Lecture 2: Markov Decision Processes

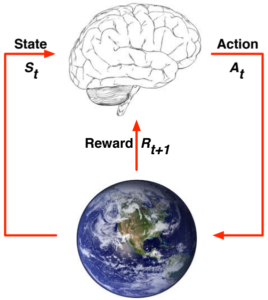
Joseph Modayil

Overview

- Last lecture: Basic ideas in reinforcement learning
- This lecture: A formal description of the RL problem
  - How is an RL problem formally specified?
  - How are the core ideas formally defined and connected?
  - What are the formal properties of the problems and solutions?
- Next lectures: Finding solutions to RL problems

Formalizing the RL interface

The conventional RL interface is an interface between a learning agent and an environment. The agent-environment interface may not coincide with a physical boundary. A learning agent may not know any other details about the environment to which it is connected (e.g. if it is an Atari game or the physical world).



We want to define learning objectives using only the signals sent across the interface.

Introduction to MDPs

- *Markov decision processes* formally describe an environment for reinforcement learning
- Conventionally where the environment is *fully observable*
- i.e. The current *state* completely characterizes the process
- Almost all RL problems can be formalized as MDPs, e.g.
  - Optimal control primarily deals with continuous MDPs
  - Partially observable problems can be converted into MDPs
  - Bandits are MDPs with one state
- References: Sutton & Barto Chapter 3.
  Slides: Moodle

Statistical notation in Reinforcement Learning

- A variety of statistical notation has been used in earlier RL textbooks and earlier versions of this course.
- We will move towards more standard notation.
- Lowercase letters are used for states s and functions f.
- Uppercase letters are used for random variables, S_t, and these are often indexed by time.
- Be aware of that various sources differ in their notation (so v^π in the slides is v_π in the book).
- An expectation $\mathbb{E}\left[G_t\right]$ is often made over future trajectories.
- A probability distribution is denoted by $\mathbb{P}\left[\right]$.

Markov Property

"The future is independent of the past given the present"
Consider a sequence of random states, $\{S_t\}_{t \in \mathbb{N}}$, indexed by time.

Definition

A random state S_t has the *Markov* property if and only if $\forall s, s' \in \mathcal{S}$

$$\mathbb{P}\left[S_{t+1} = s' \mid S_t = s\right] = \mathbb{P}\left[S_{t+1} = s' \mid S_1, \ldots, S_t = s\right]$$

for all histories (all instantiations of S_k for $k < t$).

- 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

State Transition Matrix

For random states with the Markov property, 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',

$$\mathcal{P} \;=\; \begin{array}{cc} & \textit{to} \\ \textit{from} & \begin{bmatrix} \mathcal{P}_{11} & \ldots & \mathcal{P}_{1n} \\ \vdots & & \\ \mathcal{P}_{11} & \ldots & \mathcal{P}_{nn} \end{bmatrix} \end{array}$$

where each row of the matrix sums to 1.

Markov Process

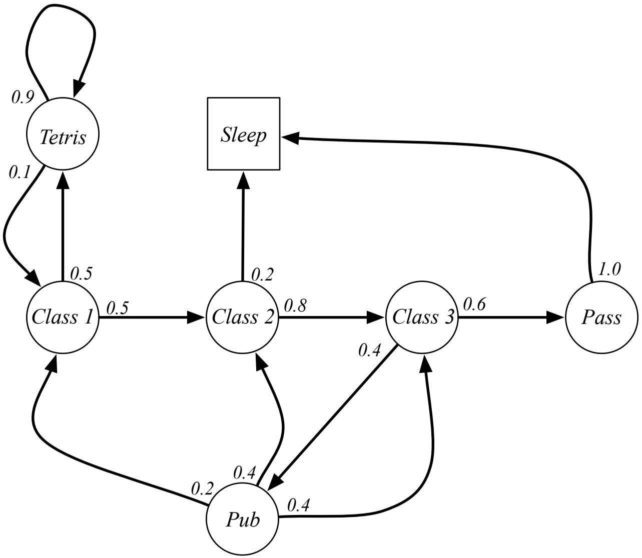
A Markov process is a memoryless random process, i.e. a sequence of random states $S_1, S_2, ...$ with the Markov property.

