Markov Decision Process (MDP)

Prof. Seungchul Lee
Industrial AI Lab.

Source

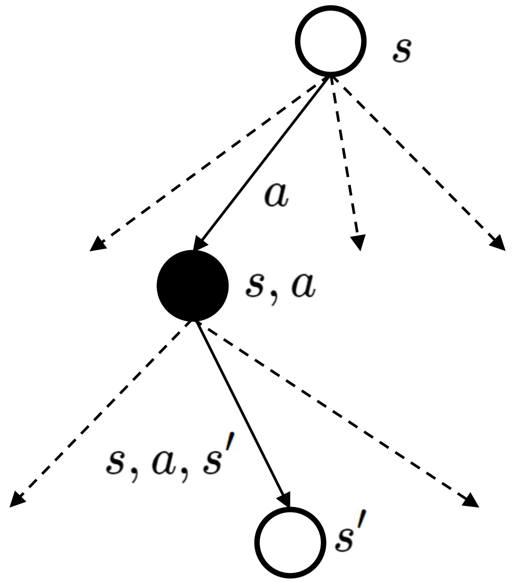
- David Silver's Lecture (DeepMind)
  - UCL homepage for slides (http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html)
  - DeepMind for RL videos (https://www.youtube.com/watch?v=2pWv7GOvuf0)
  - An Introduction to Reinforcement Learning, Sutton and Barto pdf

- CMU by Zico Kolter
  - http://www.cs.cmu.edu/~zkolter/course/15-780-s14/lectures.html
  - https://www.youtube.com/watch?v=un-FhSC0HfY&hd=1

- Deep RL Bootcamp by Rocky Duan
  - https://sites.google.com/view/deep-rl-bootcamp/home
  - https://www.youtube.com/watch?v=qO-HUo0LsO4

- Stanford Univ. by Serena Yeung
  - https://www.youtube.com/watch?v=lvoHnicueoE&list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv&index=15&t=1337s

Markov Decision Process



- So far, we analyzed the passive behavior of a Markov chain with rewards
- A Markov decision process (MDP) is a Markov reward process with decisions (or actions).
  - MDP = MRP + action

Definition: A Markov Decision Process is a tuple $\langle S, A, P, R, \gamma \rangle$

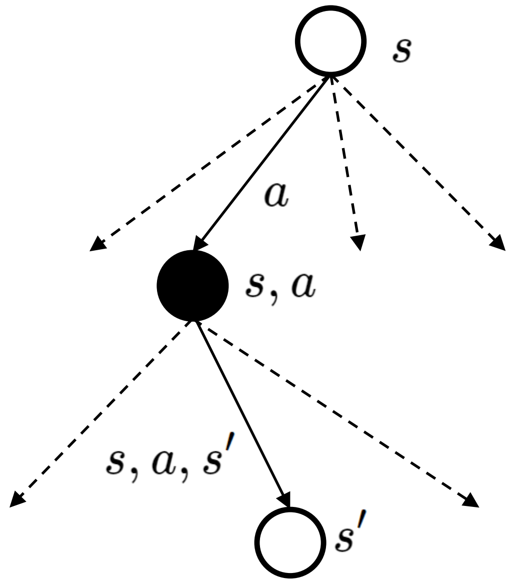
- S is a finite set of states
- A is a finite set of actions
- P is a state transition probability matrix

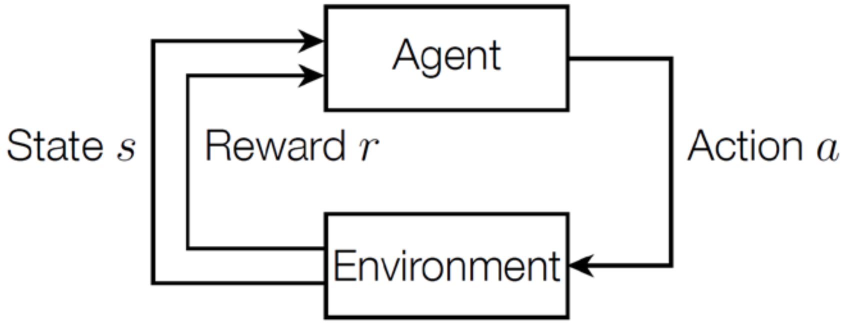
$$P_{ss'}^{a} = P\left[S_{t+1} = s' \mid S_t = s, A_t = a\right]$$

- R is a reward function, $R_s^a = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$ $\quad (= \mathbb{E}\left[R_{t+1} \mid S_t = s\right], \text{ often assumed})$
- γ is a discount factor, $\gamma \in [0, 1]$

  - It is an environment in which all states are Markov

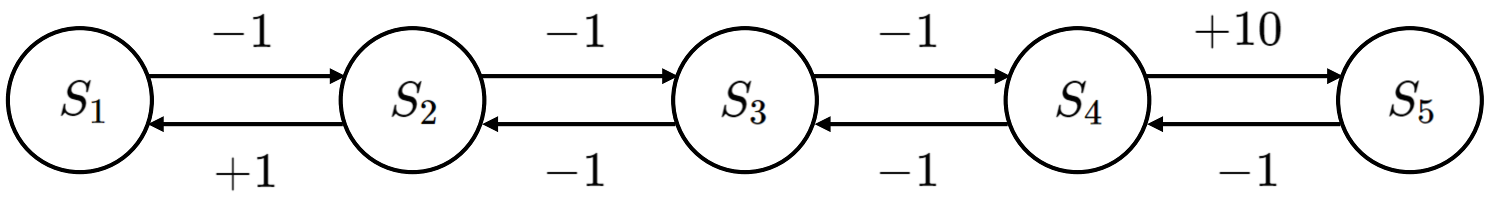
Markov Decision Process





Example: Mars Rover MDP

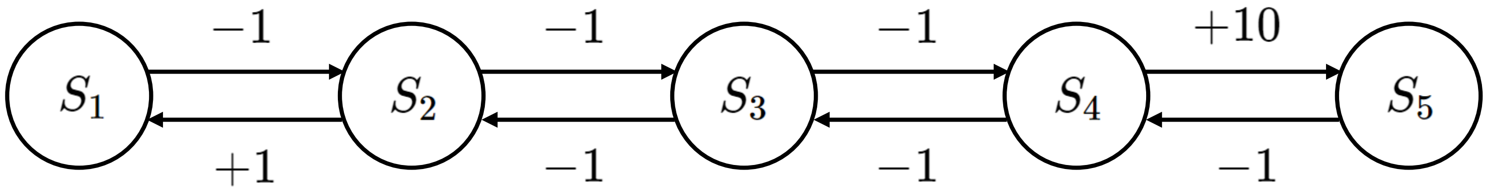
- Discount factor γ
- Two actions: Left and Right
- Reward: When the rover has an action, it achieves +1 in S_1, +10 in S_5, -1 in all others



