Lecture 2: Markov Decision Processes

Markov Processes

Introduction to MDPs

- Markov decision processes formally describe an environment for reinforcement learning
- Where the environment is fully observable
- i.e. The current state completely characterises the process
- Almost all RL problems can be formalised 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

Markov Property

A state S_t is Markov if and only if

$$P[S_{t+1}|S_t] = P[S_{t+1}|S_1, \ldots, S_t]$$

- 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

State Transition Matrix

For a Markov state s and successor state s 0 , the state transition probability is defined by

$$P_{ss'} = P[S_{t+1} = s' | S_t = s]$$

State transition matrix P defines transition probabilities from all state s to all successor state s'

$$P = \begin{bmatrix} P_{11} & \cdots & P_{1n} \\ \vdots & \ddots & P_{2n} \\ P_{n1} & \cdots & P_{nn} \end{bmatrix}$$

where each row of the matrix sums to 1

Markov Chains

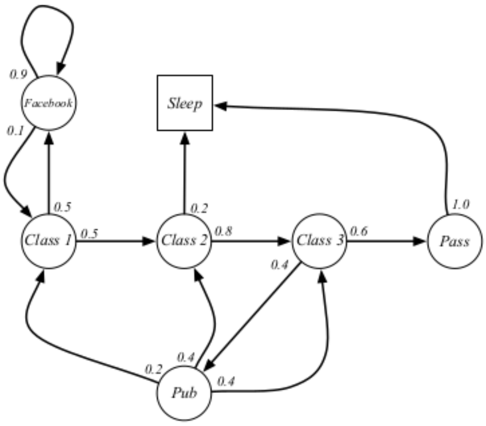
Markov Process

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

A Markov Process (or Markov Chain) is a tuple $< S, P >$

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

Example: Student Markov Chain Episodes

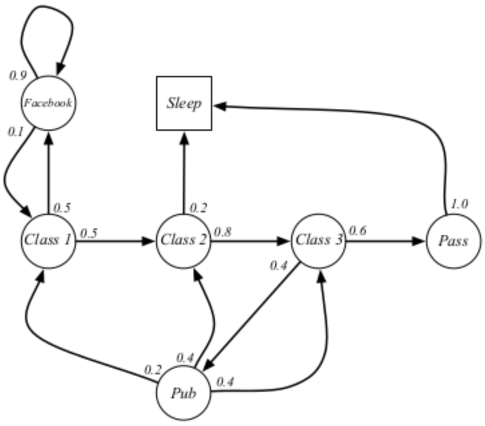


Sample episodes for Student Markov Chain starting from $S_1 = C1$

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

- C1 C2 C3 Pass Sleep
- C1 FB FB C1 C2 Sleep
- C1 C2 C3 Pub C2 C3 Pass Sleep
- C1 FB FB C1 C2 C3 Pub C1 FB FB FB C1 C2 C3 Pub C2 Sleep

Example: Student Markov Chain Transition Matrix



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

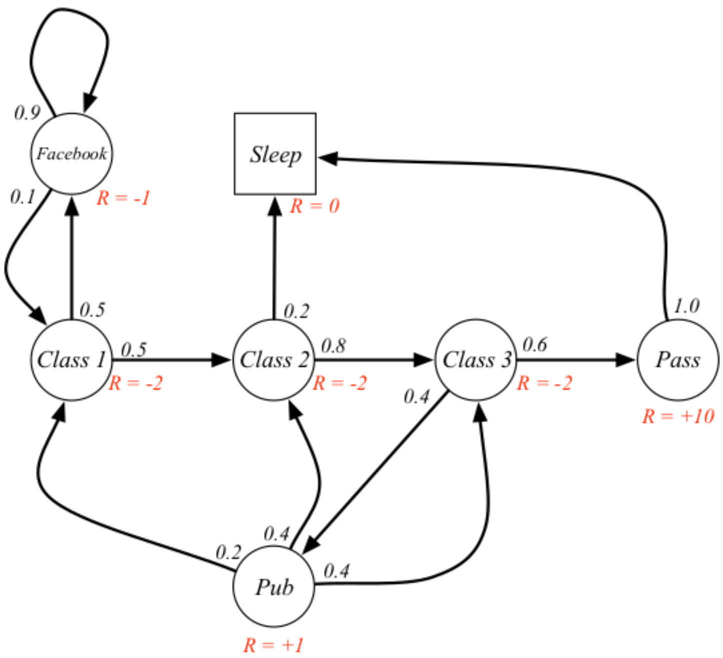
Markov Reward Processes

A Markov reward process is a Markov chain with values.

