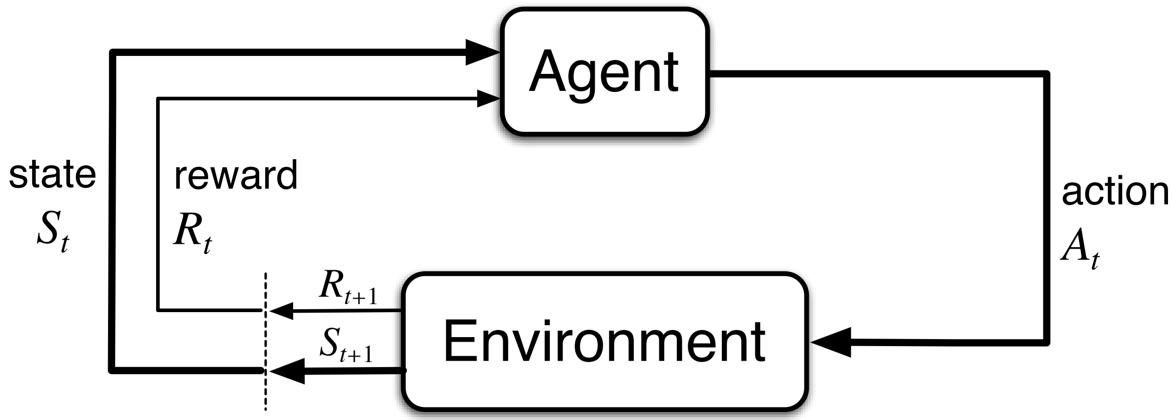
Lecture 2: Markov Decision Processes

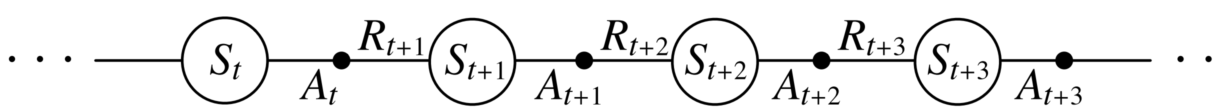
Kind acknowledgements: Dave Silver's and Rich Sutton's slide material

Outline

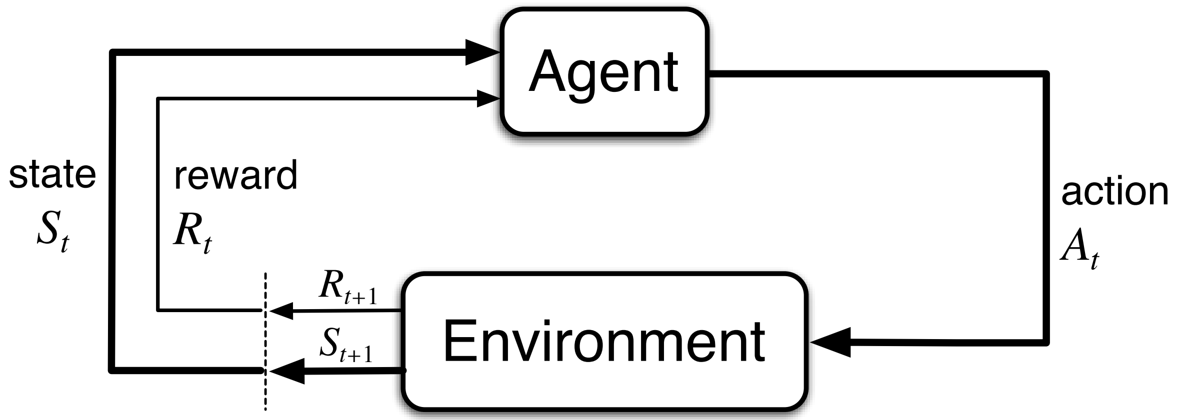
- Agents, Actions, Rewards
- Markov Decision Processes
- Value functions
- Optimal value functions

The Agent-Environment Interface



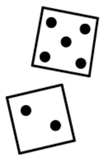


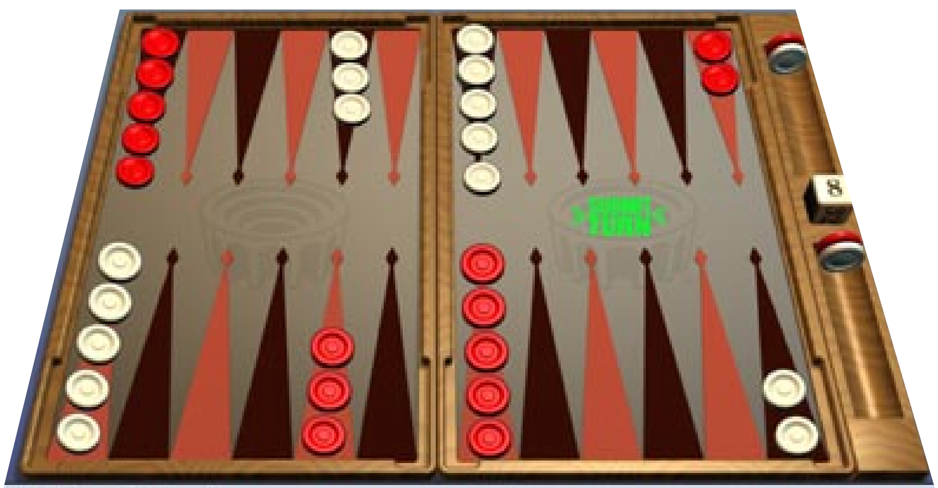
The Agent-Environment Interface



- Rewards specify what the agent needs to achieve, not how to achieve it.
- The simplest and cheapest form of supervision

