Markov Decision Processes (MDPs)

Industrial AI Lab.

Prof. Seungchul Lee

POSTECH

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

Markov Reward Process

Markov Chains with Rewards

- Suppose that each transition in a Markov chain is associated with a reward r
- As the Markov chain proceeds from state to state, there is an associated sequence of rewards
- Discount factor γ

- Later, we will study dynamic programming and Markov decision theory

  $\Rightarrow$ Markov Decision Process (MDP)

  – These topics include a *decision maker*, *policy maker*, or *control* that modify both the transition probabilities and the rewards at each trial of the Markov chain.

Markov Reward Process (MRP)

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

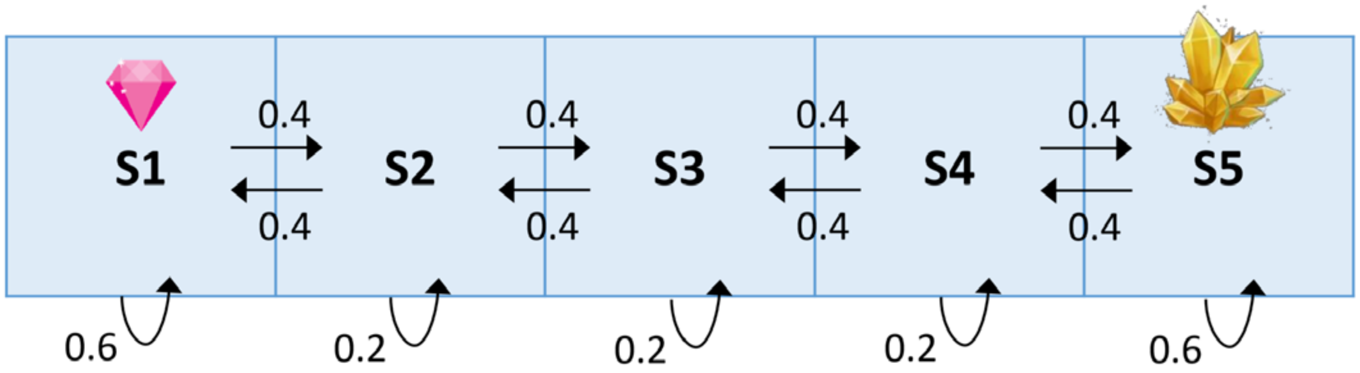
- S is a finite set of states
- P is a state transition probability matrix

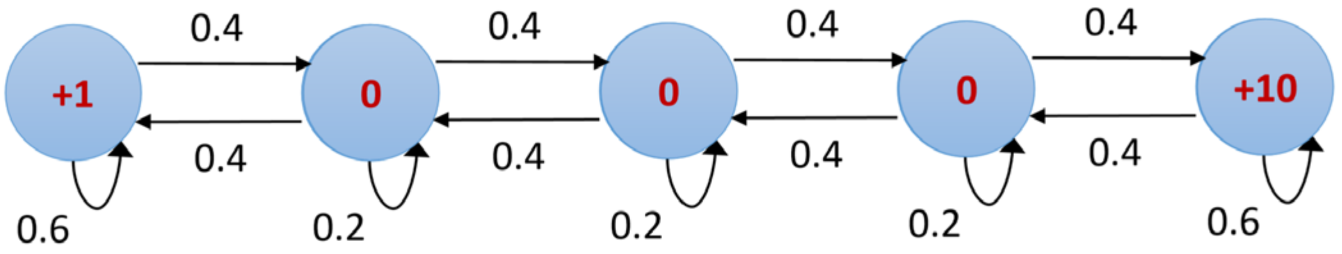
$$P_{ss'} = P\left[S_{t+1} = s' \mid S_t = s\right]$$

- R is a reward function, $R = \mathbb{E}\left[R_{t+1} \mid S_t = s\right]$
- γ is a discount factor, $\gamma \in [0, 1]$

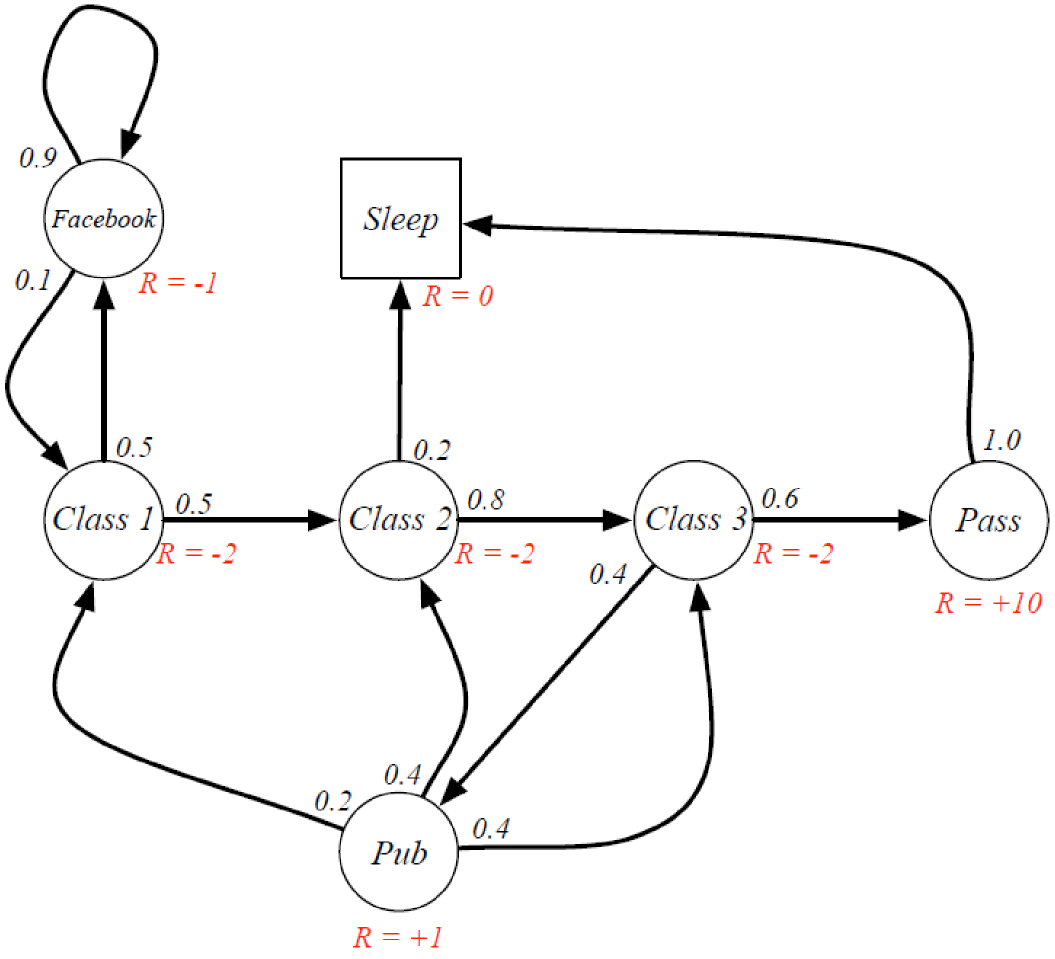
Example: Mars Rover MRP

- Reward: +1 in S_1, +10 in S_5, 0 in the other states
- Discount factor $\gamma = 0.5$





Example: Student MRP



Reward over Multiple Transitions

- Return
  - Total discounted sum of rewards from time step t

Definition: The return G_t is the total discounted reward from time-step t

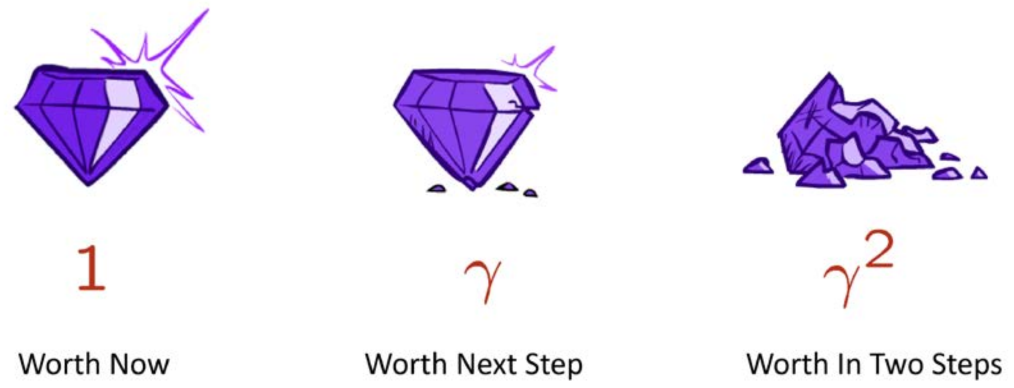
$$G_t = \underbrace{R_{t+1}}_{\text{Immediate reward}} + \underbrace{\gamma R_{t+2} + \cdots}_{\text{Discount sum of future reward}} = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- Discount factor γ is used
  - the present value of future rewards

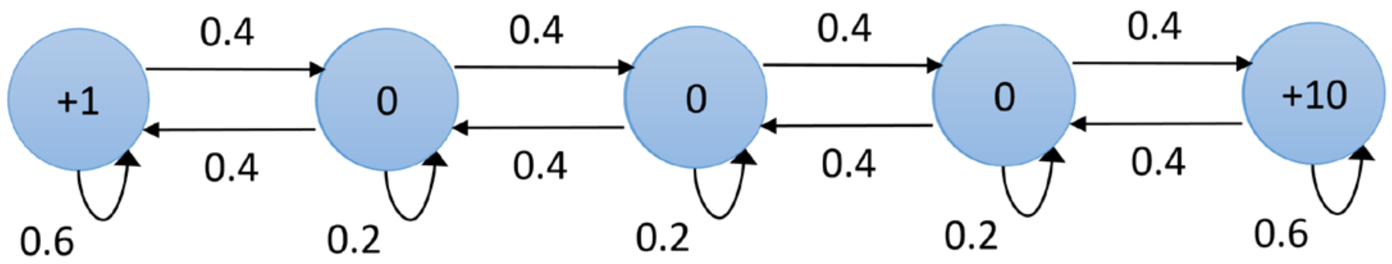
Discount factor γ

- It is reasonable to maximize the sum of rewards.
- It is also reasonable to prefer rewards now to rewards later.
- One solution: values of rewards decay exponentially
- Mathematically convenient (avoid infinite returns and values)

- Humans often act as if there's a discount factor $\gamma < 1$
  - $\gamma = 0$: Only care about immediate reward
  - $\gamma = 1$: Future reward is as beneficial as immediate reward