Example: Mars Rover MDP

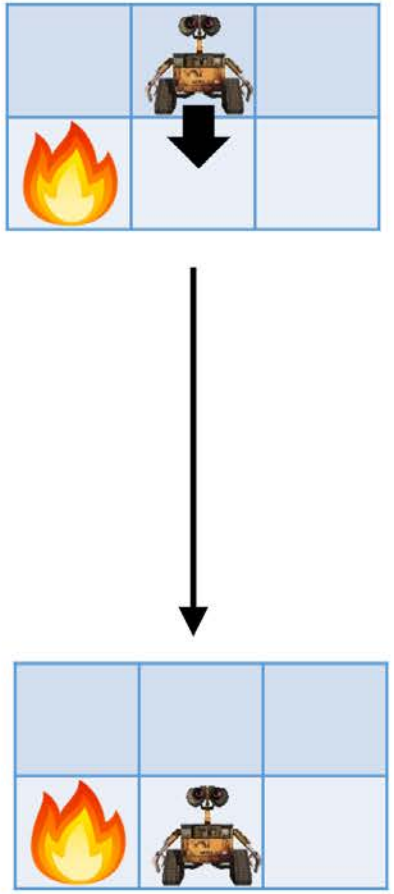
- Deterministic state transition matrix

$$P(s' \mid s, L) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \qquad P(s' \mid s, R) = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

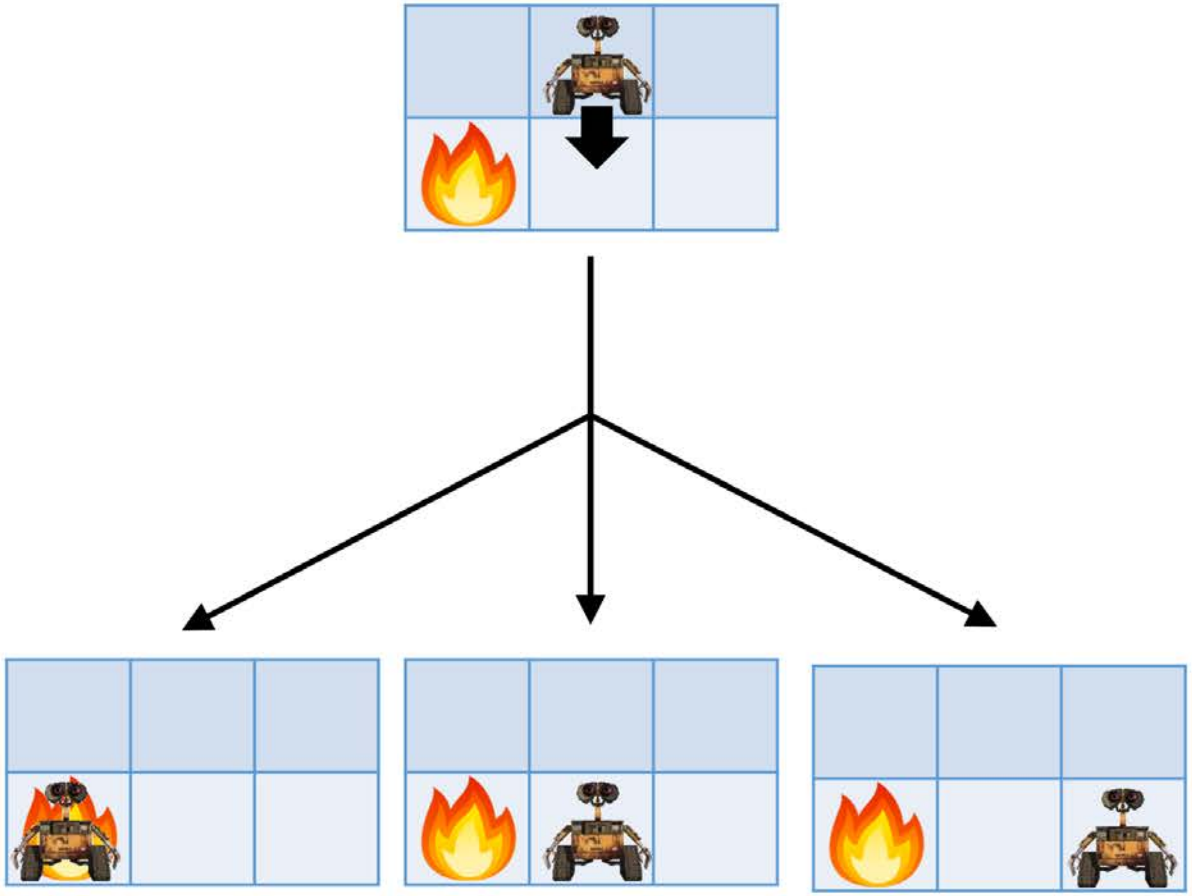


Example: Grid World Actions

Deterministic grid world



Stochastic grid world

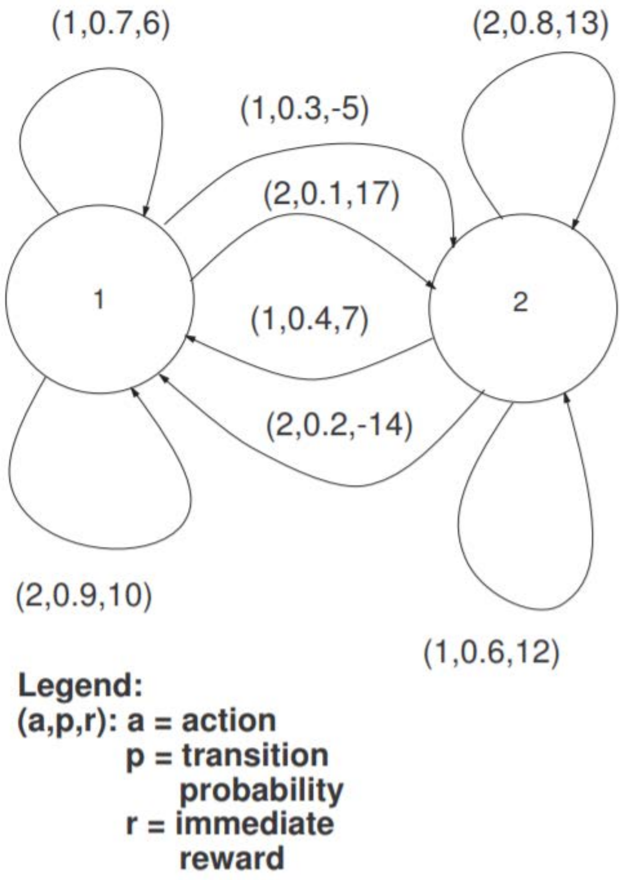


Example

- $P_a (= P^a)$: transition probability matrix for action a
- $R_a (= R^a)$: transition reward matrix for action a

