Markov Decision Processes

Lecture 2

Subir Varma

Contents

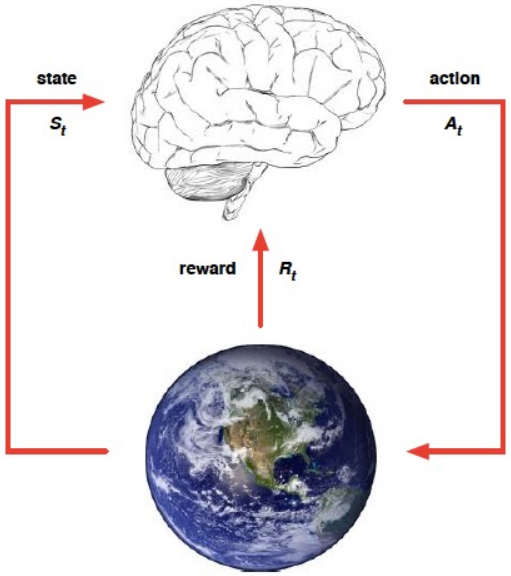
- Markov Processes
- Markov Decision Processes Part 1
  - The Bellman Expectation Equation

Given a Policy $\pi(s)$
find corresponding Value Function $v_\pi(s)$

- Markov Decision Processes Part 2
  - The Bellman Optimality Equation

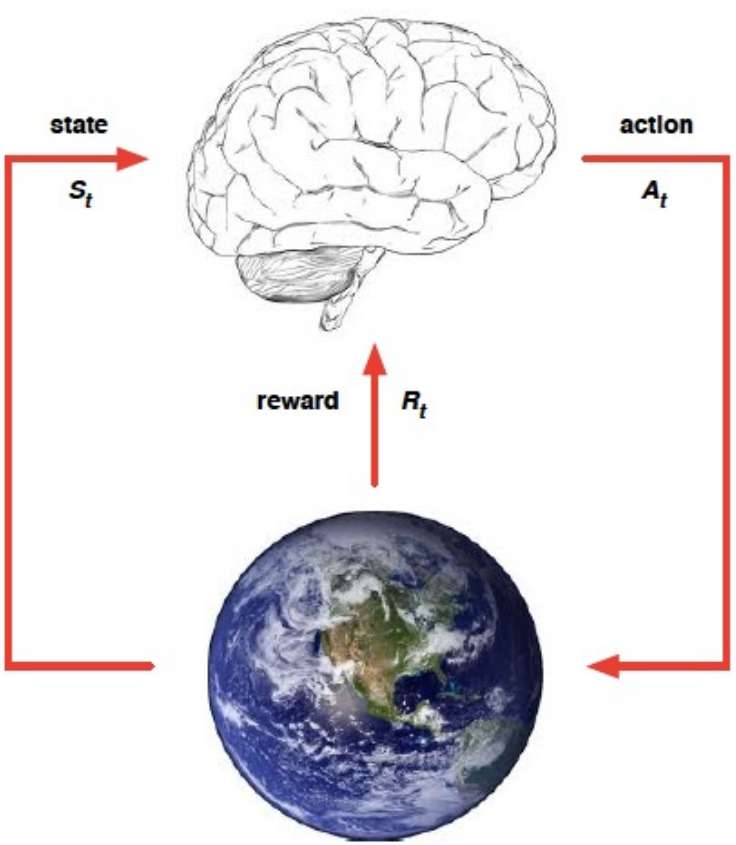
Find the Best Policy $\pi_*(s)$
by Computing the Best Value Function $v_*(s)$

Broad Assumptions

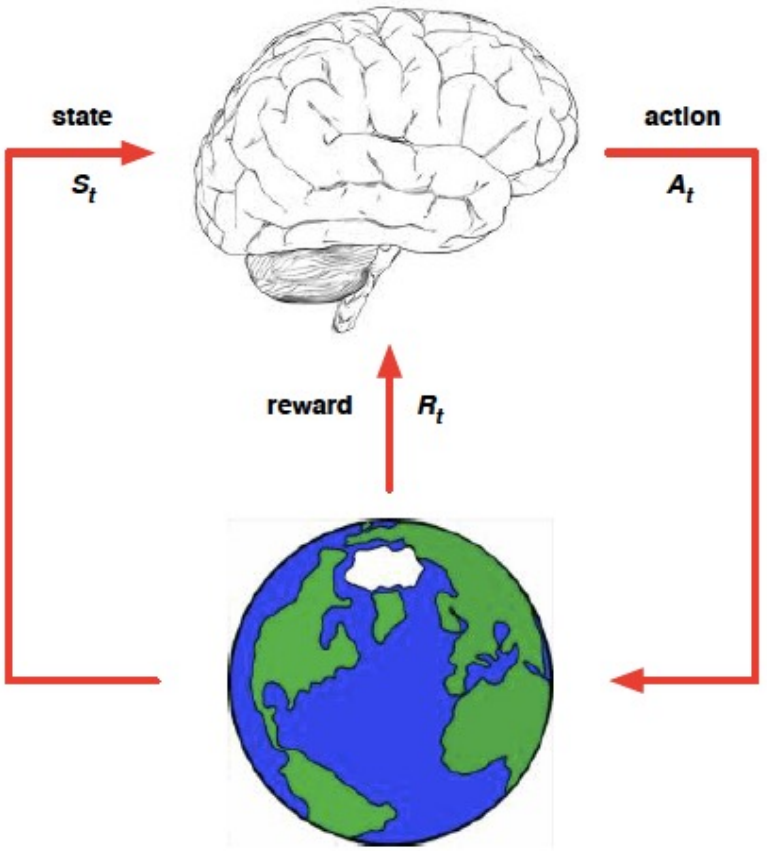


Fully Observable	Partially Observable
Model Based	Model Free

Model Free vs Model Based



Model Free



Model Based

Markov Processes

Markov Property

"The future is independent of the past given the present"

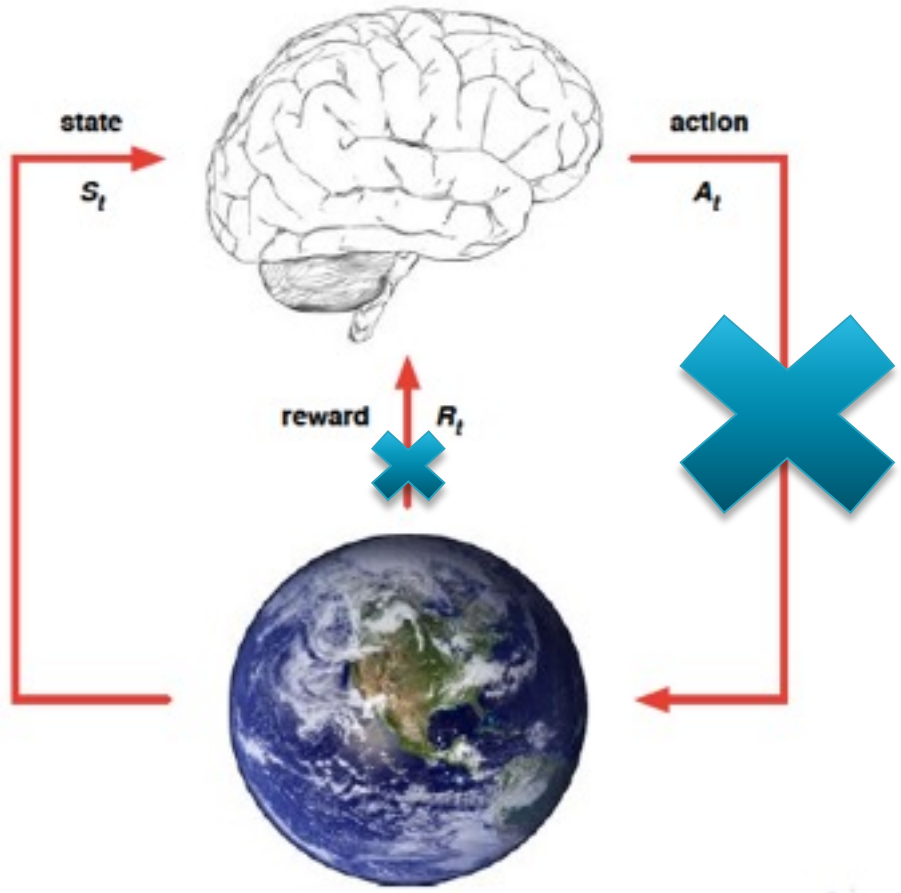
Definition

A state S_t is *Markov* if and only if

$$\mathbb{P}[S_{t+1} \mid S_t] = \mathbb{P}[S_{t+1} \mid S_1, ..., S_t]$$

- The state captures all relevant information from the history
- Once the state is known, the history may be thrown away
- i.e. The state is a sufficient statistic of the future

Markov Process



The Agent does not have any influence on the evolution of the System State

Markov Process

A Markov process is a memoryless random process, i.e. a sequence of random states $S_1, S_2, \ldots$ with the Markov property.

Definition

A *Markov Process* (or *Markov Chain*) is a tuple $\langle \mathcal{S}, \mathcal{P} \rangle$

- $\mathcal{S}$ is a (finite) set of states
- $\mathcal{P}$ is a state transition probability matrix, $\mathcal{P}_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s]$

State Transition Matrix

For a Markov state s and successor state s', the *state transition probability* is defined by

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

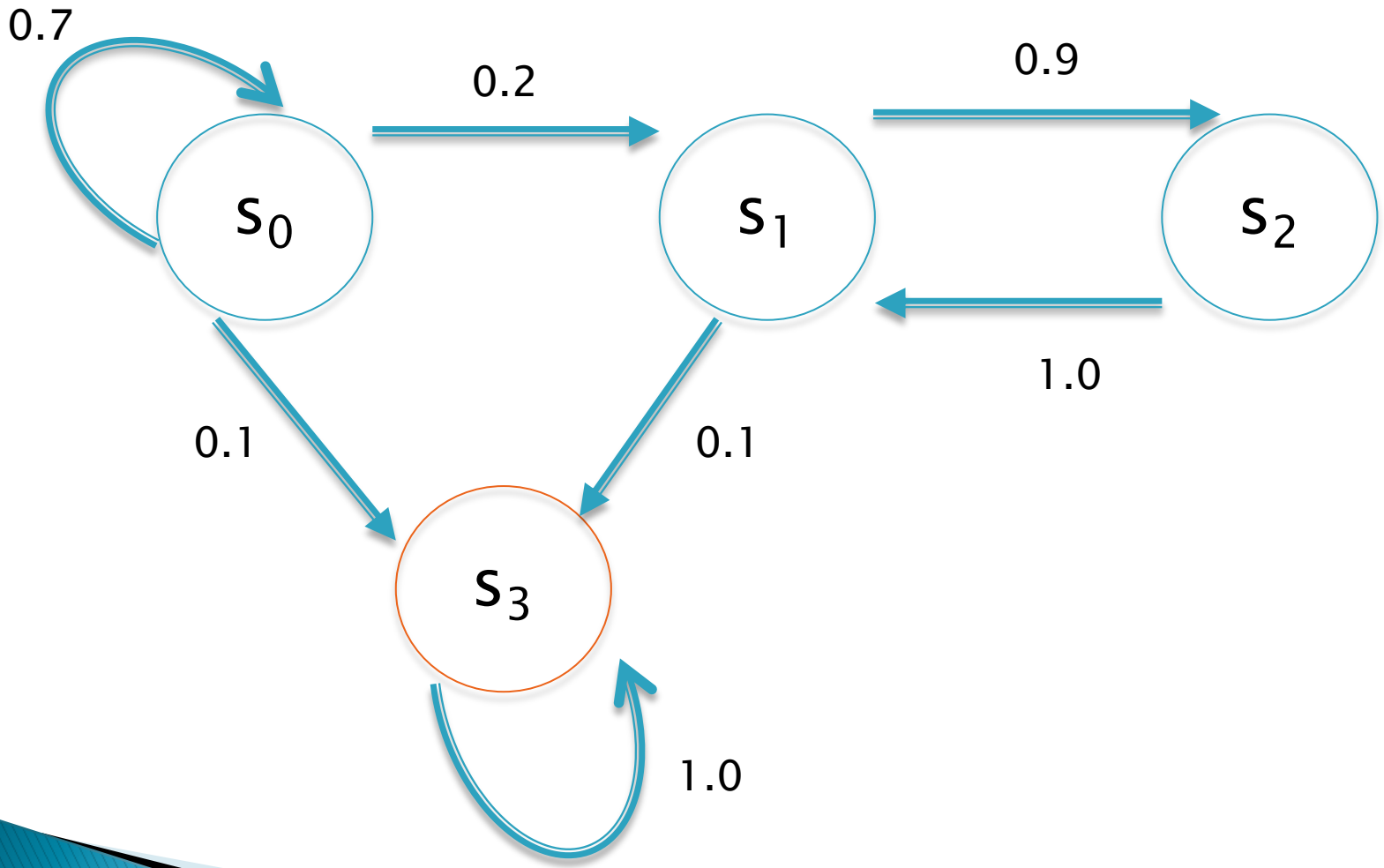
State transition matrix $\mathcal{P}$ defines transition probabilities from all states s to all successor states s',

$$\mathcal{P} = \begin{matrix} \\ \text{from} \end{matrix} \begin{matrix} \text{to} \\ \end{matrix} \begin{bmatrix} \mathcal{P}_{11} & \dots & \mathcal{P}_{1n} \\ \vdots & & \\ \mathcal{P}_{n1} & \dots & \mathcal{P}_{nn} \end{bmatrix}$$