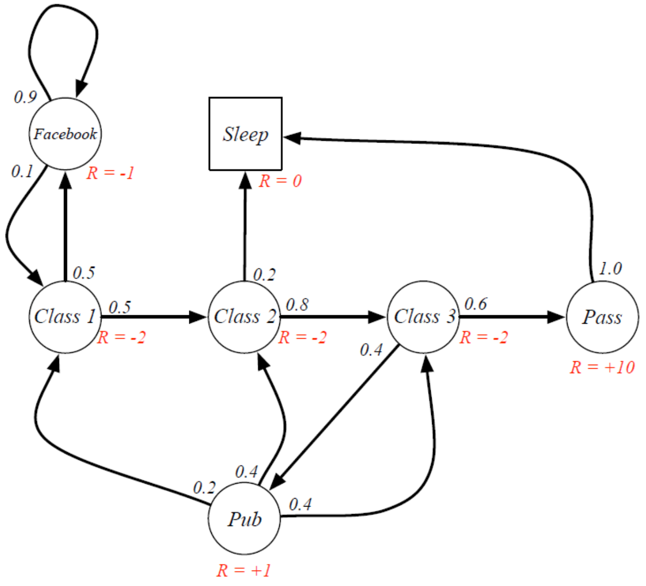


Example: Mars Rover MRP



- Reward: +1 in S_1, +10 in S_5, 0 in all other states
- Sample returns from sample episodes, $\gamma = 0.5$
  - $S_2 \rightarrow S_3 \rightarrow S_4 \rightarrow S_5 : 0 + (0.5 \times 0) + \left(0.5^2 \times 0\right) + \left(0.5^3 \times 10\right) = 1.25$
  - $S_2 \rightarrow S_3 \rightarrow S_4 \rightarrow S_3 : 0 + (0.5 \times 0) + \left(0.5^2 \times 0\right) + \left(0.5^3 \times 0\right) = 0.0$
  - $S_2 \rightarrow S_3 \rightarrow S_2 \rightarrow S_2 : 0 + (0.5 \times 0) + \left(0.5^2 \times 0\right) + \left(0.5^3 \times 1\right) = 0.125$

Example: Student MRP Returns



Sample returns for Student MRP:
Starting from $S_1 = \text{C1}$ with $\gamma = \frac{1}{2}$

$$G_1 = R_2 + \gamma R_3 + ... + \gamma^{T-2} R_T$$

C1 C2 C3 Pass Sleep	$v_1 = -2 - 2 * \frac{1}{2} - 2 * \frac{1}{4} + 10 * \frac{1}{8} = -2.25$
C1 FB FB C1 C2 Sleep	$v_1 = -2 - 1 * \frac{1}{2} - 1 * \frac{1}{4} - 2 * \frac{1}{8} - 2 * \frac{1}{16} = -3.125$
C1 C2 C3 Pub C2 C3 Pass Sleep	$v_1 = -2 - 2 * \frac{1}{2} - 2 * \frac{1}{4} + 1 * \frac{1}{8} - 2 * \frac{1}{16} \ldots = -3.41$
C1 FB FB C1 C2 C3 Pub C1 ...	$v_1 = -2 - 1 * \frac{1}{2} - 1 * \frac{1}{4} - 2 * \frac{1}{8} - 2 * \frac{1}{16} \ldots = -3.20$
FB FB FB C1 C2 C3 Pub C2 Sleep	

Value Function

- The value function $v(s)$ gives the long-term value of state s
- Definition: The state value function $v(s)$ of an MRP is the expected return starting from state s
- Expected return from starting from state s

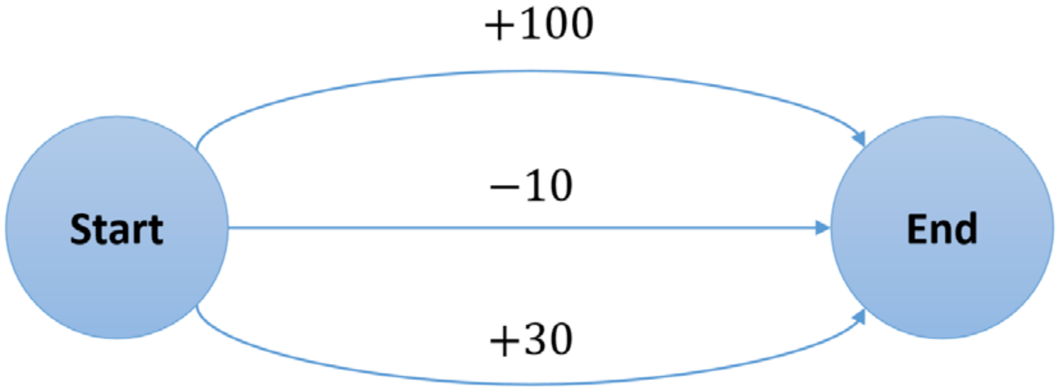
$$
\begin{aligned}
v(s) &= \mathbb{E}[G_t \mid S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \mid S_t = s]
\end{aligned}
$$

Computing Value Function of MRP (Naïve)

- Generate a large number of episodes and compute the average return

$$v(s) = \mathbb{E}[G_t \mid S_t = s]$$

- Example



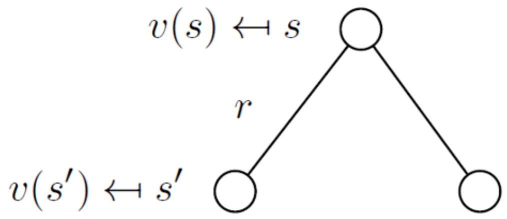
$$v(Start) = \frac{100 - 10 + 30}{3} = \frac{120}{3} = +40$$

Computing Value Function of MRP (Smart and Efficient)

- The value function $v(S_t)$ can be decomposed into two parts:
  - Immediate reward R_{t+1} at state S_t
  - Discounted value of successor state $\gamma v(S_{t+1})$

$$
\begin{aligned}
v(s) &= \mathbb{E}\left[G_t \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma\left(R_{t+2} + \gamma R_{t+3} + \cdots\right) \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma v\left(S_{t+1}\right) \mid S_t = s\right]
\end{aligned}
$$

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

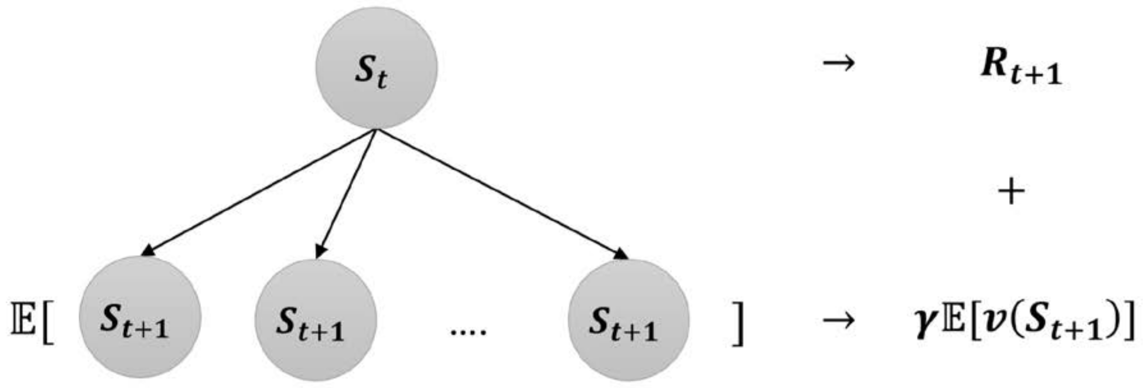


$$\boxed{v(s) = \mathcal{R}_s + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'} v(s')}$$

Computing Value Function of MRP (Smart and Efficient)

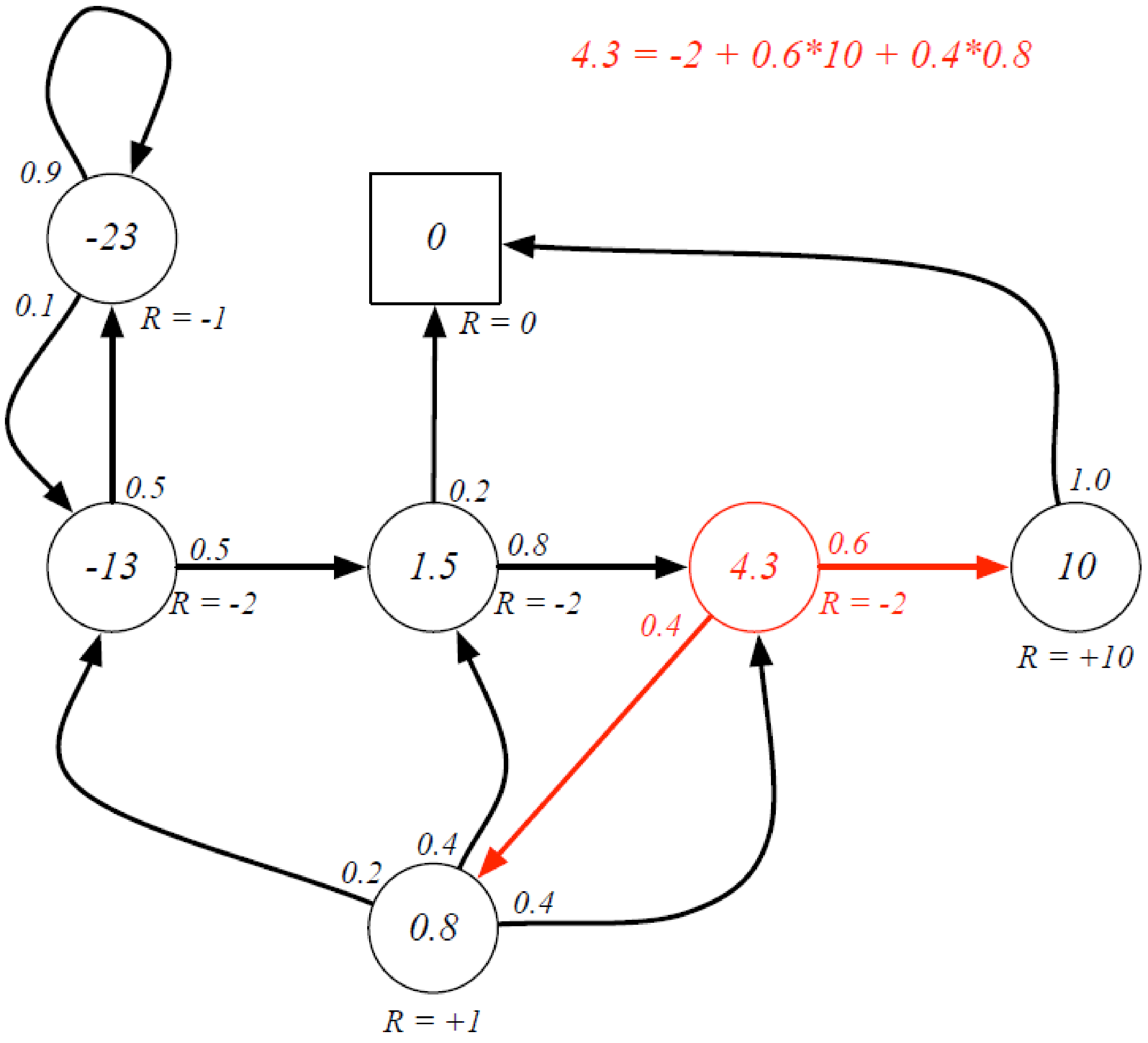
- Bellman Equations for MRP

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



$$v(s) = R(s) + \gamma \sum_{s' \in S} p(s' \mid s) v(s')$$

Example: Bellman Equation for Student MRP



Bellman Equation in Matrix Form

$$v(s) = R + \gamma \sum_{s' \in S} P_{ss'} v(s') \qquad \forall s$$

- The Bellman equation can be expressed concisely using matrices,

$$v = R + \gamma P v$$

- v is a column vector with one entry per state

$$\begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix} = \begin{bmatrix} R_1 \\ \vdots \\ R_n \end{bmatrix} + \gamma \begin{bmatrix} p_{11} & \cdots & p_{1n} \\ & \vdots & \\ p_{n1} & \cdots & p_{nn} \end{bmatrix} \begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix}$$