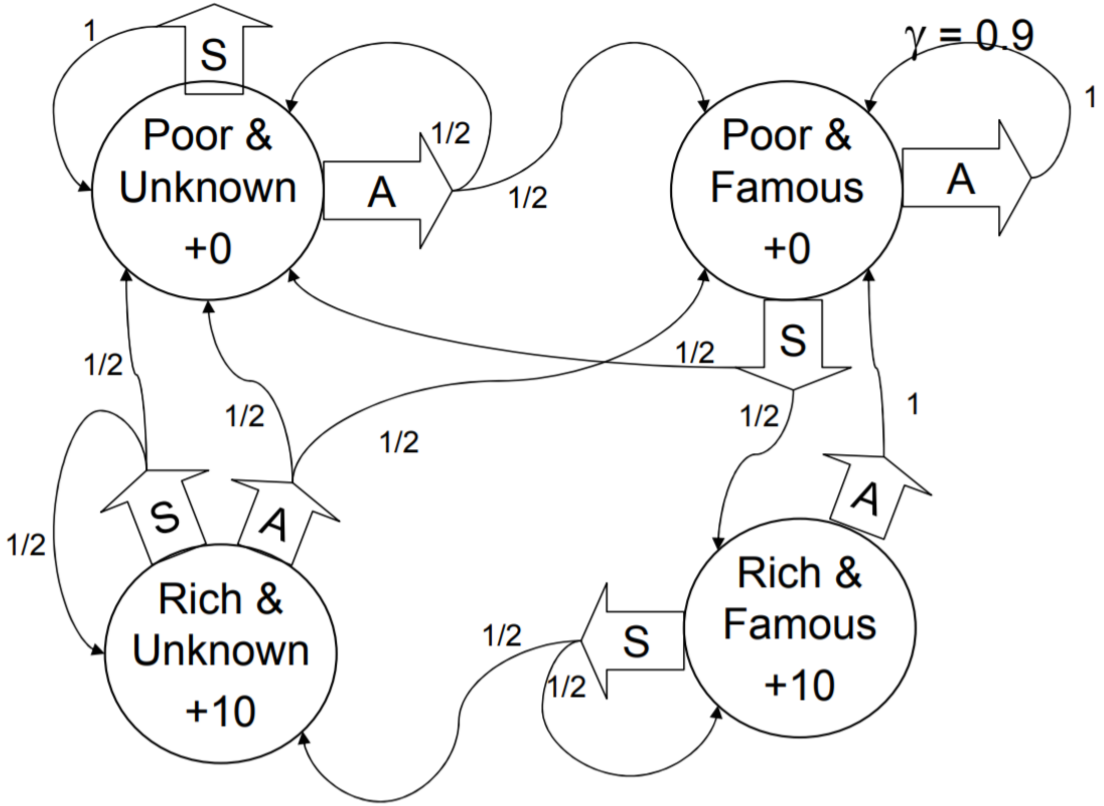
$$\mathbf{P}_1 = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix}; \mathbf{P}_2 = \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix};$$

$$\mathbf{R}_1 = \begin{bmatrix} 6 & -5 \\ 7 & 12 \end{bmatrix}; \mathbf{R}_2 = \begin{bmatrix} 10 & 17 \\ -14 & 13 \end{bmatrix}.$$



Example

- You run a startup company.
  – In every state, you must choose between Saving money or Advertising

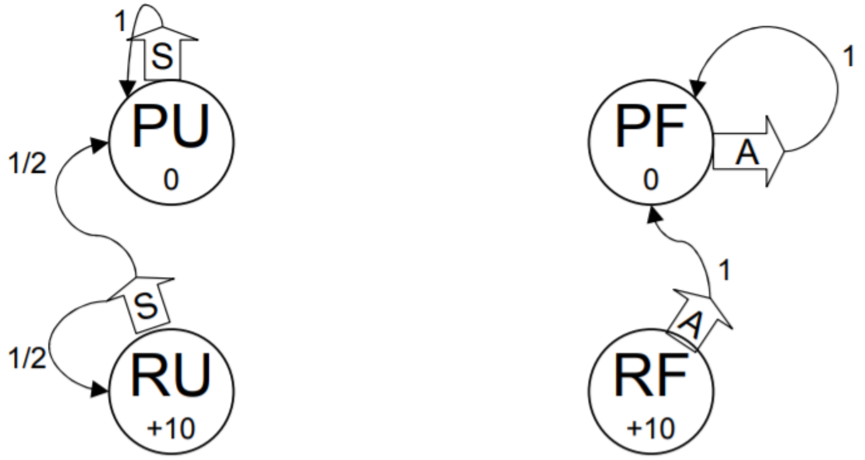


Policy

- A policy is a mapping from states to actions, $\pi: S \rightarrow A$
- Example: two policies

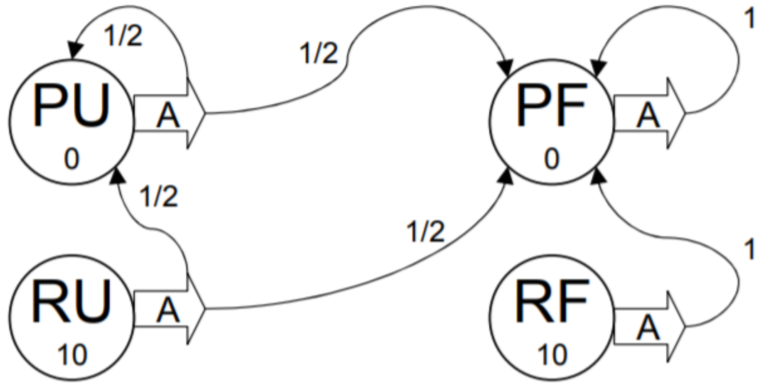
Policy Number 1:

STATE →	ACTION
PU	S
PF	A
RU	S
RF	A



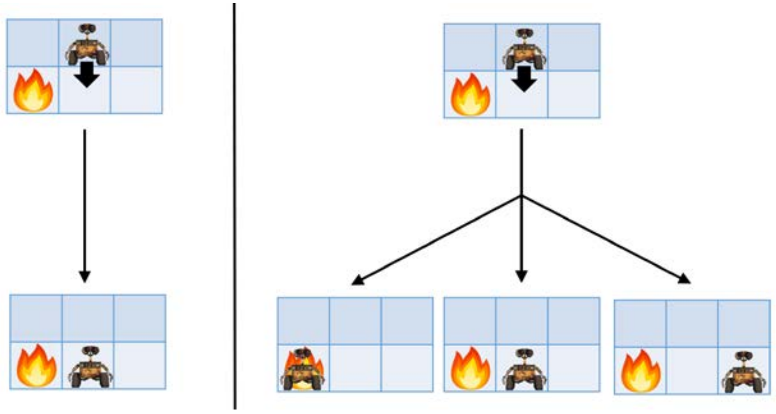
Policy Number 2:

STATE →	ACTION
PU	A
PF	A
RU	A
RF	A



Policy

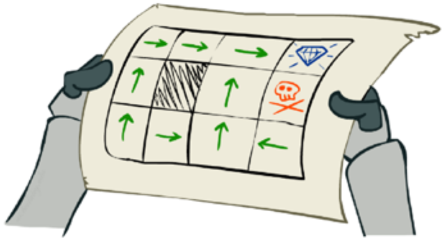
- A policy is a mapping from states to actions, $\pi: S \rightarrow A$



- A policy fully defines the behavior of an agent
  - It can be deterministic or stochastic

- Given a state, it specifies a distribution over actions

$$\pi(a \mid s) = P(A_t = a \mid S_t = s)$$



- MDP policies depend on the current state (not the history)
- Policies are stationary (time-independent, but it turns out to be optimal)

Policy

- A policy is a mapping from states to actions, $\pi: S \rightarrow A$

- A policy fully defines the behavior of an agent
  - It can be deterministic or stochastic

- Let P^{π} be a matrix containing probabilities for each transition under policy π

- Given an MDP $\mathcal{M} = \langle S, A, P, R, \gamma \rangle$ and a policy π
  - The state sequence $s_1, s_2, \cdots$ is a Markov process $\langle S, P^{\pi} \rangle$
  - The state and reward sequence is a Markov reward process $\langle S, P^{\pi}, R^{\pi}, \gamma \rangle$