Definition

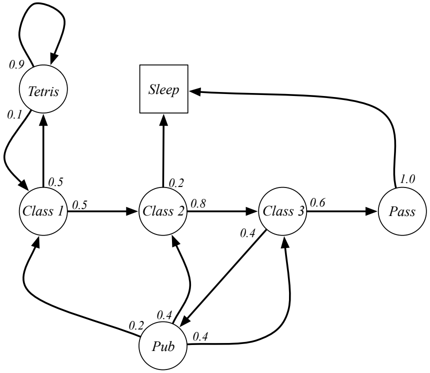
A *Markov Process* (or *Markov Chain*) is a tuple $\langle \mathcal{S}, \mathcal{P} \rangle$

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

Example: Student Markov Chain



Example: Student Markov Chain Episodes

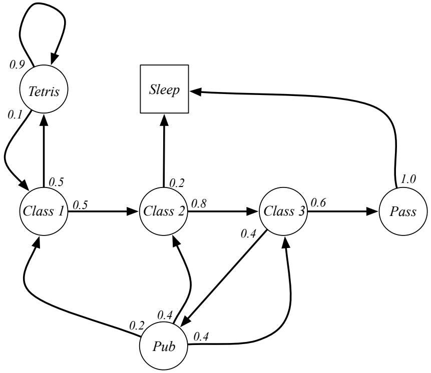


Sample episodes for Student Markov Chain starting from $S_1 = \text{C1}$

$$S_1, S_2, ..., S_T$$

- C1 C2 C3 Pass Sleep
- C1 TT TT C1 C2 Sleep
- C1 C2 C3 Pub C2 C3 Pass Sleep
- C1 TT TT C1 C2 C3 Pub C1 TT TT TT C1 C2 C3 Pub C2 Sleep

Example: Student Markov Chain Transition Matrix



$$\mathcal{P} = \begin{array}{c} \\ C1 \\ C2 \\ C3 \\ Pass \\ Pub \\ Tetris \\ Sleep \end{array} \begin{array}{c} \begin{array}{ccccccc} C1 & C2 & C3 & Pass & Pub & Tetris & Sleep \end{array} \\ \left[\begin{array}{ccccccc} & 0.5 & & & & 0.5 & \\ & & 0.8 & & & & 0.2 \\ & & & 0.6 & 0.4 & & \\ & & & & & & 1.0 \\ 0.2 & 0.4 & 0.4 & & & & \\ 0.1 & & & & & 0.9 & \\ & & & & & & 1 \end{array}\right] \end{array}$$

Markov Reward Process

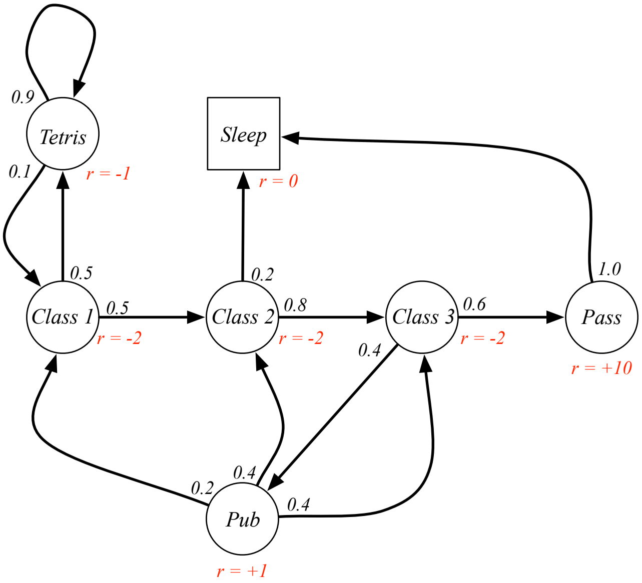
A Markov reward process is a Markov chain with values.

Definition

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

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

Example: Student MRP



Return

Definition

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

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

- 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?

Most Markov reward and decision processes are discounted. Why?

- Mathematically convenient to discount rewards
- Avoids infinite returns in cyclic Markov processes
- 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
- It sometimes possible to use *undiscounted* Markov reward processes (i.e. $\gamma = 1$), e.g. if all sequences terminate.