A **Markov Reward Process** is a tuple $< S, P, R, \gamma >$

- S is a finite set of states
- P is a state transition probability matrix, $P_{ss'} = P[S_{t+1} = s' | S_t = s]$
- R is a reward function, $R_s = E[R_{t+1} | S_t = s]$
- γ is a discount factor, $\gamma \in [0, 1]$

Example: Student MRP



Return

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

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

- 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 is 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

The state value function $v(s)$ of an MRP is the expected return starting from state s

$$v(s) = E[G_t | S = s]$$

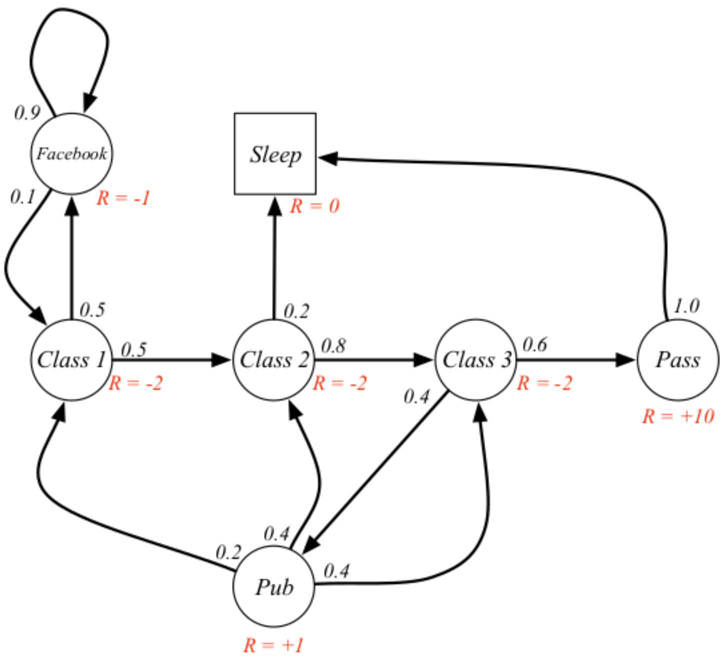
Example: Student MRP Returns

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

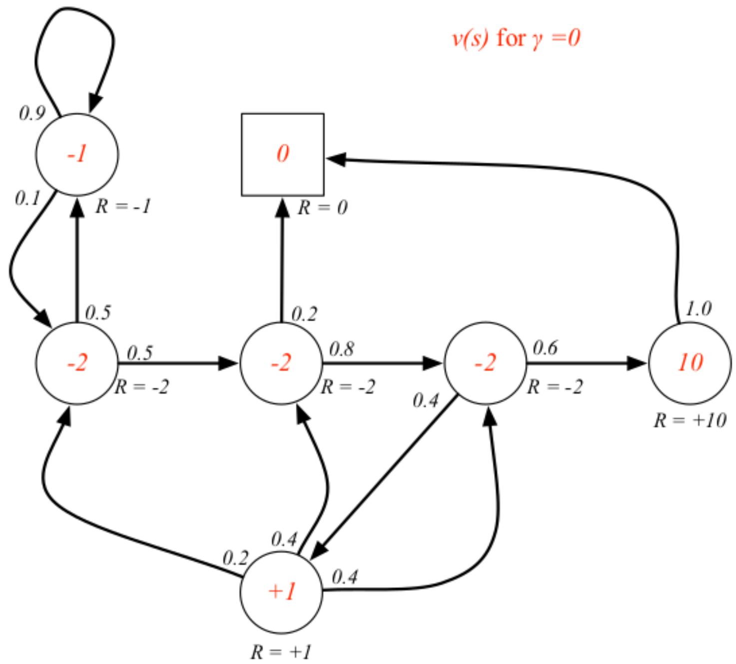
$$G_1 = R_2 + \gamma R_3 + ... + \gamma^{T-2} R_T$$

