Formalizing Reinforcement Learning

Slides from DAVIDE BACCIU - UNIVERSITÀ DI PISA

Rewards

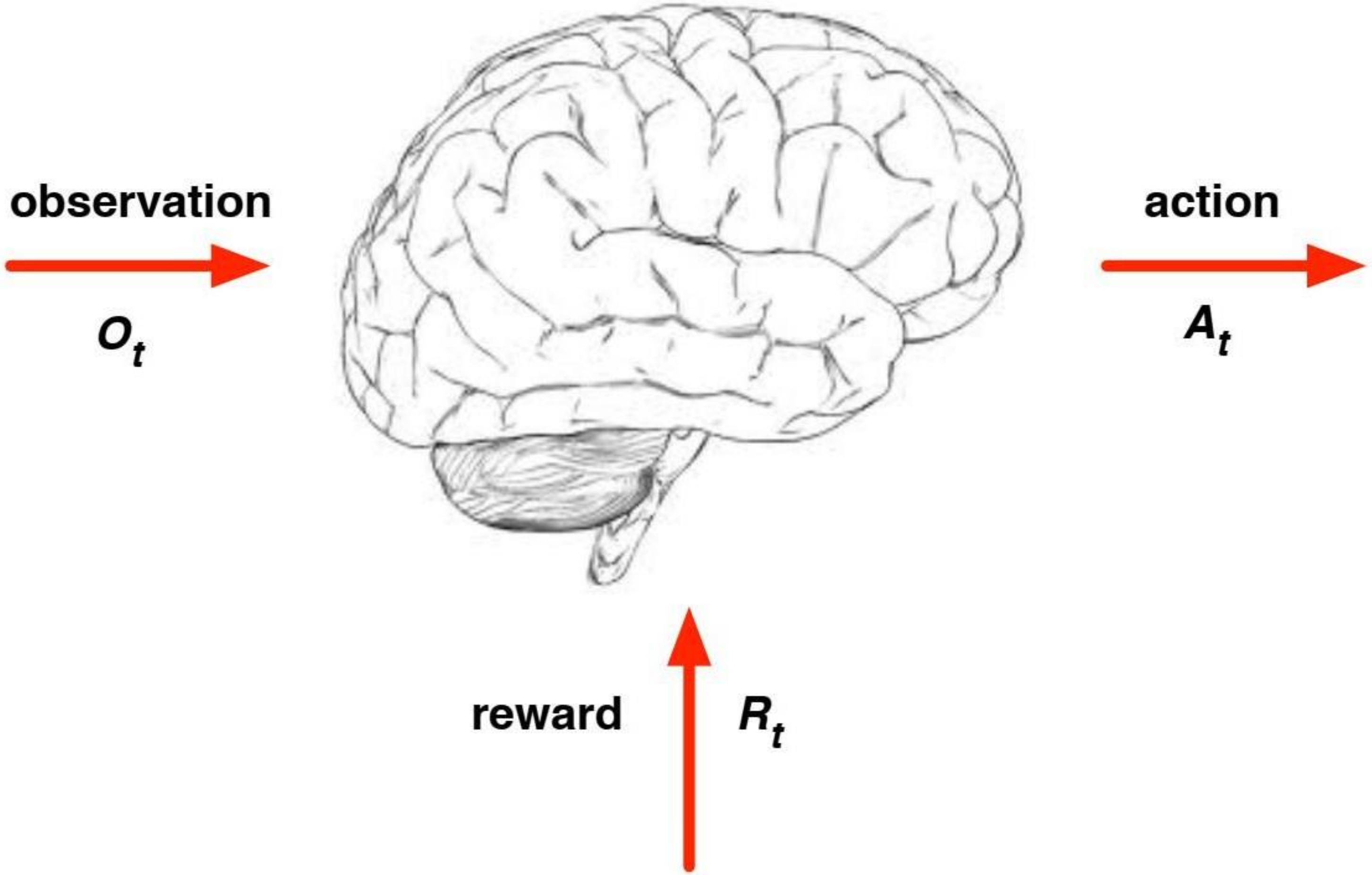
- ✓ A reward R_t is a scalar feedback signal
- ✓ Indicates how well agent is doing at step t
- ✓ The agent's job is to maximise cumulative reward

Reinforcement learning is based on the reward hypothesis

- ✓ All goals can be described by the maximisation of expected cumulative reward

Sequential Decision Making

- ✓Goal: *select actions to maximise total future reward*
- ✓Actions may have long term consequences
- ✓Reward may be delayed
- ✓It may be better to sacrifice immediate reward to gain more long-term reward
- ✓Examples:
  - ✓A financial investment (may take months to mature)
  - ✓Refuelling a helicopter (might prevent a crash in several hours)
  - ✓Blocking opponent moves (might help winning chances many moves from now)



Agent and Environment

- At each step t the agent:
  - Executes action $\mathbf{A_t}$
  - Receives observation $\boldsymbol{O_t}$
  - Receives scalar reward $\boldsymbol{R_t}$
- The Environment:
  - Receives action $\mathbf{A_t}$
  - Emits observation $\boldsymbol{O_{t+1}}$
  - Emits scalar reward $\boldsymbol{R_{t+1}}$
- t increments at environment step

History and State

The history is the sequence of observations, actions, rewards

$$H_t = O_1; R_1; A_1 \dots A_{t-1}; O_r; R_t$$

- ✓ i.e. all observable variables up to time t
- ✓ i.e. the sensorimotor stream of a robot or embodied agent

What happens next depends on the history:

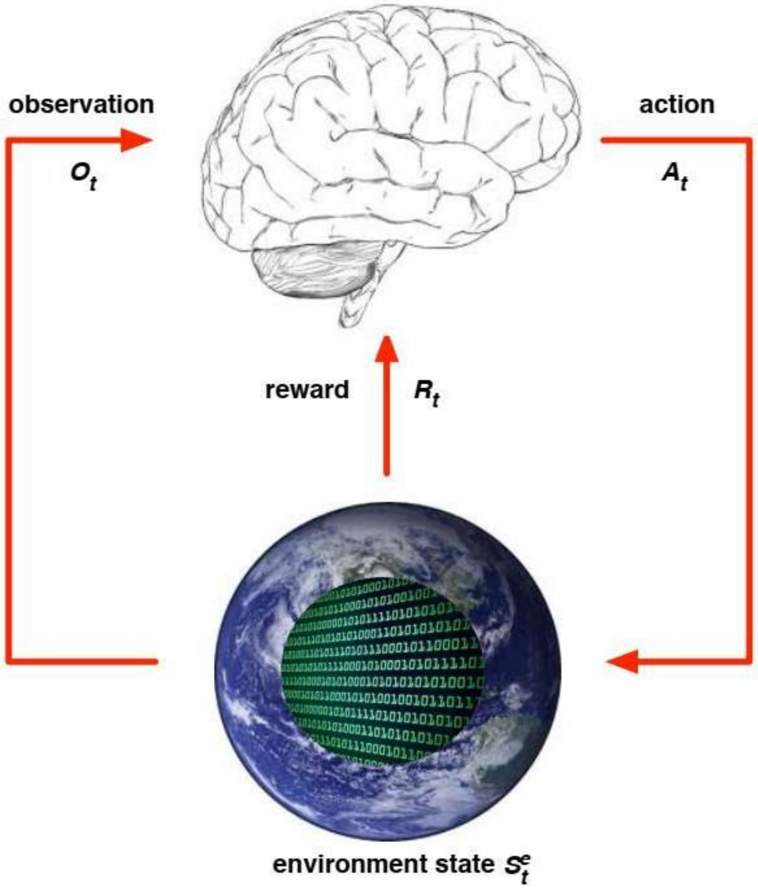
- ✓ The agent selects actions
- ✓ The environment selects observations/rewards