Backgammon





States: configurations of the playing board

($\approx 10^{20}$)

Actions: moves

Rewards win: +1

lose: –1

else: 0

Visual Attention

States: road traffic, weather, time of day

Actions: Visual glimpses from mirrors/cameras/ front

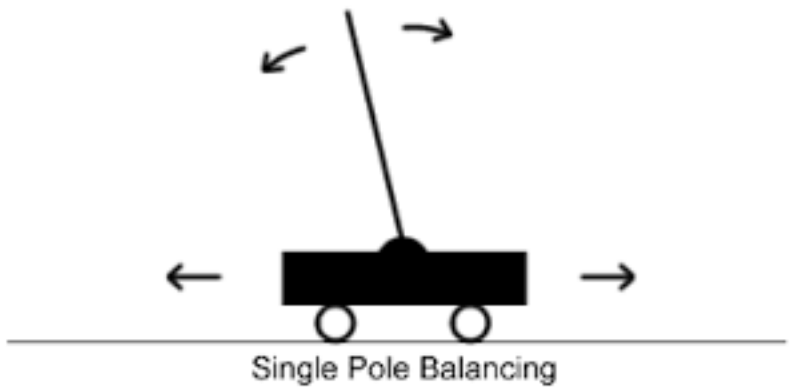
Rewards: +1 safe driving, not over-tired

-1: honking from surrounding drivers

conservative exploration strategy

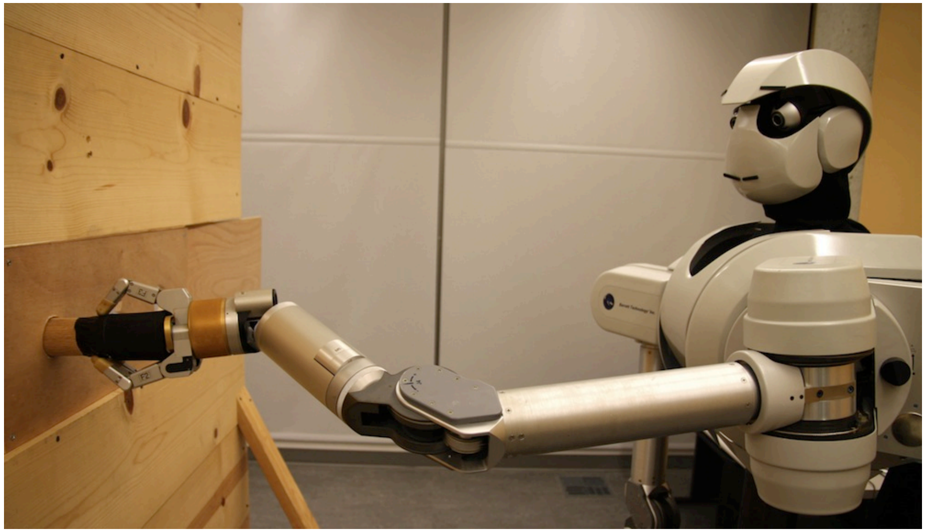
conservative exploration strategy

Cart pole



States: Pole angle and
angular velocity
Actions: move left right
Rewards: 0 while balancing,
-1 for imbalance

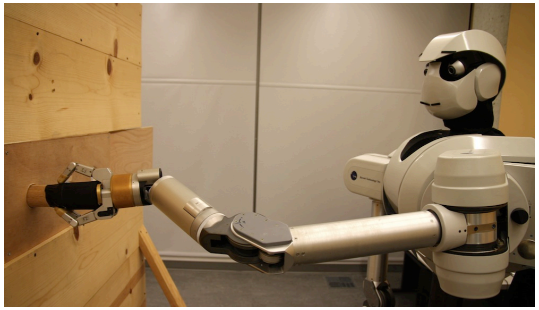
Peg in hole insertion task

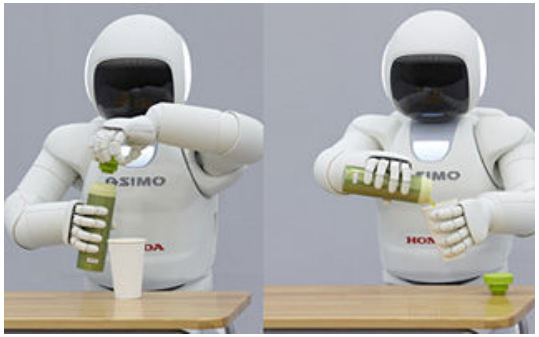


States: joint configurations (7DOF)
Actions: torques on joints
Rewards: penalize jerky motions, inversely proportional to distance from target pose

Detecting success

- The agent should be able to measure its success explicitly





We often times cannot automatically detect whether the task has been achieved!