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

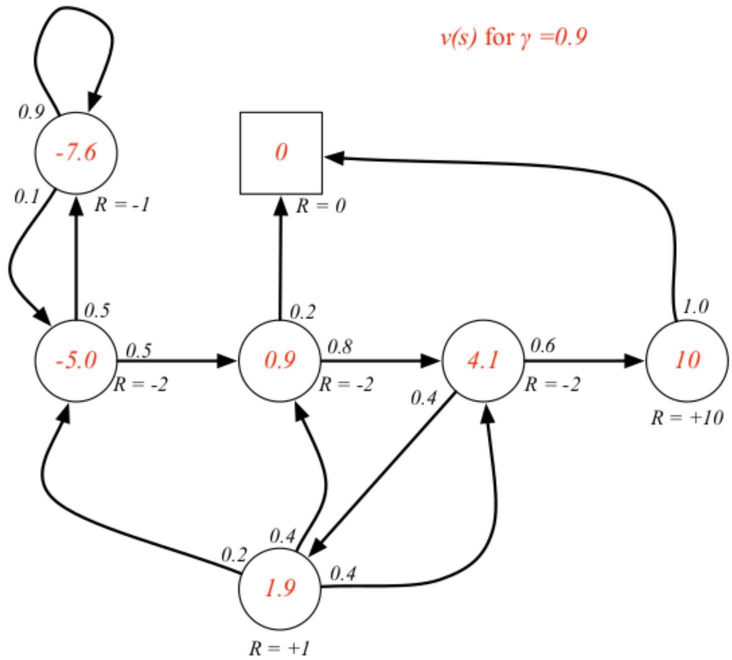
Example: Student MRP



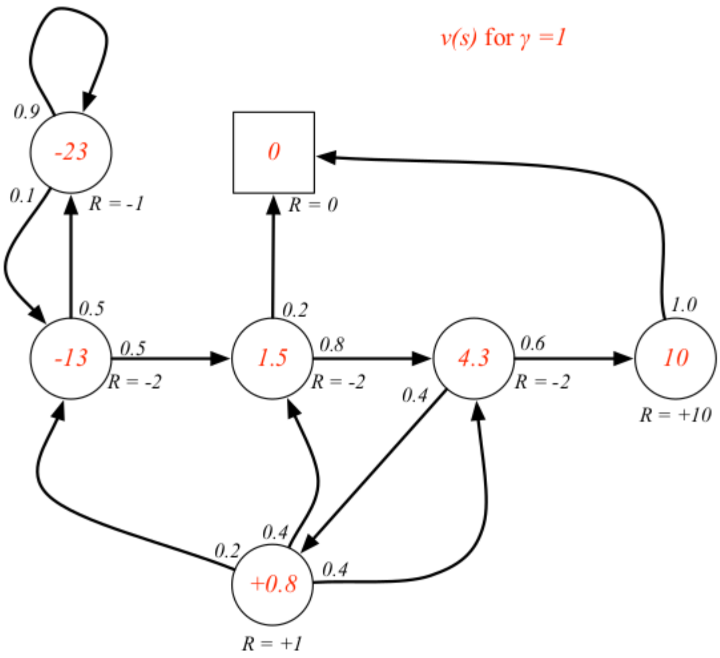
Example: State-Value Function for Student MRP (1)



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



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



Bellman Equation for MRPs

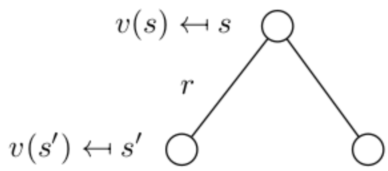
The value function can be decomposed into two parts:

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

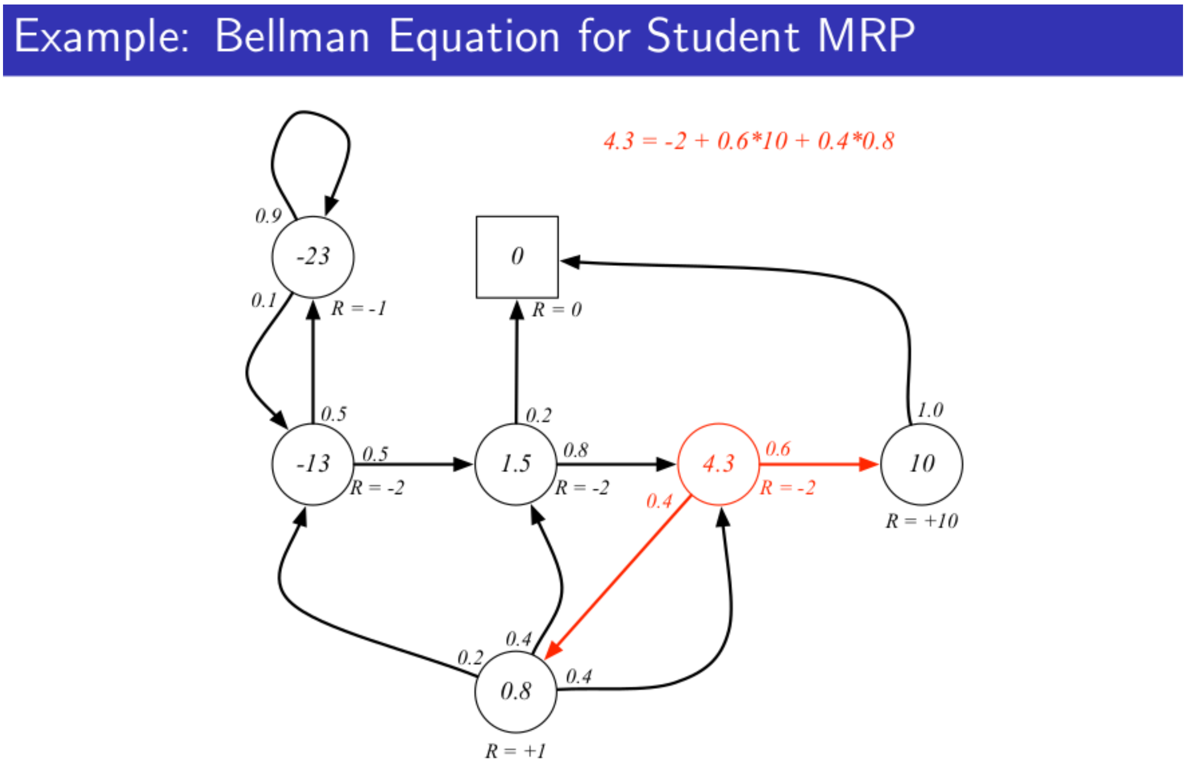
$$
\begin{aligned}
v(s) &= E[G_t|S_t = s] \\
&= E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots|S_t = s] \\
&= E[R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + \ldots)|S_t = s] \\
&= E[R_{t+1} + \gamma G_{t+1}|S_t = s] \\
&= E[R_{t+1} + \gamma v(S_{t+1})|S_t = s]
\end{aligned}
$$

Bellman Equation for MRPs (2)

