

# 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

“The future is independent of the past given the present”

## Definition

A state  $S_t$  is *Markov* if and only if

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

- 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  $s'$ , the *state transition probability* is defined by

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

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

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

where each row of the matrix sums to 1.

# Return

## Definition

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

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

- 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.
  - $\gamma$  close to 0 leads to "myopic" evaluation
  - $\gamma$  close to 1 leads to "far-sighted" evaluation

# 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  $s$

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

# Markov Decision Process

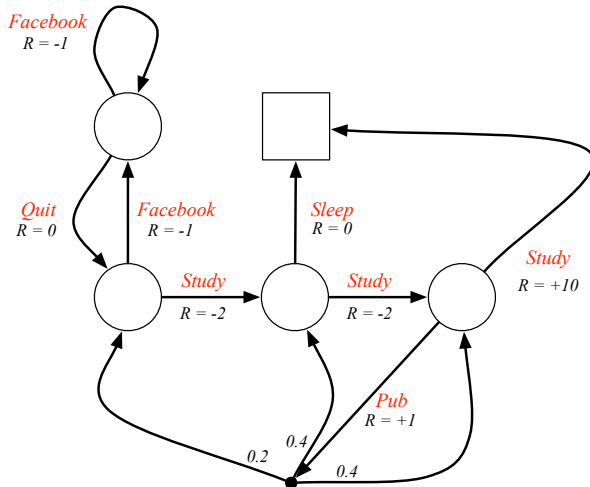
A Markov decision process (MDP) is a Markov reward process with decisions. It is an *environment* in which all states are Markov.

## 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,  
 $\mathcal{P}_{ss'}^a = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$
- $\mathcal{R}$  is a reward function,  $\mathcal{R}_s^a = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$
- $\gamma$  is a discount factor  $\gamma \in [0, 1]$ .

# Example: Student MDP



# Policies (1)

## Definition

A *policy*  $\pi$  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),  
 $A_t \sim \pi(\cdot|S_t), \forall t > 0$



## Policies (2)

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

$$\mathcal{P}_{s,s'}^\pi = \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{P}_{ss'}^a$$

$$\mathcal{R}_s^\pi = \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{R}_s^a$$

# 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  $\pi$

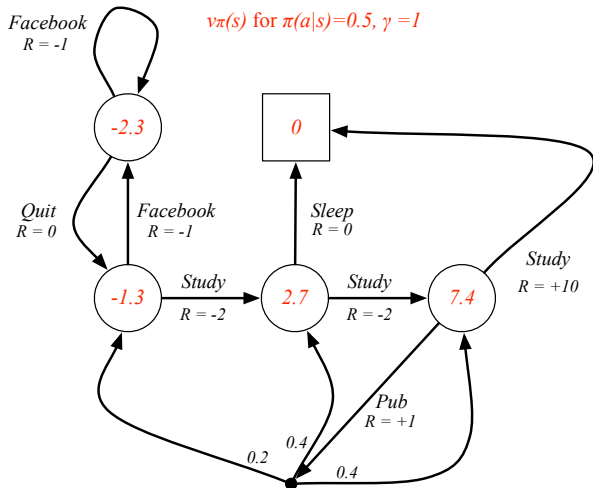
$$v_{\pi}(s) = \mathbb{E}_{\pi} [G_t \mid S_t = s]$$

## Definition

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

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

# Example: State-Value Function for Student MDP



# Bellman Expectation Equation

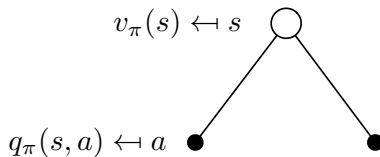
The state-value function can again be decomposed into immediate reward plus discounted value of successor state,

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

The action-value function can similarly be decomposed,

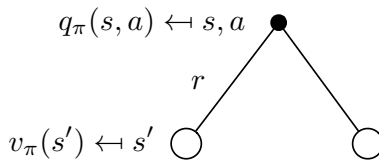
$$q_{\pi}(s, a) = \mathbb{E}_{\pi} [R_{t+1} + \gamma q_{\pi}(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a]$$