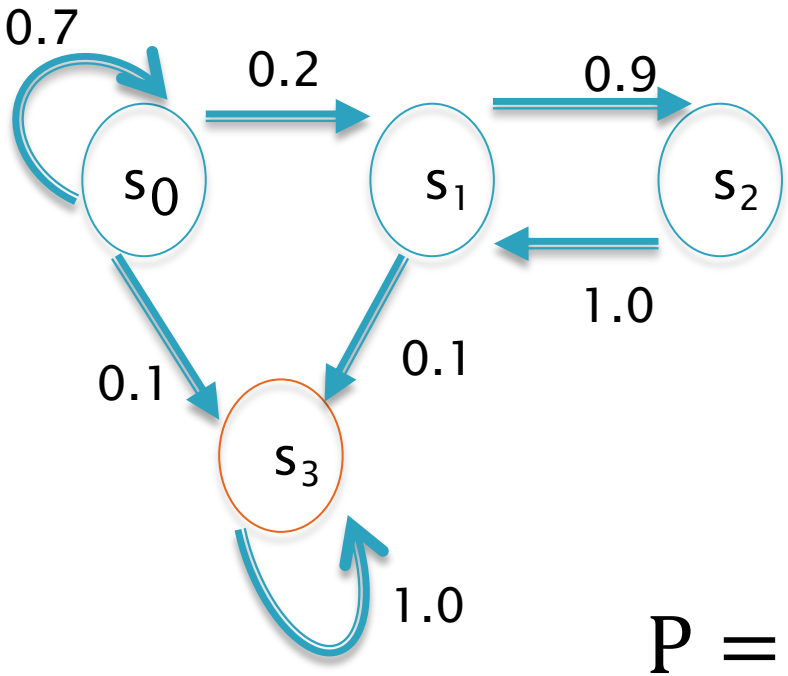
where each row of the matrix sums to 1.

The Matrix P completely characterizes the model!

Example of a Markov Chain

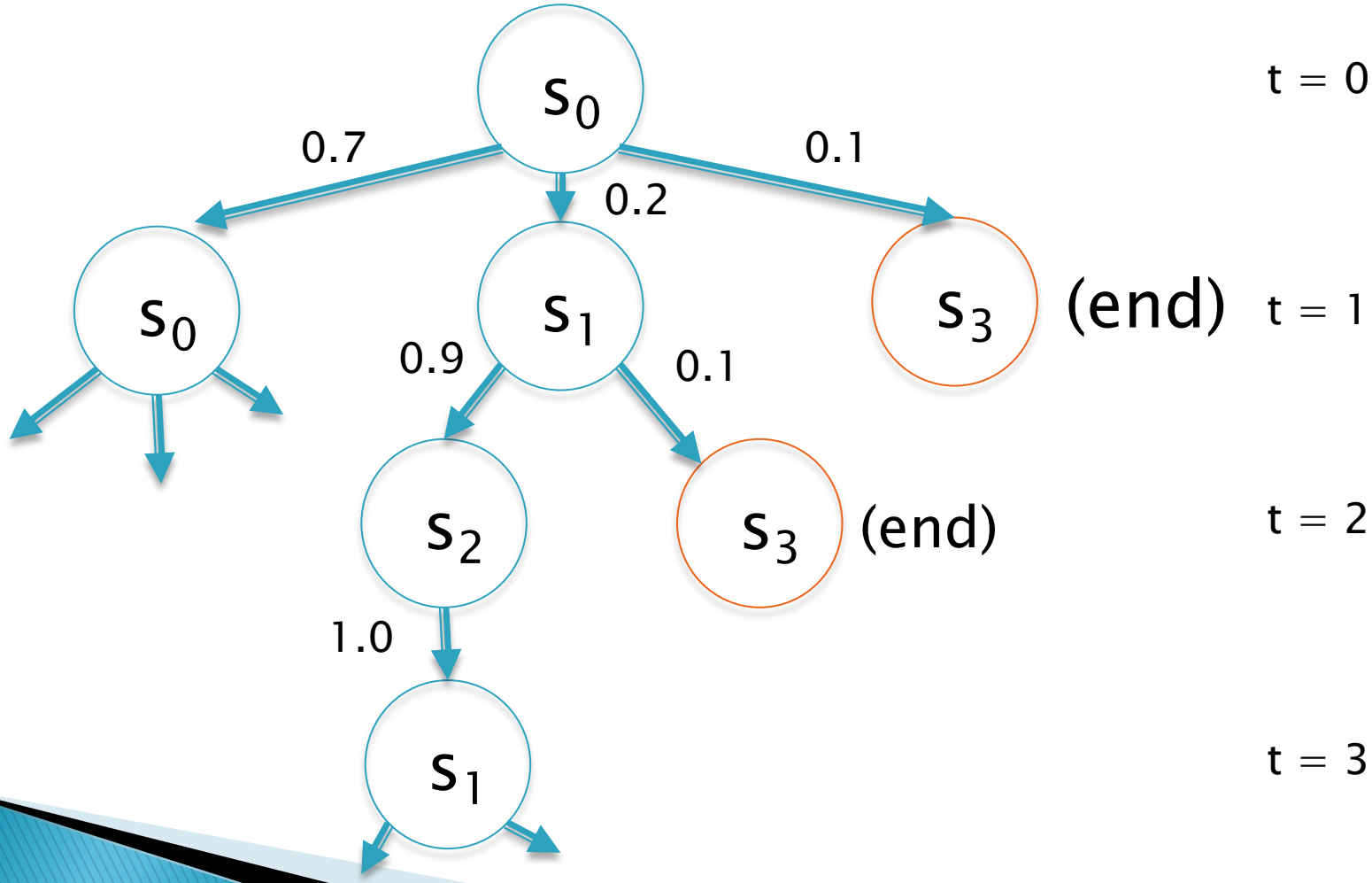


Example: Sate Transition Representation



$$P = \begin{array}{c} \begin{matrix} s_0 & s_1 & s_2 & s_3 \end{matrix} \\ \begin{pmatrix} 0.7 & 0.2 & 0 & 0.1 \\ 0 & 0 & 0.9 & 0.1 \\ 0 & 1.0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array} \begin{matrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{matrix}$$

Example: Tree Representation (Time Evolution)



Example: Computing Episodes

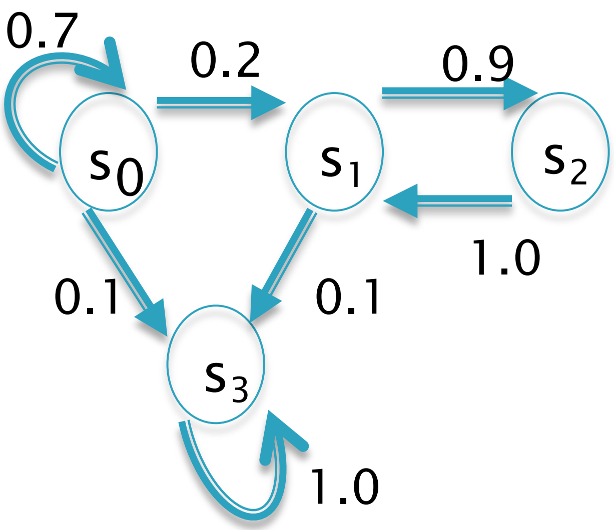
```
import numpy as np
import random as rnd
transition_probabilities = [
        [0.7, 0.2, 0.0, 0.1],  # from s0 to s0, s1, s2, s3
        [0.0, 0.0, 0.9, 0.1],  # from s1 to ...
        [0.0, 1.0, 0.0, 0.0],  # from s2 to ...
        [0.0, 0.0, 0.0, 1.0],  # from s3 to ...
    ]

n_max_steps = 50

def print_sequence(start_state=0):
    current_state = start_state
    print("States:", end=" ")
    for step in range(n_max_steps):
        print(current_state, end=" ")
        if current_state == 3:
            break
        current_state = np.random.choice(range(4), p=transition_probabilities[current_state])
    else:
        print("...", end="")
    print()

for _ in range(10):
    print_sequence()
```

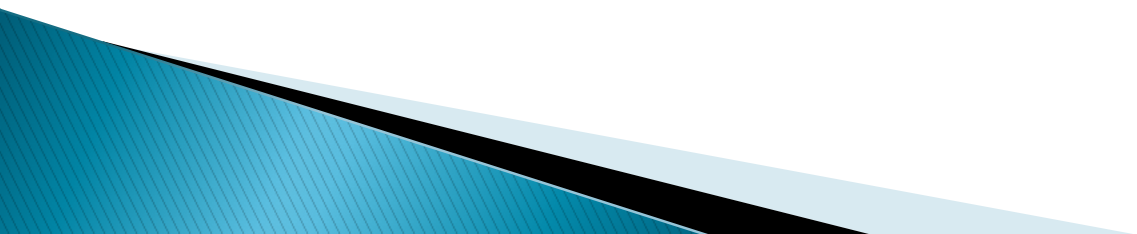
Example: Episodes



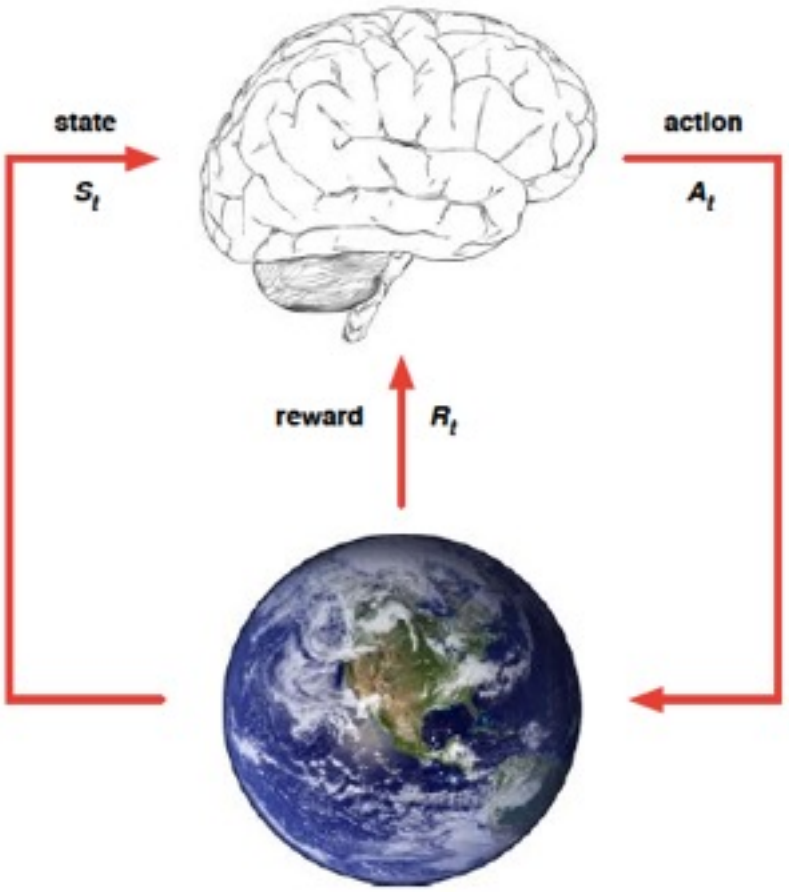
Sample Episodes starting from state s_0

```
States: 0 0 0 3
States: 0 0 0 0 0 0 0 0 0 1 2 1 2 1 2 1 2 1 3
States: 0 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 3
States: 0 0 3
States: 0 0 1 2 1 2 1 2 1 3
States: 0 0 0 0 0 0 0 0 0 0 3
States: 0 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 3
States: 0 0 1 2 1 2 1 2 1 2 1 3
States: 0 0 3
States: 0 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 ...
```