Questions on MDP Policy

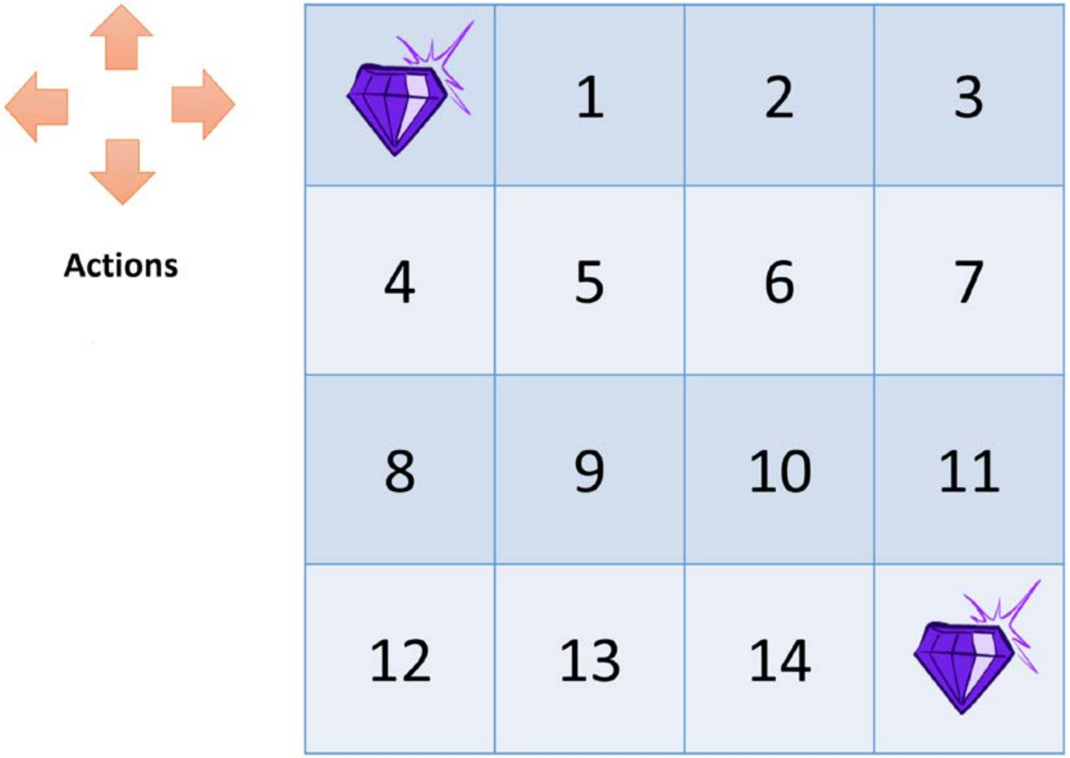
- How many possible policies in our example?
- Which of the above two policies is best?
- How do you compute the *optimal* policy?

Examples: Mars Rover Polices

- How many possible policies in our example?

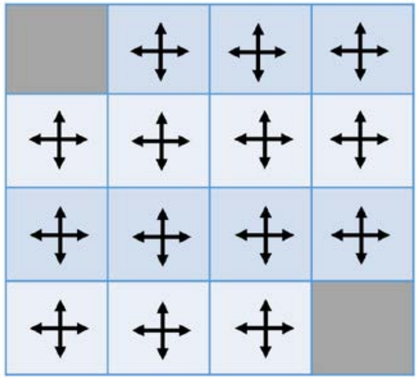
- Which of the above policies is best?

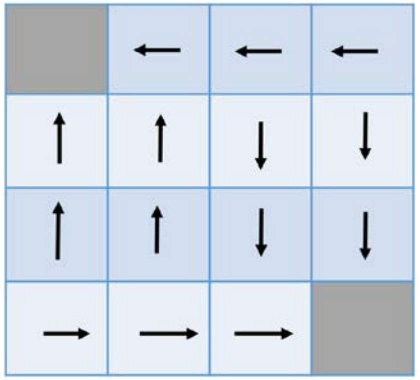
Example: Small Grid World

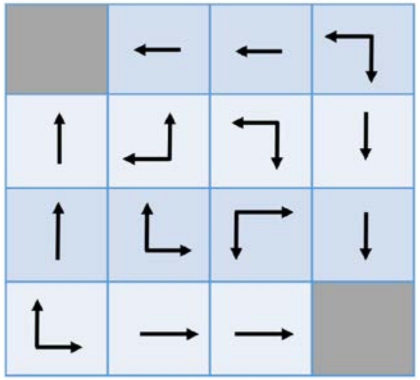


Example: Possible Policies

- How many possible policies are there in the grid world?
  - For every state, assume that the probabilities of actions are equal.





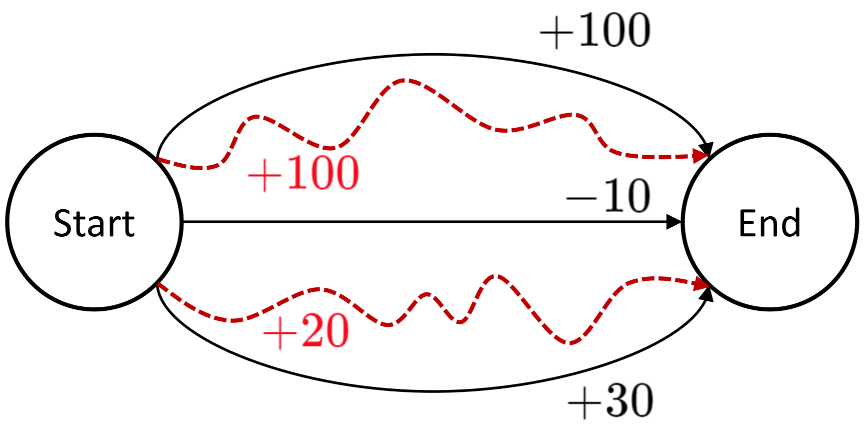


Value Function: State-Value Function

- The state-value function $v_\pi(s)$ of an MDP is the expected return starting from state s, and then following policy π

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi\left[G_t \mid S_t = s\right] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] \end{aligned}$$

- Example



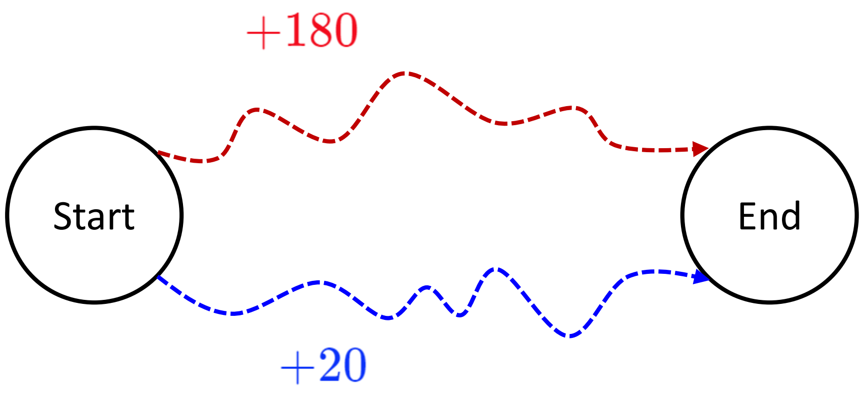
$$v(\text{Start}) = \frac{100 - 10 + 30}{3} = +40$$

$$v_\pi(\text{Start}) = \frac{100 + 20}{2} = +60$$

Value Function: Action-Value Function

- The action-value function $q_\pi(s, a)$ of an MDP is the expected return starting from state s, taking action a, and then following policy π

$$\begin{aligned} q_\pi(s,a) &= \mathbb{E}_\pi \left[G_t \mid S_t = s, A_t = a \right] \\ &= \mathbb{E}_\pi \left[R_{t+1} + \gamma q_\pi \left(S_{t+1}, A_{t+1} \right) \mid S_t = s, A_t = a \right] \end{aligned}$$



