reinforcement learning.

  D. They only apply to deterministic planning tasks.

Summary

- An MDP provides a mathematical framework for sequential decision making when outcomes are partly random and partly under the agent's control.
  - It formalizes how a rational agent should act to maximize long-term expected utility.
- Components of an MDP are States (S), Actions (A), Transition Model ($P(s' \mid s, a)$), Reward Function ($R(s, a, s')$) and Discount Factor ($0 \le \gamma < 1$).
- The Objective is to find a policy **π** that maps each state to the optimal action, maximizing the expected discounted sum of future rewards
- The Bellman Equations give the optimal value function satisfies the recursive relationship: $U^*(s) = \max_a \Sigma_{s'} P(s'|s,a)[R(s,a,s') + \gamma\, U^*(s')]$
  - The utility of a state equals its immediate reward plus the expected discounted value of the next state.
- Solving an MDP
  - Value Iteration: iteratively apply Bellman updates until values converge.
  - Policy Iteration: alternate between evaluating a policy and improving it.