# Bellman Expectation Equation for $V^\pi$



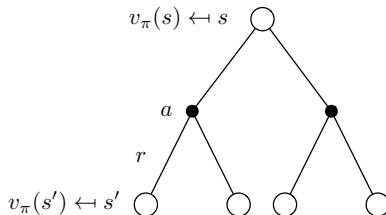
$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s, a)$$

# Bellman Expectation Equation for $Q^\pi$



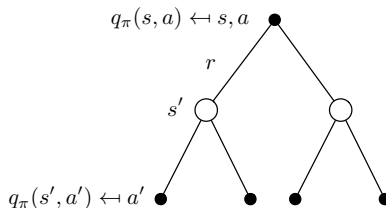
$$q_\pi(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s')$$

# Bellman Expectation Equation for $v_\pi$ (2)



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

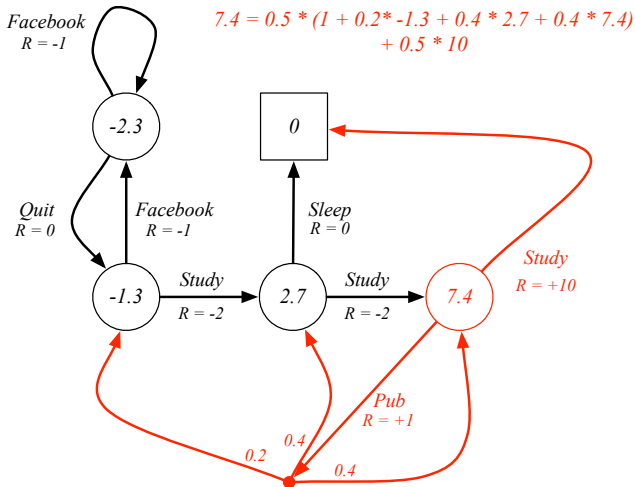
# Bellman Expectation Equation for $q_\pi$ (2)



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



# 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

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

# Optimal Value Function

## Definition

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

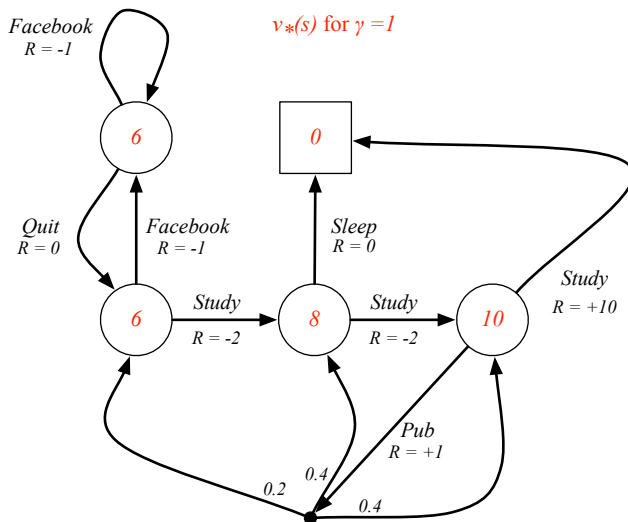
$$v_*(s) = \max_{\pi} v_{\pi}(s)$$

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

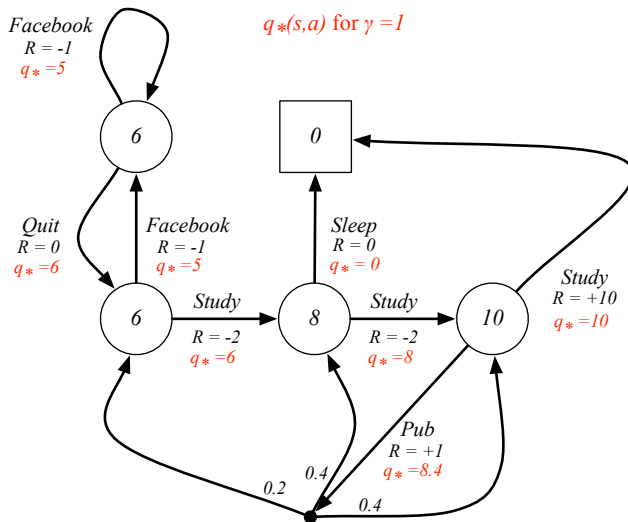
$$q_*(s, a) = \max_{\pi} q_{\pi}(s, a)$$