- Can we think of goal directed behavior learning problems that are not meaningful within a trial-and-error agent-environment interaction based learning framework?

- Can we think of goal directed behavior learning problems that are not meaningful within a trial-and-error agent-environment interaction based learning framework?
- The agent should have the chance to try (and fail) enough times
- This is impossible if episode takes too long, e.g., reward="obtain a great Ph.D."
- This is impossible when safety is a concern: we can't learn to drive via reinforcement learning in the real world, failure cannot be tolerated

Markov Decision Processes

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 $\gamma \in [0, 1]$.

Actions

- For now we assume discrete actions.
- Actions can have many different temporal granularities.

States

- A state captures whatever information is available to the agent at step *t* about its environment. The state can include immediate "sensations," highly processed sensations, and structures built up over time from sequences of sensations, memories etc.
- A state should summarize past sensations so as to retain all "essential" information, i.e., it should have the **Markov Property**:

$$\mathbf{Pr}\{R_{t+1} = r, S_{t+1} = s' \mid S_0, A_0, R_1, \ldots, S_{t-1}, A_{t-1}, R_t, S_t, A_t\} = \mathbf{Pr}\{R_{t+1} = r, S_{t+1} = s' \mid S_t, A_t\}$$

for all $s' \in \mathcal{S}^+, r \in \mathcal{R},$ and all histories

- We should be able to throw away the history once state is known

States

A state captures whatever information is available to the agent at step *t* about its environment. The state can include immediate “sensations,” highly processed sensations, and structures built up over time from sequences of sensations, memories etc.

Example: what would you expect to be the state information of a self driving car?

An agent cannot be blamed for missing information that is unknown, but for forgetting relevant information.

States

A state captures whatever information is available to the agent at step t about its environment. The state can include immediate “sensations,” highly processed sensations, and structures built up over time from sequences of sensations, memories etc.

Example: how would you expect to be the state information of a vacuum-cleaner robot ?

Dynamics

- How the state changes given the actions of the agent
- Model based: dynamics are known or are estimated
- Model free: we do not know the dynamics of the MDP

Since in practice the dynamics are unknown, the state representation should be such that is easily predictable from neighboring states