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



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



Bellman Equation in Matrix Form

The Bellman equation can be expressed concisely using matrices

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

where v is a column vector with one entry per state

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

Solving the Bellman Equation

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

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

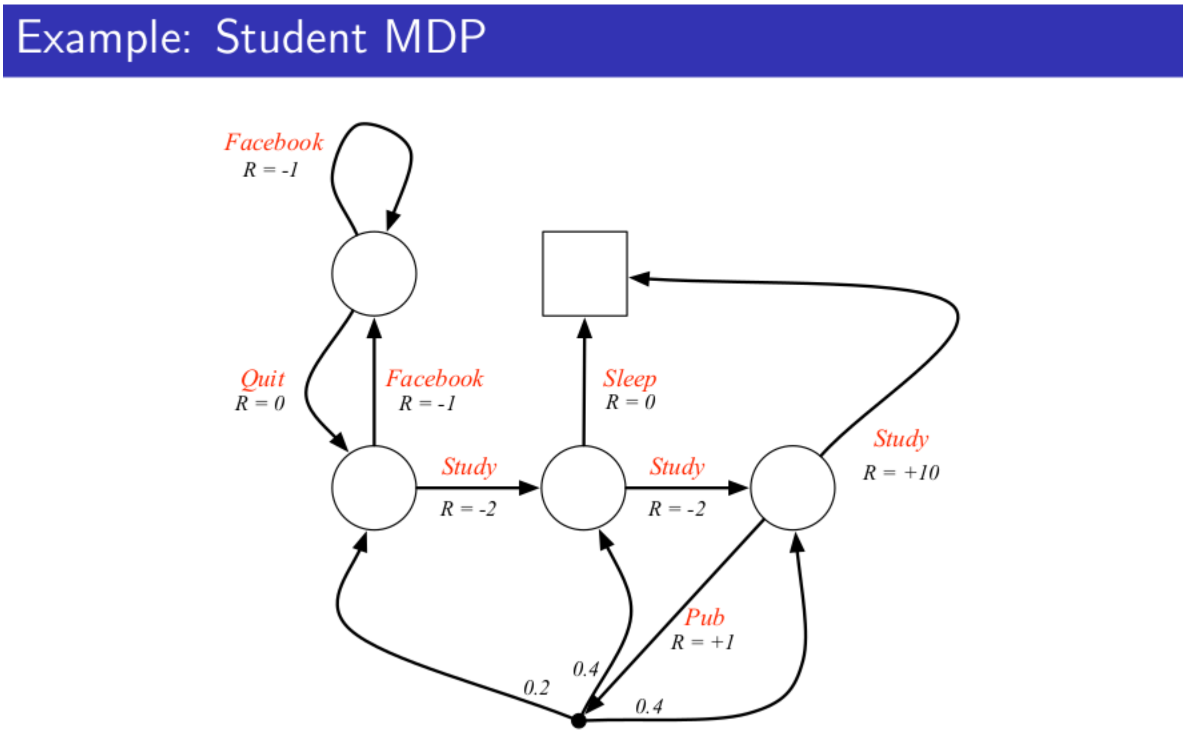
- Computational complexity is $O(n^3)$ for n states
- Direct solution only possible for small MRPs
- There are many iterative methods for large MRPs, e.g.
  - Dynamic programming
  - Monte-Carlo evaluation
  - Temporal-Difference learning

Markov Decision Processes

A Markov decision process (MDP) is a Markov reward process with decisions. It is an environment in which all states are Markov.

A Markov Decision Process is a tuple $< S, A, P, R, \gamma >$

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



Policies

A Policy π is a distribution over actions given states

$$\pi(a|s) = P[A_t = a | S_t = s]$$

- 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), $A_t \sim \pi(\cdot|S_t), \forall t > 0$
- Given an MDP $M =< S, A, P, R, \gamma >$ and a policy π
- The state sequence $S_1, S_2, \ldots$ is a Markov process $< S, P^\pi >$
- The state and reward sequence $S_1, R_2, S_2, \ldots$ is a Markov reward process $< S, P^\pi, R^\pi, \gamma >$
- where

$$P_{s,s'}^\pi = \sum_{a \in A} \pi(a|s) P_{ss'}^a$$
$$R_s^\pi = \sum_{a \in A} \pi(a|s) R_s^a$$

Value Function

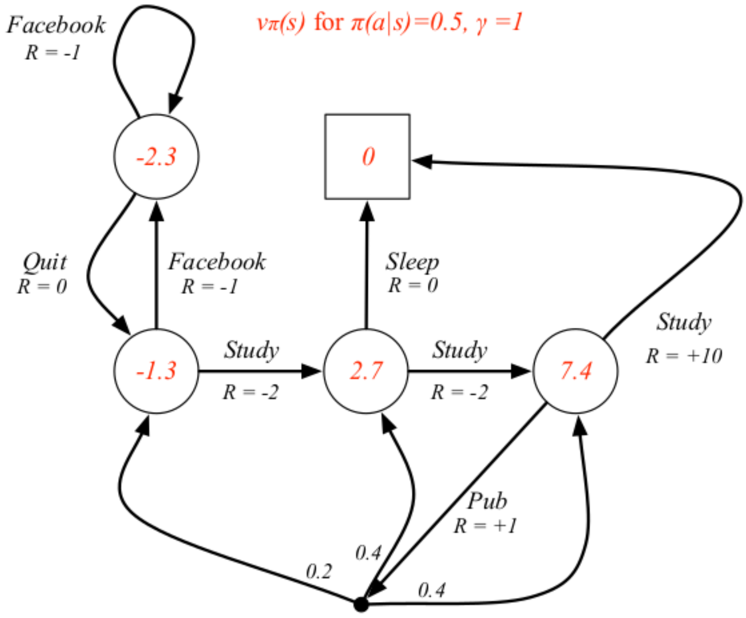
The state-value function $v_\pi(s)$ of an MDP is the expected return starting from state s, and then following policy π