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

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

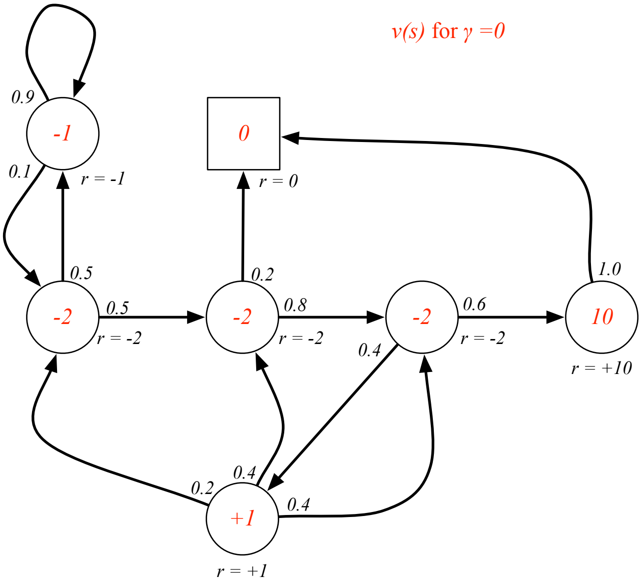
Example: Student Markov Chain Returns

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

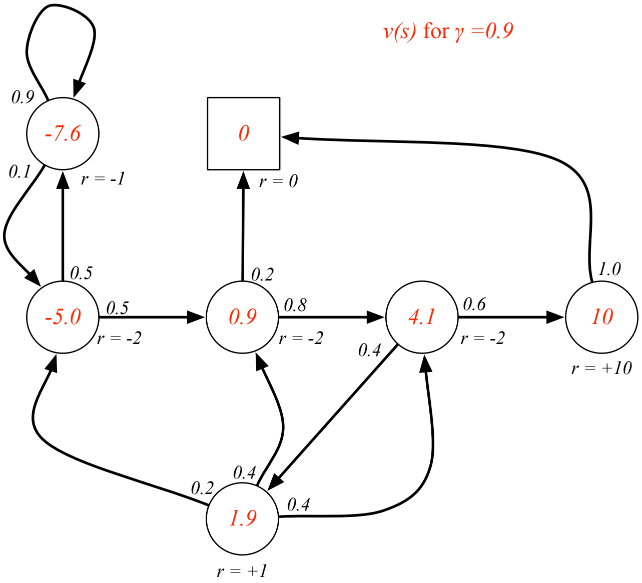
$$G_1 = R_1 + \gamma R_2 + ... + \gamma^{T-1} R_T$$

C1 C2 C3 Pass Sleep	$G_1 = -2 - 2 * \frac{1}{2} - 2 * \frac{1}{4} + 10 * \frac{1}{8}$	$=$	-2.25
C1 TT TT C1 C2 Sleep	$G_1 = -2 - 1 * \frac{1}{2} - 1 * \frac{1}{4} - 2 * \frac{1}{8} - 2 * \frac{1}{16}$	$=$	-3.125
C1 C2 C3 Pub C2 C3 Pass Sleep	$G_1 = -2 - 2 * \frac{1}{2} - 2 * \frac{1}{4} + 1 * \frac{1}{8} - 2 * \frac{1}{16} \dots$	$=$	-3.41
C1 TT TT C1 C2 C3 Pub C1 ...	$G_1 = -2 - 1 * \frac{1}{2} - 1 * \frac{1}{4} - 2 * \frac{1}{8} - 2 * \frac{1}{16} \dots$	$=$	-3.20
TT TT TT C1 C2 C3 Pub C2 Sleep			

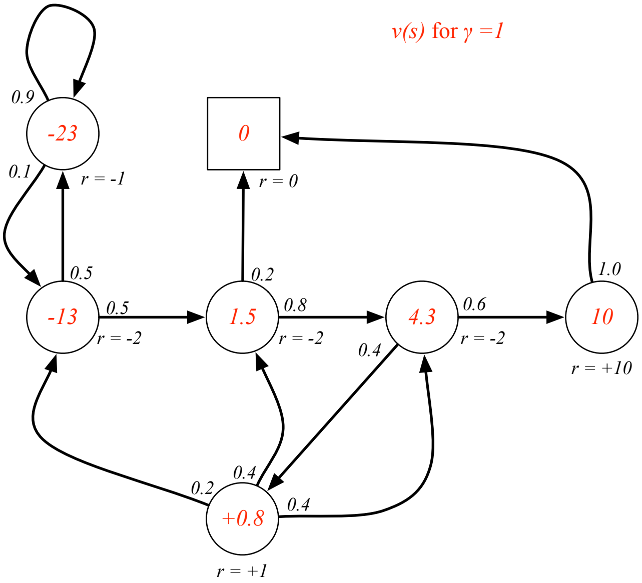
Example: State-Value Function for Student MRP (1)



Example: State-Value Function for Student MRP (2)



Example: State-Value Function for Student MRP (3)



Bellman Equation for MRPs

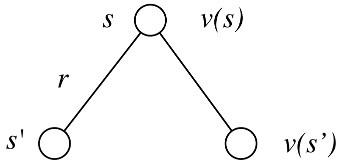
The value function can be decomposed into two parts:

- immediate reward r
- discounted value of successor state $\gamma v(s')$

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

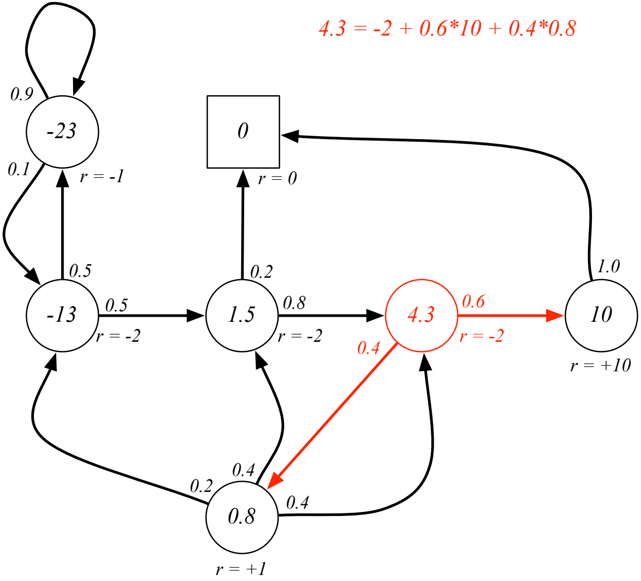
Bellman Equation for MRPs (2)

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



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

Example: Bellman Equation for Student MRP



Bellman Equation in Matrix Form

The Bellman equation can be expressed concisely using matrices,

$$v = \mathcal{R} + \gamma \mathcal{P} v$$

where v is a column vector with one entry per state

$$\begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix} = \begin{bmatrix} \mathcal{R}_1 \\ \vdots \\ \mathcal{R}_n \end{bmatrix} + \gamma \begin{bmatrix} \mathcal{P}_{11} & \dots & \mathcal{P}_{1n} \\ \vdots & & \\ \mathcal{P}_{n1} & \dots & \mathcal{P}_{nn} \end{bmatrix} \begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix}$$

Solving the Bellman Equation

- The Bellman equation is a linear equation
- It can be solved directly:

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

- Computational complexity is $O(n^3)$ for n states
- Direct solution only possible for small MRPs
- There are many iterative methods for large MRPs, e.g.
  - Dynamic programming
  - Monte-Carlo evaluation
  - Temporal-Difference learning

Markov Decision Process

A Markov decision process (MDP) is a Markov reward process with decisions. It is an *environment* in which all random 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, $\mathcal{P}^{a}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s, A_t = a\right]$
- $\mathcal{R}$ is a reward function, $\mathcal{R}^{a}_{s} = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$
- γ is a discount factor $\gamma \in [0, 1]$.

Markov Decision Process: New notation

A more precise way to describe the state and reward dynamics of a MDP is to give the probability for each possible next state and reward.

$$p(s', r \mid s, a) \equiv \mathbb{P}\left[S_{t+1} = s', R_{t+1} = r | S_t = s, A_t = a\right]$$

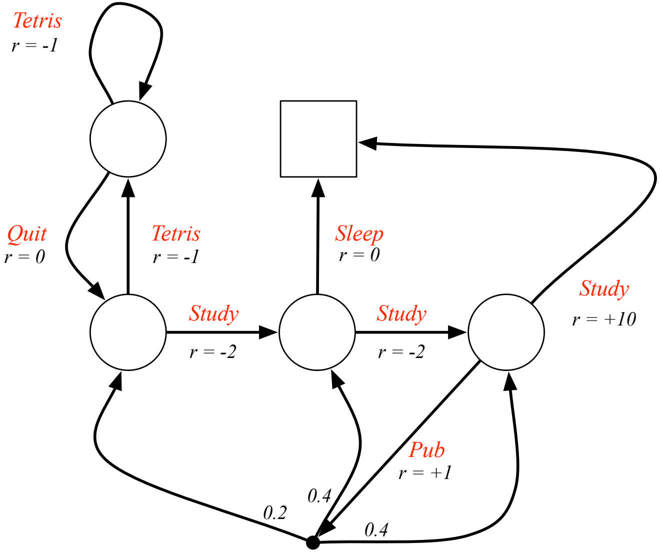
We can then write the state transistion probabilities as

$$\mathcal{P}_{ss'}^{a} \equiv \mathbb{P}\left[S_{t+1} = s' | S_t = s, A_t = a\right] = \sum_{r \in \mathcal{R}} p(s', r | s, a).$$

We can then write the state transistion probabilities as

$$\mathcal{R}_{s}^{a} \equiv \mathbb{E}\left[R_{t+1} = r | S_t = s, A_t = a\right] = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r | s, a).$$

Example: Student MDP



Break and Thought Excercises

- How would you formalise the game of rock, paper, scissors? Tetris?
- Give an MDP abstraction of another problem.