Rewards

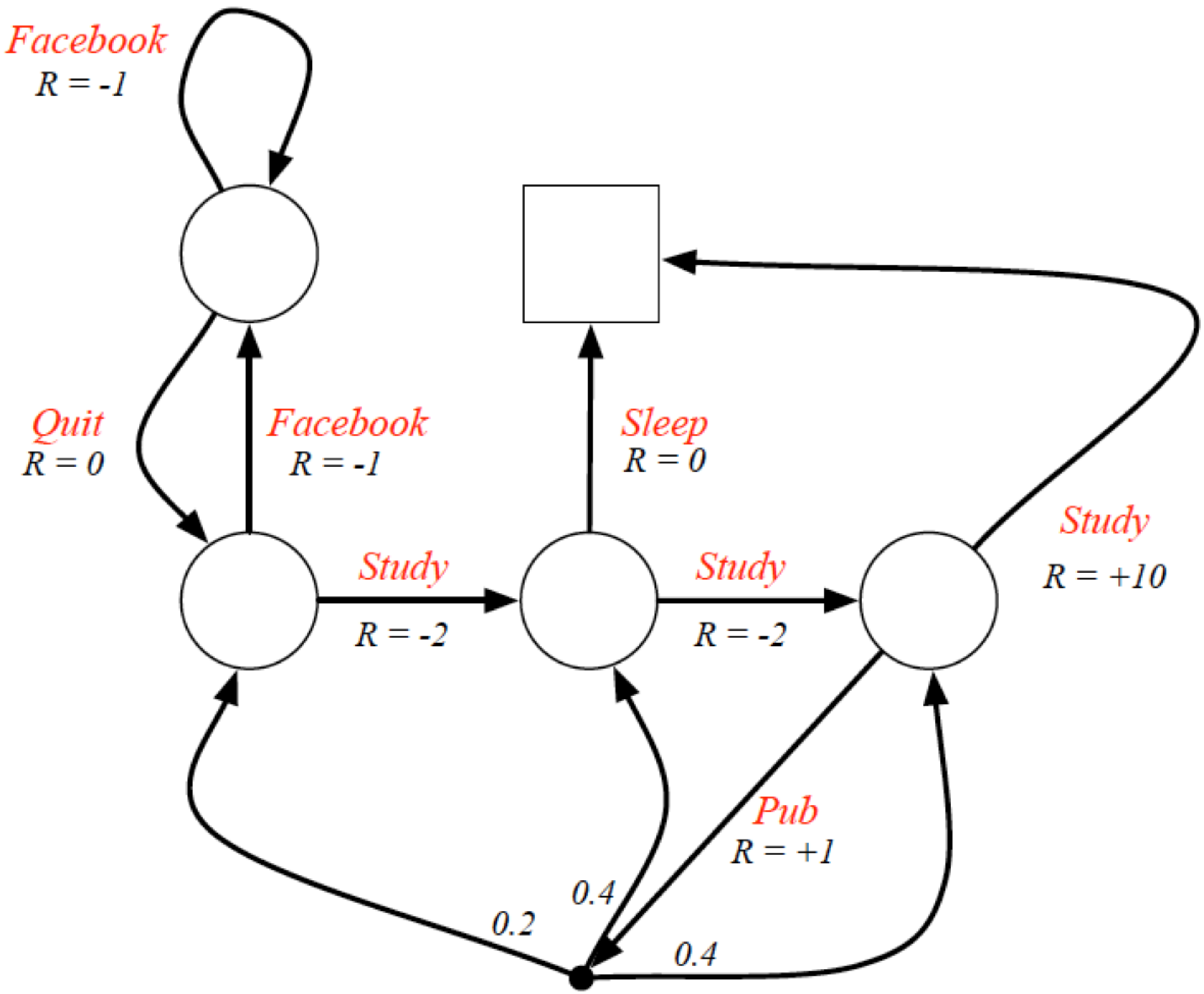
Definition

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 objective in RL is to maximize long-term future reward
- That is, to choose A_t so as to maximize $R_{t+1}, R_{t+2}, R_{t+3}, \ldots$
- Episodic tasks - finite horizon VS continuous tasks - infinite horizon
- In episodic tasks we can consider undiscounted future rewards

The student MDP



Agent learns a Policy

Definition

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

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

- 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), $A_t \sim \pi(\cdot|S_t), \forall t > 0$

Solving Markov Decision Processes

- Find the optimal policy
- Prediction: For a given policy, estimate value functions of states and states/action pairs
- Control: Estimate the value function of states and state/action pairs for the optimal policy.

Value functions

	state values	action values
prediction	v_π	q_π
control	v_*	q_*

- Value functions measure the goodness of a particular state or state/action pair: how good is for the agent to be in a particular state or execute a particular action at a particular state. Of coarse that depends on the policy!
- Optimal value functions measure the best possible goodness of states or state/action pairs under *any* policy.

Value functions are cumulative expected rewards

Definition

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

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

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

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

Optimal value functions are best achievable cumulative expected rewards

Definition

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

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

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

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

Value functions are cumulative expected rewards

Definition

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

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

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

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

Bellman Expectation Equation

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

The value function can be decomposed into two parts:

- immediate reward R_{t+1}
- discounted value of successor state $\gamma v(S_{t+1})$

Bellman Expectation Equation

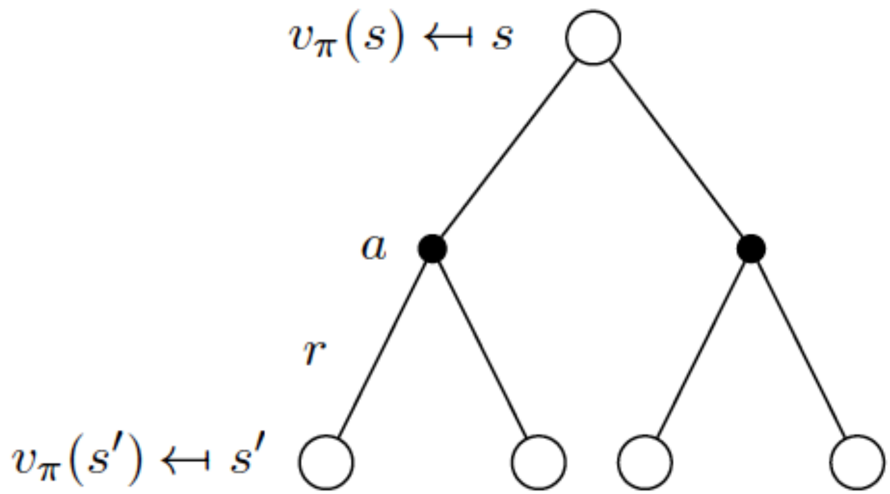
The state-value function can be decomposed into immediate reward plus discounted value of successor state,