State S_t is the information used to determine what happens next and is a function of history

$$S_t = f(H_t)$$

Environment State

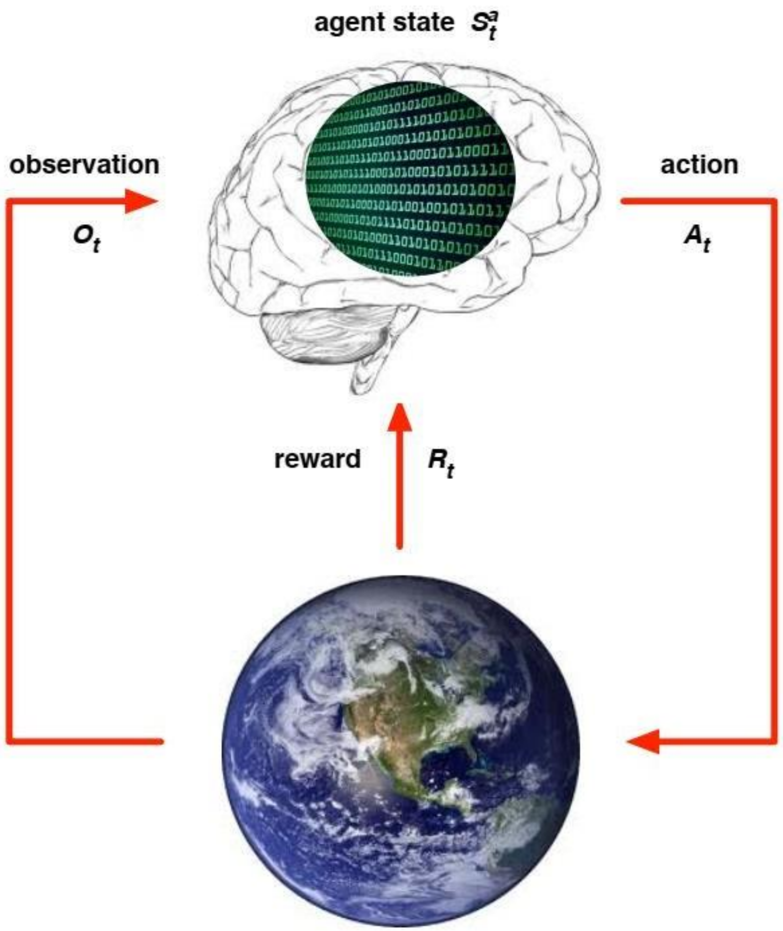
- ✓The environment state S_t^e is the environment e private representation at time t
  - ✓Whatever information the environment uses to generate the next observation/reward
- ✓The environment state is not usually visible to the agent (unobservable environment)
- ✓ Even if S_t^e is visible, it may contain irrelevant information



Agent State

- The agent state S_t^a the internal representation owned by agent a
  - Whatever information the agent uses to select next action
- The agent state is the information used by reinforcement learning algorithms
- Generally speaking a function of history

$$S_t^a = f(H_t)$$



Information (Markov) State

An information state (Markov state) contains all useful information from the history

Definition (Markov State)

A state S_t is Markov if and only if

$$P(S_{t+1}|S_1, \dots, S_t) = P(S_{t+1}|S_t)$$

- ✓ The future is independent of the past given present (d-separation)

$$H_{1:t} \rightarrow S_t \rightarrow H_{t+1:\infty}$$

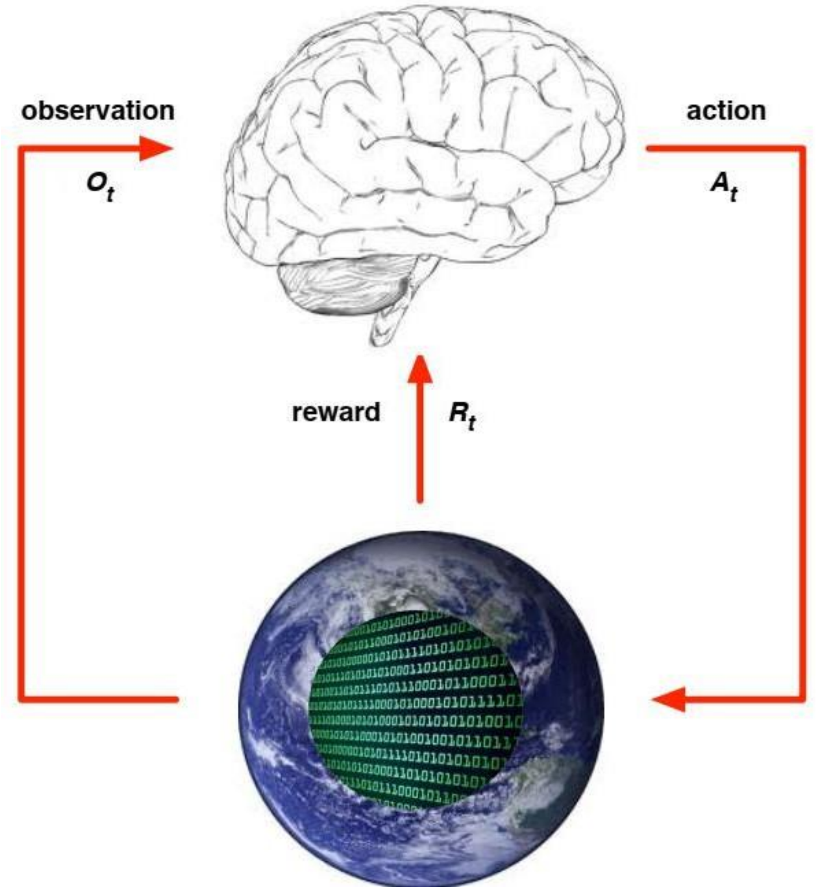
- ✓ The state is a sufficient statistics for the future
- ✓ The environment state S_t^e is Markov
- ✓ The history H_t is Markov

Fully Observable Environment

- ✓ Full observability $\Longrightarrow$ Agent directly observes the environment state

$$O_t = S_t^a = S_t^e$$

- ✓ Formally this is a Markov Decision Process (MDP)
- ✓ Next lecture (and much of the RL literature)



Partially Observable Environment

- Partial observability $\Longrightarrow$ Agent indirectly observes the environment
  - A robot with camera vision only may not know absolute location
  - A trading agent only observes current prices
  - A poker player only observes public cards
- Formally $S_t^a \neq S_t^e$ and the problem is a Partially Observable Markov Decision Process (POMDP)
- The agent needs to build its own state representation S_t^a
  - History: $S_t^a = H_t$
  - Beliefs on environment state: $S_t^a = [P(S_t^e = s^1) \dots P(S_t^e = s^N)]$
  - A dynamic memory (RNN): $S_t^a = \sigma(W_s S_{t-1}^a + W_o O^t)$

Components of a Reinforcement Learning Agent

Key Components of an RL Agent

- ✓ **Policy**: agent's behaviour function
- ✓ **Value function**: how good is each state and/or action
- ✓ **Model**: agent's representation of the environment

An RL agent may include one or more of the above

Policy

- ✓ A policy π is the agent's behaviour
- ✓ It is a map from state s to action a
  - ✓ Deterministic policy: $\mathrm{a} = \pi(s)$
  - ✓ Stochastic policy: $\pi(a|s) = P(A_t = a|S_t = s)$

Value Function

How “good” is a specific state/action for an agent?