Solving the Bellman Equation

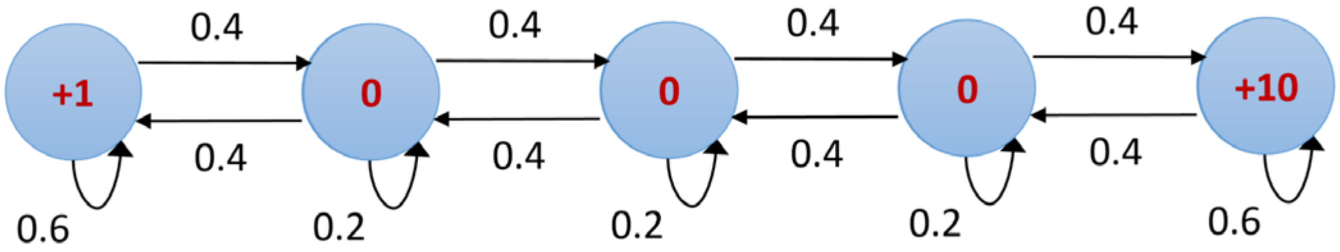
- Analytic solution for value function
- The Bellman equation is a linear equation
- It can be solved directly:

$$
\begin{aligned}
v &= R + \gamma P v \\
(I - \gamma R) v &= R \\
v &= (I - \gamma P)^{-1} R
\end{aligned}
$$

- Direct solution only possible for small MRP
- Computational complexity is $O(n^3)$ for n states

Example: Mars Rover MRP

- Reward: +1 in S_1, +10 in S_5, 0 in the other states
- Discount factor $\gamma = 0.5$



$$v = R + \gamma P v$$

$$\begin{bmatrix} v(1) \\ v(2) \\ v(3) \\ v(4) \\ v(5) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 10 \end{bmatrix} + 0.5 \begin{bmatrix} 0.6 & 0.4 & 0.0 & 0.0 & 0.0 \\ 0.4 & 0.2 & 0.4 & 0.0 & 0.0 \\ 0.0 & 0.4 & 0.2 & 0.4 & 0.0 \\ 0.0 & 0.0 & 0.4 & 0.2 & 0.4 \\ 0.0 & 0.0 & 0.0 & 0.4 & 0.6 \end{bmatrix} \begin{bmatrix} v(1) \\ v(2) \\ v(3) \\ v(4) \\ v(5) \end{bmatrix}$$

Iterative Algorithm for Value Function

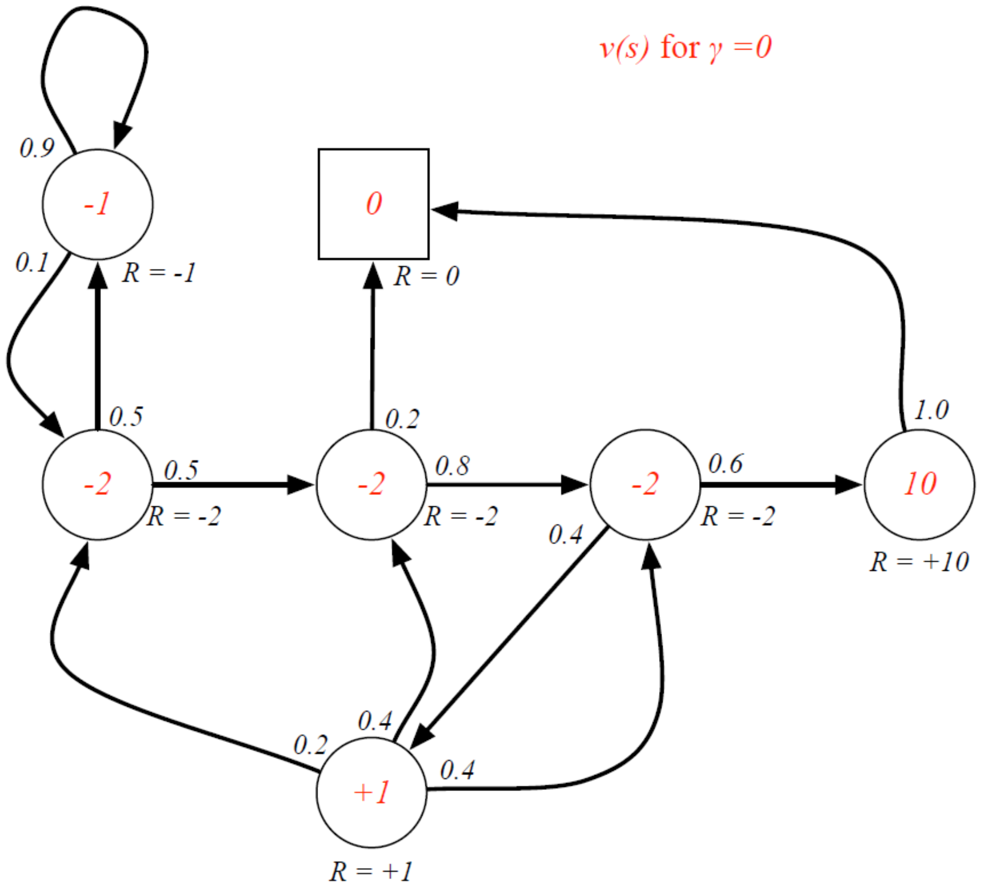
- There are many *iterative* methods for large MRP
  - Dynamic programming
  - Monte-Carlo simulation
  - Temporal-difference learning

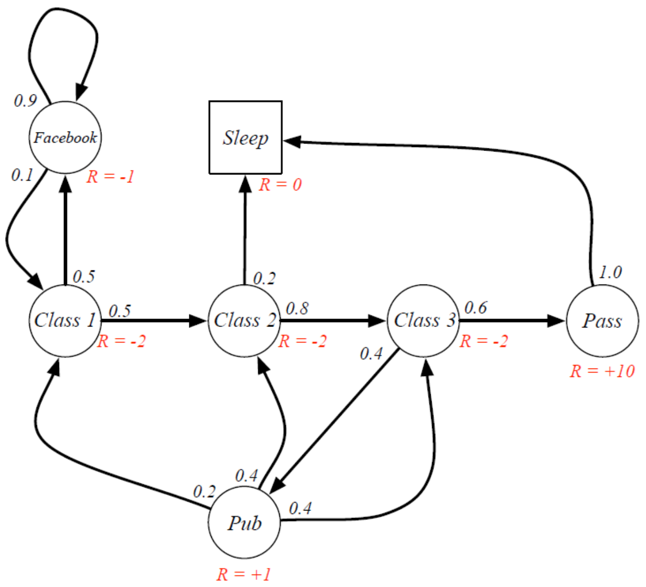
- Iterative algorithm for value function (Value Iteration)

> – Initialize $V_1(s) = 0$ for all s
> – For $k = 1$ until convergence
>   - For all s in S
>
> $$v_{k+1}(s) \longleftarrow R(s) + \gamma \sum_{s' \in S} p\left(s' \mid s\right) v_k\left(s'\right)$$

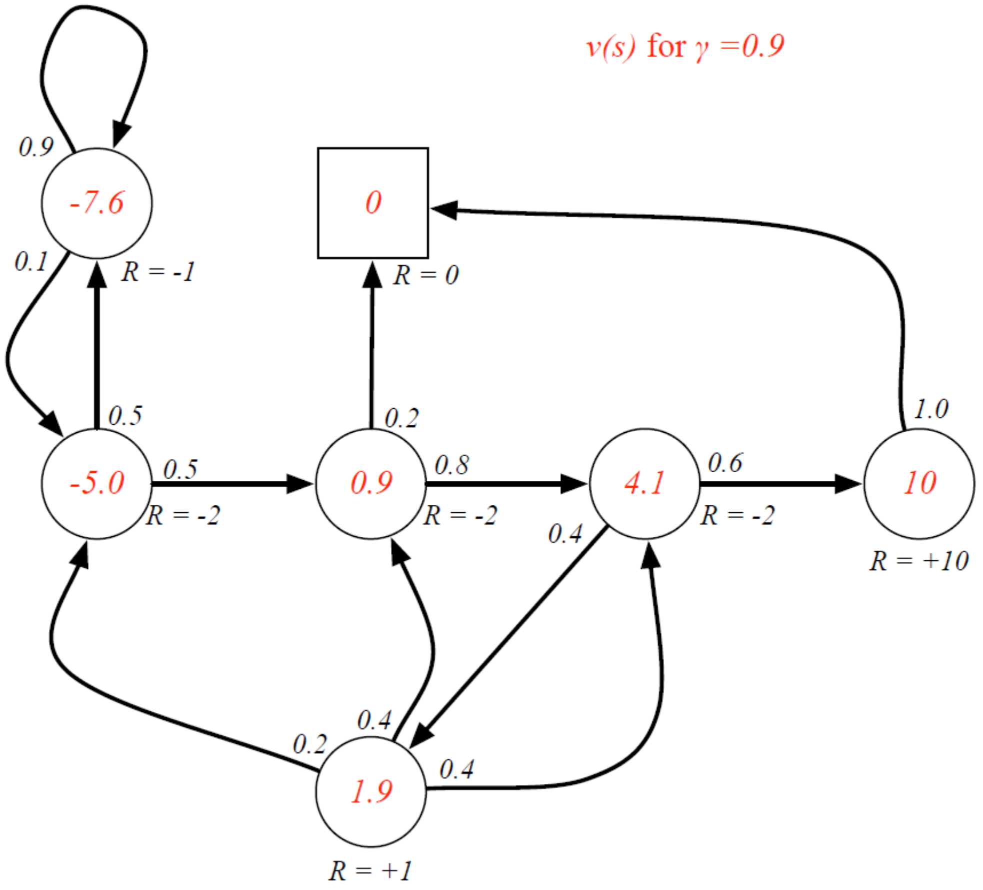
- Computational complexity: $O\left(n^2\right)$ for each k

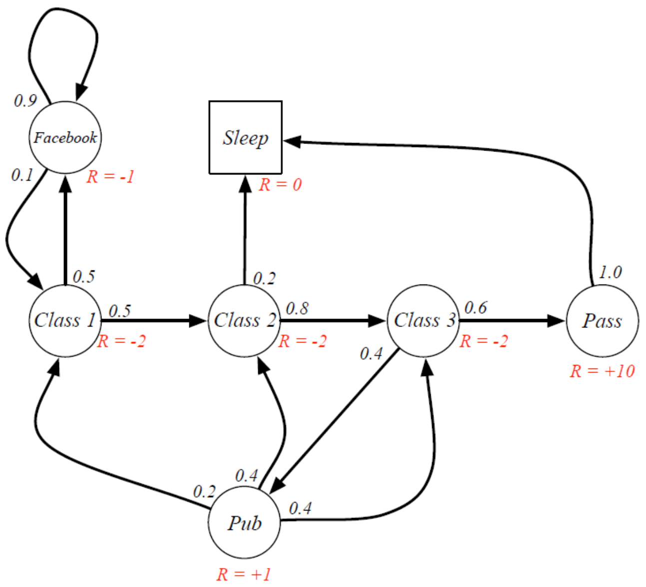
State-Value Function for Student MRP (1/3)