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

The action-value function can similarly be decomposed,

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

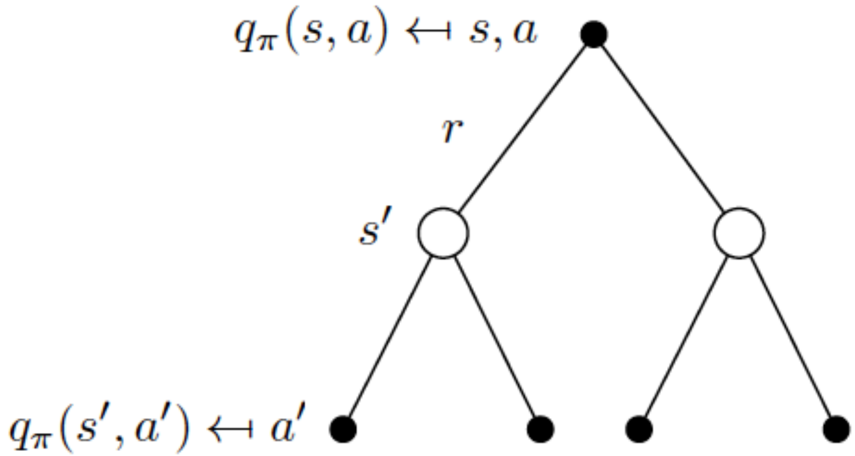
Looking inside the Expectations



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

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

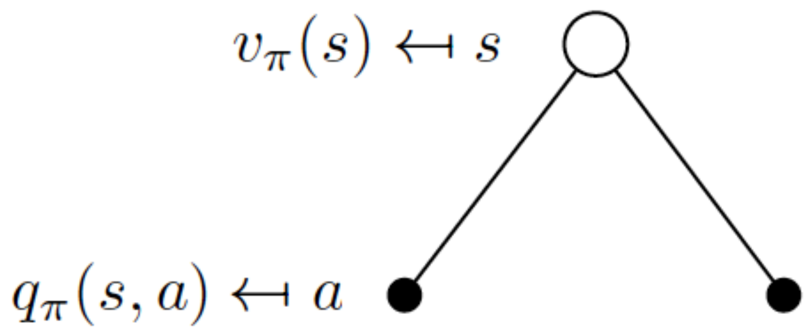
Looking inside the Expectations



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

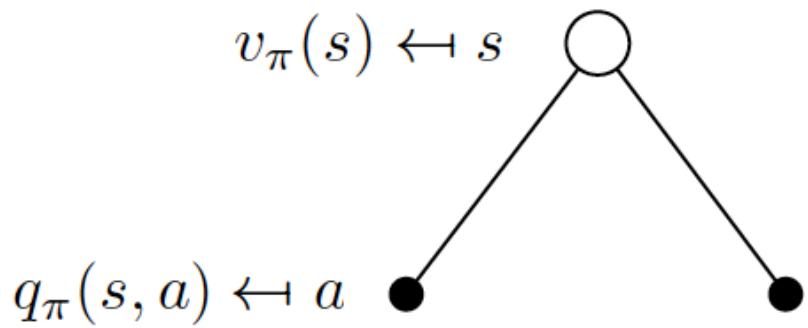
$$q_\pi(s,a) = \mathbb{E}_\pi \left[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a \right]$$