Policies (1)

Definition

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

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

- 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), $A_t \sim \pi(S_t, \cdot), \forall t > 0$
- The policy $\pi(s, a)$ is sometimes written as $\pi(a|s)$

Policies (2)

- Given an MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ and a policy π
- The state sequence $S_1, S_2, ...$ is a Markov process $\langle \mathcal{S}, \mathcal{P}^\pi \rangle$
- The state and reward sequence $S_1, R_2, S_2, ...$ is a Markov reward process $\langle \mathcal{S}, \mathcal{P}^\pi, \mathcal{R}^\pi, \gamma \rangle$
- where

$$\mathcal{P}^\pi_{s,s'} = \sum_{a \in \mathcal{A}} \pi(s,a) \mathcal{P}^a_{s,s'}$$
$$\mathcal{R}^\pi_s = \sum_{a \in \mathcal{A}} \pi(s,a) \mathcal{R}^a_s$$

Policy Conditional Expectations

Definition

The expectation of a random variable F_t conditional on π being used to select future actions is written as $\mathbb{E}_\pi [F_t]$.

Value Function

Definition

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

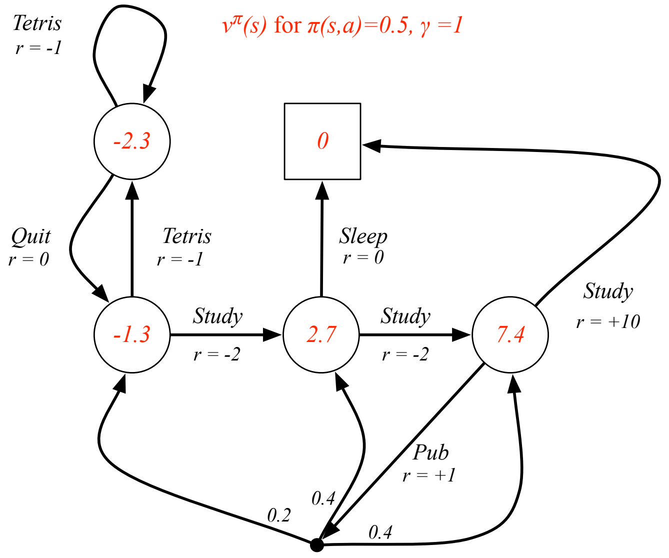
$$v^\pi(s) = \mathbb{E}_\pi \left[G_t \mid S_t = s \right]$$

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

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

Example: State-Value Function for Student MDP



Bellman Expectation Equation

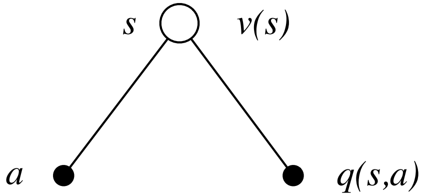
The state-value function can again be decomposed into immediate reward plus discounted value of successor state,

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

The action-value function can similarly be decomposed,

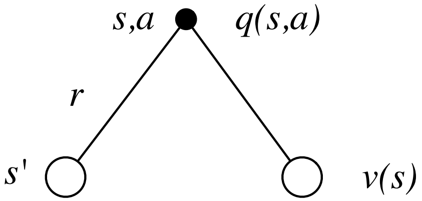
$$q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[R_{t+1} + \gamma q^{\pi}(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a\right]$$

Bellman Expectation Equation for v^π



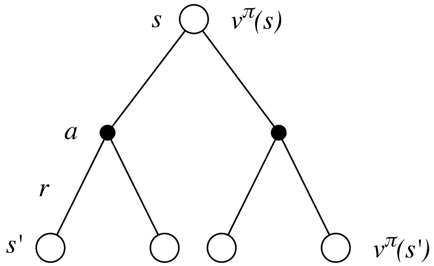
$$v^\pi(s) = \sum_{a \in \mathcal{A}} \pi(s,a) q^\pi(s,a)$$

Bellman Expectation Equation for q^π



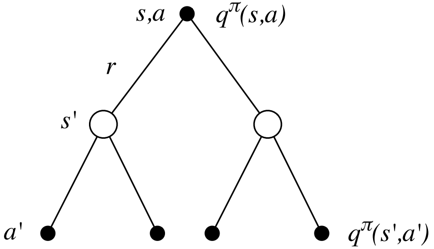
$$q^\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v^\pi(s')$$