Value Function

- ✓The value function v is a predictor of future reward
- ✓Used to evaluate the goodness/badness of states
- ✓And therefore to select between actions, e.g

$$v_\pi(s) = \mathrm{D}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+2} + \cdots | S_t = s]$$

Expected (discounted) future reward following policy π from state s

Model

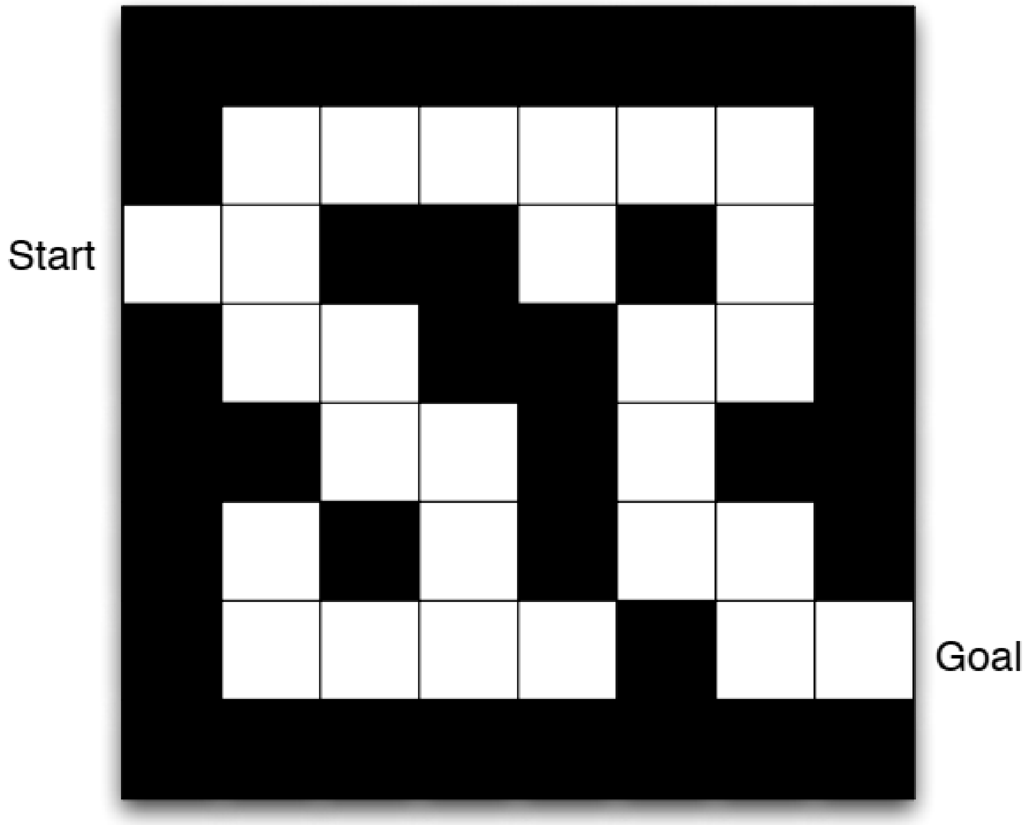
- A model predicts what the environment will do next
- Predict next state s' following an action a