Relating state and action value functions



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

Relating state and action value functions

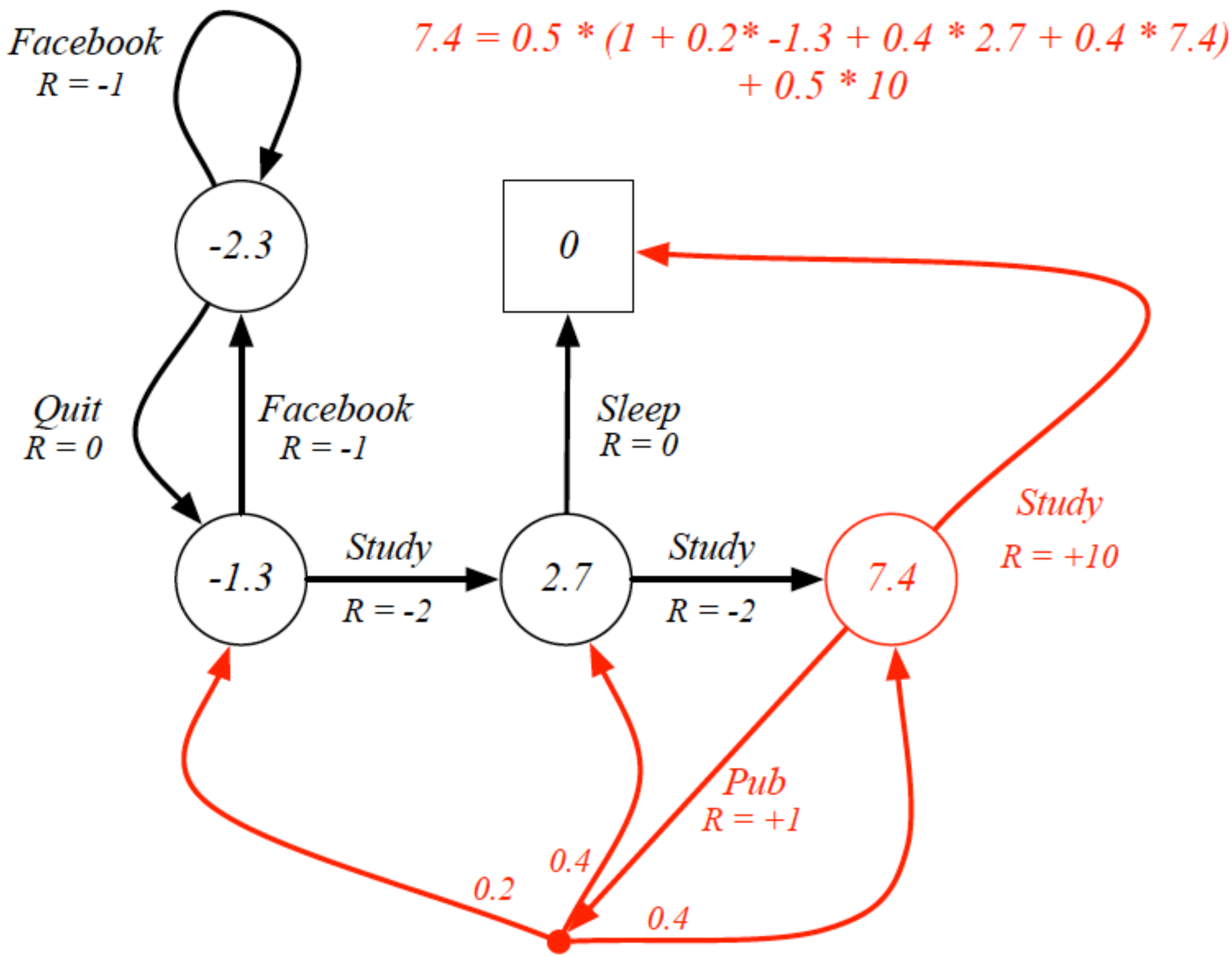


$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s,a)$$

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

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

Value function for the student MDP under the uniform policy



Linear system of Equations

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

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

Optimal value functions are best achievable cumulative expected rewards

Definition

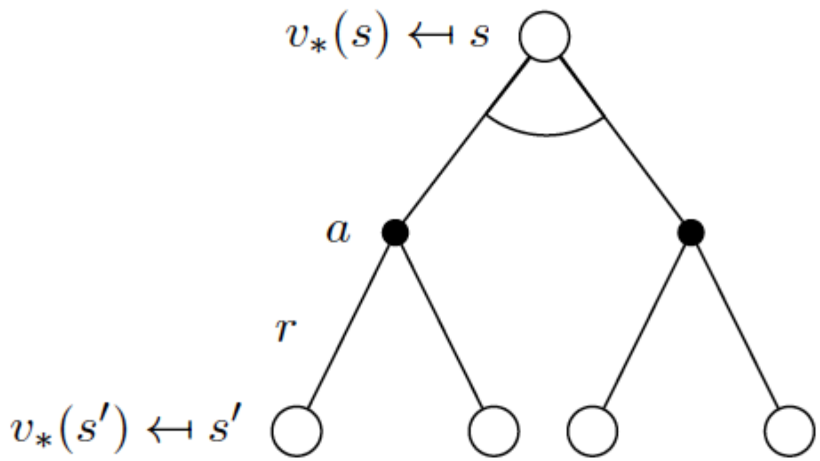
The *optimal state-value function* $v_*(s)$ is the maximum value function over all policies

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

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

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

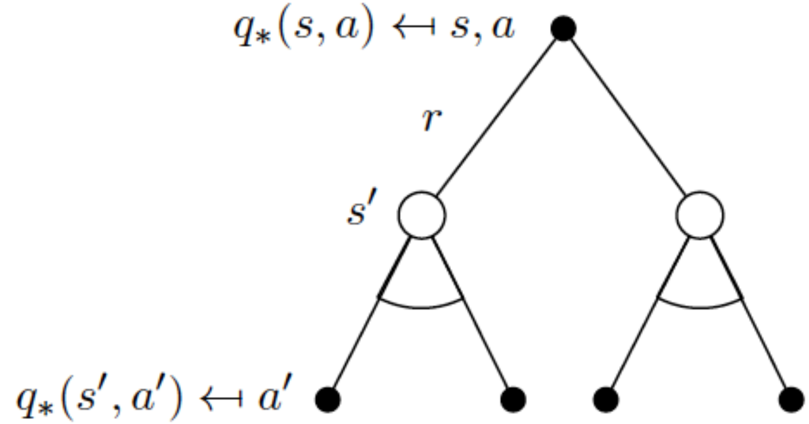
Bellman optimality equations for state value functions