$$v_\pi(s) = E_\pi[G_t | S_t = s]$$

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

$$q_\pi(s, a) = E_\pi[G_t | S_t = s, A_t = a]$$

Example: State-Value Function for Student MDP



Bellman Expectation Equation

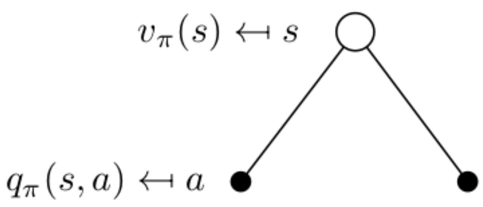
The state-value function can again be decomposed into immediate reward plus discounted value of successor state,

$$v_\pi(s) = E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s]$$

The action-value function can similarly be decomposed,

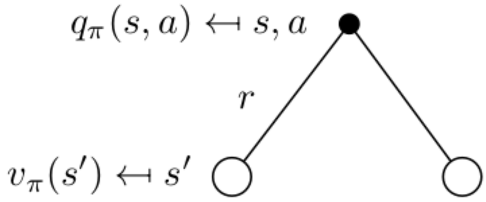
$$q_\pi(s,a) = E_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1})|S_t = s, A_t = a]$$

Bellman Expectation Equation for V^π



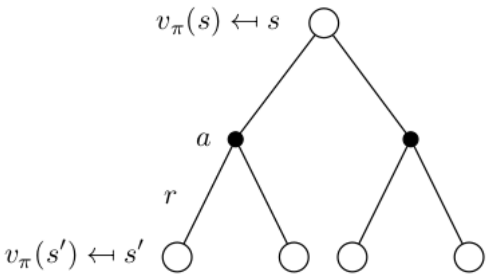
$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s,a)$$

Bellman Expectation Equation for Q^π



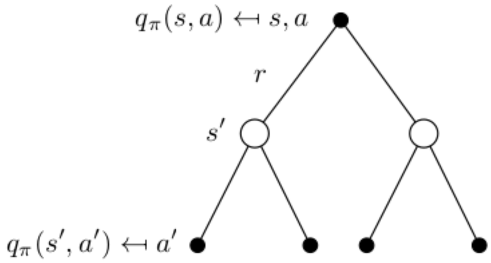
$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s')$$

Bellman Expectation Equation for v_π (2)

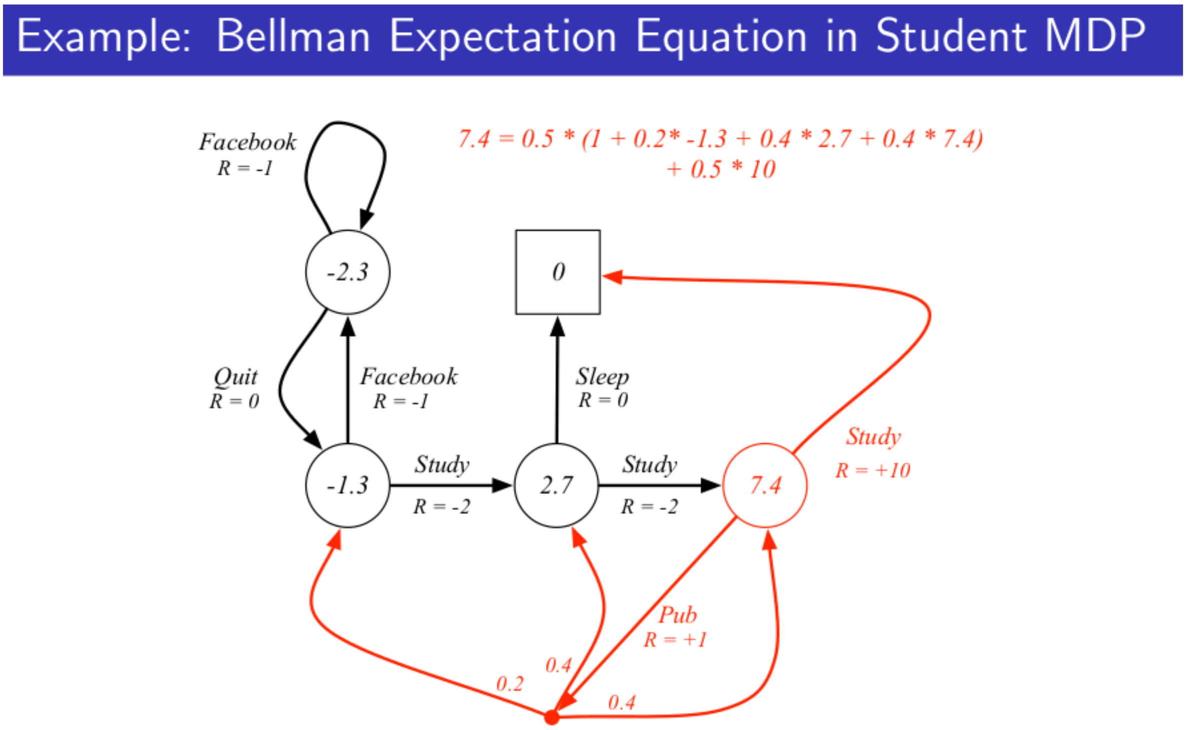


$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s') \right)$$

Bellman Expectation Equation for q_π (2)