$$P_{ss'}^{a} = P(S_{t+1} = s' | S_t = s, A_t = a)$$

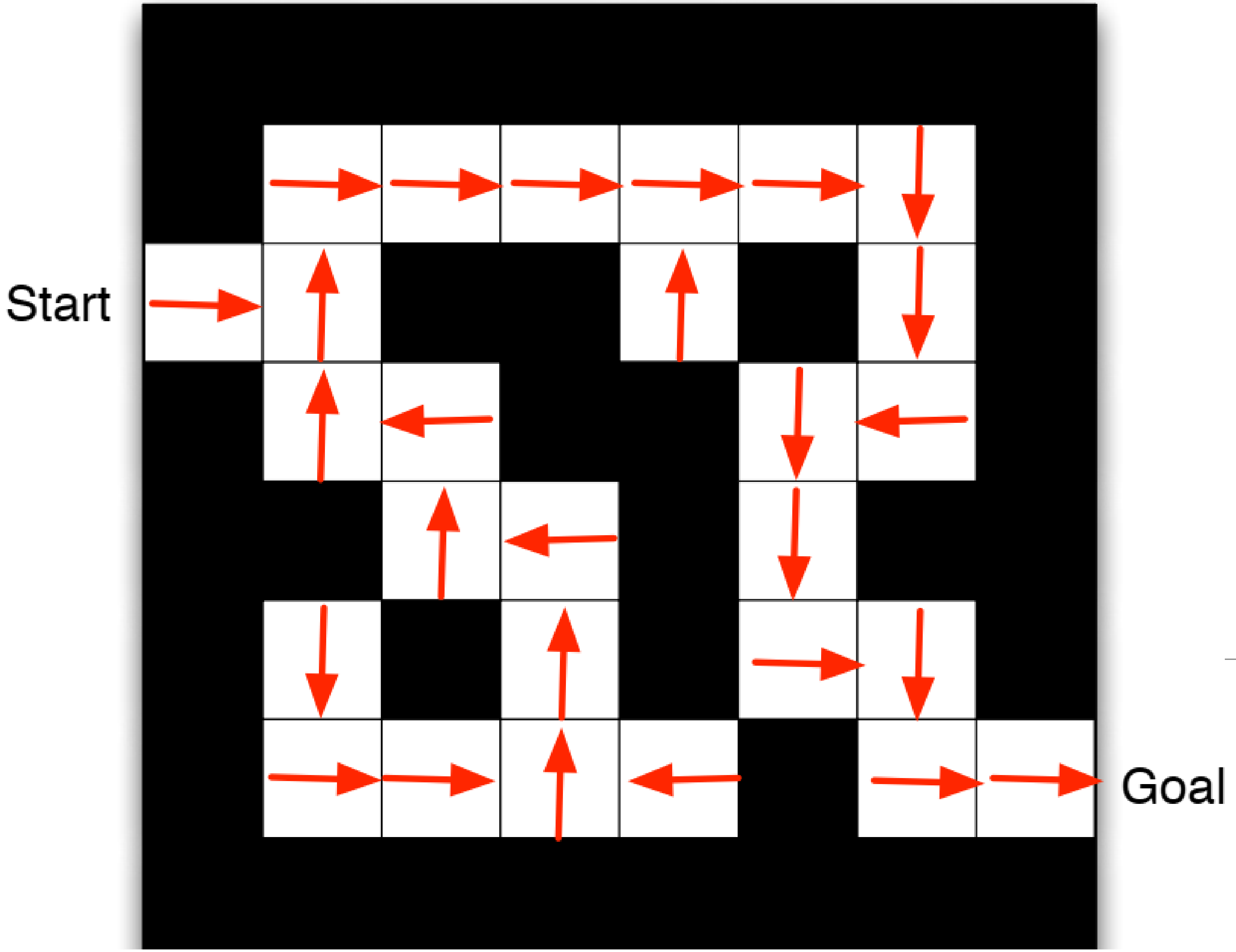
- Predict next reward

$$\mathcal{R}_s^a = \mathrm{D}[R_{t+1} | S_t = s, A_t = a]$$

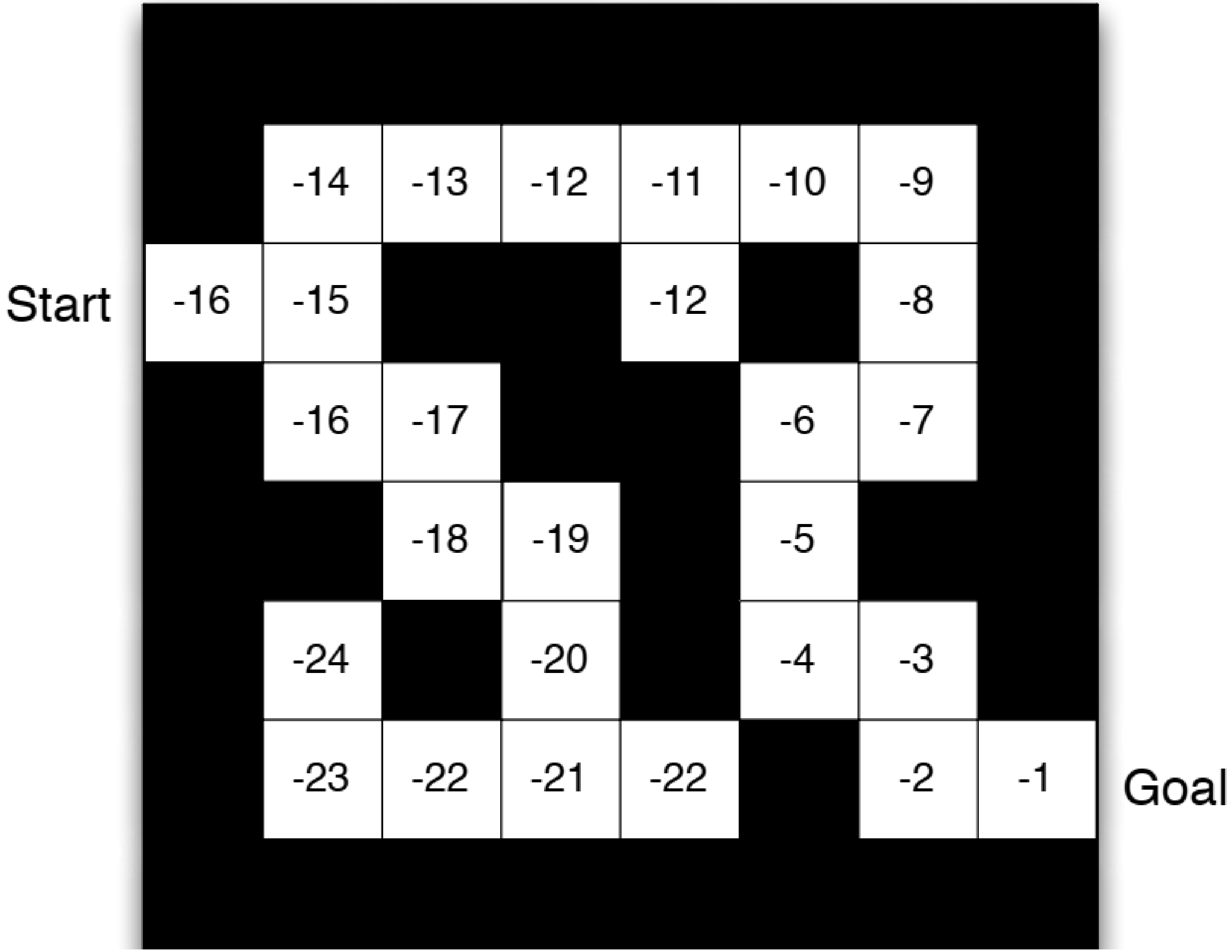
A Forever Classic - The Maze Example



- ✓ Rewards: -1 per time-step
- ✓ Actions: N, E, S, W
- ✓ States: Agent location



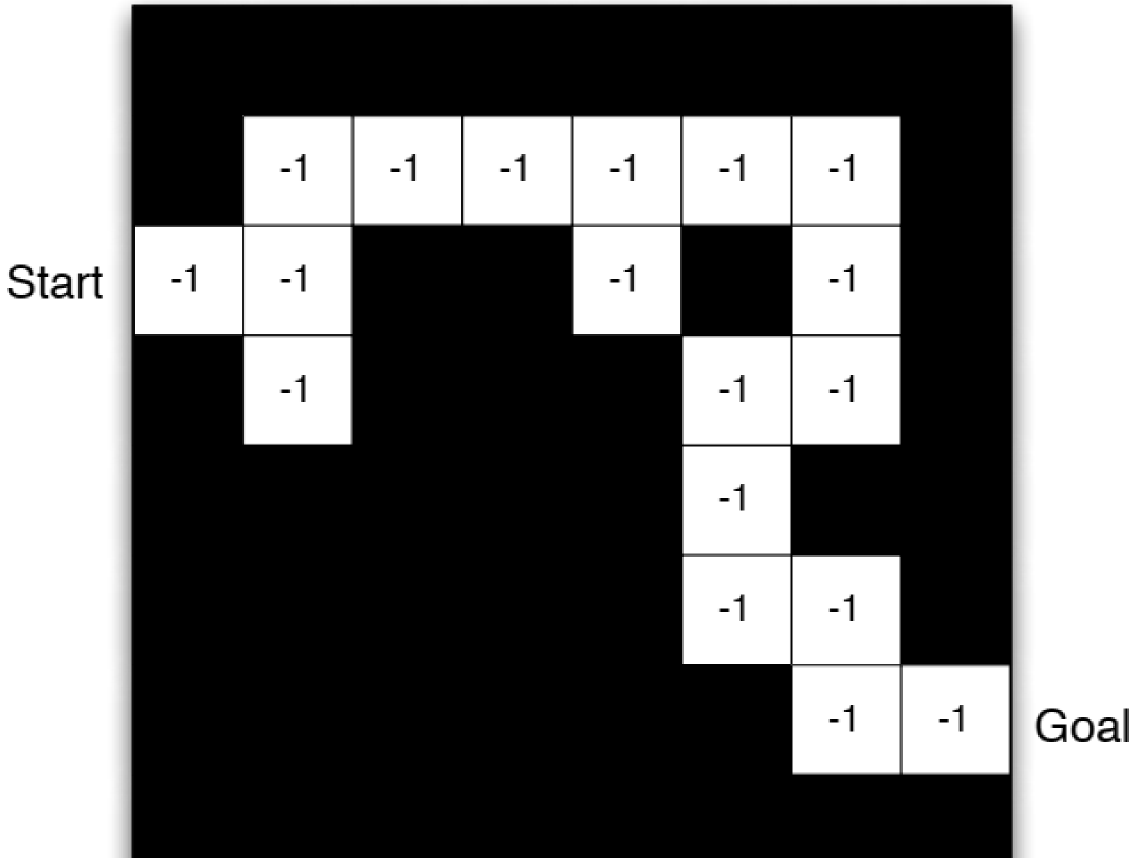
Arrows represent policy $\pi(s)$ for each state s