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

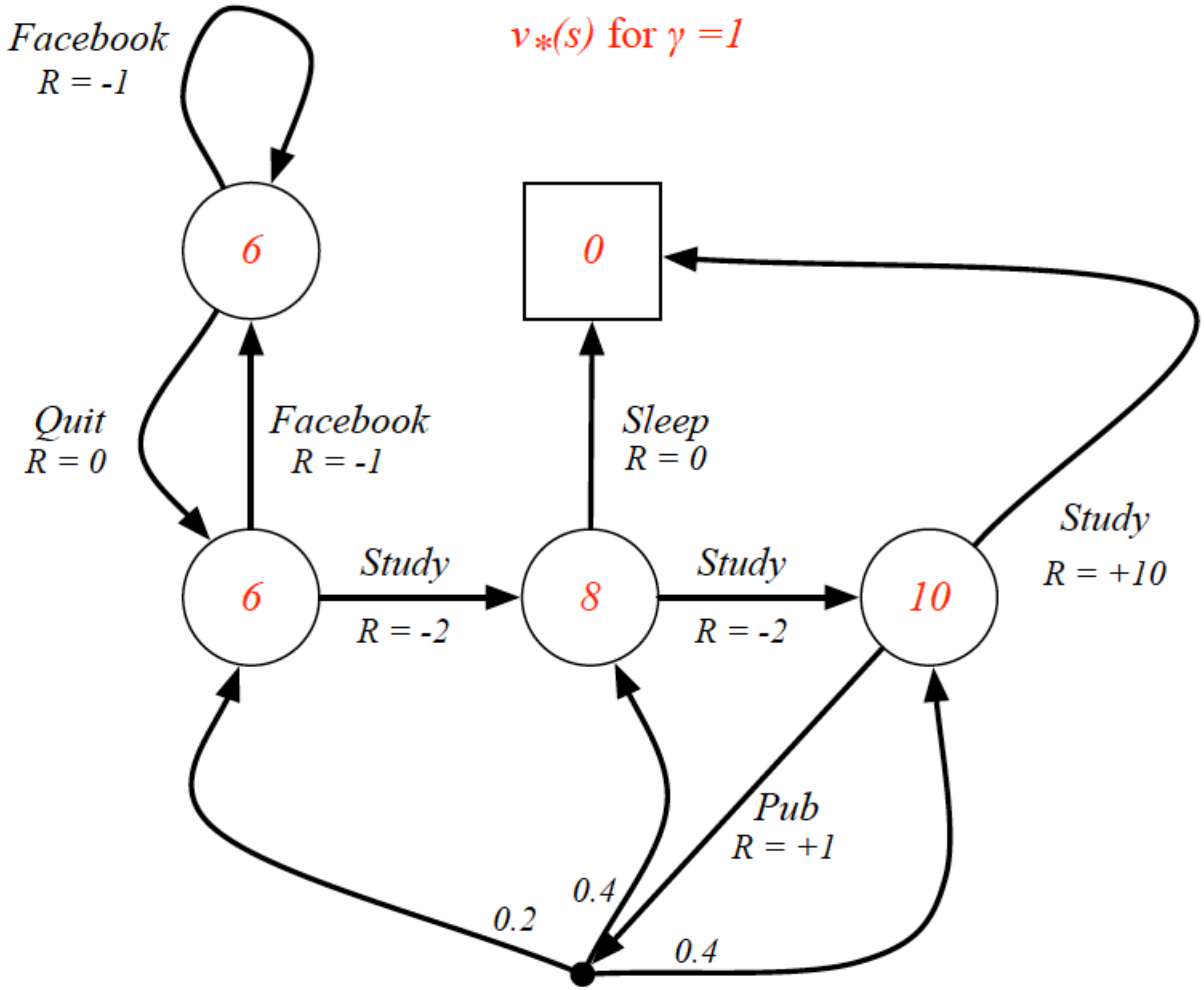
Principle of Optimality: An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision. (See Bellman, 1957, Chap. III.3).