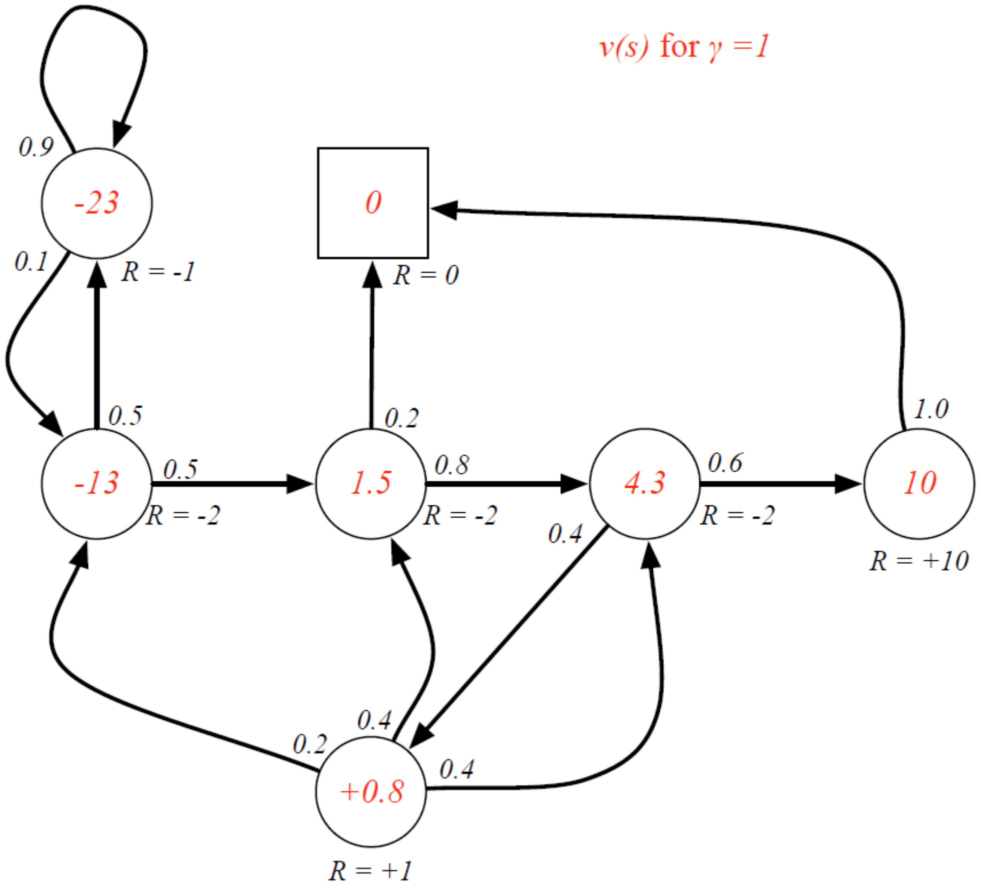


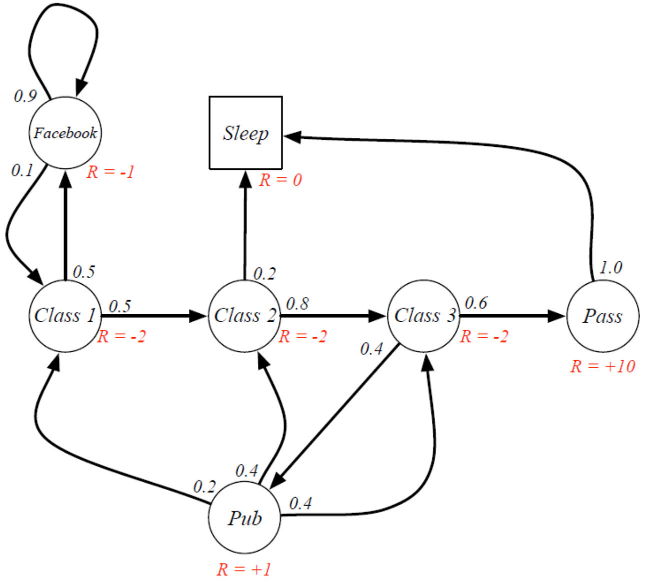
State-Value Function for Student MRP (2/3)





State-Value Function for Student MRP (3/3)





Markov Decision Process

Markov Decision Process

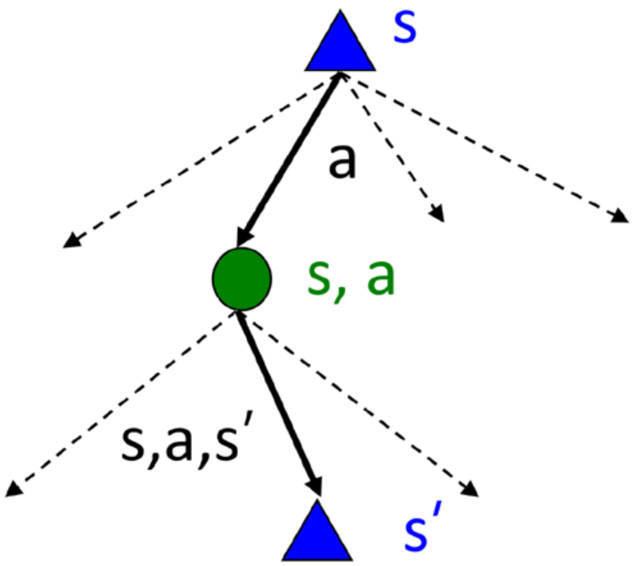
- So far, we analyzed the passive behavior of a Markov chain with rewards

- 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 $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

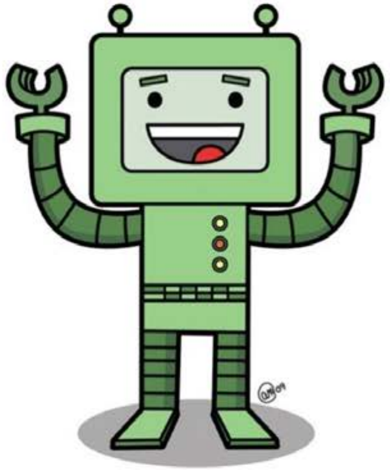
- $\mathcal{S}$ is a finite set of states
- $\mathcal{A}$ is a finite set of actions
- $\mathcal{P}$ is a state transition probability matrix, $\mathcal{P}_{ss'}^{a} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s, A_t = a\right]$
- $\mathcal{R}$ is a reward function, $\mathcal{R}_s^a = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$
- γ is a discount factor $\gamma \in [0, 1]$.



  - It is an environment in which all states are Markov

Markov Decision Process

Agent



Policy

Value

State s_t

Action a_t

Reward r_{t+1}

State s_{t+1}

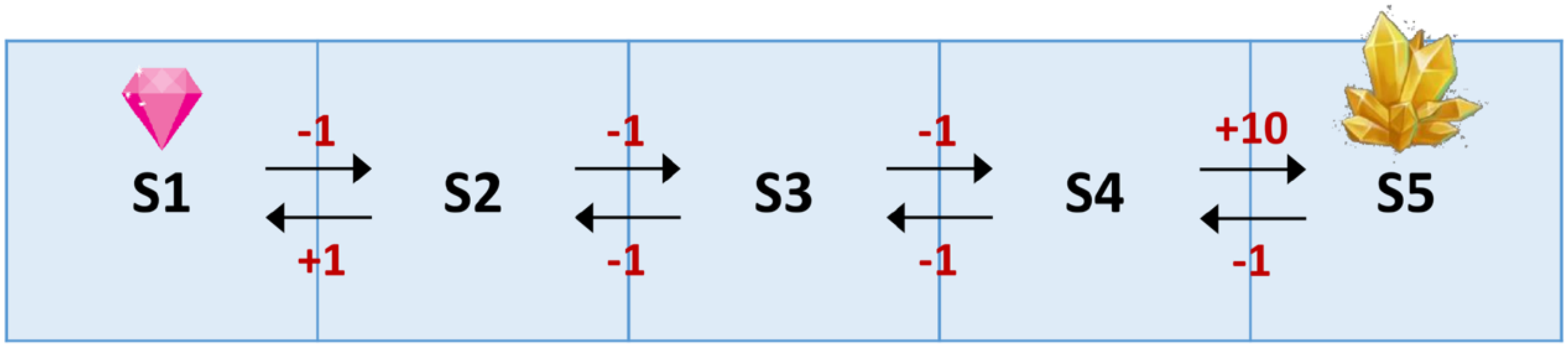
Environment

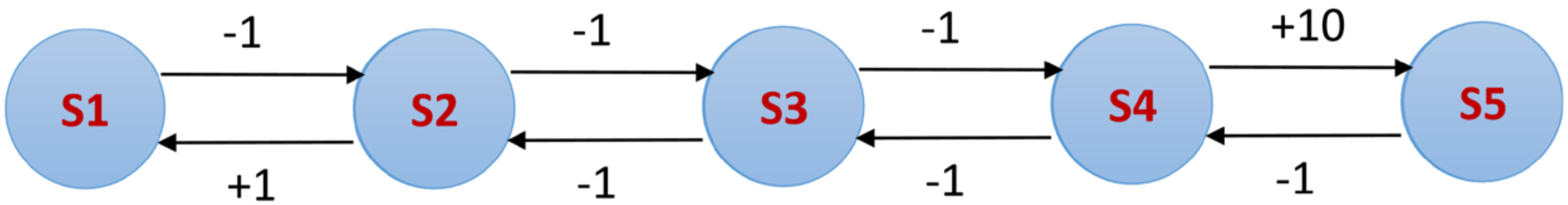
MDP

Note: Reward is a function of state s_t and action a_t.