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

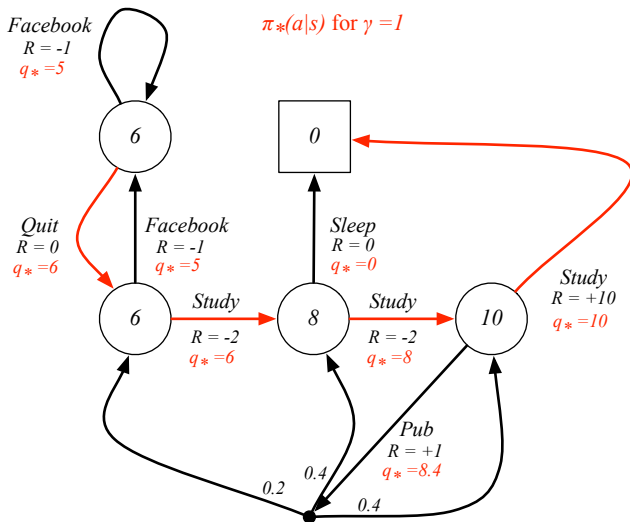
# Finding an Optimal Policy

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

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

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

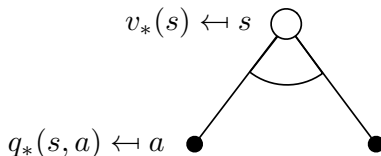
# Example: Optimal Policy for Student MDP





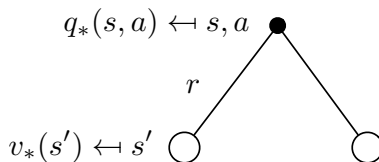
# Bellman Optimality Equation for $v_*$

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



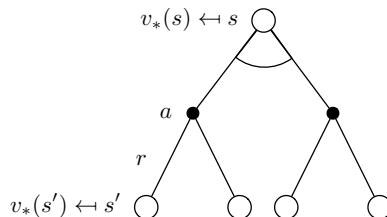
$$v_*(s) = \max_a q_*(s, a)$$

# Bellman Optimality Equation for $Q^*$



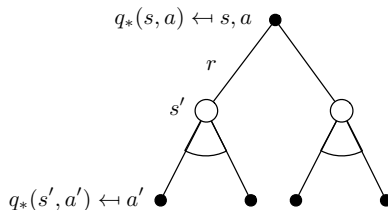
$$q_*(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s')$$

# Bellman Optimality Equation for $V^*$ (2)



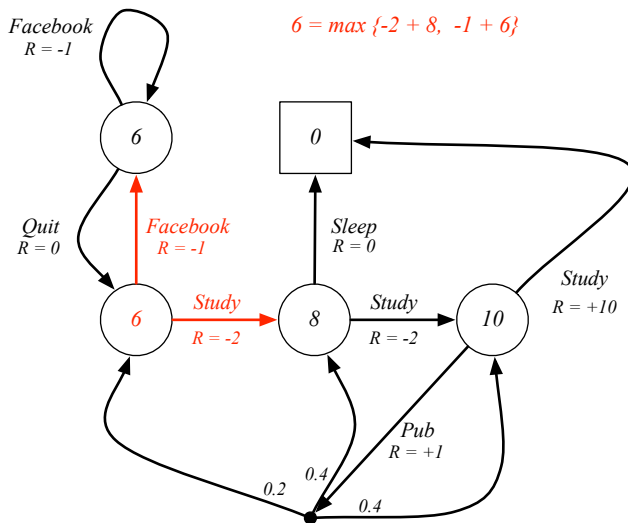
$$v_*(s) = \max_a \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s')$$

# Bellman Optimality Equation for $Q^*$ (2)



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

# 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