Markov Decision Processes



Markov Decision Process



The Agent is now able to influence the evolution of future states

The difference between watching a Movie vs playing a video game

Markov Decision Processes

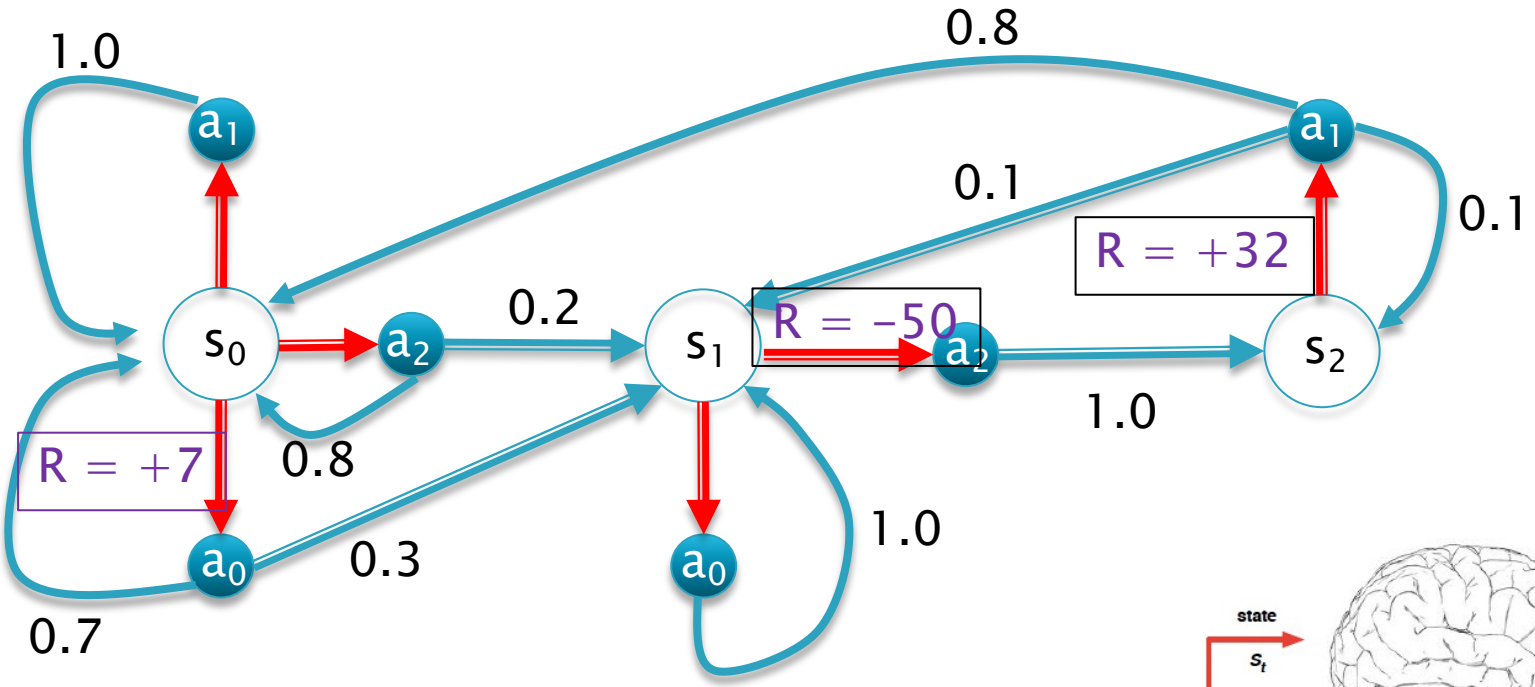
A Markov Decision Process is a Markov Process with Actions and Rewards. It is in an environment in which all states are Markov.

Definition

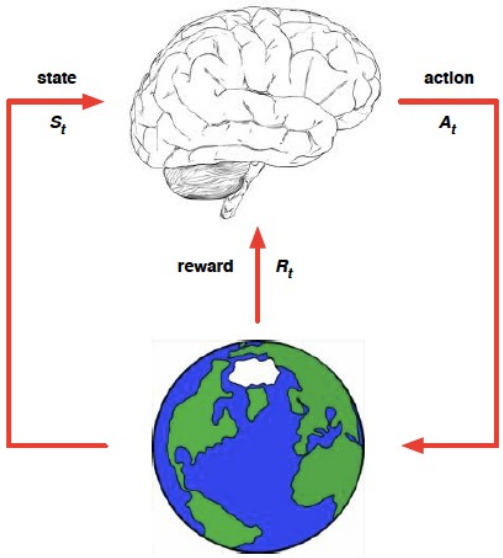
A *Markov Decision Process* is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