Maze Example (Value Function)

Numbers denote the value $v_\pi(s)$ for each s

Expected time to reach the goal



Maze Example (Model)

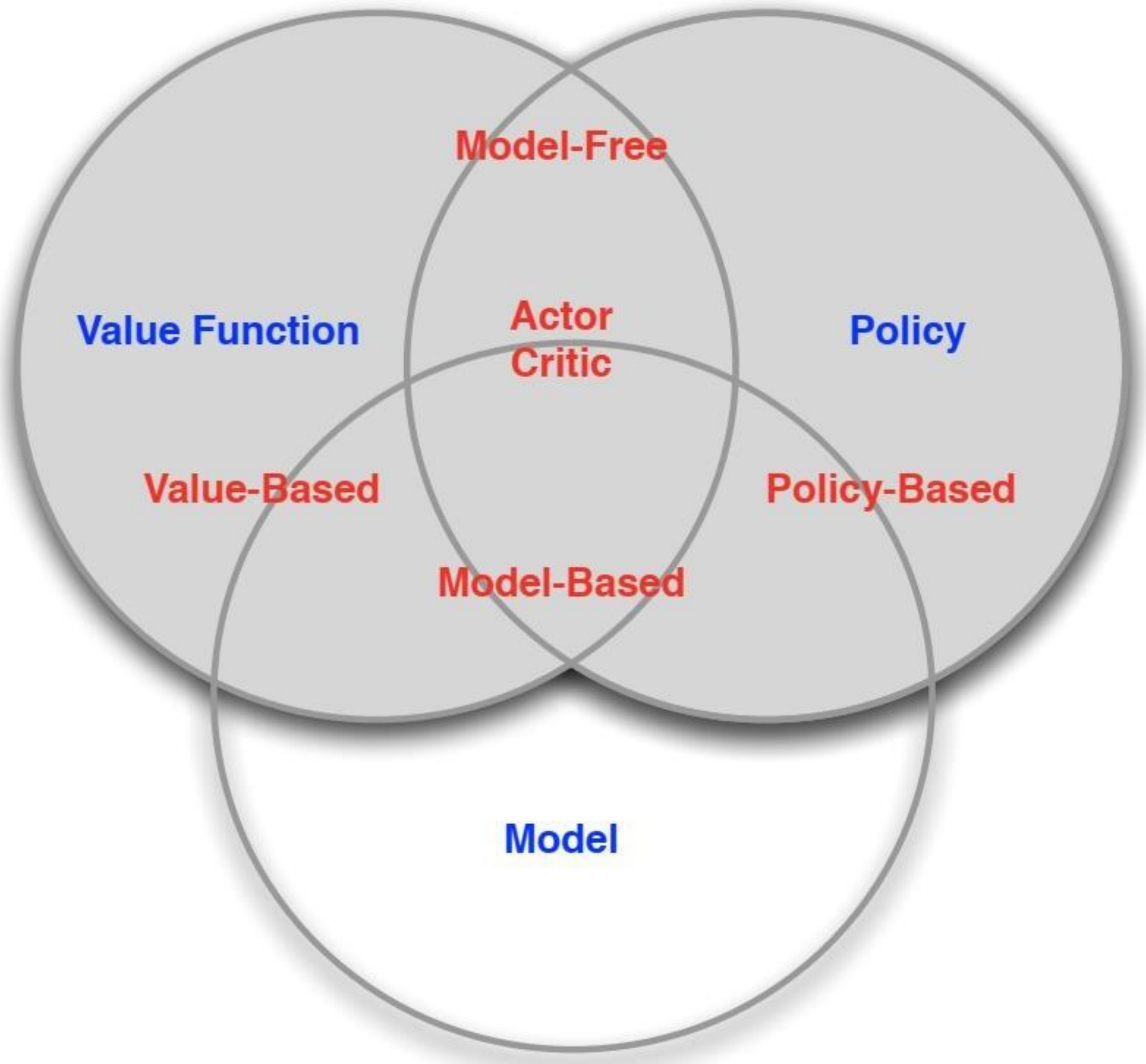
- Agent may have an internal (imperfect) model of the environment
  - How actions change the state
  - How much reward from each state
- Grid Layout: transition model $\mathcal{P}_{ss'}^{a}$
- Numbers: immediate reward model $\mathcal{R}_{s}^{a}$

Characterizing RL Agents (I)

- ✓ Value Based
  - ✓ Policy (Implicit)
  - ✓ Value Function
- ✓ Policy Based
  - ✓ Policy
  - ✓ Value Function
- ✓ Actor Critic
  - ✓ Policy
  - ✓ Value Function

Characterizing RL Agents (II)

- ✓ Model Free
  - ✓ ~~Model~~
  - ✓ Policy and/or Value Function
- ✓ Model Based
  - ✓ Model
  - ✓ Policy and/or Value Function


Model-Free
Value Function
Actor
Critic
Policy
Value-Based
Policy-Based
Model-Based
Model