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



Bellman Expectation Equation (Matrix Form)

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

$$v_\pi = R^\pi + \gamma P^\pi v_\pi$$

with direct solution

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

Optimal Value Function

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

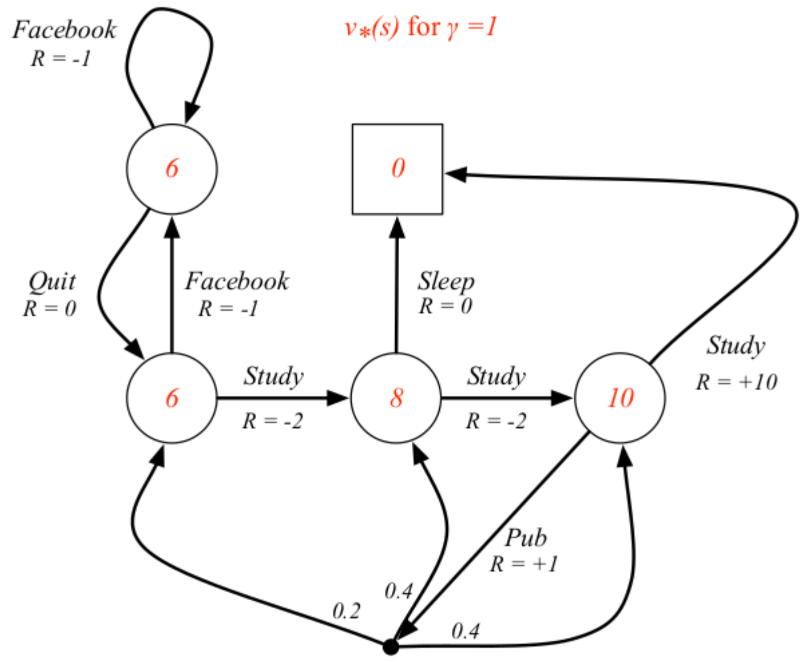
$$v_*(s) = \max_\pi v_\pi(s)$$

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

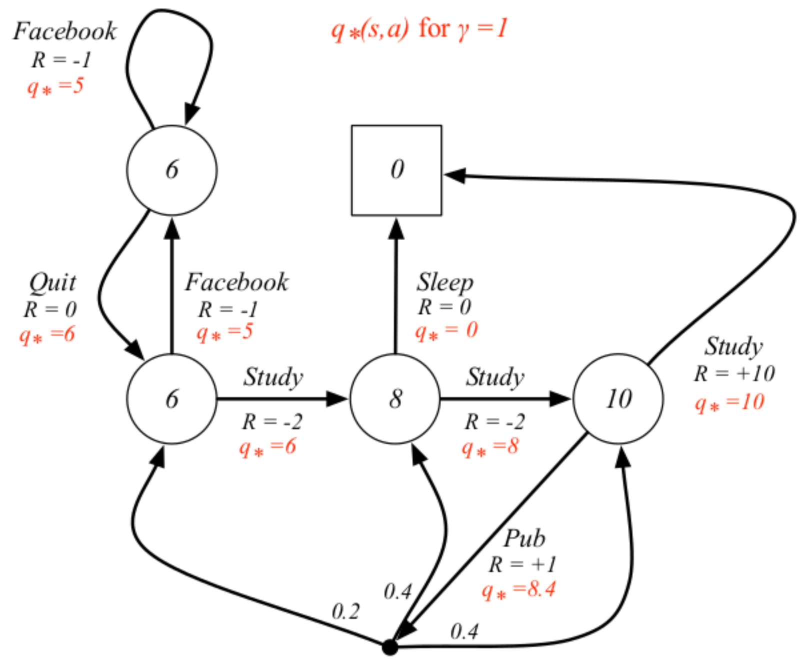
$$q_*(s, a) = \max_\pi q_\pi(s, a)$$

- 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

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

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

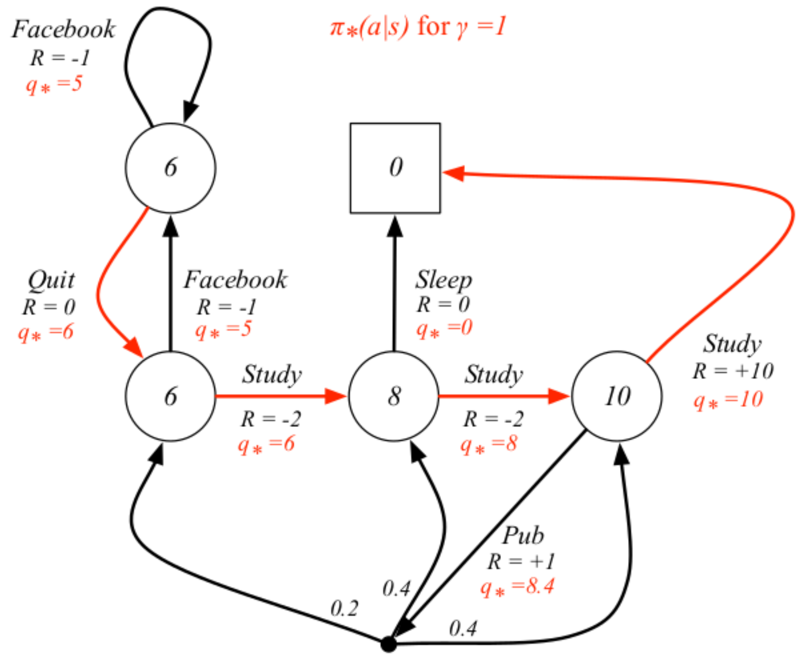
Finding an Optimal Policy