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

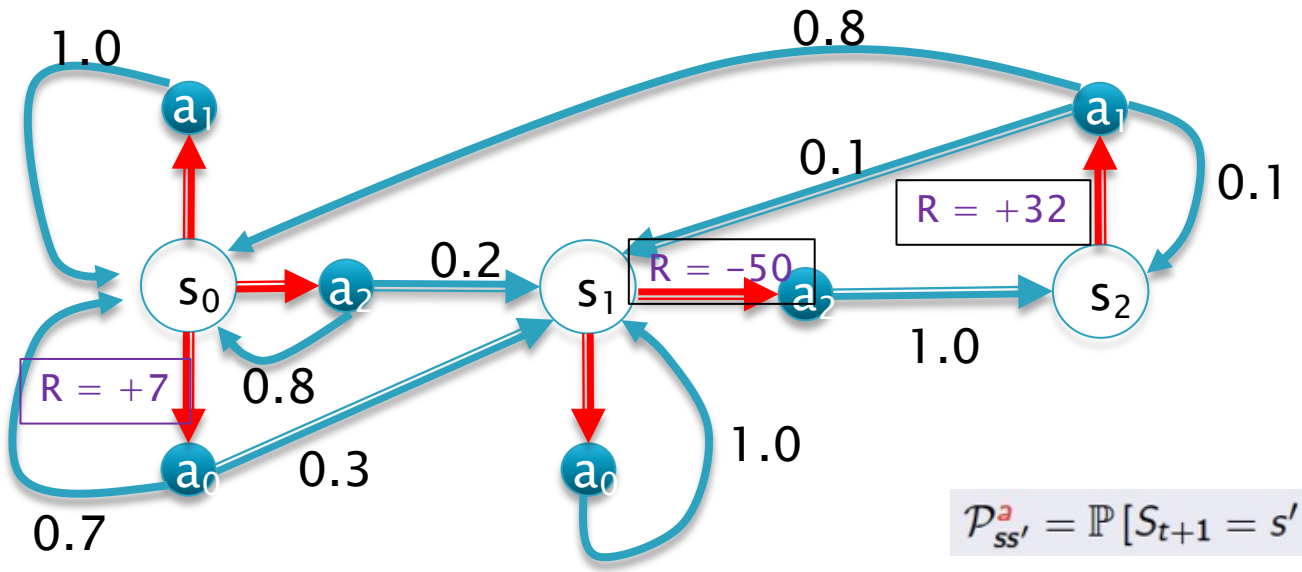
Example of a MDP



This is an example of a Model based RL



MDP Matrix Representation



$$\mathcal{P}^{a}_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$$

When in state s_0

	s_0	s_1	s_2
a_0	0.7	0.3	0
a_1	1.0	0	0
a_2	0.8	0.2	0

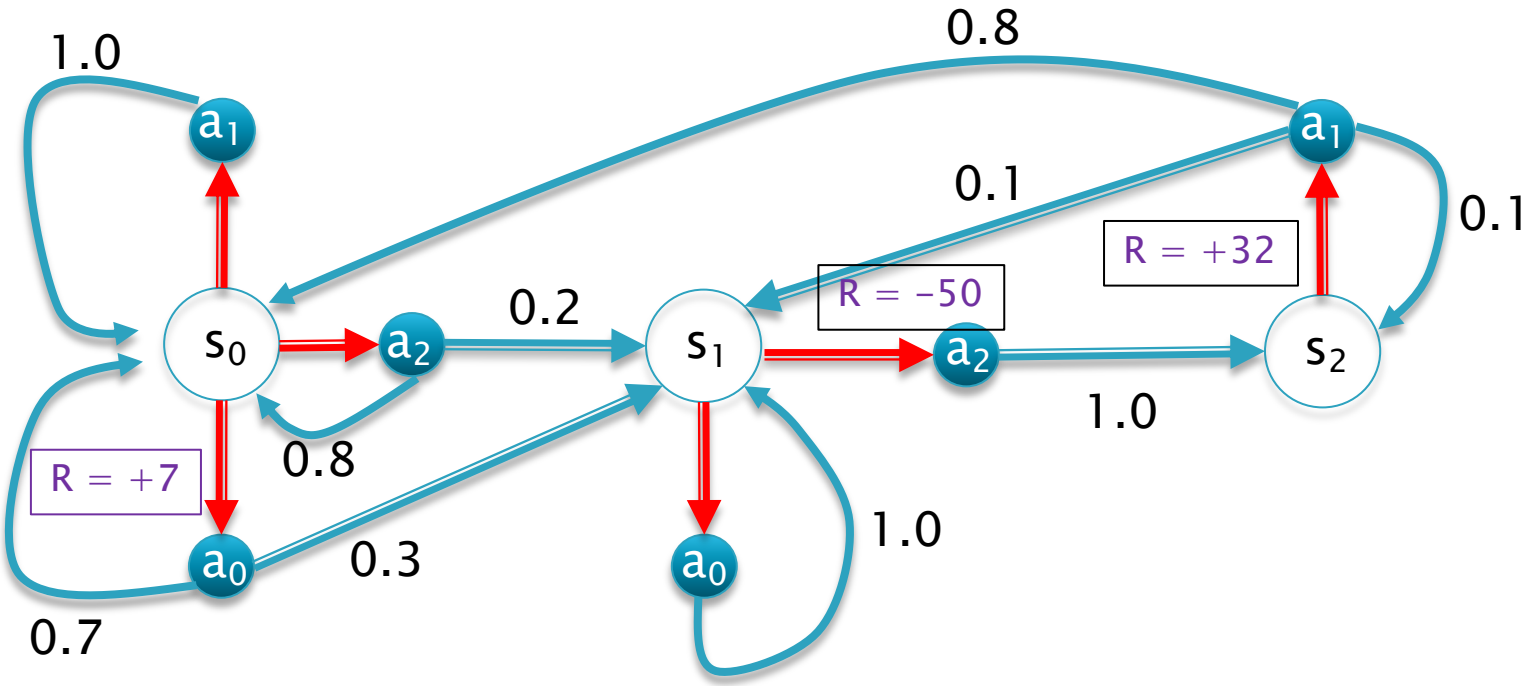
When in state s_1

	s_0	s_1	s_2
a_0	0	1.0	0
a_1	—	—	—
a_2	0	0	1.0

When in state s_2

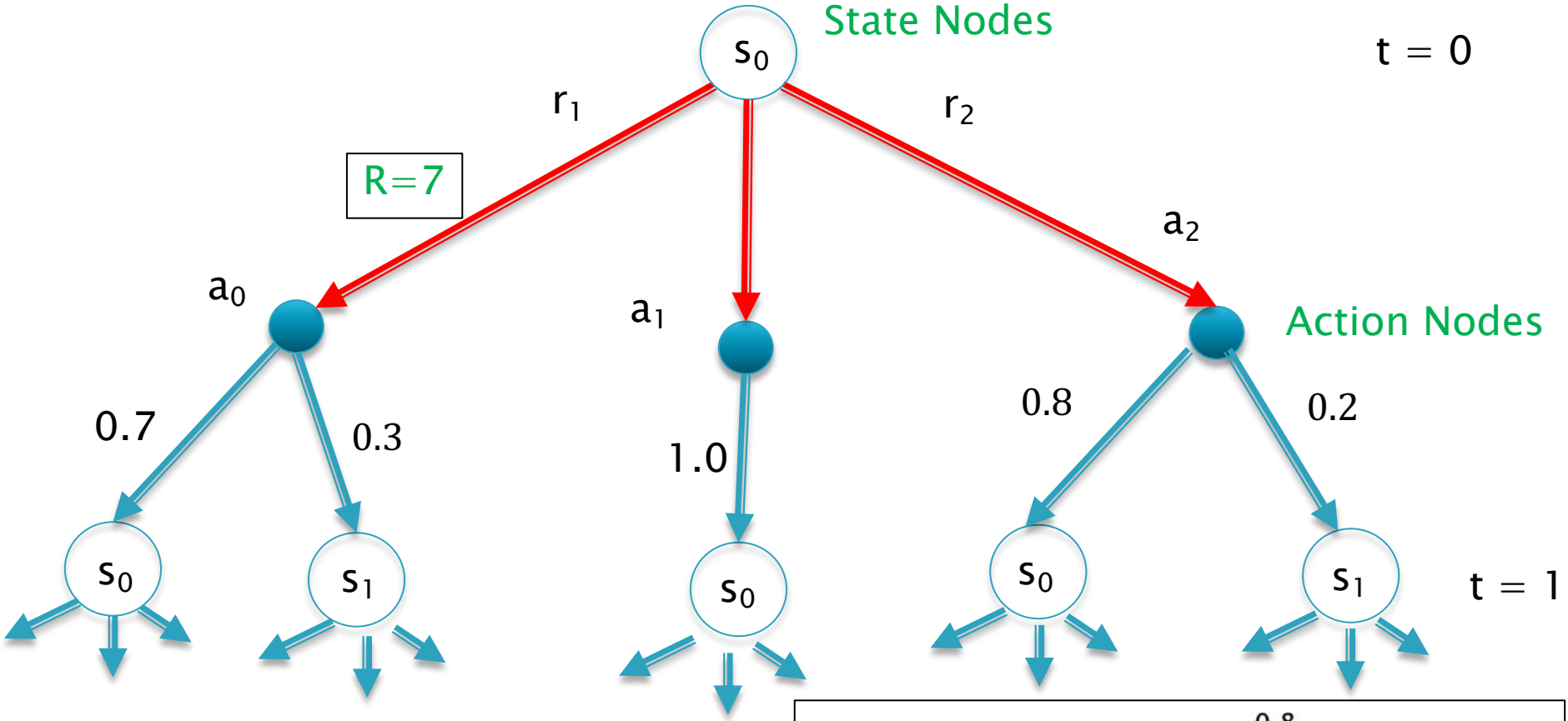
	s_0	s_1	s_2
a_0	—	—	—
a_1	0.8	0.1	0.1
a_2	—	—	—

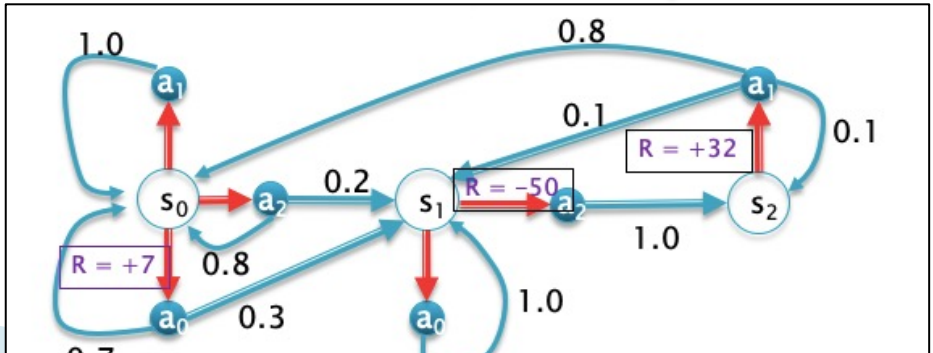
MDP: Matrix Representation



$$P^{a_0} = \begin{array}{c} \\ s_0 \\ s_1 \end{array} \begin{array}{c} \begin{array}{ccc} s_0 & s_1 & s_2 \end{array} \\ \begin{pmatrix} 0.7 & 0.3 & 0 \\ 0 & 1.0 & 0 \end{pmatrix} \end{array} \quad P^{a_1} = \begin{array}{c} \begin{array}{ccc} s_0 & s_1 & s_2 \end{array} \\ \begin{pmatrix} 1.0 & 0 & 0 \\ 0.8 & 0.1 & 0.1 \end{pmatrix} \end{array} \begin{array}{c} s_0 \\ s_2 \end{array} \quad P^{a_2} = \begin{array}{c} \begin{array}{ccc} s_0 & s_1 & s_2 \end{array} \\ \begin{pmatrix} 0.8 & 0.2 & 0 \\ 0 & 0 & 1.0 \end{pmatrix} \end{array} \begin{array}{c} s_0 \\ s_1 \end{array}$$

MDP Tree Representation (Rollouts)





Policies

Definition

A *policy* π is a distribution over actions given states,

$$\pi(a|s) = \mathbb{P}\left[A_t = a \mid S_t = s\right]$$

- A policy fully defines the behaviour of an agent
- MDP policies depend on the current state (not the history)
- i.e. Policies are *stationary* (time-independent),
  $A_t \sim \pi(\cdot|S_t), \forall t > 0$

MDP: Examples of Policies

- Policy: BoomBust

Policy Matrix

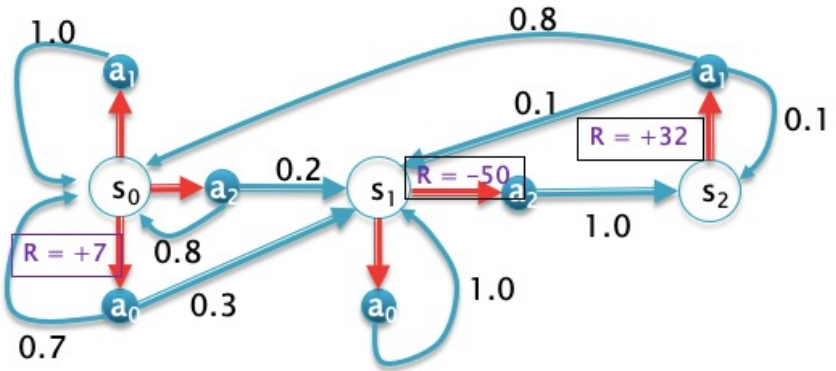
$$\pi(s) = \begin{array}{c} \\ s_0 \\ s_1 \\ s_2 \end{array} \begin{array}{c} \begin{array}{ccc} a_0 & a_1 & a_2 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \end{array}$$

OR

$$\pi(s_0) = a_0$$
$$\pi(s_1) = a_2$$
$$\pi(s_2) = a_1$$

Resulting Markov Chain Transition Matrix

$$P^{\pi} = \begin{array}{c} \\ s_0 \\ s_1 \\ s_2 \end{array} \begin{array}{c} \begin{array}{ccc} s_0 & s_1 & s_2 \end{array} \\ \begin{pmatrix} 0.7 & 0.3 & 0 \\ 0 & 0 & 1 \\ 0.8 & 0.1 & 0.1 \end{pmatrix} \end{array}$$

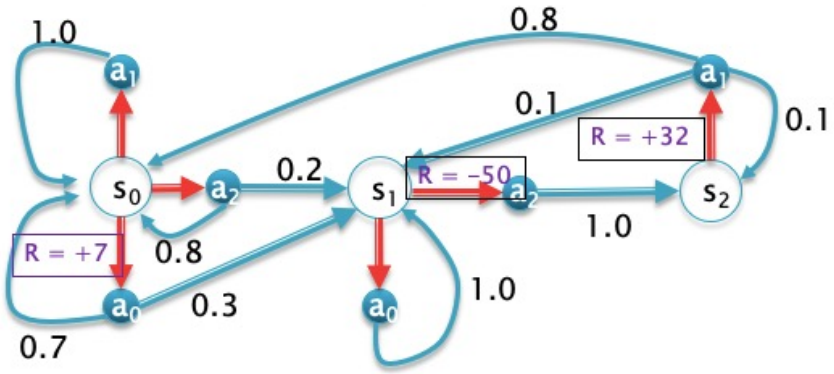


MDP: Examples of Policies

- Policy: Safe

$$\pi(s) = \begin{matrix} a_0 & a_1 & a_2 \end{matrix} \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{matrix} s_0 \\ s_1 \\ s_2 \end{matrix}$$

$$P^{\pi} = \begin{matrix} s_0 & s_1 & s_2 \end{matrix} \begin{pmatrix} 0.7 & 0.3 & 0 \\ 0 & 1.0 & 0 \\ 0.8 & 0.1 & 0.1 \end{pmatrix} \begin{matrix} s_0 \\ s_1 \\ s_2 \end{matrix}$$



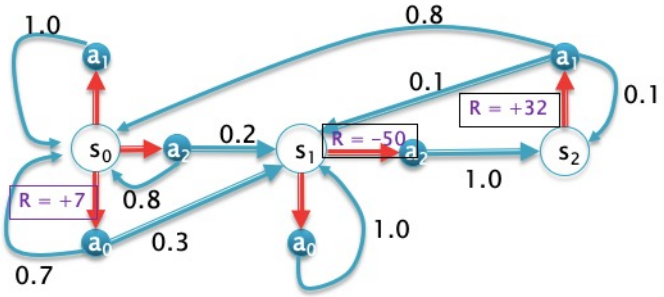
MDP: Examples of Policies

- Policy: Random

Policy Matrix

$$\pi(s) = \begin{pmatrix} 1/3 & 1/3 & 1/3 \\ 1/2 & 0 & 1/2 \\ 0 & 1 & 0 \end{pmatrix}$$