A Taxonomy

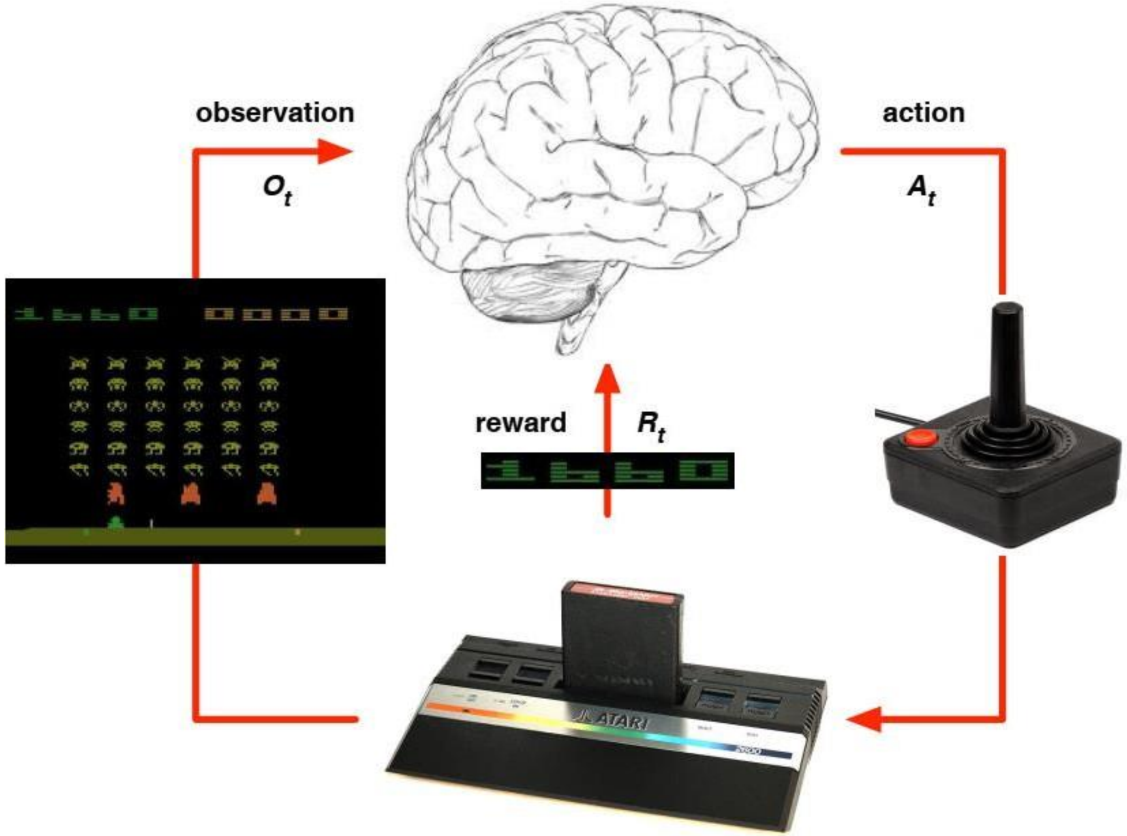
Problems within Reinforcement Learning

Learning and Planning

Two fundamental problems in sequential decision making

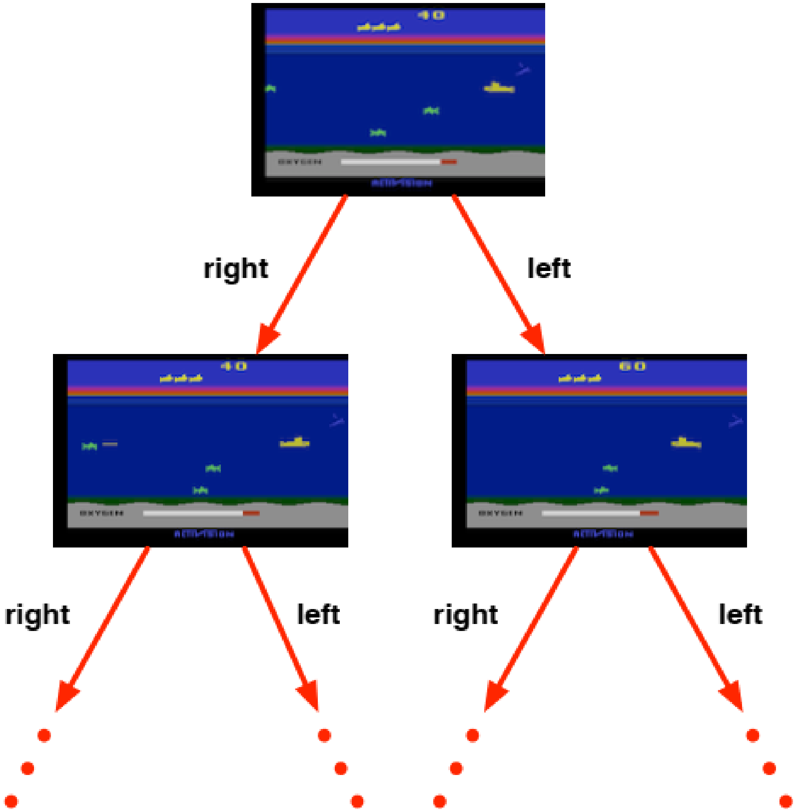
- ✓ Reinforcement Learning
  - ✓ The environment is initially unknown
  - ✓ The agent interacts with the environment
  - ✓ The agent improves its policy
- ✓ Planning (reasoning, introspection, search,…)
  - ✓ A model of the environment is known
  - ✓ The agent performs computations with its model (no external interaction)
  - ✓ The agent improves its policy

Atari Example – Reinforcement learning



- ✓Rules of the game are unknown
- ✓Learn directly from interactive game-play
- ✓Pick actions on joystick, see pixels and scores

Atari Example – Planning



- ✓
- ✓
- ✓
  - ✓ what would the next state be?
  - ✓ what would the score be?
- ✓
  - ✓ e.g. tree search

Exploration Vs Exploitation

- ✓ Reinforcement Learning follows a trial-and-error process
- ✓ The agent should discover a good policy
- ✓ From its experiences of the environment
- ✓ Without losing too much reward along the way

- ✓ Exploration finds more information about the environment
- ✓ Exploitation exploits known information to maximise reward

Effective reinforcement learning requires to trade between exploration and exploitation

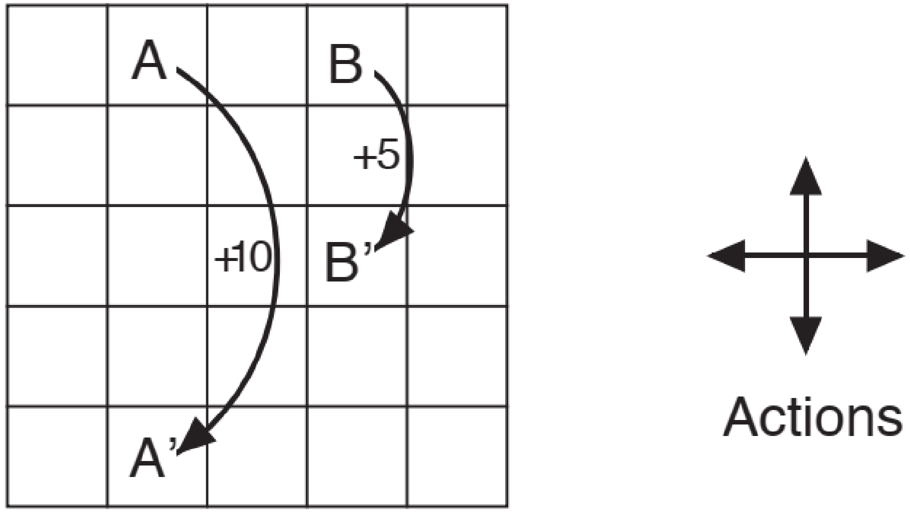
Examples

- ✓ Restaurant Selection
  - ✓ Exploitation - Go to your favourite restaurant
  - ✓ Exploration - Try a new restaurant
- ✓ Holiday planning
  - ✓ Exploitation – The camping site you go to since you are born
  - ✓ Exploration – Hitchhike and follow the flow
- ✓ Game Playing
  - ✓ Exploitation - Play the move you believe is best
  - ✓ Exploration - Play an experimental move