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

$$\pi_*(a|s) = \begin{cases} 1 & if \quad a = \underset{a \in A}{\arg\max}\, q_*(s,a) \\ 0 & otherwise \end{cases}$$

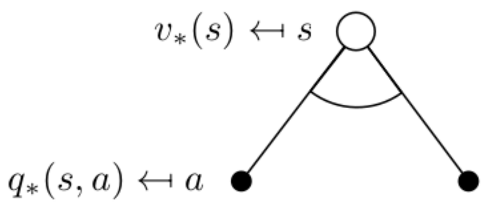
- There is always a deterministic optimal policy for any MDP
- If we know $q_*(s,a)$, we immediately have the optimal policy

Example: Optimal Policy for Student MDP



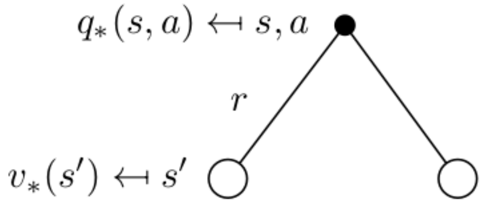
Bellman Optimality Equation for v_*

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



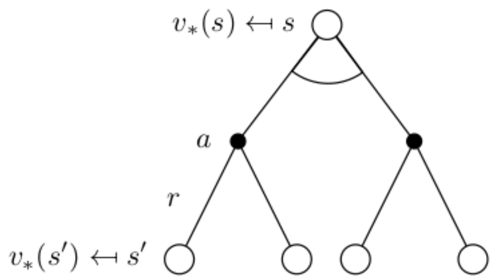
$$v_*(s) = \max_a q_*(s,a)$$

Bellman Optimality Equation for Q^*



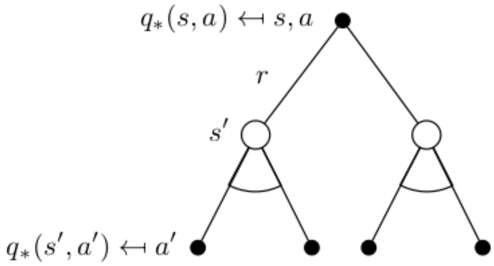
$$q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s')$$

Bellman Optimality Equation for V^* (2)