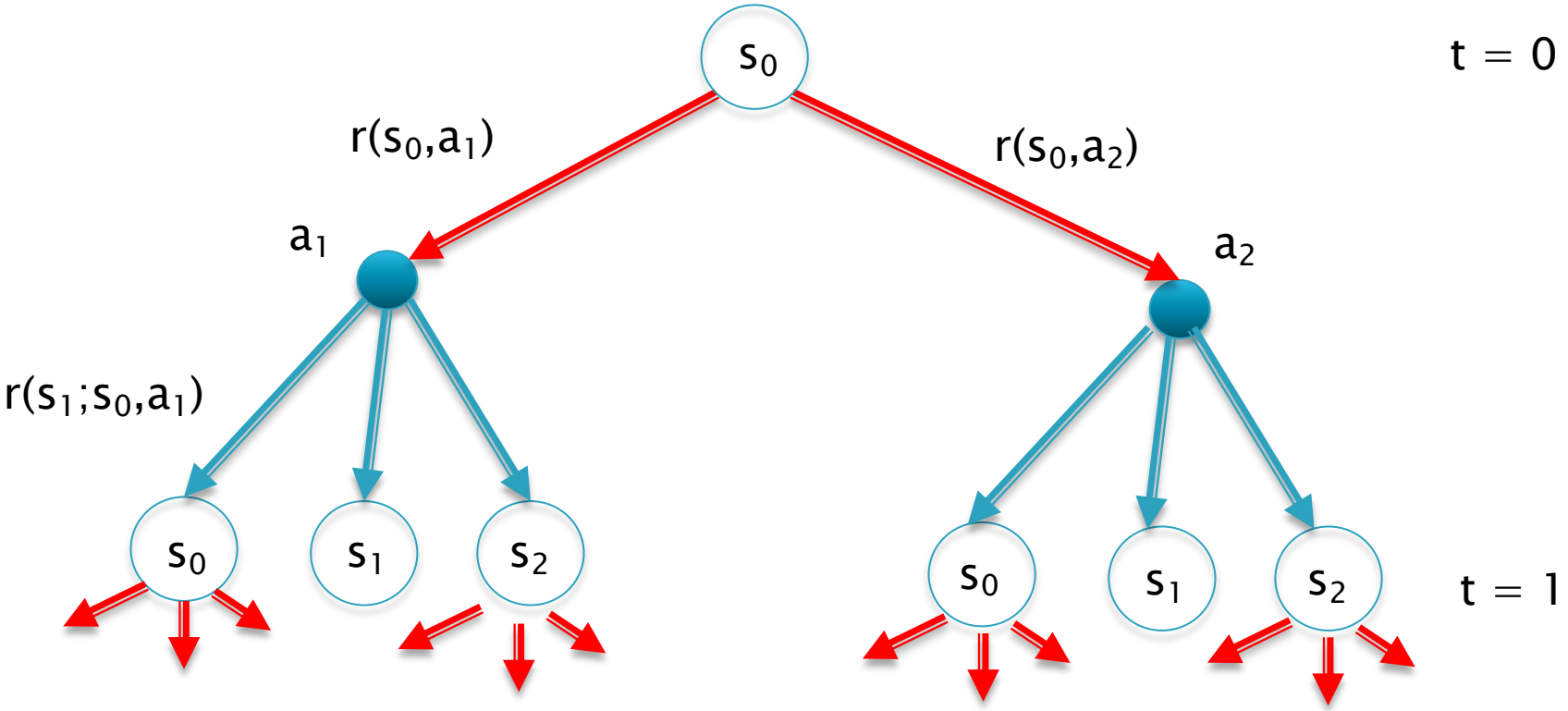
a_0 a_1 a_2 — s_0 s_1 s_2



$$P^{\pi} = ?$$

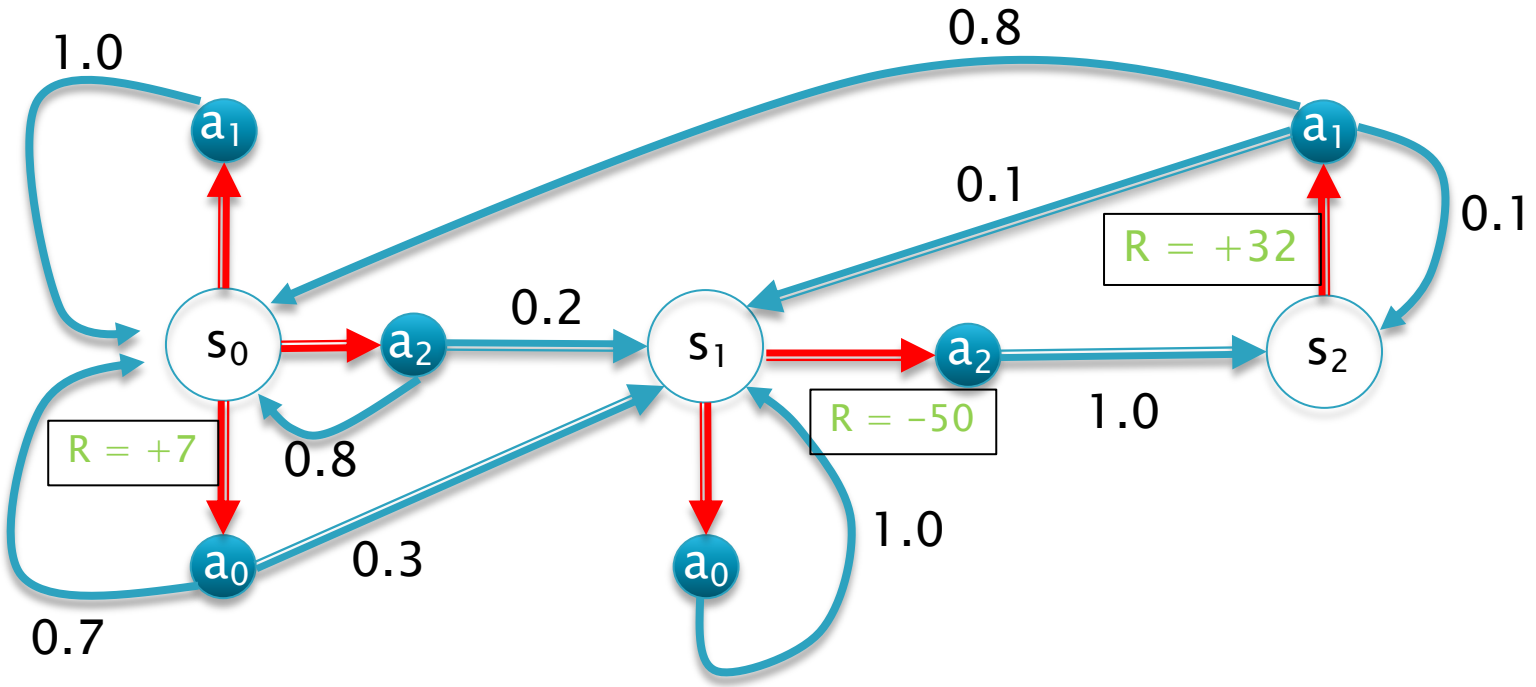
$$P^{\pi}(s'|s) = \sum_{a} \pi(a|s) P^{a}(s'|s)$$

MDP: Rewards



Two ways of specifying Rewards

Example: Specifying Rewards



Rewards r(s,a) when transitioning from s_0, s_1 and s_2 respectively

$$
\begin{array}{c} \\ s_0 \\ s_1 \\ s_2 \end{array}
\begin{array}{c}
\begin{array}{ccc} a_0 & a_1 & a_2 \end{array} \\
\begin{pmatrix} +7 & 0 & 0 \\ 0 & 0 & -50 \\ 0 & +32 & 0 \end{pmatrix}
\end{array}
$$

MDP: Return

Definition

The *return* G_t is the total discounted reward from time-step t.

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- The *discount* $\gamma \in [0, 1]$ is the present value of future rewards
- The value of receiving reward R after $k+1$ time-steps is $\gamma^k R$.
- This values immediate reward above delayed reward.
  - γ close to 0 leads to "myopic" evaluation
  - γ close to 1 leads to "far-sighted" evaluation

Why Discount?

- Uncertainty about the future may not be fully represented
- If the reward is financial, immediate rewards may earn more interest than delayed rewards
- Animal/human behaviour shows preference for immediate reward

- Mathematically convenient to discount rewards
- Avoids infinite returns in cyclic Markov processes

- It is sometimes possible to use *undiscounted* Markov reward processes (i.e. $\gamma = 1$), e.g. if all sequences terminate.

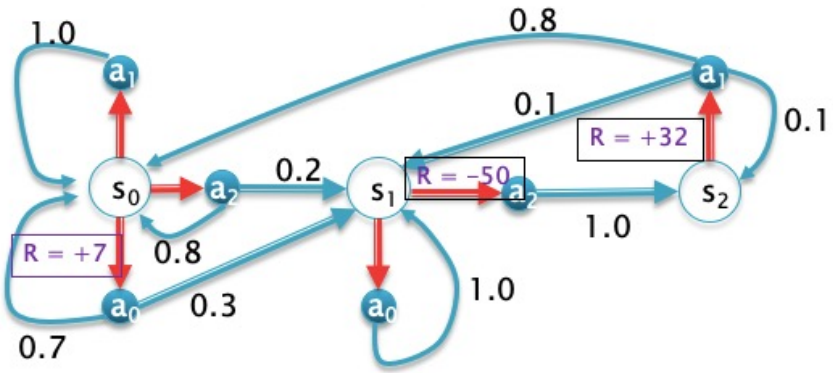
Example: MDP Returns

Sample returns, starting from state s_0 and $\gamma = 1$,

Sample returns computed from 1000 episodes and 100 steps per episode

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

```
policy_fire
States (+rewards): 0 1 (-50) 2 (32) 0 (10) 0 (10) 0 (10) 0 1 (-50) 2 (32) 0 ... Total rewards = -220
States (+rewards): 0 (10) 0 (10) 0 (10) 0 (10) 0 (10) 0 (10) 0 (10) 0 (10) 0 ... Total rewards = 40
States (+rewards): 0 (10) 0 (10) 0 (10) 0 1 (-50) 2 (32) 0 (10) 0 1 (-50) 2 (32) ... Total rewards = 160
States (+rewards): 0 (10) 0 (10) 0 (10) 0 (10) 0 (10) 0 1 (-50) 2 (32) 0 (10) 0 (10) ... Total rewards = 280
States (+rewards): 0 (10) 0 1 (-50) 2 1 (-50) 2 (32) 0 (10) 0 (10) 0 (10) 0 (10) ... Total rewards = 190
Summary: mean=122.2, std=134.956674, min=-340, max=490
```



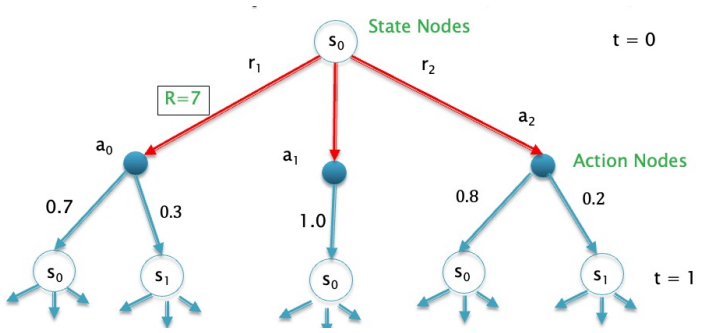
State Value Function: $v_\pi(s)$

Definition

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

$$v_\pi(s) = \mathbb{E}_\pi [G_t \mid S_t = s]$$

Answers the question: How good is it to be in state s, while following Policy π



Action Value Function: $q_\pi(s,a)$

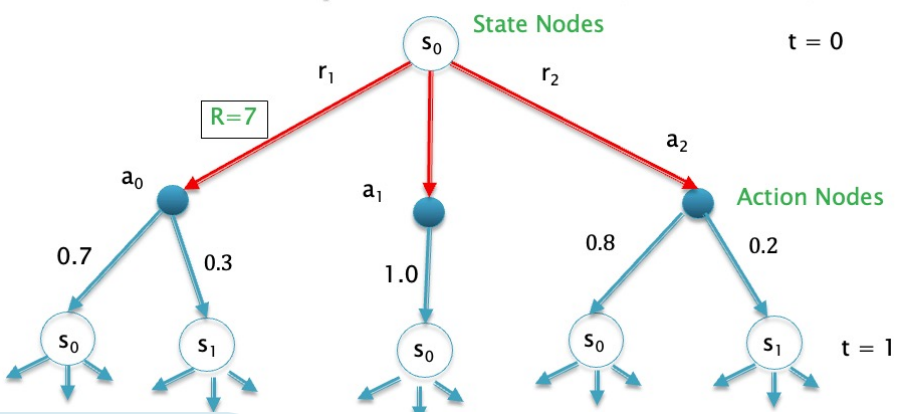
Definition

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

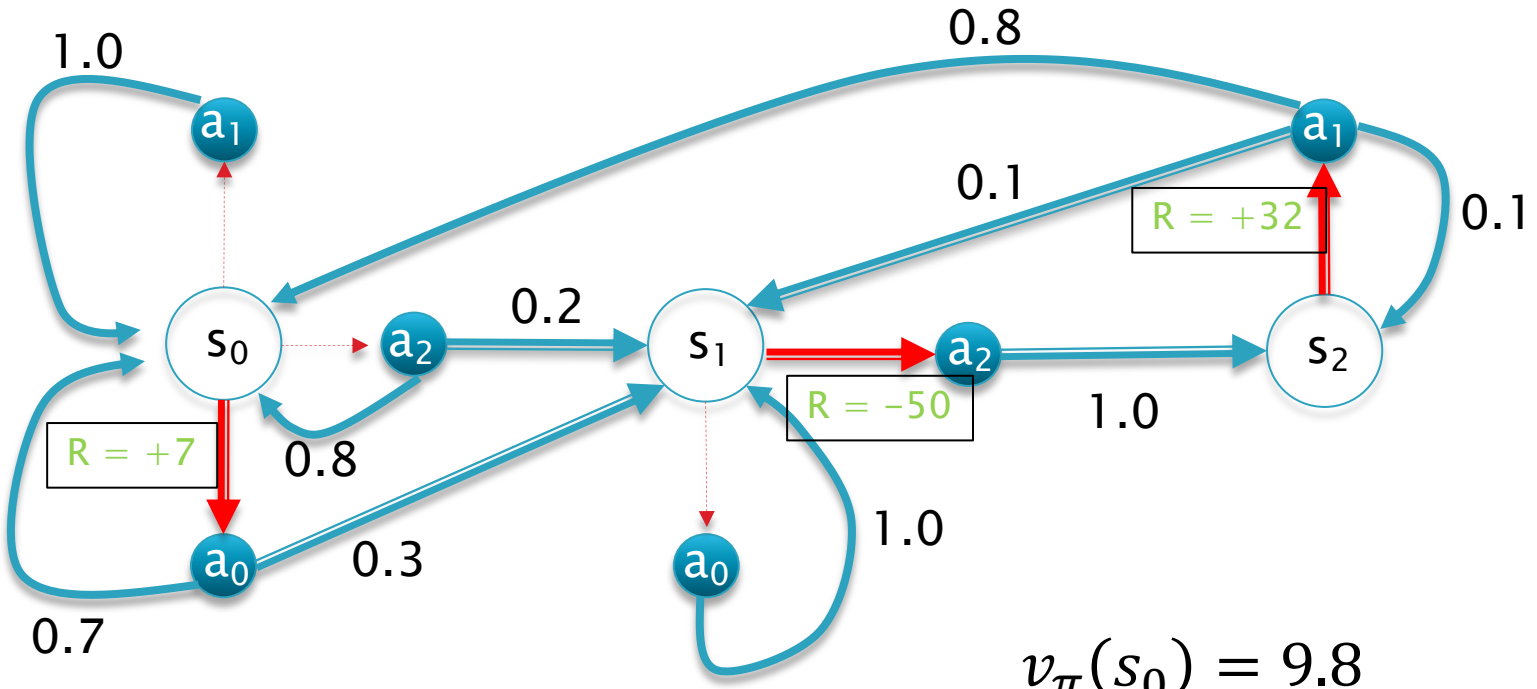
$$q_\pi(s, a) = \mathbb{E}_\pi [G_t \mid S_t = s, A_t = a]$$

Answers the question: How good is it to take Action a in state s, while following Policy π

If $q_\pi(s, a_1) > q_\pi(s, a_2)$, then choose Action a_1



State Value Function: Boom-Bust Policy



$\gamma = 0.9$

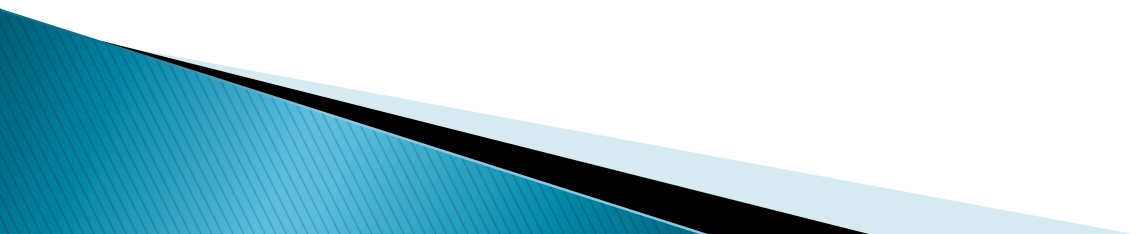
$v_\pi(s_0) = 9.8$

$v_\pi(s_1) = -12.5$

$v_\pi(s_2) = 41.7$

How to compute these?