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

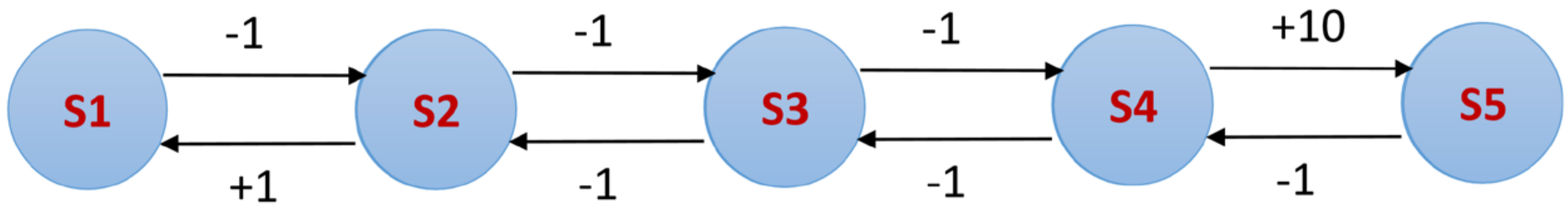




Example: Mars Rover MDP

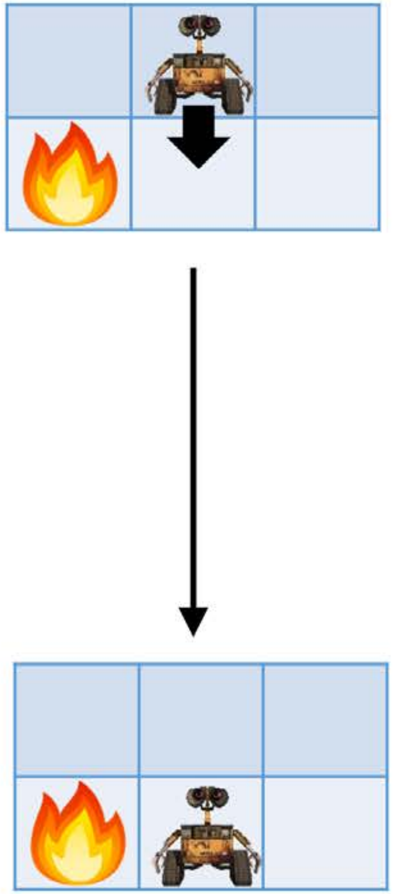
- Deterministic state transition matrix

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

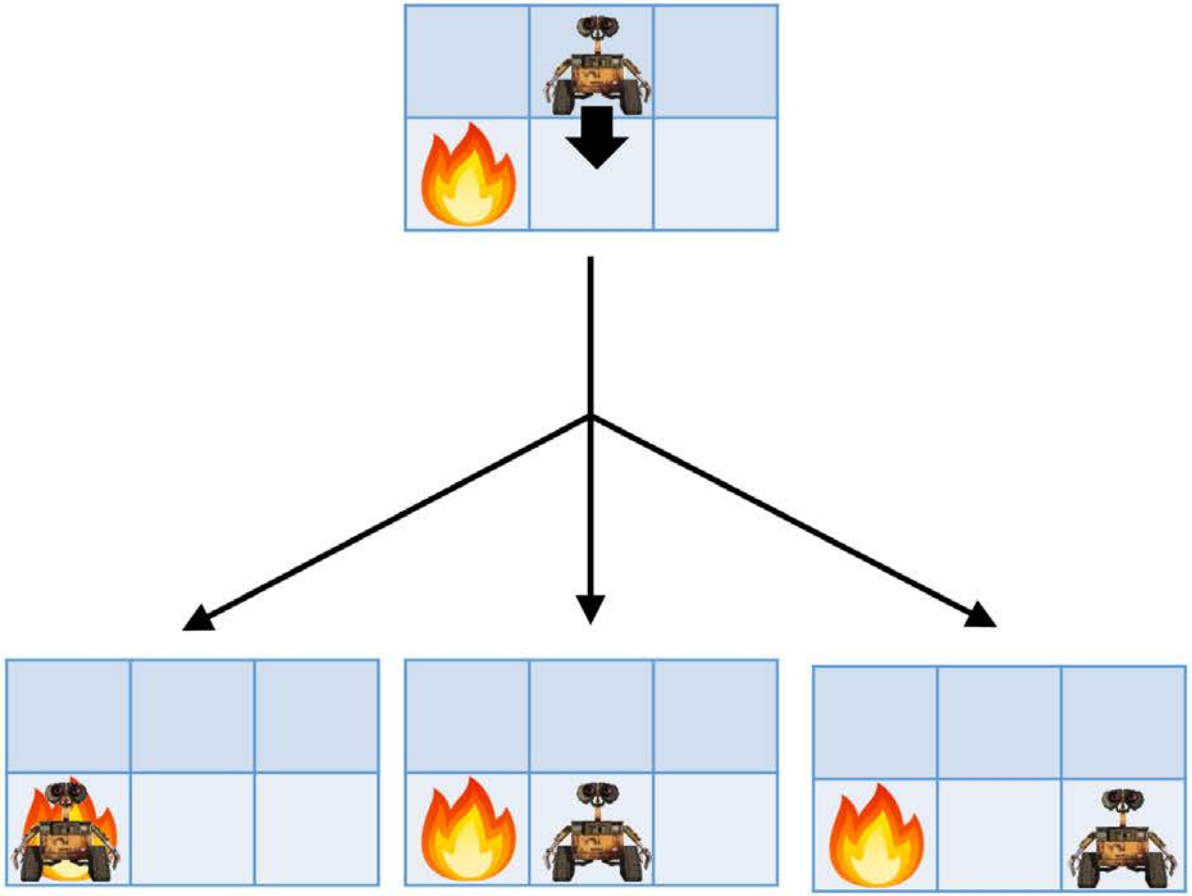


Example: Grid World Actions

Deterministic grid world



Stochastic grid world

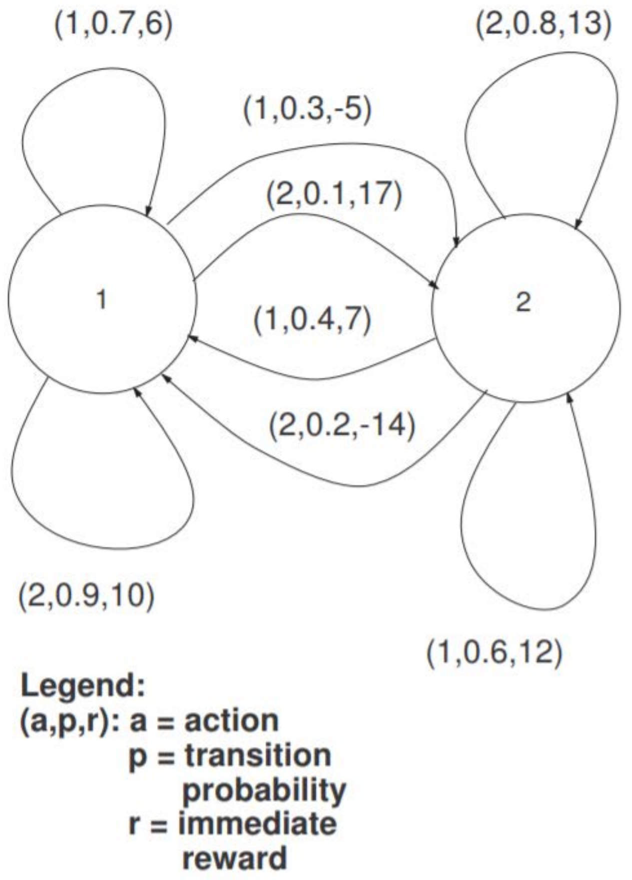


Example

- P_a: transition probability matrix for action a
- R_a: transition reward matrix for action a

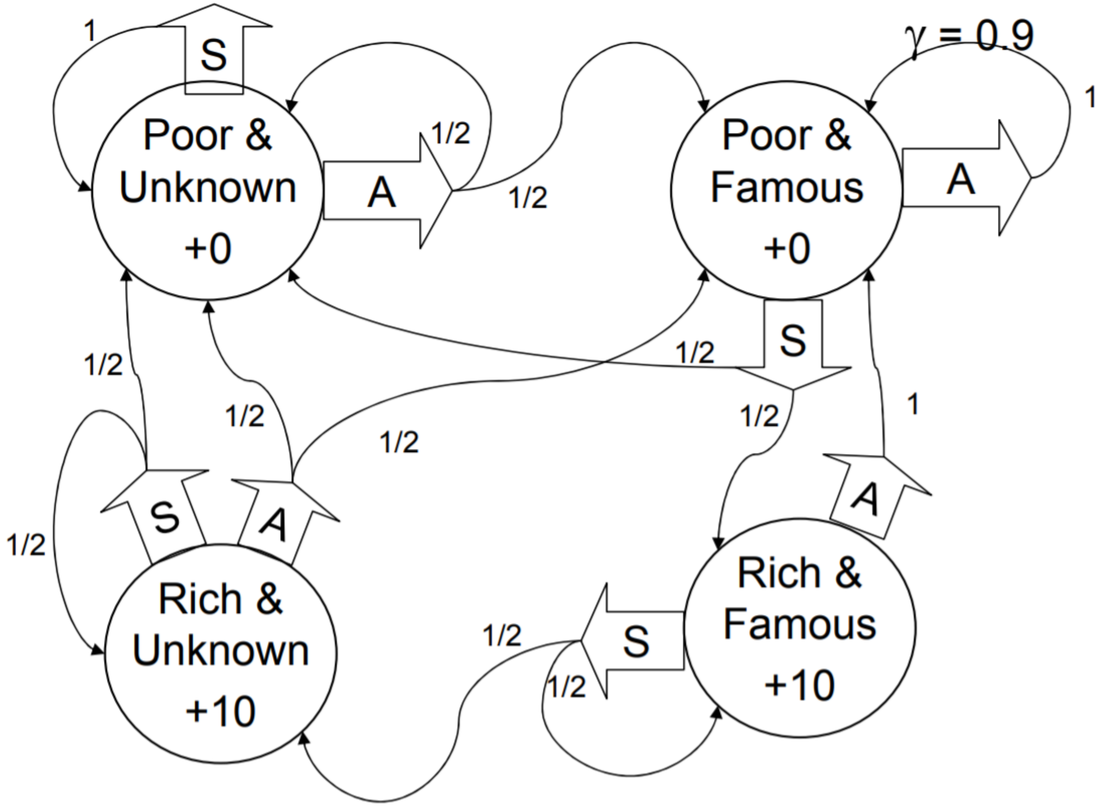
$$\mathbf{P}_1 = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix}; \mathbf{P}_2 = \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix};$$

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



Example

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

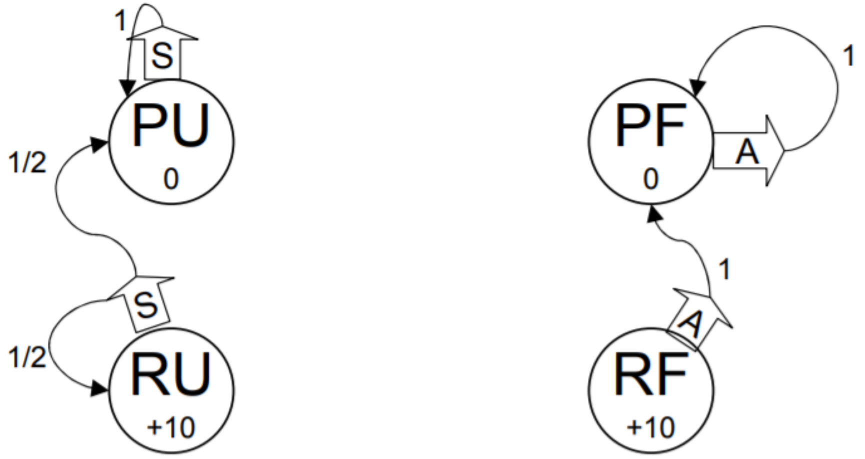


Policy

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

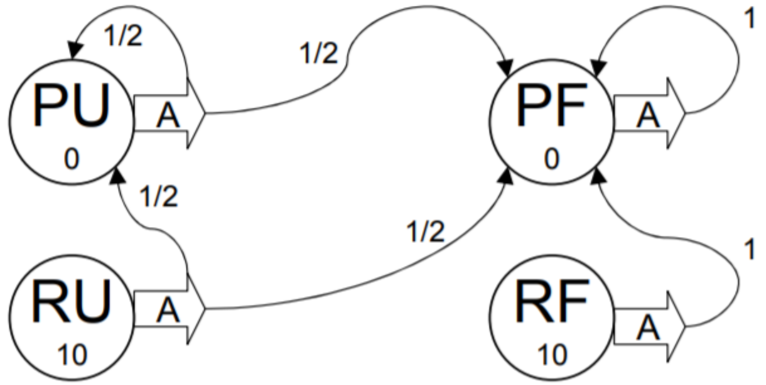
Policy Number 1:

STATE →	ACTION
PU	S
PF	A
RU	S
RF	A



Policy Number 2:

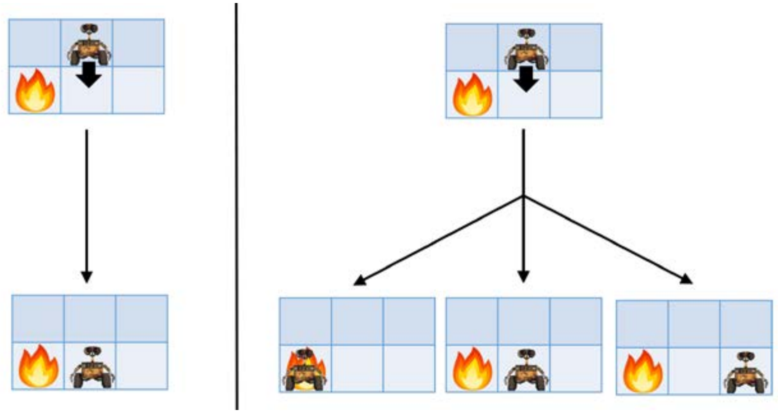
STATE →	ACTION
PU	A
PF	A
RU	A
RF	A



Policy

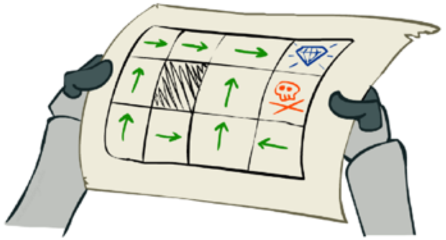
- A policy is a mapping from states to actions, $\pi: S \rightarrow A$