Bellman optimality equations for action value functions

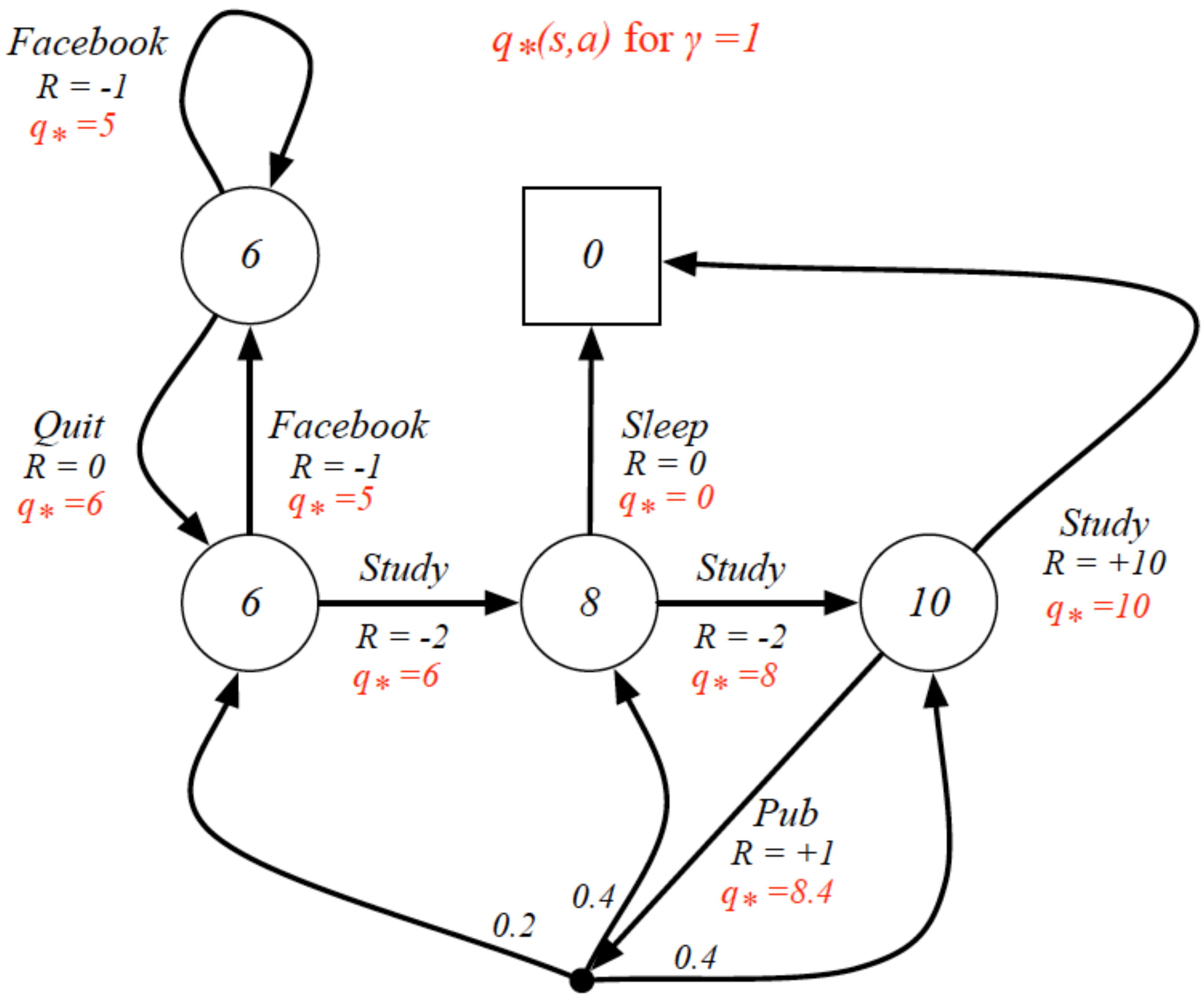


$$q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \max_{a'} q_*(s',a')$$

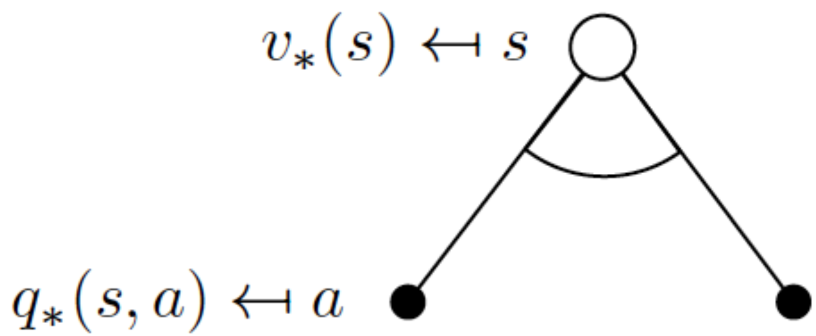
Optimal value function for the student MDP



Optimal state/action value function for the student MDP

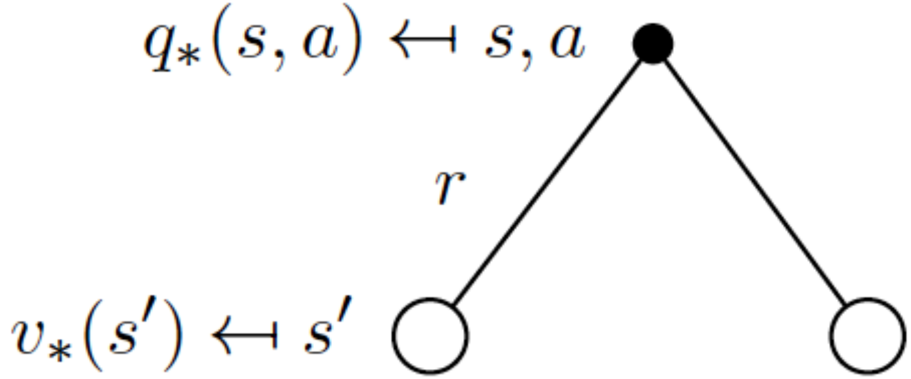


Relating optimal state and action value functions



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

Relating optimal state and action value functions



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

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

From optimal state value functions to optimal policies

- An optimal policy can be found from $v_*(s)$ and the model dynamics using one step look ahead, that is, acting greedily w.r.t. $v_*(s)$

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

From optimal action value functions to optimal policies

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

Solving the Bellman Optimality Equation

- Finding an optimal policy by solving the Bellman Optimality Equation requires the following:
  - accurate knowledge of environment dynamics;
  - we have enough space and time to do the computation;
  - the Markov Property.

Solving the Bellman Optimality Equation

- Finding an optimal policy by solving the Bellman Optimality Equation requires the following:
  - accurate knowledge of environment dynamics;
  - we have enough space and time to do the computation;
  - the Markov Property.
- How much space and time do we need?
  - polynomial in number of states (tabular methods)
  - BUT, number of states is often huge
  - So exhaustive sweeps of the state space are not possible
- We usually have to settle for approximations. Approximate dynamic programming has been introduced by D. P. Bertsekas and J. N. Tsitsiklis with the use of artificial neural networks for approximating the Bellman function. This is an effective mitigation strategy for reducing the impact of dimensionality by replacing the memorization of the complete function mapping for the whole space domain with the memorization of the sole neural network parameters.

Approximation and Reinforcement Learning

- RL methods: Approximating Bellman optimality equations
- Balancing reward accumulation and system identification (model learning) in case of unknown dynamics
- The on-line nature of reinforcement learning makes it possible to approximate optimal policies in ways that put more effort into learning to make good decisions for frequently encountered states, at the expense of less effort for infrequently encountered states. This is not the case, e.g., in control.

Summary

- Markov Decision Processes
- Value functions and Optimal Value functions
- Bellman Equations

So far finite MDPs with known dynamics