Bellman Expectation Equation for v^π (2)



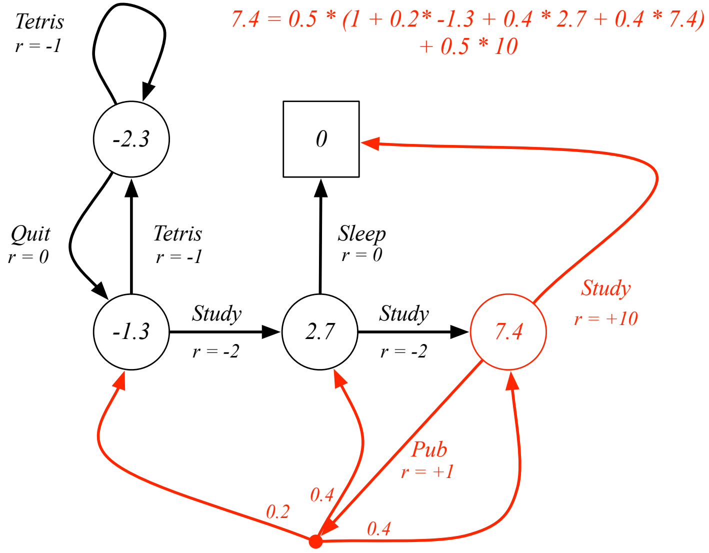
$$v^\pi(s) = \sum_{a \in \mathcal{A}} \pi(s,a) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v^\pi(s') \right)$$

Bellman Expectation Equation for q^π (2)



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

Example: Bellman Expectation Equation in Student MDP



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

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

Optimal Value Function

Definition

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

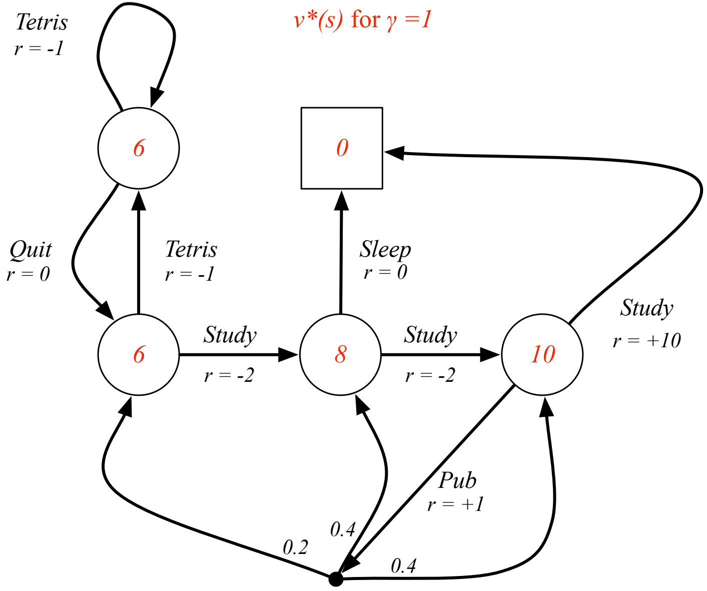
$$v^*(s) = \max_\pi v^\pi(s)$$

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

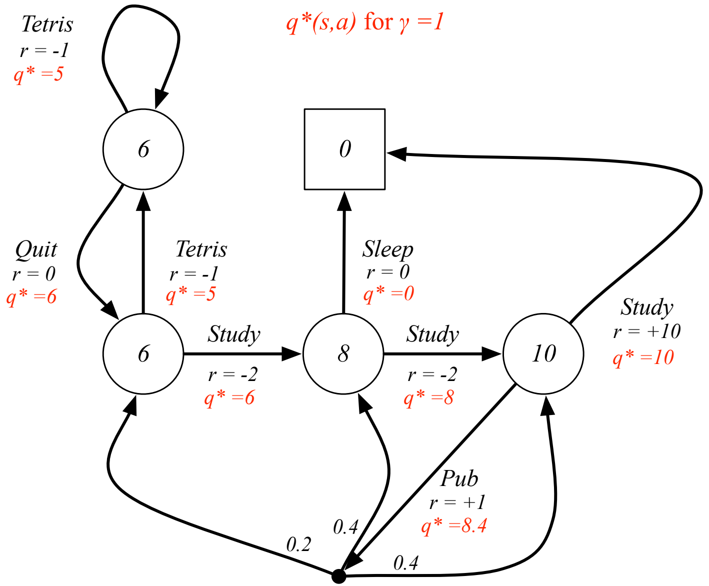
$$q^*(s, a) = \max_\pi q^\pi(s, a)$$

- The optimal value function specifies the best possible performance in the MDP.
- An MDP is "solved" when we know the optimal value fn.