- 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

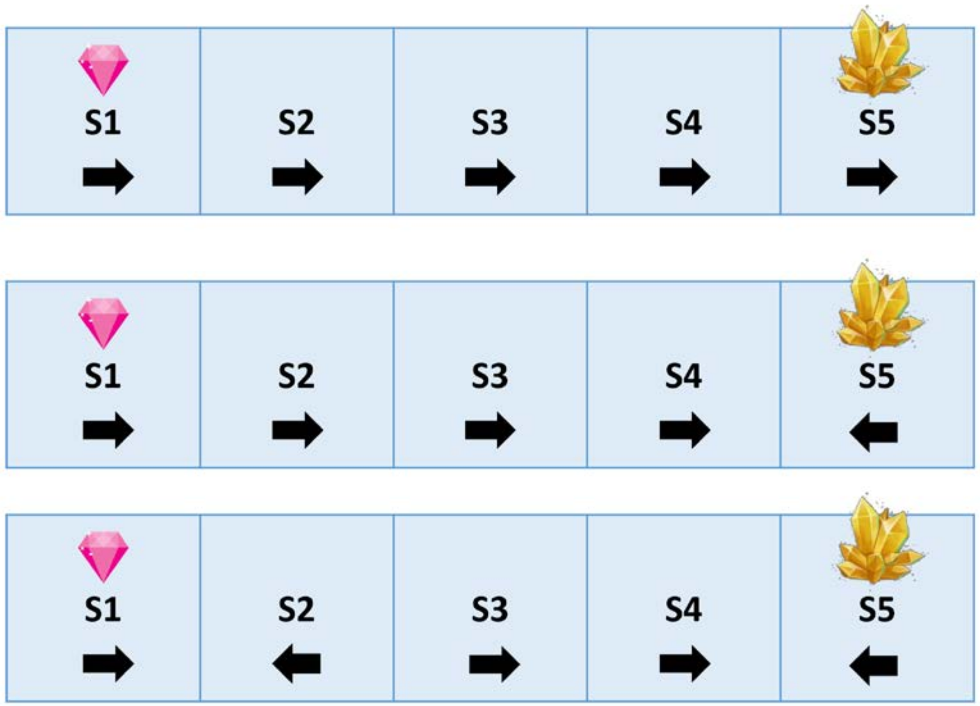
- Given an MDP $\mathcal{M} = \langle S, A, P, R, \gamma \rangle$ and a policy π
  - The state sequence $s_1, s_2, \cdots$ is a Markov process $\langle S, P^{\pi} \rangle$
  - The state and reward sequence is a Markov reward process $\langle S, P^{\pi}, R^{\pi}, \gamma \rangle$

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?

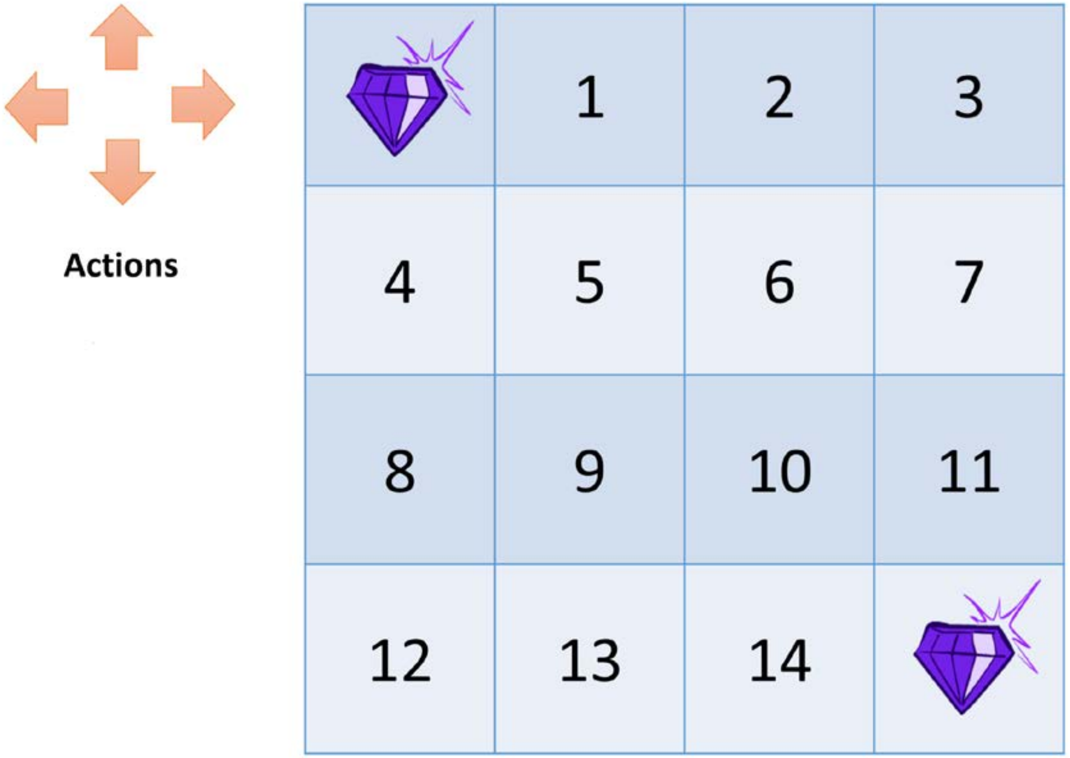
Examples: Mars Rover Polices

- How many possible policies in our example?



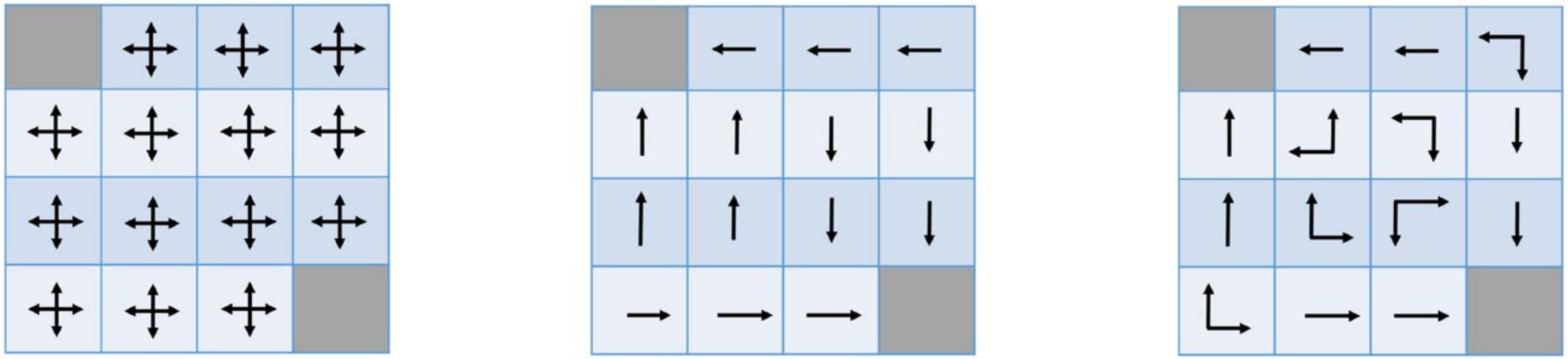
- Which of the above two policies is best?

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.