Markov Decision Processes: The Bellman Expectation Equation

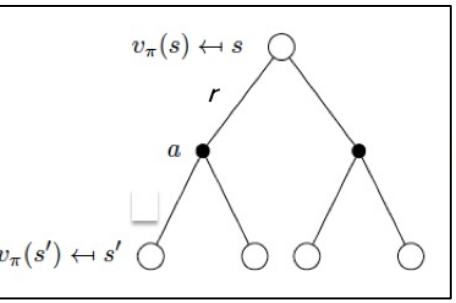


Bellman Expectation Equation

Problem: Given a policy π, Compute the Value Functions $v_\pi(s)$ and $q_\pi(s,a)$

Bellman Expectation Equation

The state-value function can again be decomposed into immediate reward plus discounted value of successor state,



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

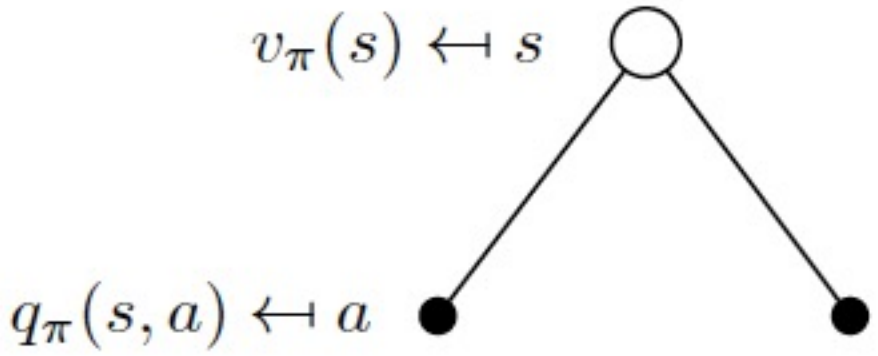
The action-value function can similarly be decomposed,

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

The Next State

The Next Action

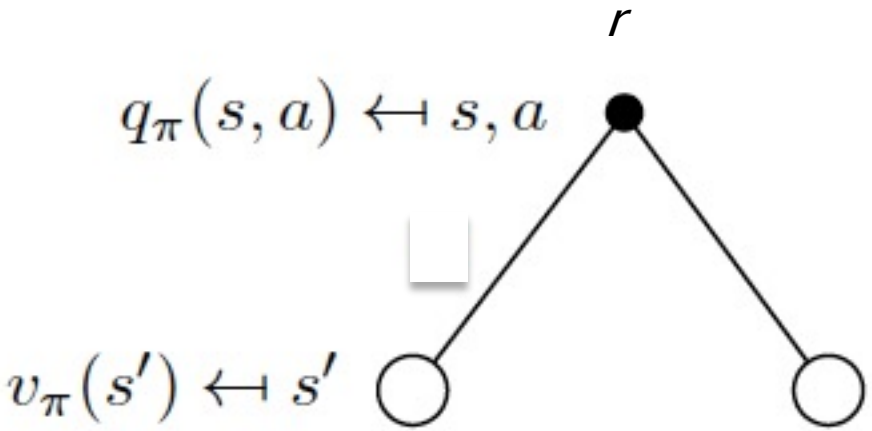
Bellman Expectation Equation for v_π



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

Assume this is known

Bellman Expectation Equation for q_π



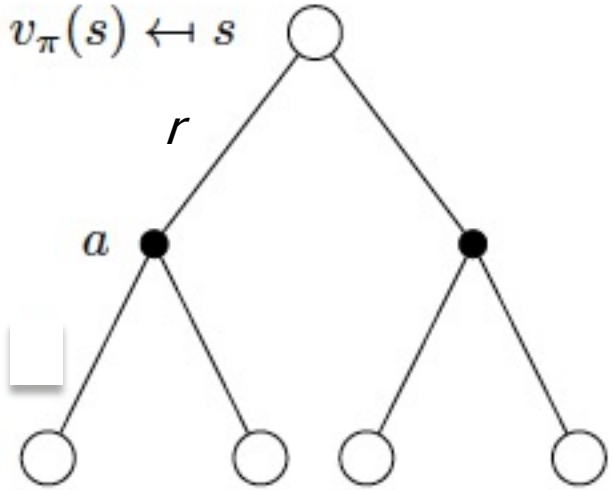
$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s')$$

Assume this is known

Bellman Expectation Equation for v_π

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

$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s')$$



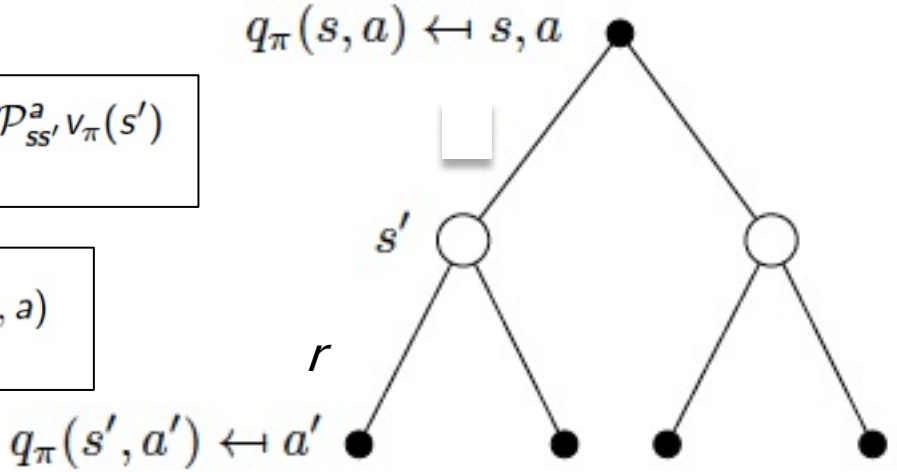
$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s') \right)$$

A Set of Linear Equation for v_π!

Bellman Expectation Equation for q_π

$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s')$$

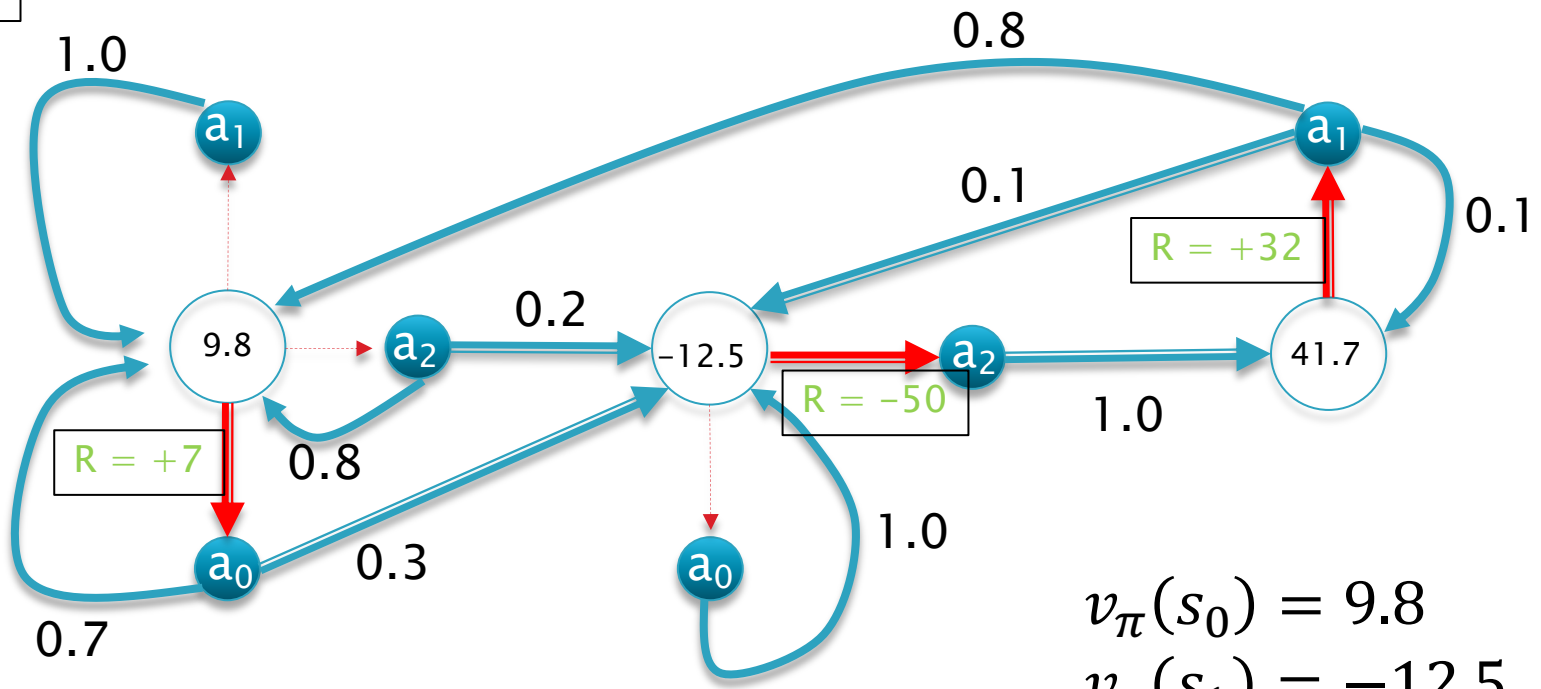
$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s,a)$$



$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \sum_{a' \in \mathcal{A}} \pi(a'|s') q_\pi(s',a')$$

Example: Bellman Expectation Equation for Boom-Bust Policy (Value Functions)