$q_\pi(\text{Start}, \text{Right}) = +180$

$q_\pi(\text{Start}, \text{Left}) = +20$

Bellman Expectation Equation

Richard Ernest Bellman

Value Functions for policy π

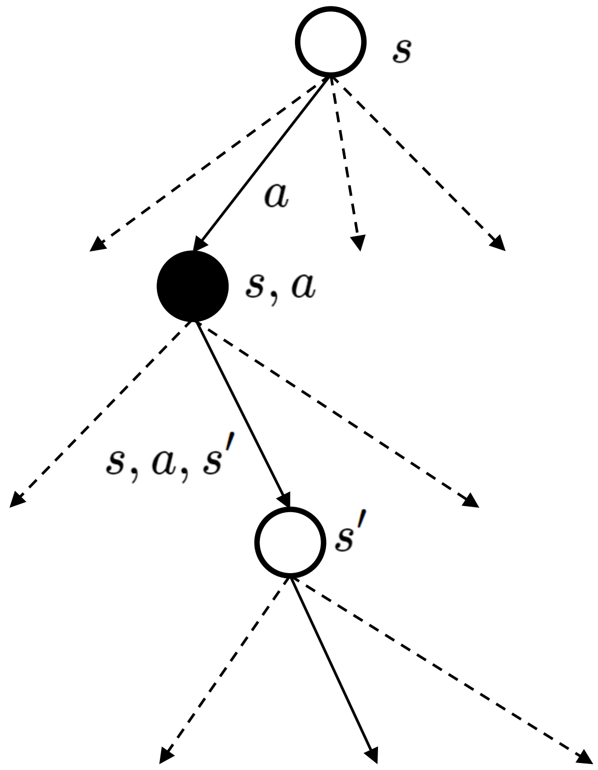
- Given the policy π, the value function can again be decomposed into immediate reward plus discounted value of successor state (recursively)

- The state-value function $v_\pi(s)$ for policy π
  - Expected return from staring in state under policy π

$$v_\pi(s) = \mathbb{E}\left[R_{t+1} + \gamma v_\pi\left(S_{t+1}\right) \mid S_t = s\right]$$

- The action-value function $q_\pi(s, a)$ for policy π
  - Expected return from starting in state s, taking action a under policy π

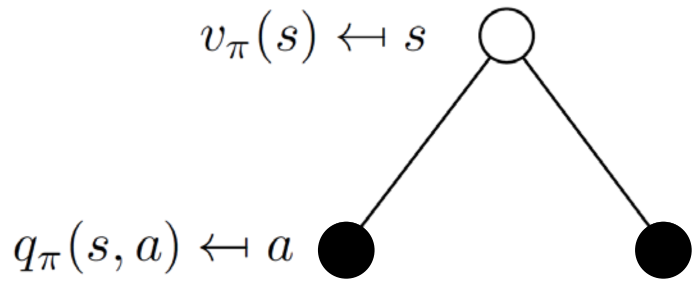
$$q_\pi(s, a) = \mathbb{E}\left[R_{t+1} + \gamma q_\pi\left(S_{t+1}, A_{t+1}\right) \mid S_t = s, A_t = a\right]$$

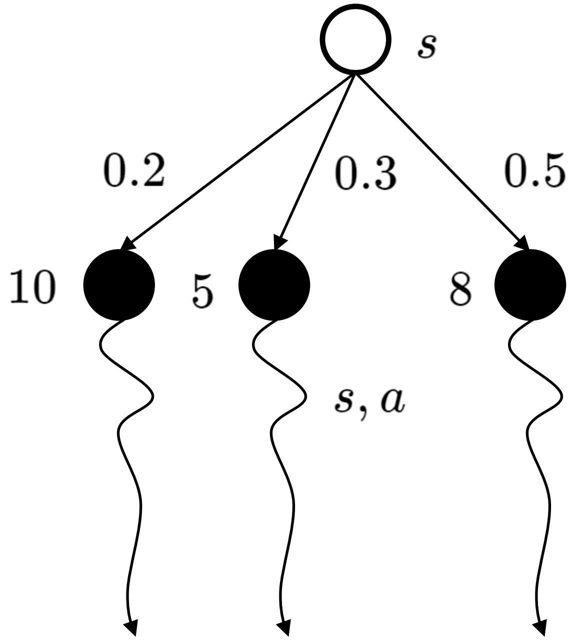


Relationship between $v_\pi(s)$ and $q_\pi(s,a)$

- State-value function using policy π

$$v_\pi(s) = \sum_{a \in A} \pi(a \mid s) q_\pi(s,a)$$



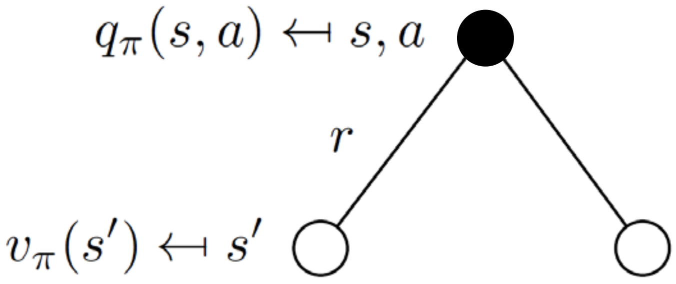


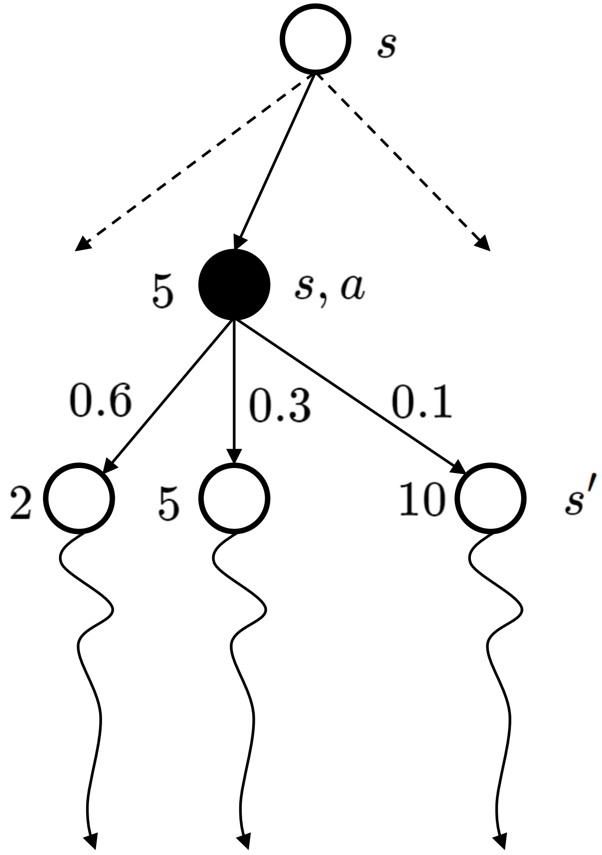
- $v_\pi(s) = 0.2 \times 10 + 0.3 \times 5 + 0.5 \times 8 = 7.5$