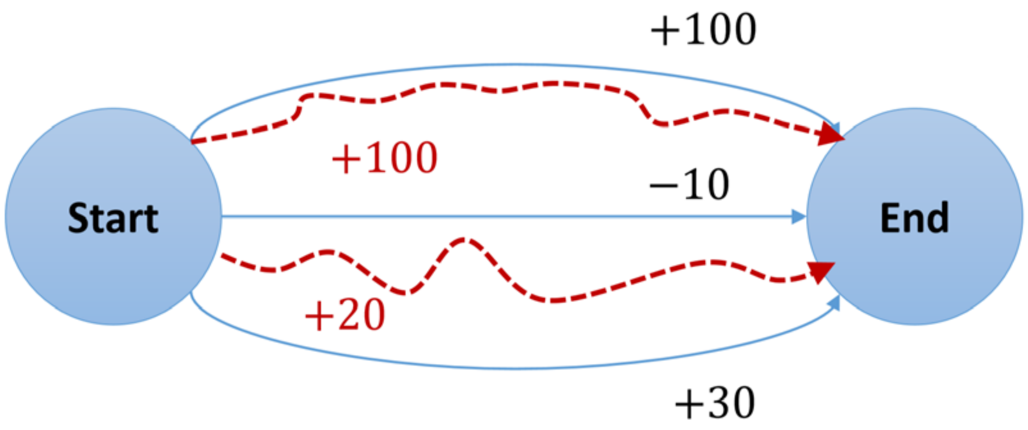


Value Function: State-Value Function

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

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

- Example



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

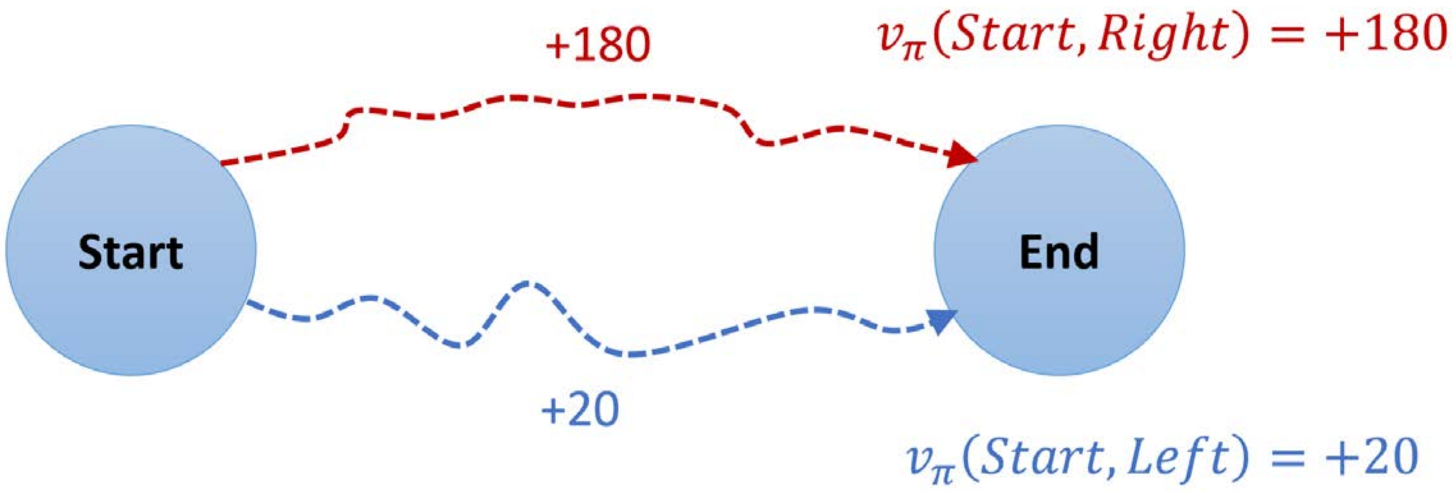
$$\Downarrow$$

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

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

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



Bellman Equation

Richard Ernest Bellman

Value Functions for policy π

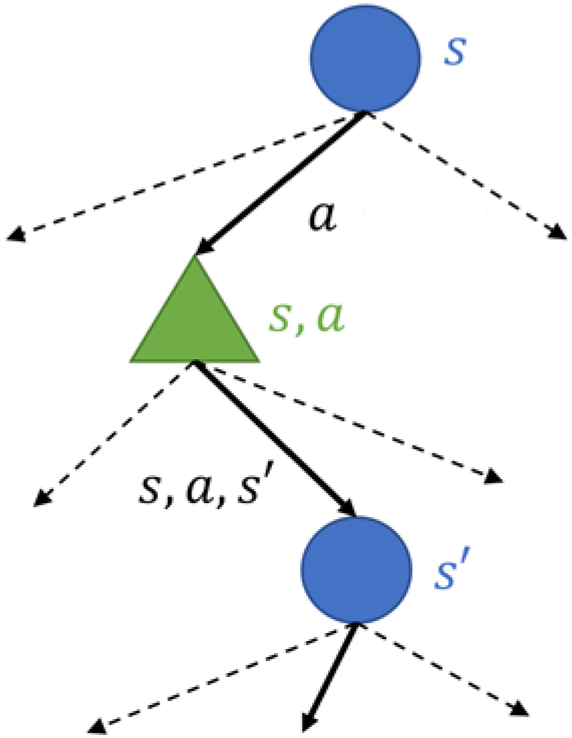
- Given the policy π, the value function can again be decomposed into immediate reward plus discounted value of successor state (recursively)

- 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

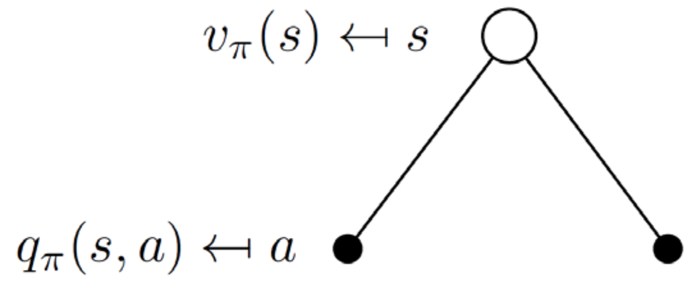
$$q_\pi(s, a) = \mathbb{E}\left[R_{t+1} + \gamma q_\pi\left(S_{t+1}, A_{t+1}\right) \mid S_t = s, A_t = a\right]$$



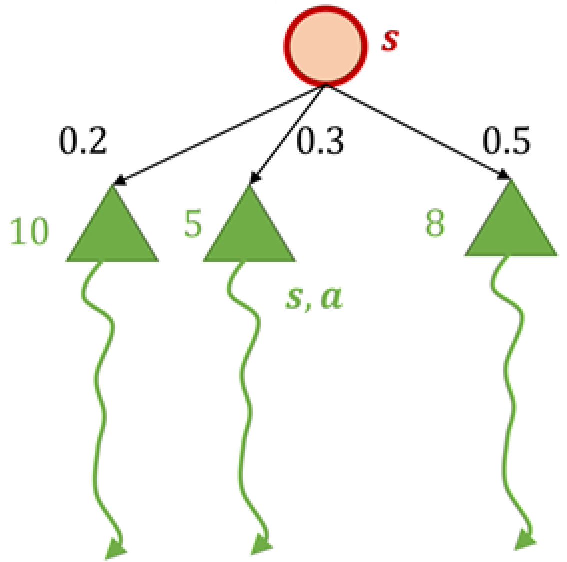
Relationship between $v_\pi(s)$ and $q_\pi(s,a)$

- State-value function using policy π

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



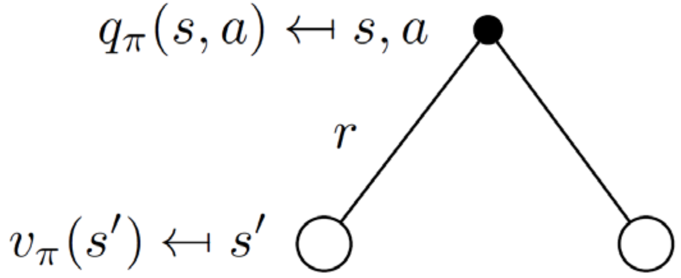
- $v_\pi(s) = 0.2 \times 10 + 0.3 \times 5 + 0.5 \times 8 = 7.5$



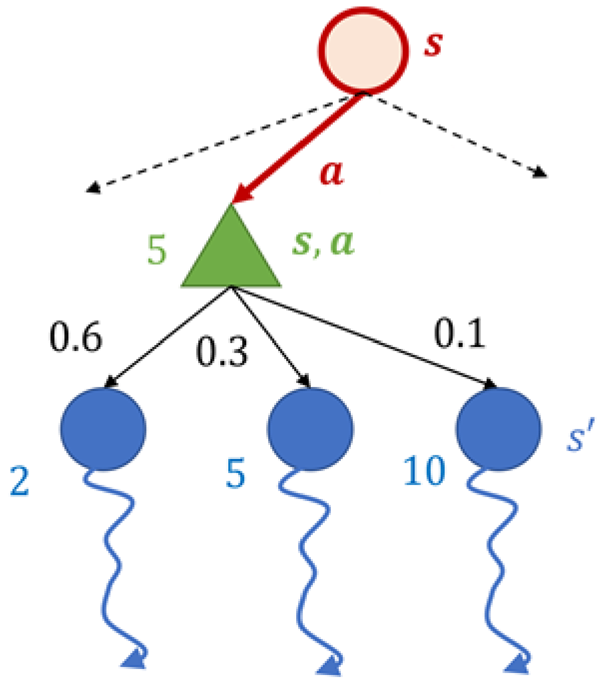
Relationship between $v_\pi(s)$ and $q_\pi(s, a)$

- Action-value function using policy π

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

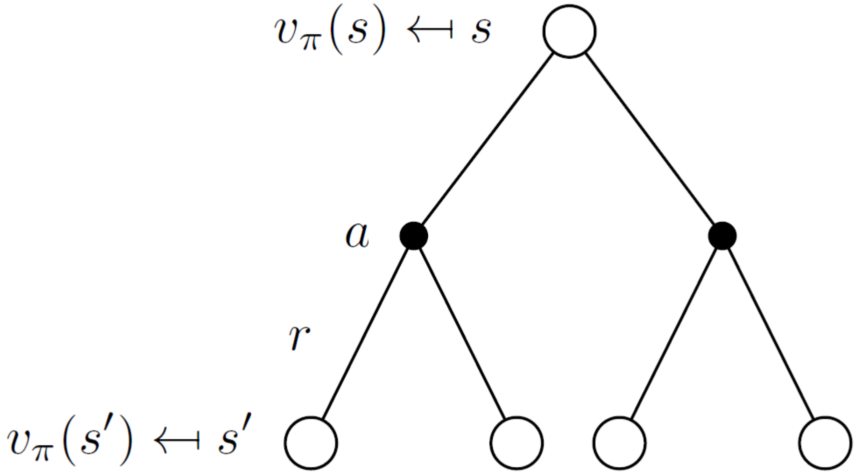


- $q_\pi(s,a) = 5 + \gamma \times (0.6 \times 2 + 0.3 \times 5 + 0.1 \times 10)$



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

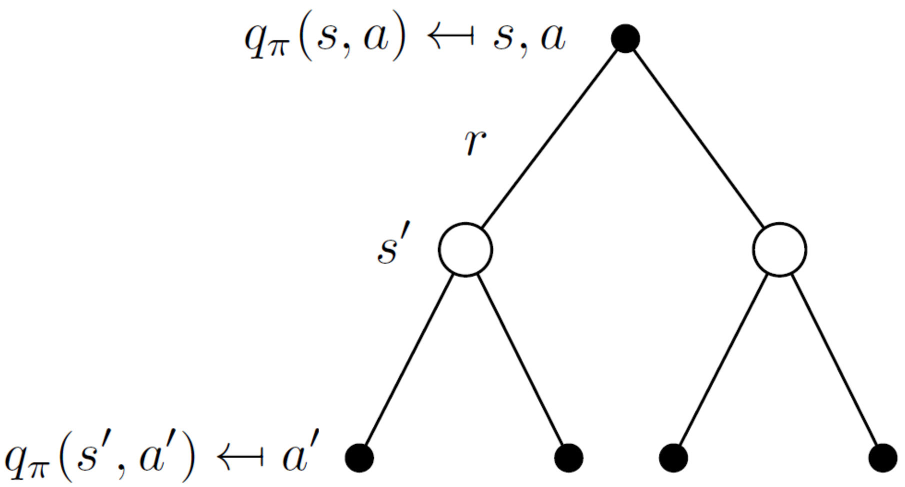
$$
\begin{aligned}
v_\pi(s) &= \sum_{a \in A} \pi(a \mid s) \underline{q_\pi(s, a)} \\
&= \sum_{a \in A} \pi(a \mid s) \left(R(s) + \gamma \sum_{s' \in S} P(s' \mid s, a) \, v_\pi(s') \right)
\end{aligned}
$$



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

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

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



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

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

Optimal Value Function

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

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

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

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

Optimal Policy

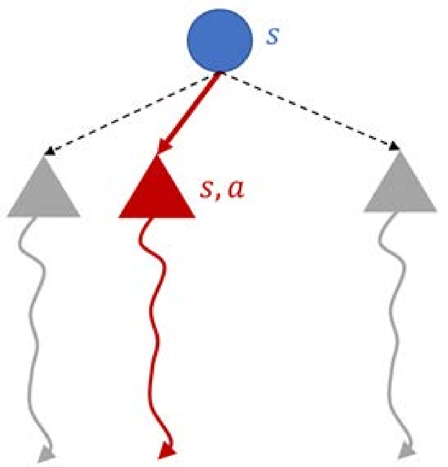
- The optimal policy is the policy that achieves the highest value for every state

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

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

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

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

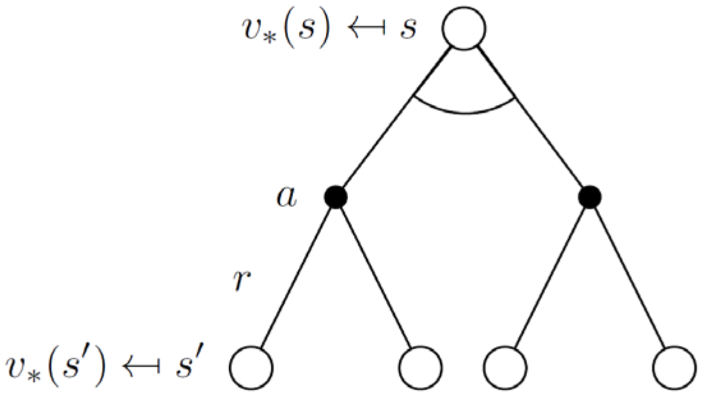


- There is always a deterministic optimal policy for any MDP
- If we know $q_*(s, a)$, we can have the optimal policy