Example: Optimal Value Function for Student MDP



Example: Optimal Action-Value Function for Student MDP



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, $q^{\pi^*}(s, a) = q^*(s, a)$*

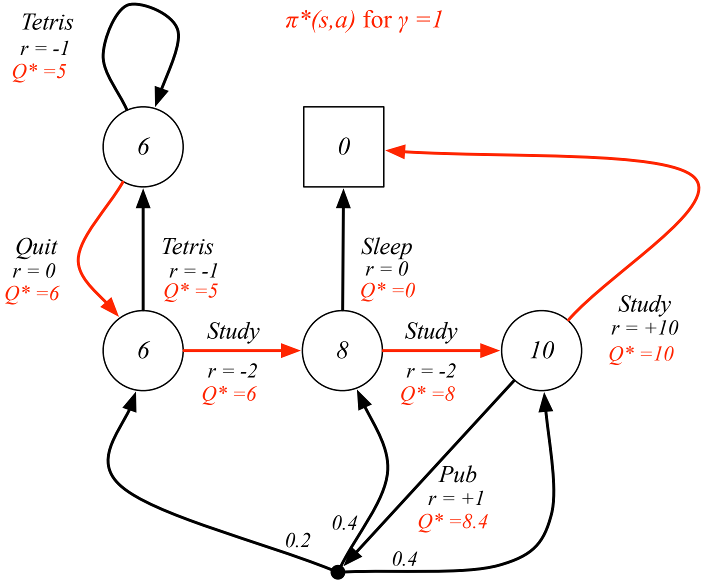
Finding an Optimal Policy

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

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

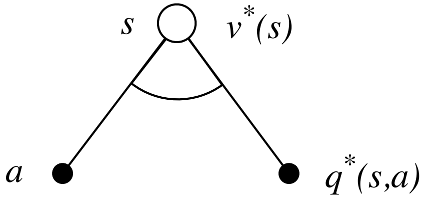
- There is always a deterministic optimal policy for any MDP
- If we know $q^*(s, a)$, we immediately have the optimal policy
- There can be multiple optimal policies

Example: Optimal Policy for Student MDP



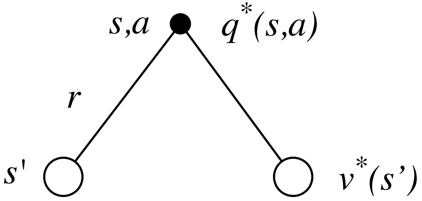
Bellman Optimality Equation for v^*

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



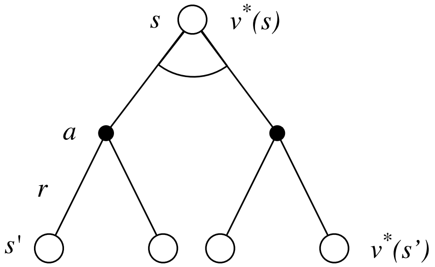
$$v^*(s) = \max_a q^*(s, a)$$

Bellman Optimality Equation for q^*



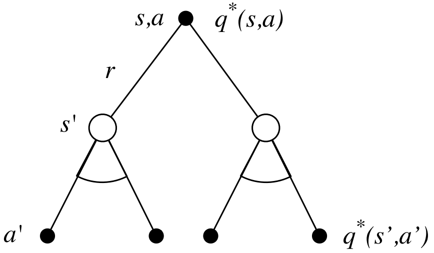
$$q^*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v^*(s')$$

Bellman Optimality Equation for v^* (2)