Prediction & Control

- Prediction: evaluate the future
  - Given a policy
- Control: optimise the future
  - Find the best policy

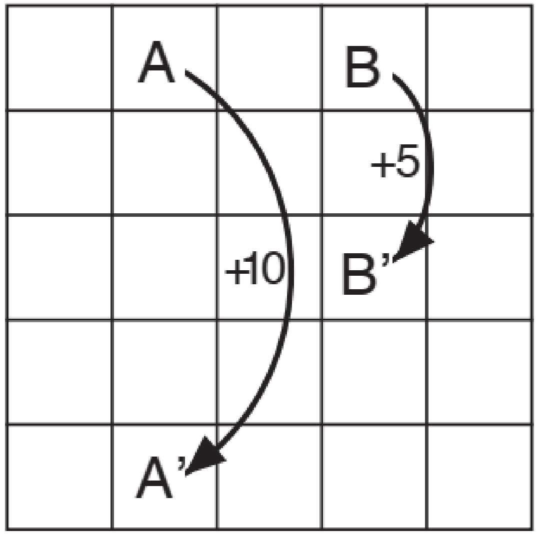
Gridworld Example - Prediction



3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

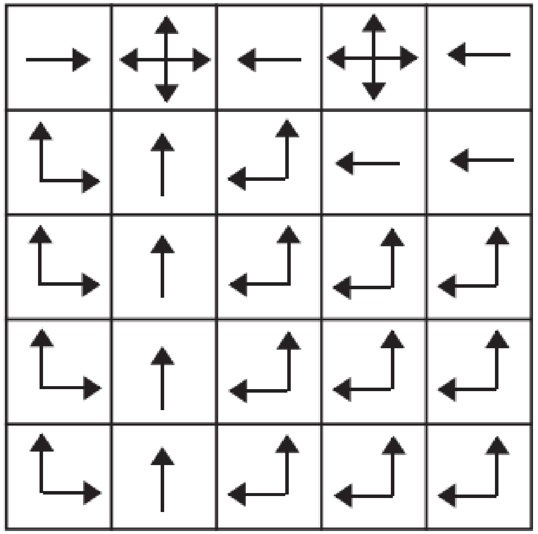
What is the value function for the uniform random policy?

Gridworld Example - Control



22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

v_*



π_*

What is the optimal value function over all possible policies?

What is the optimal policy?

Wrap-up

Take home messages

- ✓ Reinforcement learning is a general-purpose framework for decision-making
- ✓ Reinforcement learning is for an agent with the capacity to act and observe
- ✓ The state is the sufficient statistics to characterize the future
  - ✓ Depends on the history of actions and observations
  - ✓ Environment state Vs Agent state
- ✓ Success is measured by a scalar reward signal
  - ✓ The goal is to select actions to maximise future reward (exploit)
  - ✓ In order to be effective we should not forget to explore

Markov Decision Processes

Introduction

Outline

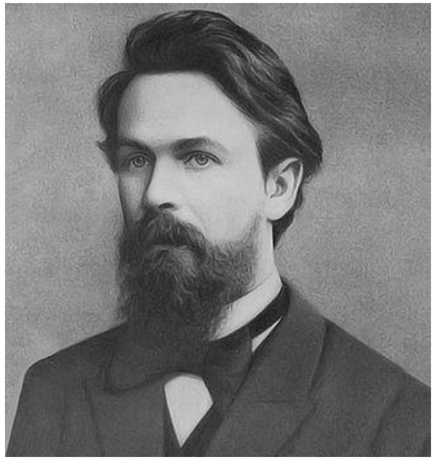
- ✓ Formalizing reinforcement learning with fully observable environment
  - ✓ Markov Processes
  - ✓ Markov Rewards
  - ✓ Markov Decision Processes
- ✓ A recursive formulation for value functions
- ✓ Extensions of the Markov decision process

Introduction to MDPs

- ✓ Markov decision processes formally describe an environment for reinforcement learning
  - ✓ 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 Process

Markov Property



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

Definition (Markov State)

A state S_t is Markov if and only if

$$P(S_{t+1}|S_1, \dots, S_t) = P(S_{t+1}|S_t)$$

- ✓ The state captures all relevant information from the history
- ✓ Once the state is known, the history may be thrown away
- ✓ The state is a sufficient statistics for the future

State Transition Matrix

✓For a starting state s and successor state s', the state transition probability is defined by

$$P_{ss'} = P(S_{t+1} = s' \mid S_t = s)$$

✓The state transition matrix **P** defines the transition probabilities from all states s to all successor states s'

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

where each row of the matrix sums to 1 (marginalization)

Markov Process

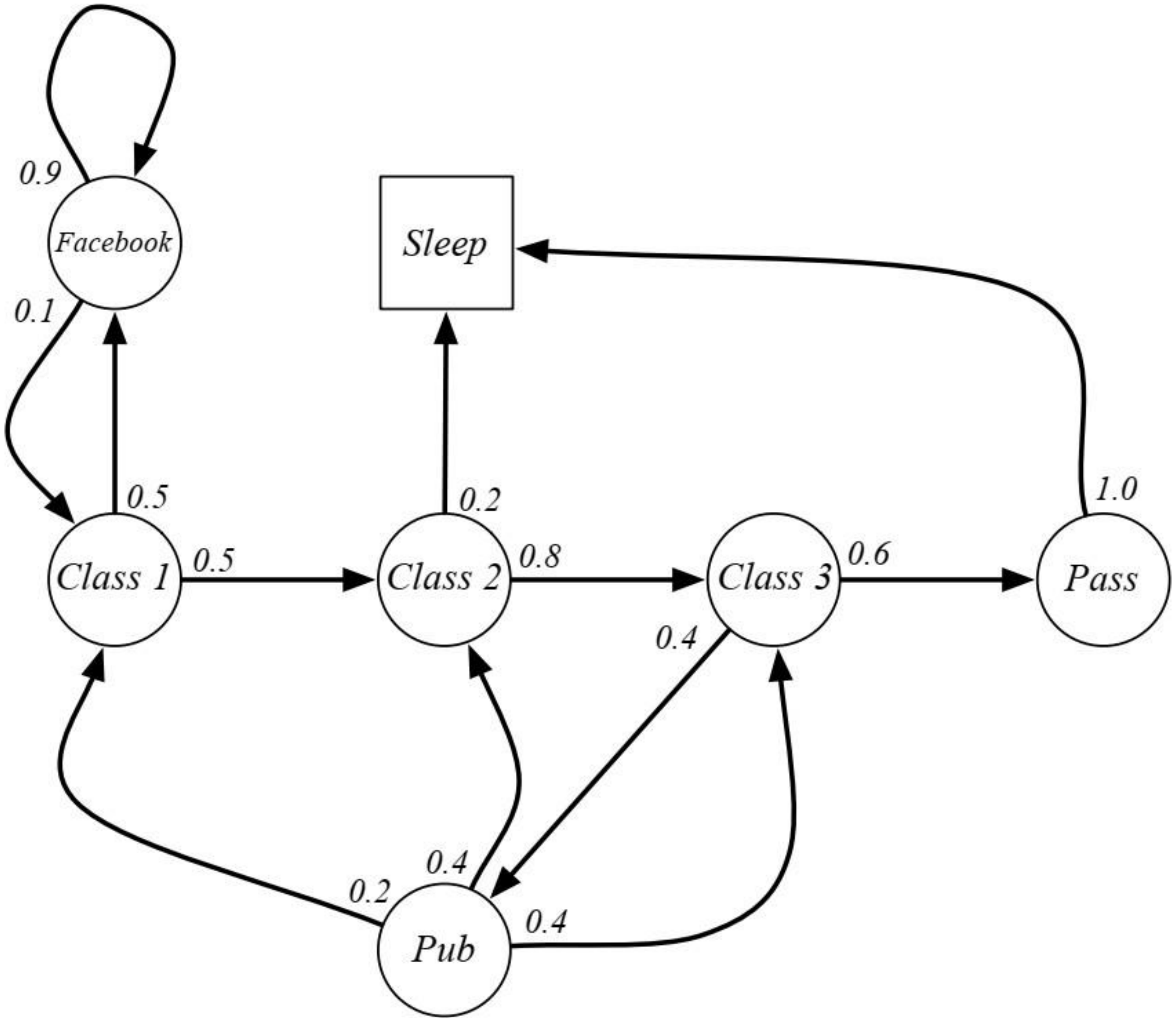
A Markov process is a memoryless random process, i.e. a sequence of random states $S_1, S_2, \ldots$ with the Markov property