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

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

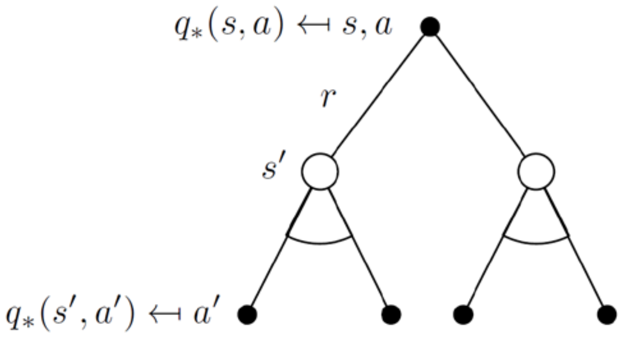
$$
\begin{aligned}
v_*(s) &= \max_\pi v_\pi(s) \\
&= \max_a q_\pi(s,a) \\
&= \max_a \left(R(s) + \gamma \sum_{s' \in S} P(s' \mid s,a) v_\pi(s') \right) \\
&= R(s) + \gamma \max_a \sum_{s' \in S} P(s' \mid s,a) v_\pi(s')
\end{aligned}
$$

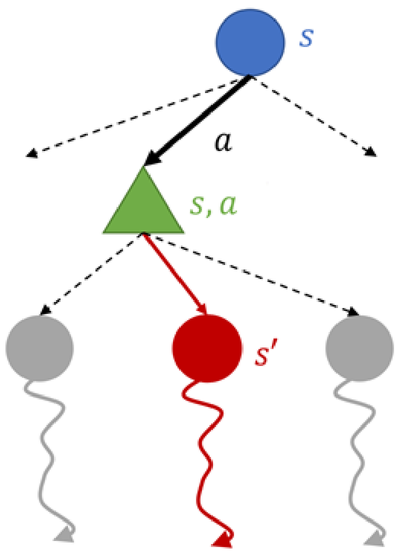


Bellman Optimality Equation for $q_\pi(s, a)$

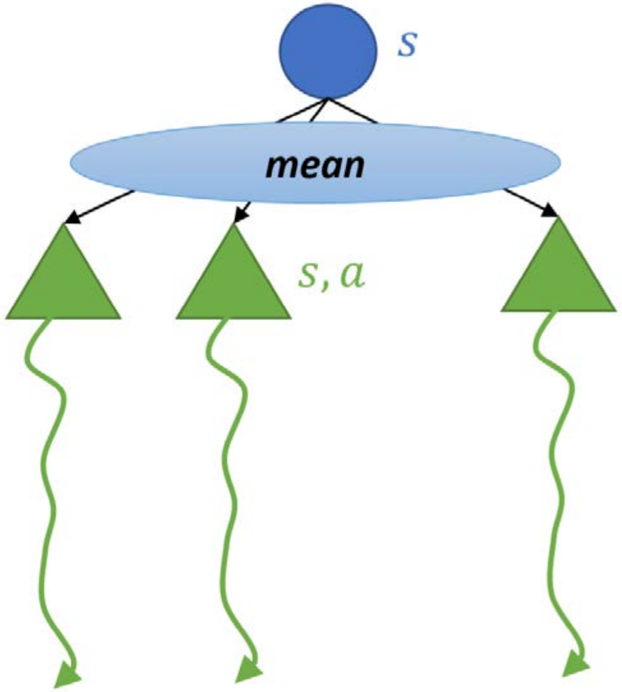
- The optimal action-value function $q_*(s, a)$ is the maximum action-value function over all policies.

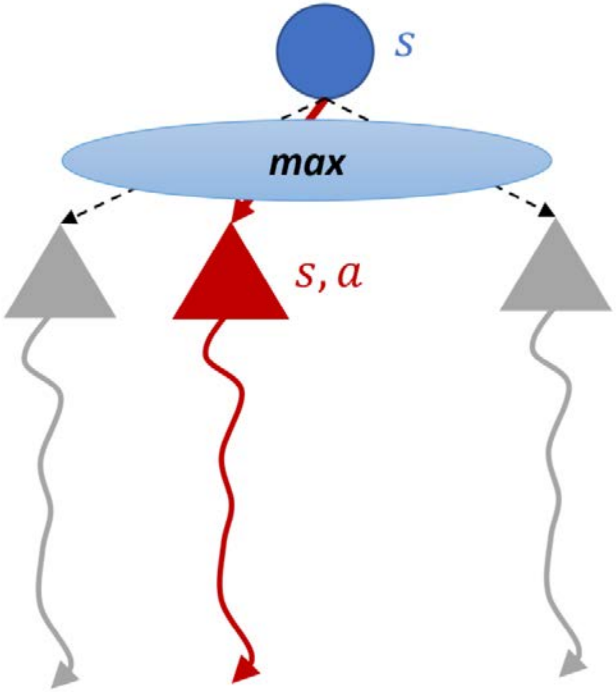
$$
\begin{aligned}
q_*(s,a) &= \max_\pi q_\pi(s,a) \\
&= \max_\pi \left(R(s) + \gamma \sum_{s' \in S} P(s' \mid s,a) v_\pi(s') \right) \\
&= R(s) + \gamma \sum_{s' \in S} P(s' \mid s,a) \max_\pi v_\pi(s')
\end{aligned}
$$





Summary: Expectation vs. Optimality



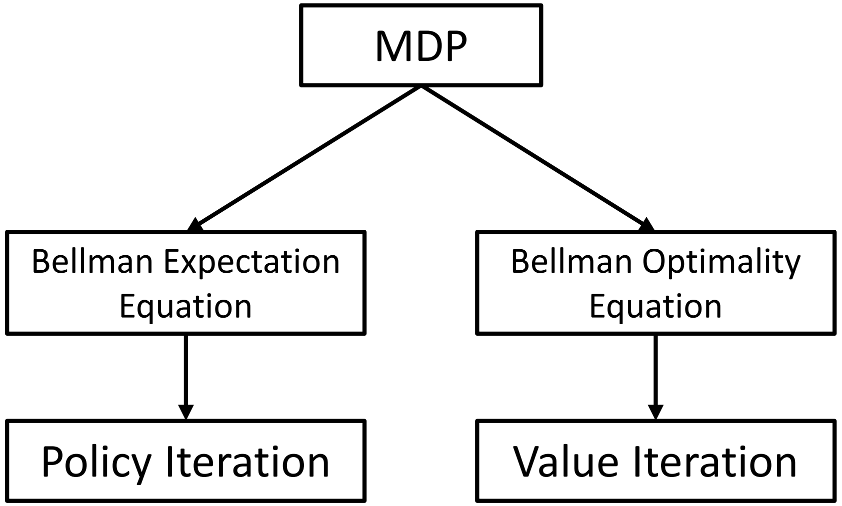


Solving the Bellman Equation

Solving the Bellman Optimality Equation

- Bellman optimality equation is nonlinear
  - Not possible to use a linear equation
- No closed form solution (in general)

- Many iterative solution methods
  - Value iterations
  - Policy iterations
  - SARSA
  - Q-learning



Dynamic Programming (DP)

- DP is a general solution method for problems which have two properties
  - Optimal substructure
    - Optimal solution can be decomposed into sub-problems
  - Overlapping sub-problems
    - Sub-problems recur many times
    - Solutions can be cached and reused

- MDP satisfy both properties

Dynamic Programming (DP)

- DP can compute optimal policies given a perfect model of the environment as a MDP

- For policy prediction (or evaluation)
  - Input: MDP and policy
  - Output: value function

- For policy control (or improvement)
  - Input: MDP
  - Output: optimal value function and optimal policy

Optimal Policy and Optimal Value Function

- 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

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

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

- Algorithm

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)

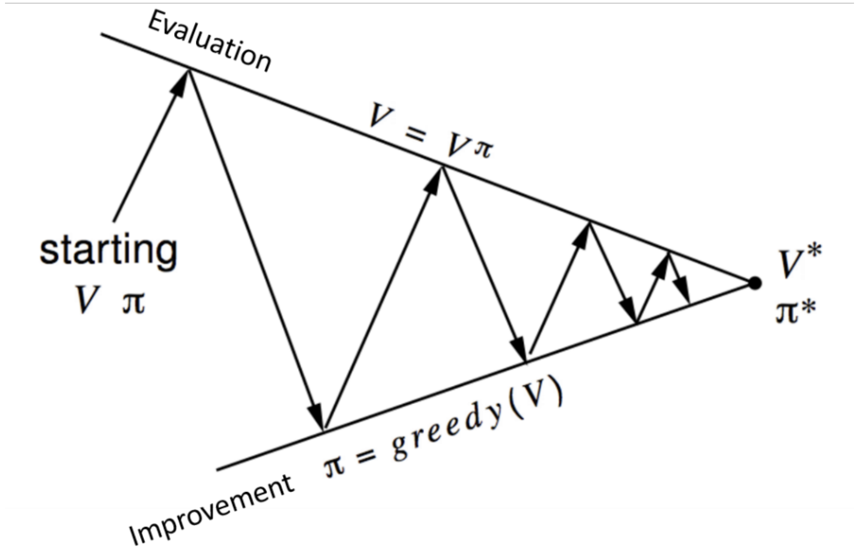
3. Update π to be *greedy* policy with respect to v_π

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

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

Policy Iteration

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



- Policy iteration requires fewer iterations that value iteration, but each iteration requires solving a linear system instead of just applying Bellman operator

- In practice, policy iteration is often faster, especially if the transition probabilities are structured (e.g., sparse) to make solution of linear system efficient

Example

Define MDP as a two-level dictionary

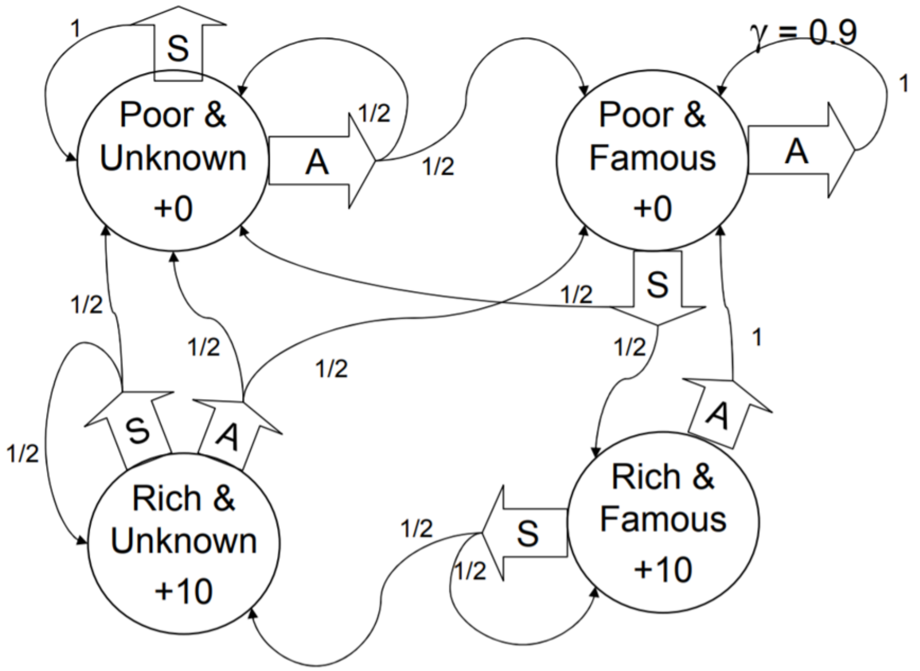
`P` is a two-level dictionary where the first key is the state and the second key is the action.

- 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
- Taking an action that would bump into a wall leaves agent where it is