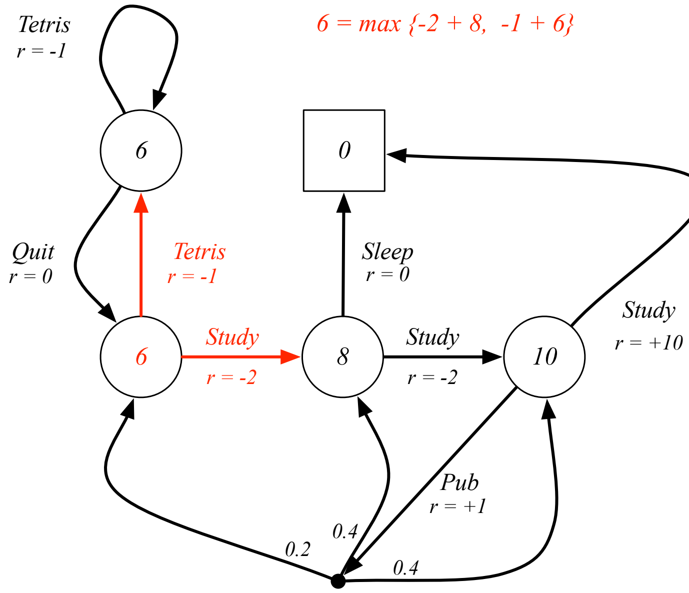
$$v^*(s) = \max_a \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v^*(s')$$

Bellman Optimality Equation for q^* (2)



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

Example: Bellman Optimality Equation in Student MDP



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
    - Q-learning
    - Sarsa

Extensions to MDPs

- Infinite and continuous MDPs
- Partially observable MDPs
- Undiscounted, average reward MDPs
- Function approximation

Infinite MDPs

The following extensions are all possible:

- Countably infinite state and/or action spaces
  - Straightforward
- Continuous state and/or action spaces
  - Closed form for linear quadratic model (LQR)
- Continuous time
  - Requires partial differential equations
  - Hamilton-Jacobi-Bellman (HJB) equation
  - Limiting case of Bellman equation as time-step $\rightarrow 0$

POMDPs

A POMDP is an MDP with hidden states. It is a hidden Markov model with actions.

Definition

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

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

Belief States

Definition

A *history* H_t is a sequence of actions, rewards, and observations

$$H_t = A_1, R_2, O_2, ..., A_{t-1}, R_t, O_t$$

Definition

A *belief state* $b(H_t)$ is a probability distribution over states, conditioned on the history h_t

$$b(H_t) = \left(\mathbb{P}\left[S_t = s^1 \mid H_t\right], ..., \mathbb{P}\left[S_t = s^n \mid H_t\right]\right)$$

- The history H_t satisfies the Markov property
- The belief state $b(H_t)$ satisfies the Markov property

Ergodic Markov Process

An ergodic Markov process is

- *Recurrent*: each state is visited an infinite number of times
- *Aperiodic*: each state is visited without any systematic period

Theorem

An ergodic Markov process has a limiting stationary distribution $d^{\pi}(s)$ with the property

$$d^{\pi}(s) = \sum_{s' \in \mathcal{S}} \mathcal{P}^{k}_{ss'} d^{\pi}(s')$$

Ergodic MDP

Definition

An MDP is ergodic if for every policy π the Markov chain induced by π is ergodic.

For any policy π, an ergodic MDP has an *average reward per time-step* ρ^π that is independent of start state.

$$\rho^\pi = \lim_{T \to \infty} \frac{1}{T} \mathbb{E}_\pi \left[\sum_{t=1}^{T} R_t \right]$$

Average Reward Value Function

- The value function of an undiscounted, ergodic MDP can be expressed in terms of average reward.
- $\tilde{v}^{\pi}(s)$ is the extra reward due to starting from state s,

$$\tilde{v}^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{k=1}^{\infty}(R_{t+k} - \rho^{\pi}) \mid S_t = s\right]$$

There is a corresponding average reward Bellman equation,

$$\begin{aligned}\tilde{v}^{\pi}(s) &= \mathbb{E}_{\pi}\left[(R_{t+1} - \rho^{\pi}) + \sum_{k=1}^{\infty}(R_{t+k+1} - \rho^{\pi}) \mid S_t = s\right] \\ &= \mathbb{E}_{\pi}\left[(R_{t+1} - \rho^{\pi}) + \tilde{v}^{\pi}(S_{t+1}) \mid S_t = s\right]\end{aligned}$$

MDPs with function approximation

- What happens when the state, actions, values, or policy can not be represented exactly? The agent approximates them!
- There are many approaches to function approximation.
  e.g. state aggregation, linear functions of features, neural nets
- Function approximation solution methods are needed to handle large MDPs (and extensions).
  e.g. Use a recurrent net to approximate an agent's history.
- Function approximation solution methods often have weaker theoretical guarantees.

Summary

- Markov processes, MRPs, MDPs
- States, Actions, Transitions
- Rewards, Returns, Values, Policies
- Bellman equations, Optimality

Questions?

The only stupid question is the one you were afraid to ask but never did.
-Rich Sutton

For questions that arise outside of class, please use the Moodle.