$\gamma = 0.9$



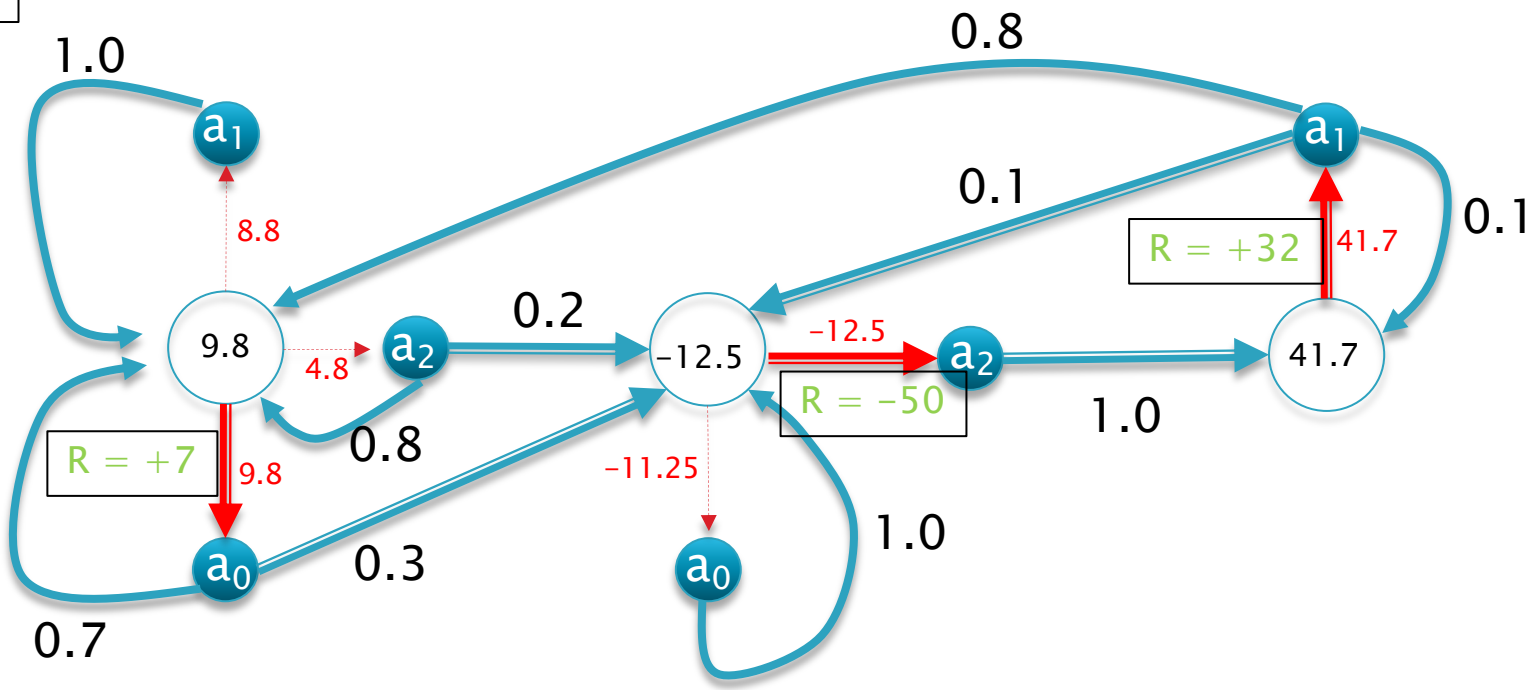
$v_\pi(s_0) = 9.8$
$v_\pi(s_1) = -12.5$
$v_\pi(s_2) = 41.7$

$9.8 = 7 + 0.9(0.7 * 9.8 + 0.3 * (-12.5))$

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s') \right)$$

Example: Bellman Expectation Equation for Fire Policy (Action Value Functions)

$\gamma = 0.9$



$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s')$$

Bellman Expectation Equation (Matrix Form)

The Bellman expectation equation can be expressed concisely using the induced MRP,

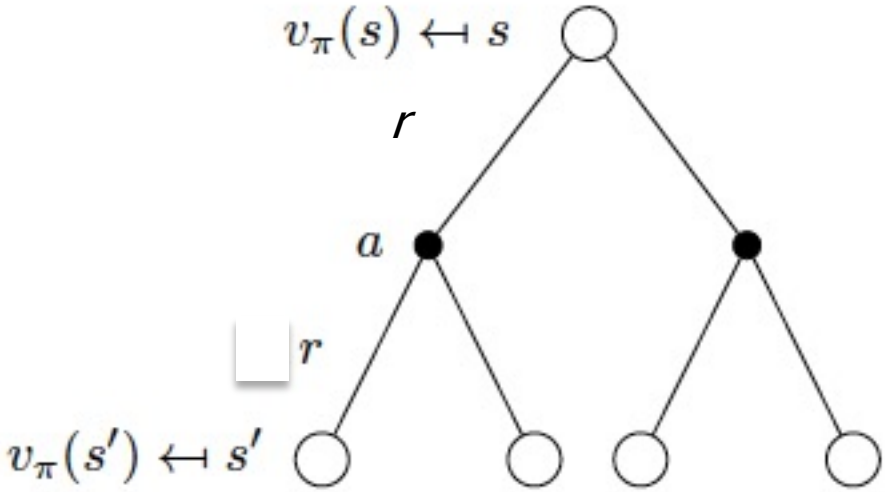
$$v_\pi = \mathcal{R}^\pi + \gamma \mathcal{P}^\pi v_\pi$$

with direct solution

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

Bellman Expectation Equation for v_π

Principle of Optimality
Decompose the problem into:
(1) A smaller problem that is easy to solve, and
(2) A bigger problem, that is assumed to be solved
(3) Put the 2 parts together to solve original problem

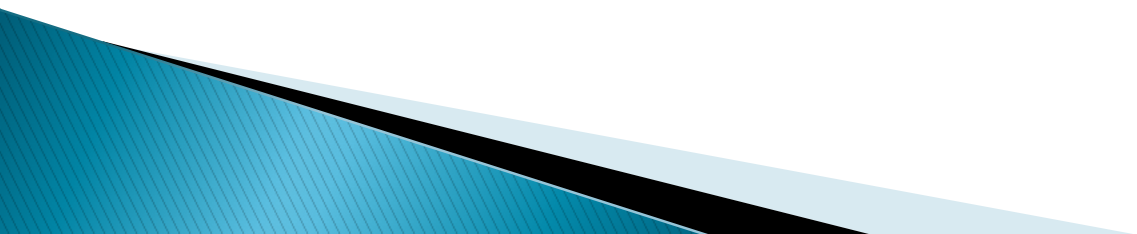


$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s') \right)$$

Value Function for State s = One Step Reward +Value Function for Next State s'

If $v_{\pi_1}(s) \geq v_{\pi_2}(s)$ for all s, then $\pi_1 \geq \pi_2$

Markov Decision Processes: The Bellman Optimality Equation



Next Step

The Bellman Expectation Equations evaluate how good a State is for a particular Policy, but
They don't tell us the Optimal Policy

How do we obtain the Optimal Policy?

Optimal State Value Function

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

$$v_*(s) = \max_\pi v_\pi(s)$$

$v_*(s)$ tells us the maximum reward that can be extracted from the system, when starting in State s

It doesn't tell us what Policy to follow

Optimal Action Value Function

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

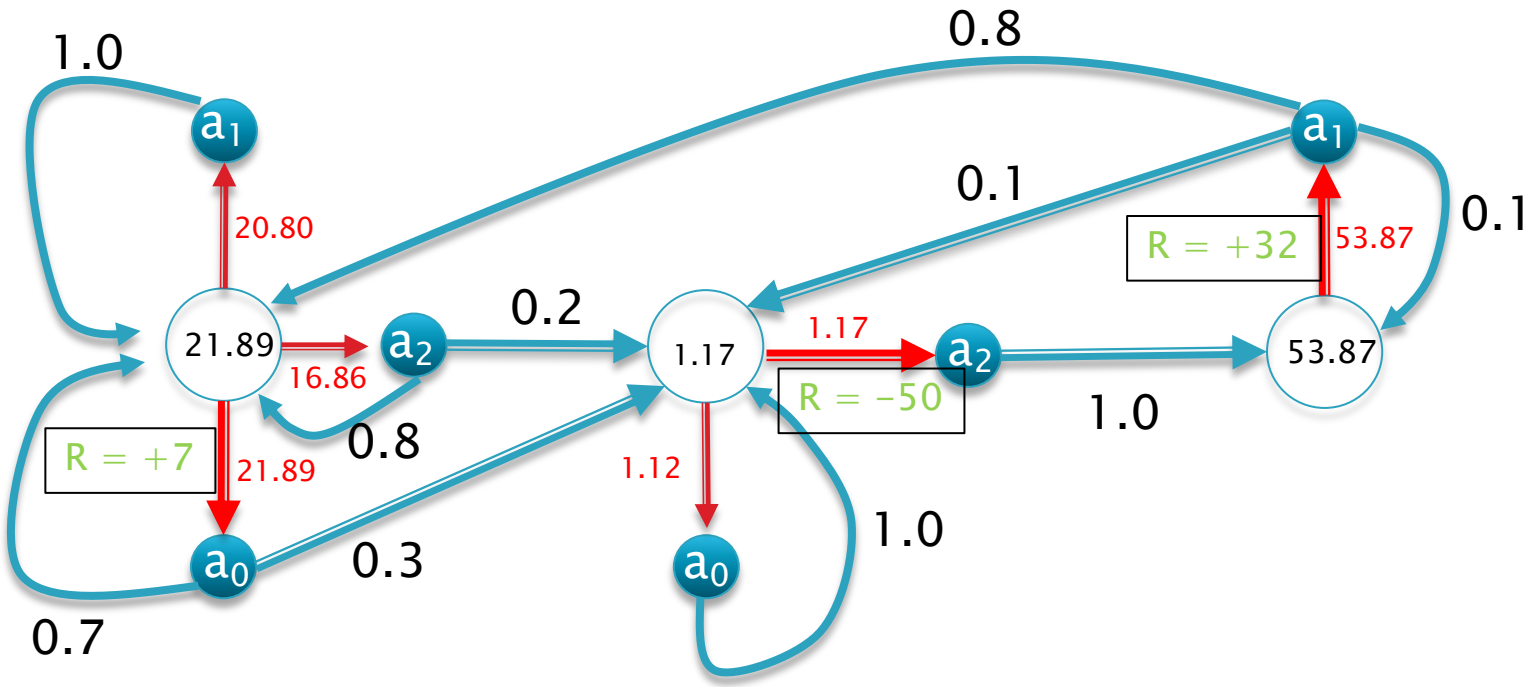
$$q_*(s, a) = \max_{\pi} q_\pi(s, a)$$

$q_*(s,a)$ tells us the maximum reward that can be extracted from the system, If Action a is taken while in State s

Can we obtain the Optimal Policy from $q_*(s,a)$?

Example: Given v_* and q_*

$\gamma = 0.95$



What is the Optimal Policy?

Optimal Policy

In order to find optimal policy we have to define the notion of optimality

What does it mean for one policy to be better than another

Define a partial ordering over policies

$$\pi \geq \pi' \text{ if } v_\pi(s) \geq v_{\pi'}(s), \forall s$$

Optimal Policy

Define a partial ordering over policies

$$\pi \geq \pi' \text{ if } v_\pi(s) \geq v_{\pi'}(s), \forall s$$

Theorem

For any Markov Decision Process

- *There exists an optimal policy π_* that is better than or equal to all other policies, $\pi_* \geq \pi, \forall \pi$*
- *All optimal policies achieve the optimal value function, $v_{\pi_*}(s) = v_*(s)$*
- *All optimal policies achieve the optimal action-value function, $q_{\pi_*}(s, a) = q_*(s, a)$*

Finding an Optimal Policy

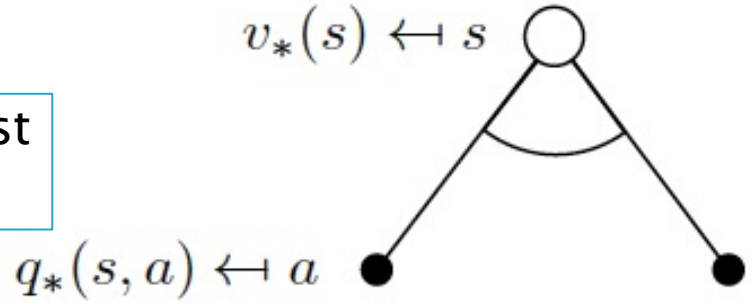
An optimal policy can be found by maximising over $q_*(s, a)$,

$$\pi_*(a|s) = \begin{cases} 1 & \text{if } a = \underset{a \in \mathcal{A}}{\text{argmax}}\ q_*(s, a) \\ 0 & \textit{otherwise} \end{cases}$$

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

Bellman Optimality Equation for v_*

The optimal value functions are recursively related by the Bellman optimality equations:



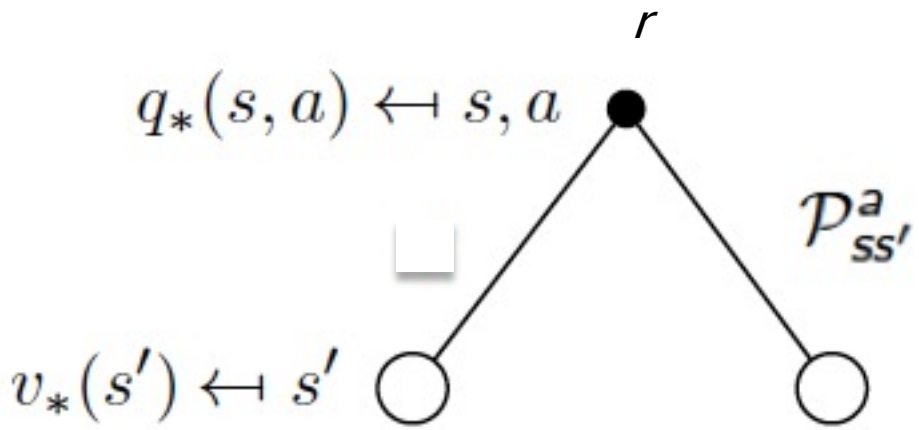
Choose the best action

$$v_*(s) = \max_a q_*(s,a)$$

Assume this is known

How do we compute q_* values?