Relationship between $v_\pi(s)$ and $q_\pi(s,a)$

- Action-value function using policy π

$$q_\pi(s,a) = R(s) + \gamma \sum_{s' \in S} P(s' \mid s,a) v_\pi(s')$$



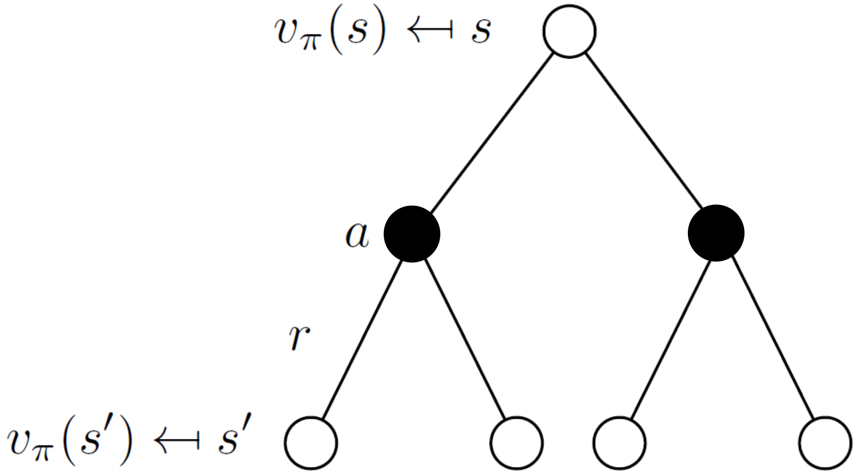


- $q_\pi(s,a) = 5 + \gamma \times (0.6 \times 2 + 0.3 \times 5 + 0.1 \times 10)$

Bellman Expectation Equation for $v_\pi(s)$

$$v_\pi(s) = \sum_{a \in A} \pi(a \mid s) \underline{q_\pi(s, a)}$$

$$= \sum_{a \in A} \pi(a \mid s) \left(R(s) + \gamma \sum_{s' \in S} P\left(s' \mid s, a\right) v_\pi\left(s'\right) \right)$$

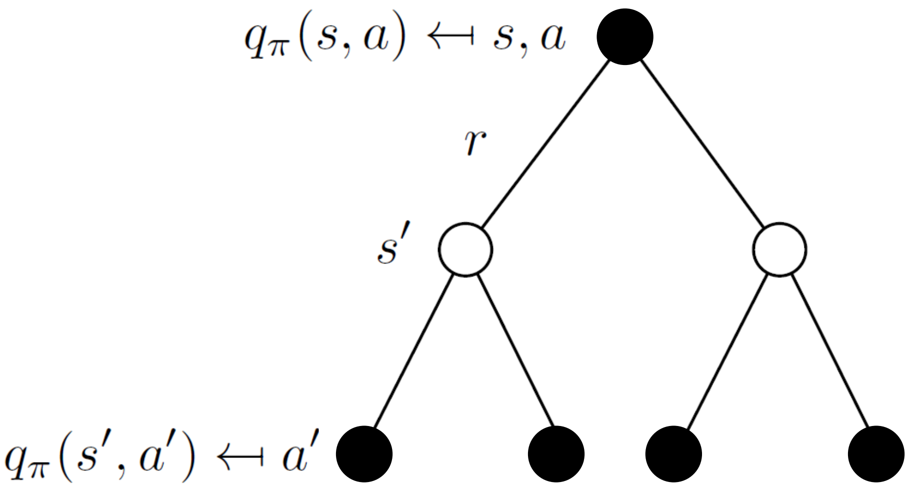


$$\boxed{q_\pi(s, a) = R(s) + \gamma \sum_{s' \in S} P(s' \mid s, a) v_\pi(s')}$$

Bellman Expectation Equation for $q_\pi(s, a)$

$$q_\pi(s,a) = R(s) + \gamma \sum_{s' \in S} P(s' \mid s, a)\, \underline{v_\pi(s')}$$

$$= R(s) + \gamma \sum_{s' \in S} P(s' \mid s, a) \left(\sum_{a' \in A} \pi(a' \mid s')\, q_\pi(s', a') \right)$$



$$\boxed{v_\pi(s) = \sum_{a \in A} \pi(a \mid s) q_\pi(s,a)}$$

Solving the Bellman Expectation Equation

- The Bellman expectation equation can be expressed concisely in a matrix form

$$v_\pi = R + \gamma P^\pi v_\pi \quad \Longrightarrow \quad v_\pi = (I - \gamma P^\pi)^{-1} R$$

- Iterative

$$v_\pi(s) \leftarrow R(s) + \gamma \sum_{s' \in S} P(s' \mid s, a) \; v_\pi(s')$$

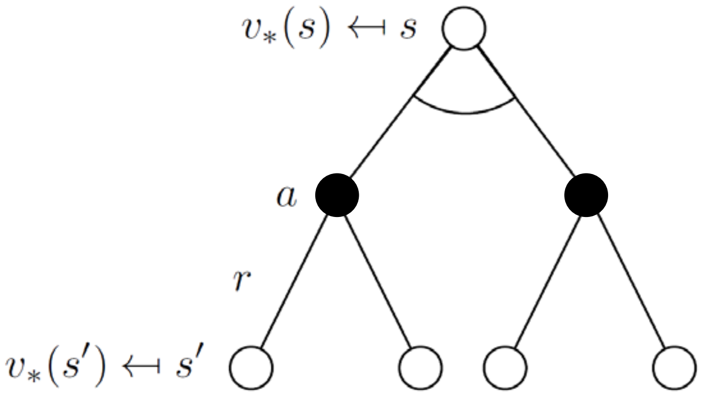
Bellman Optimality Equation

Richard Ernest Bellman

Bellman Optimality Equation for $v_\pi(s)$

- The optimal state-value function $v_*(s)$ is the maximum value function over all polices

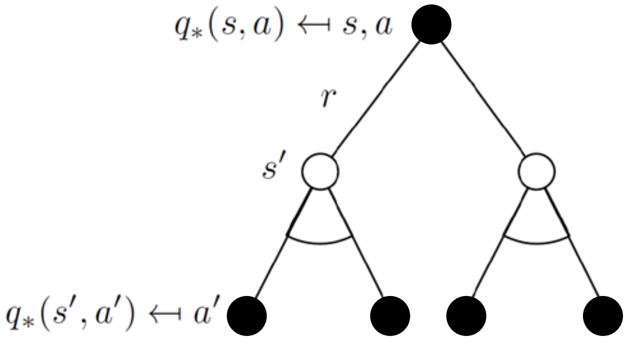
$$
\begin{aligned}
v_*(s) &= \max_\pi v_\pi(s) \\
&= \max_a q_\pi(s,a) \\
&= \max_a \left(R(s) + \gamma \sum_{s' \in S} P(s' \mid s,a) v_\pi(s') \right) \\
&= R(s) + \gamma \max_a \sum_{s' \in S} P(s' \mid s,a) v_\pi(s')
\end{aligned}
$$