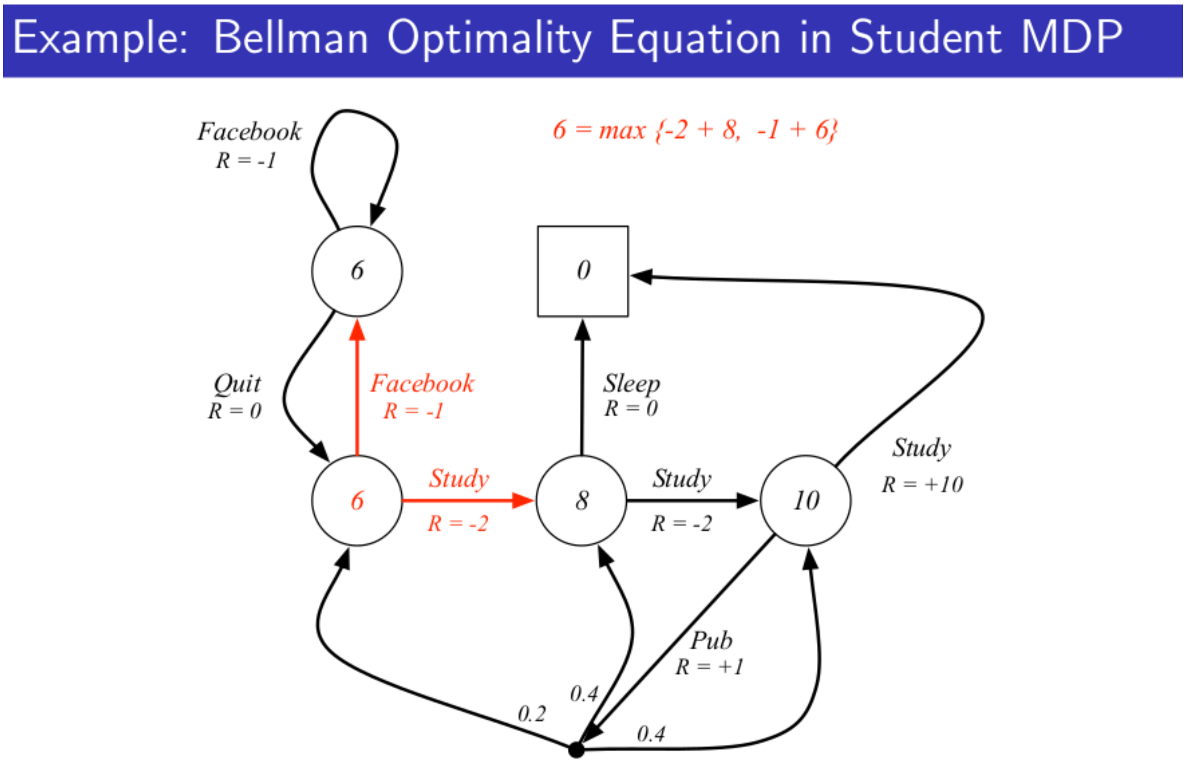


$$v_*(s) = \max_a \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s')$$

Bellman Optimality Equation for Q^* (2)



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



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
  - Q-learning
  - Sarsa

Extensions to MDPs

Infinite and continuous 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$\rightarrow 0$

Partially observable MDPs

A Partially Observable Markov Decision Process is an MDP with hidden states. It is a hidden Markov model with actions.

A POMDP is a tuple $< S, A, O, P, R, Z, \gamma >$

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

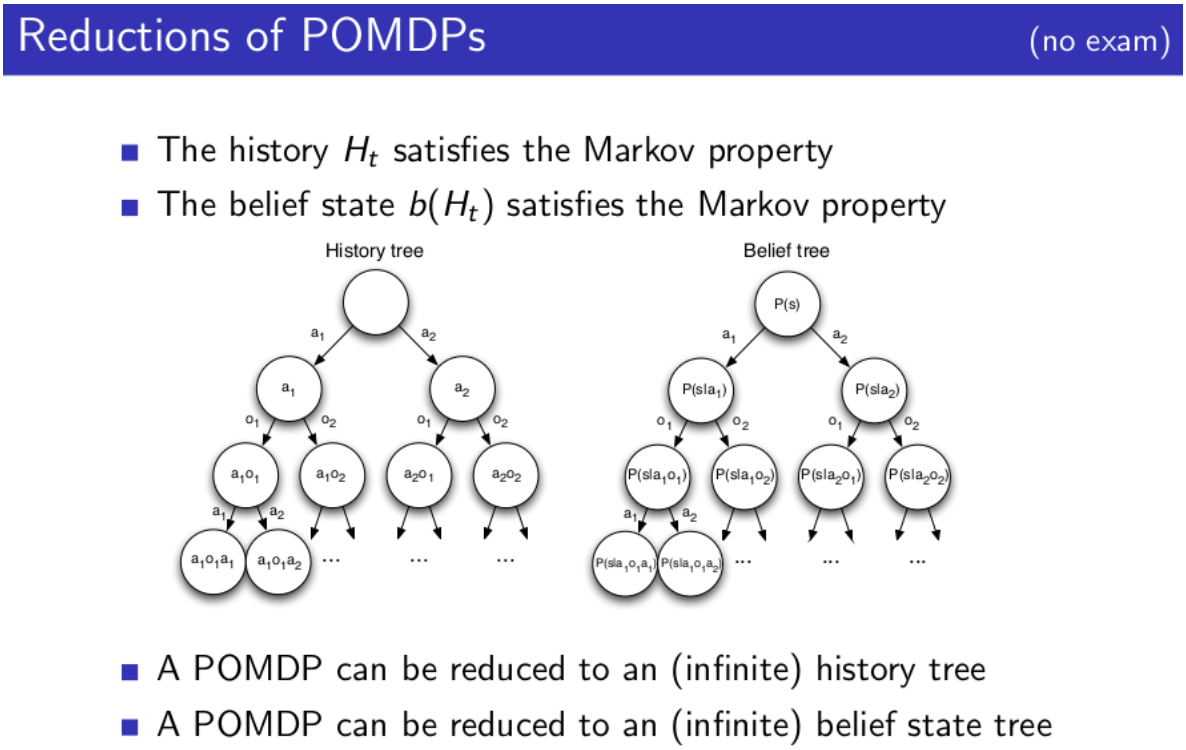
Belief States

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

$$H_t = A_0, O_1, R_1, \dots, A_{t-1}, O_t, R_t$$

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

$$b(h) = (P[S_t = s^1 | H_t = h], \dots, P[S_t = s^n | H_t = h])$$



Undiscounted, average reward MDPs

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

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

$$d^\pi(s) = \sum_{s' \in S} d^\pi(s') P_{s's}$$

Ergodic MDP

An MDP is ergodic if the Markov chain induced by any policy is ergodic.

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

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

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,

$$\tilde{v}_{\pi}(s) = E_{\pi}[\sum_{k=1}^{\infty}(R_{t+k} - \rho^{\pi})|S_t = s]$$

There is a corresponding average reward Bellman equation,

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