Bellman Optimality Equation for q_*



$$q_*(s,a) = \mathcal{R}^a_s + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} v_*(s')$$

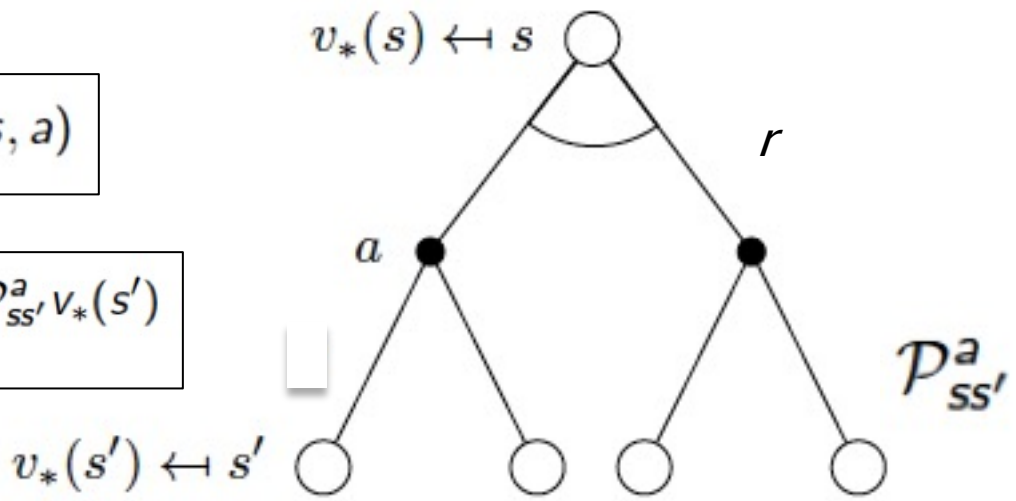
Assume this is known

Bellman Optimality Equation for v_* (2)

Putting the 2 pieces together

$$v_*(s) = \max_a q_*(s,a)$$

$$q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s')$$



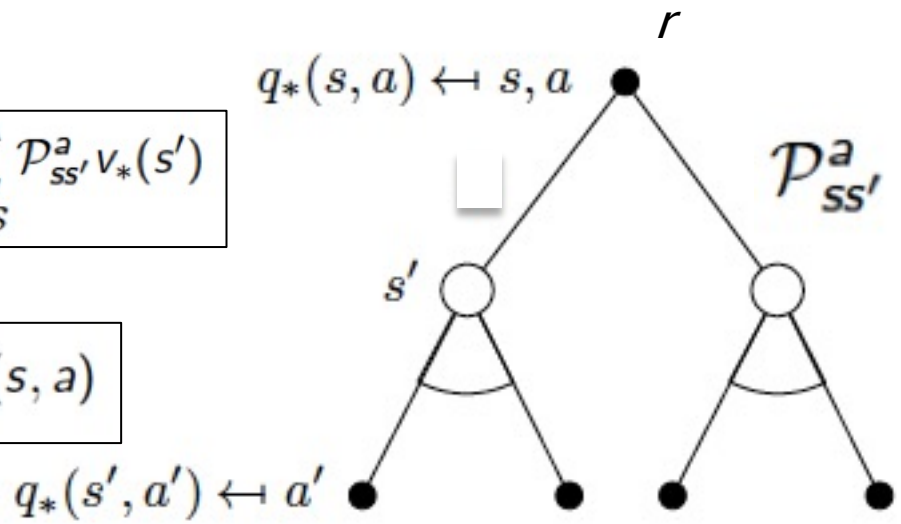
$$v_*(s) = \max_a \left[\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s') \right]$$

No dependency on the policy anymore, solely a function of the environment randomness

Bellman Optimality Equation for q_* (2)

$$q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s')$$

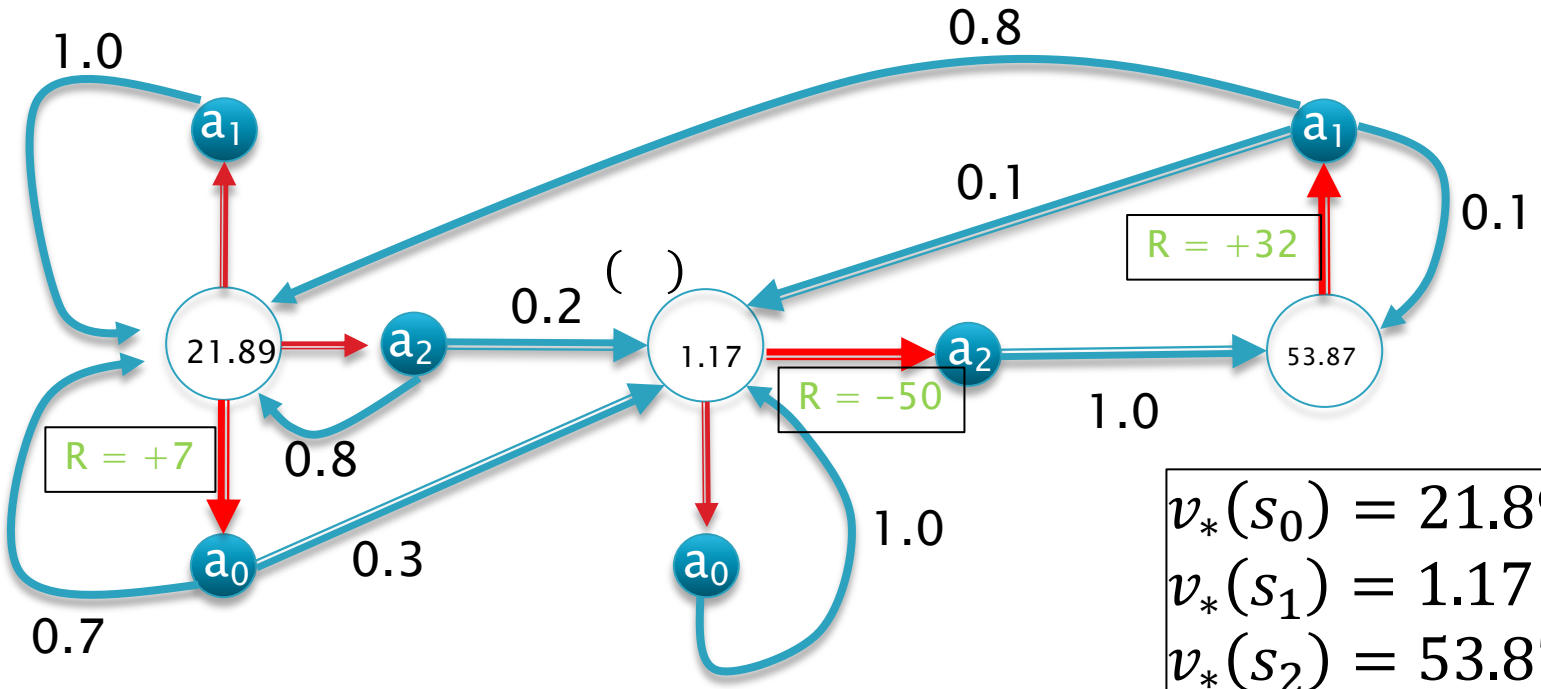
$$v_*(s) = \max_a q_*(s,a)$$



$$q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \max_{a'} q_*(s',a')$$

Example: Optimal Value Function

$\gamma = 0.95$



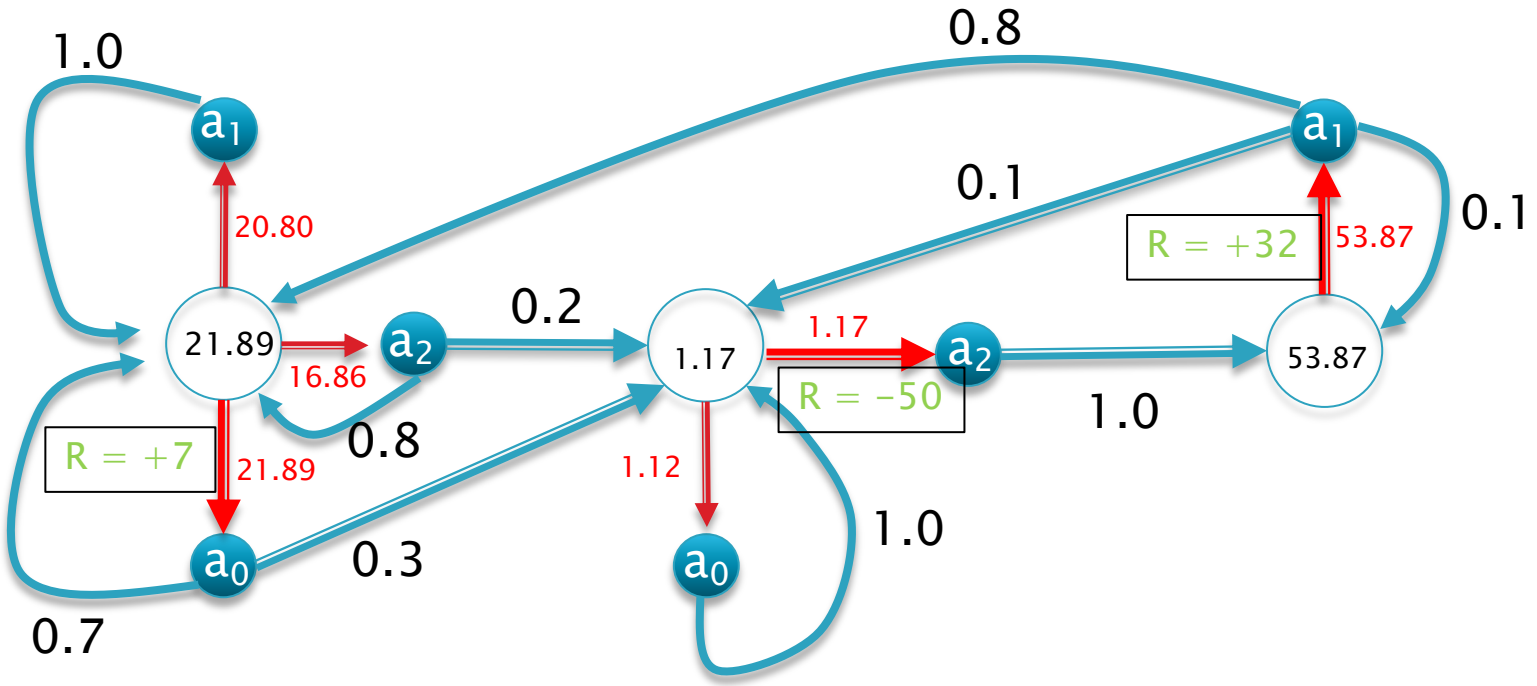
$v_*(s_0) = 21.89$
$v_*(s_1) = 1.17$
$v_*(s_2) = 53.87$

21.89 = max(7 + 0.95(0.7 * 21.89 + 0.3 * (1.17)),
0 + 0.95(21.89),
0 + 0.95(0.8*21.89 + 0.2 *(1.17)))

$$v_*(s) = \max_a \left[\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s') \right]$$

Example: Optimal Action-Value Function

$\gamma = 0.95$



$$q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s')$$

Summary

Bellman Expectation Equations

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s') \right)$$

$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \sum_{a' \in \mathcal{A}} \pi(a'|s') q_\pi(s',a')$$

Bellman Optimality Equations

$$v_*(s) = \max_a \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s') \right)$$

$$q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \max_{a'} q_*(s',a')$$

$$v_*(s) = \max_a q_*(s,a)$$

$$q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s')$$

Summary

Finding the Optimal Policy

$$v_*(s) = \max_a \left[\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s') \right]$$

$$q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s')$$

$$\boldsymbol{\pi_* = argmax_a(q_*(s,a))}$$

Solving the Bellman Optimality Equation

- Bellman Optimality Equation is non-linear
- No closed form solution (in general)
- Many iterative solution methods
  - Value Iteration
  - Policy Iteration

  Model Based, Lecture 3
  - Q-learning
  - Sarsa

  Model Free, Lectures 4,5

Further Reading

Sutton and Barto:

- Chapter 3: Sections 3.5 – 3.10