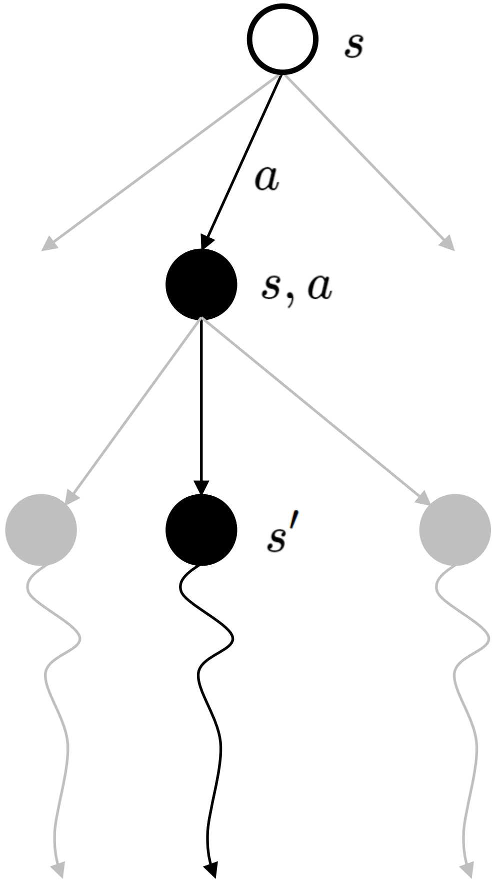


Bellman Optimality Equation for $q_\pi(s, a)$

- The optimal action-value function $q_*(s, a)$ is the maximum action-value function over all policies.

$$
\begin{aligned}
q_*(s,a) &= \max_\pi q_\pi(s,a) \\
&= \max_\pi \left(R(s) + \gamma \sum_{s' \in S} P(s' \mid s,a) v_\pi(s') \right) \\
&= R(s) + \gamma \sum_{s' \in S} P(s' \mid s,a) \max_\pi v_\pi(s')
\end{aligned}
$$





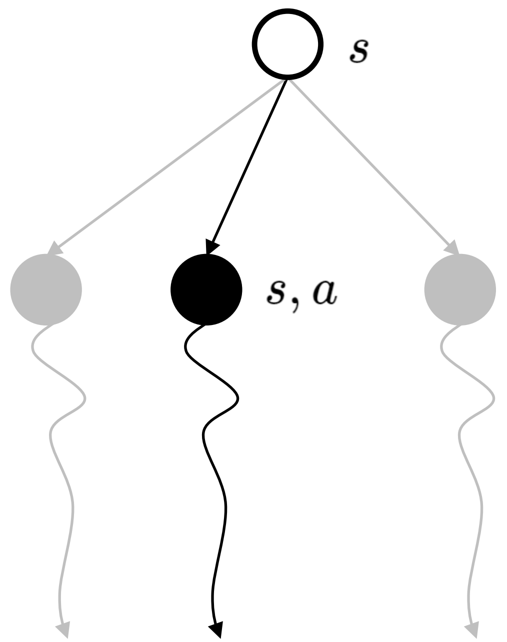
Optimal Policy

- The optimal policy is the policy that achieves the highest value for every state

$$\pi_*(s) = \arg\max_{\pi} v_\pi(s)$$

  and its optimal value function is written $v_*(s)$

- An optimal action for each state can be found by maximizing over $q_*(s,a)$



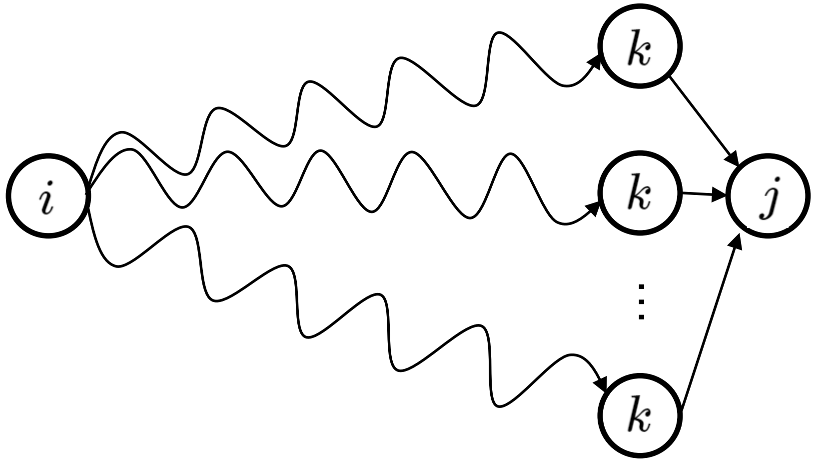
$$\pi_*(a \mid s) = \begin{cases} 1 & \text{if } a = \arg\max_{a \in A} q_*(s,a) \\ 0 & \text{otherwise} \end{cases}$$

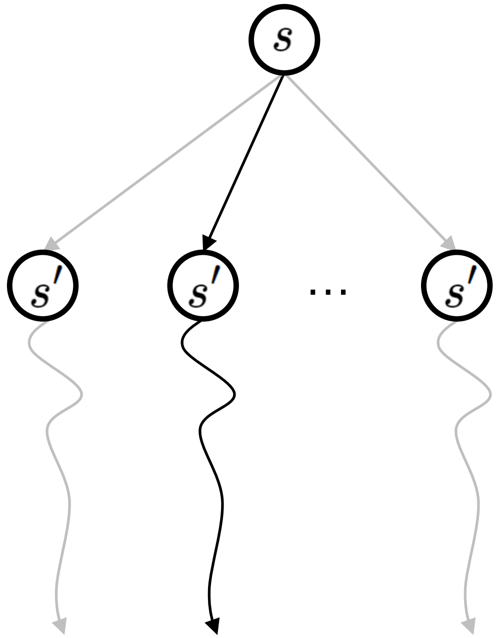
- There is always a deterministic optimal policy for any MDP
- If we know $q_*(s,a)$, we can have the optimal policy

The Principle of Optimality

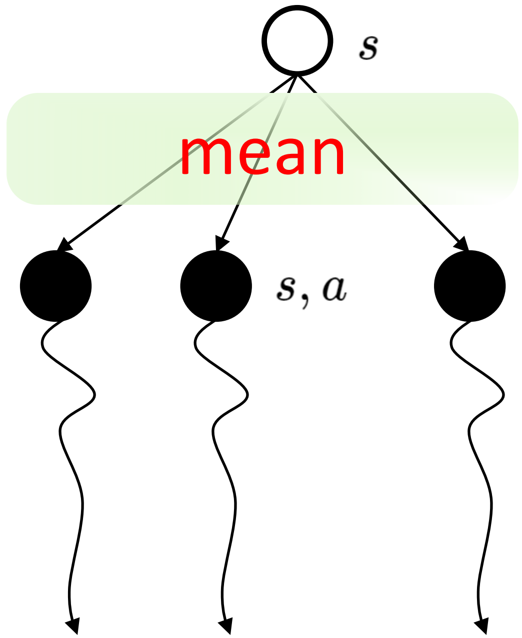
- Shortest path

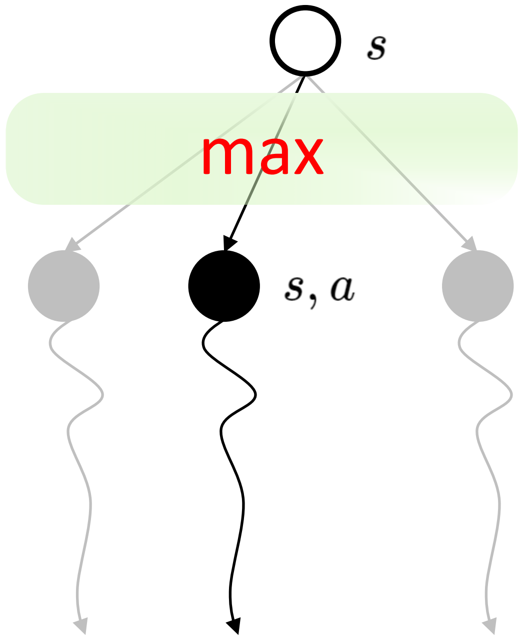
$$v_*(s) = \max_a \left(R(s) + \gamma \sum_{s' \in S} P(s' \mid s, a) v_\pi(s') \right)$$
$$= R(s) + \gamma \max_a \sum_{s' \in S} P(s' \mid s, a) v_\pi(s')$$





Summary: Expectation vs. Optimality





Solving the Bellman Optimality Equation

Optimal Policy and Optimal Value Function (1/2)

- The optimal policy is the policy that achieves the highest value for every state

$$\pi_*(s) = \arg\max_{\pi} v_\pi(s)$$

and its optimal value function is written $v_*(s)$

- We can directly define the *optimal value function* using Bellman optimality equation

$$v_*(s) = R(s) + \gamma \max_a \sum_{s' \in S} P(s' \mid s, a)\, v_*\,(s')$$

and *optimal policy* is simply the action that attains this max

$$\pi_*(s) = \arg\max_a \sum_{s' \in S} P(s' \mid s, a)\, v_*(s')$$

Optimal Policy and Optimal Value Function (2/2)

- We can directly define the *optimal value function* using Bellman optimality equation

$$v_*(s) = R(s) + \gamma \max_a \sum_{s' \in S} P(s' \mid s, a)\, v_*(s')$$

and *optimal policy* is simply the action that attains this max

$$\pi_*(s) = \arg\max_a \sum_{s' \in S} P(s' \mid s, a)\, v_*(s')$$

$$q_\pi(s, a) = R(s) + \gamma \sum_{s' \in S} P(s' \mid s, a) v_\pi(s')$$

$$\pi_*(a \mid s) = \begin{cases} 1 & \text{if } a = \arg\max_{a \in A} q_*(s, a) \\ 0 & \text{otherwise} \end{cases}$$

Value Iteration

- Algorithm

1. Initialize an estimate for the value function arbitrarily (or zeros)

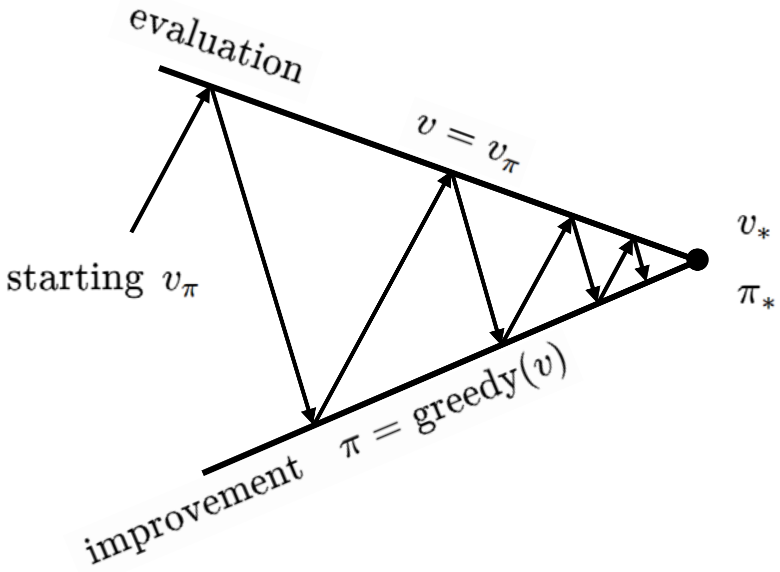
$$v(s) \leftarrow 0 \quad \forall s \in S$$

2. Repeat, update

$$v(s) \leftarrow R(s) + \gamma \max_a \sum_{s' \in S} P(s' \mid s, a)\, v(s'), \quad \forall s \in S$$

Policy Iteration

- Given a policy π, then evaluate the policy π
- Improve the policy by acting greedily with respect to v_π



1. initialize policy $\hat{\pi}$ (e.g., randomly)

2. Compute a value function of policy, v_π (e.g., via solving linear system or Bellman expectation equation iteratively)

3. Update π to be *greedy* policy with respect to v_π

$$\pi(s) \leftarrow \arg\max_a \sum_{s' \in S} P\left(s' \mid s, a\right) v_\pi\left(s'\right)$$

4. If policy π changed in last iteration, return to step 2

Example

Define MDP as a two-level dictionary

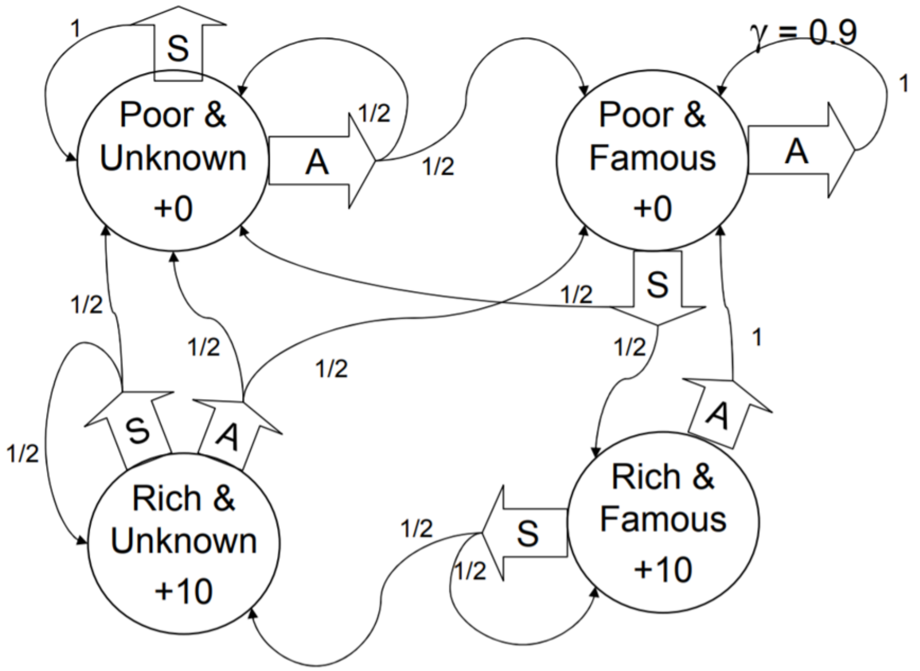
`P` is a two-level dictionary where the first key is the state and the second key is the action.

- State indices `[0, 1, 2, 3]` correspond to [PU, PF, RU, RF]
- Action indices `[0, 1]` correspond to [Saving momey, Advertising]

`P[state][action]` is a list of tuples `(probability, nextstate)`.

For example,

- the transition information for s = 0, a = 0 is `P[0][0] = [(1, 0)]`
- the transition information for s = 3, a = 0 is `P[3][0] = [(0.5, 2), (0.5, 3)]`



Example: Gridworld Domain

- Simple grid world with a goal state with reward and a “bad state” with reward -100
- Actions move in the desired direction with probably 0.8, in one of the perpendicular directions with 0.1
- Taking an action that would bump into a wall leaves agent where it is