Definition (Markov Process)

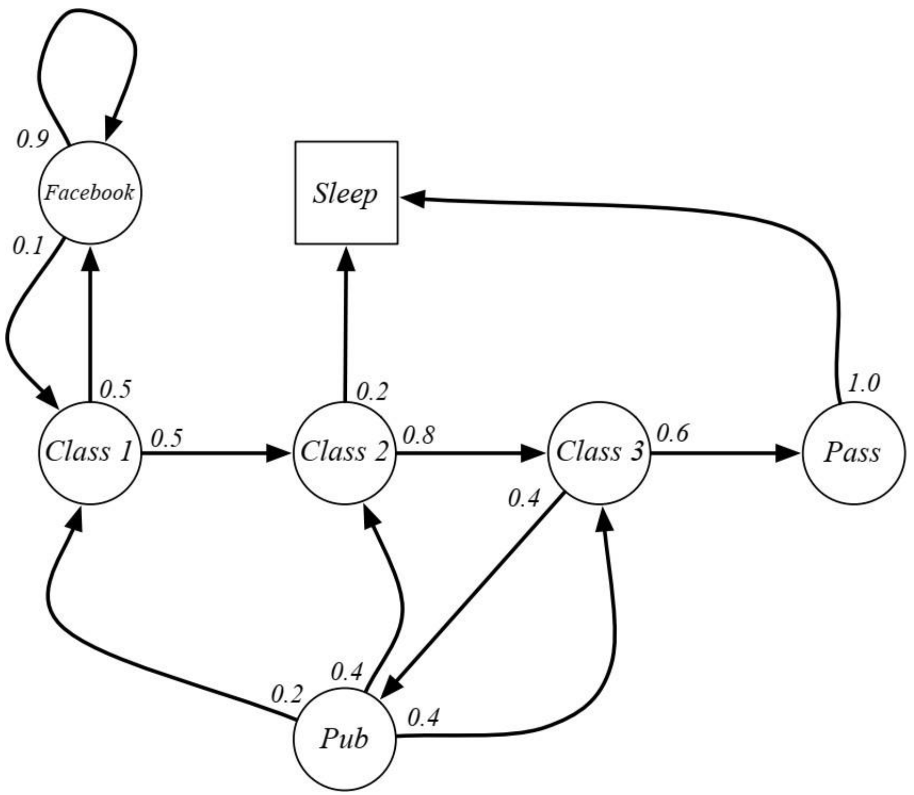
A Markov Process (or Markov Chain) is a tuple $\langle \mathcal{S}, \boldsymbol{P} \rangle$

- $\mathcal{S}$ is a finite set of states
- $\boldsymbol{P}$ is a state transition matrix, such that. $P_{ss'} = P(S_{t+1} = s' \mid S_t = s)$

Example – Student Markov Chain



Example – Student Markov Chain Episodes

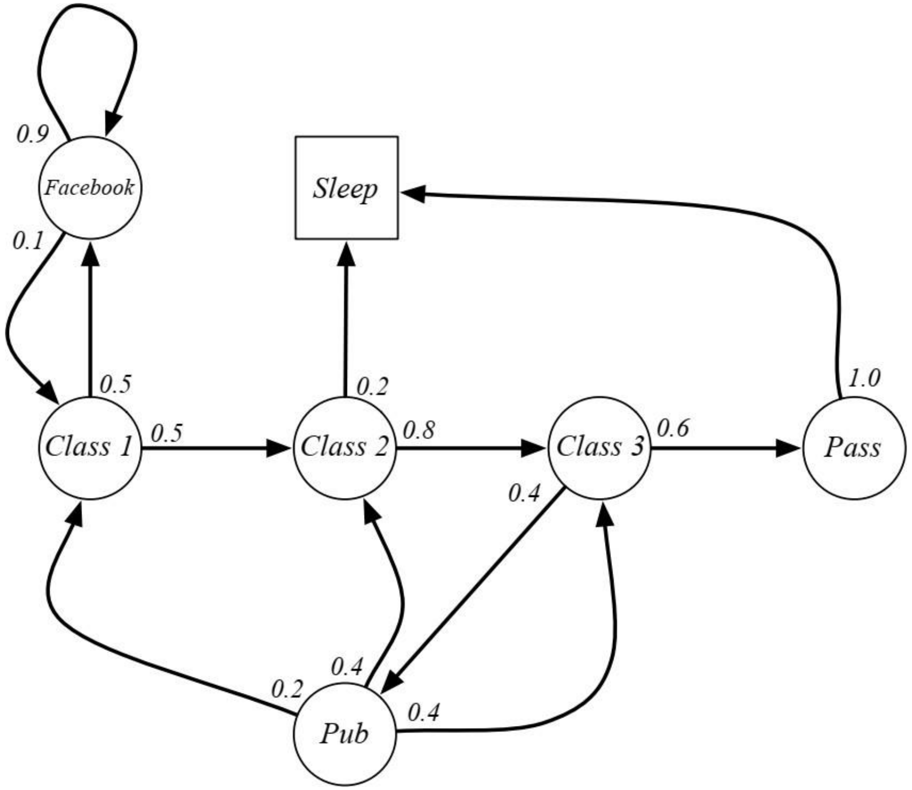


Sample episodes for Student Markov Chain starting from $S_1 = "Class\ 1"\ (C1)$

$$S_1, S_2, \dots, S_t$$

- ✓ 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



Future

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

Current

$$P_{ss'} = P(S_{t+1} = s' \mid S_t = s)$$

Markov Rewards

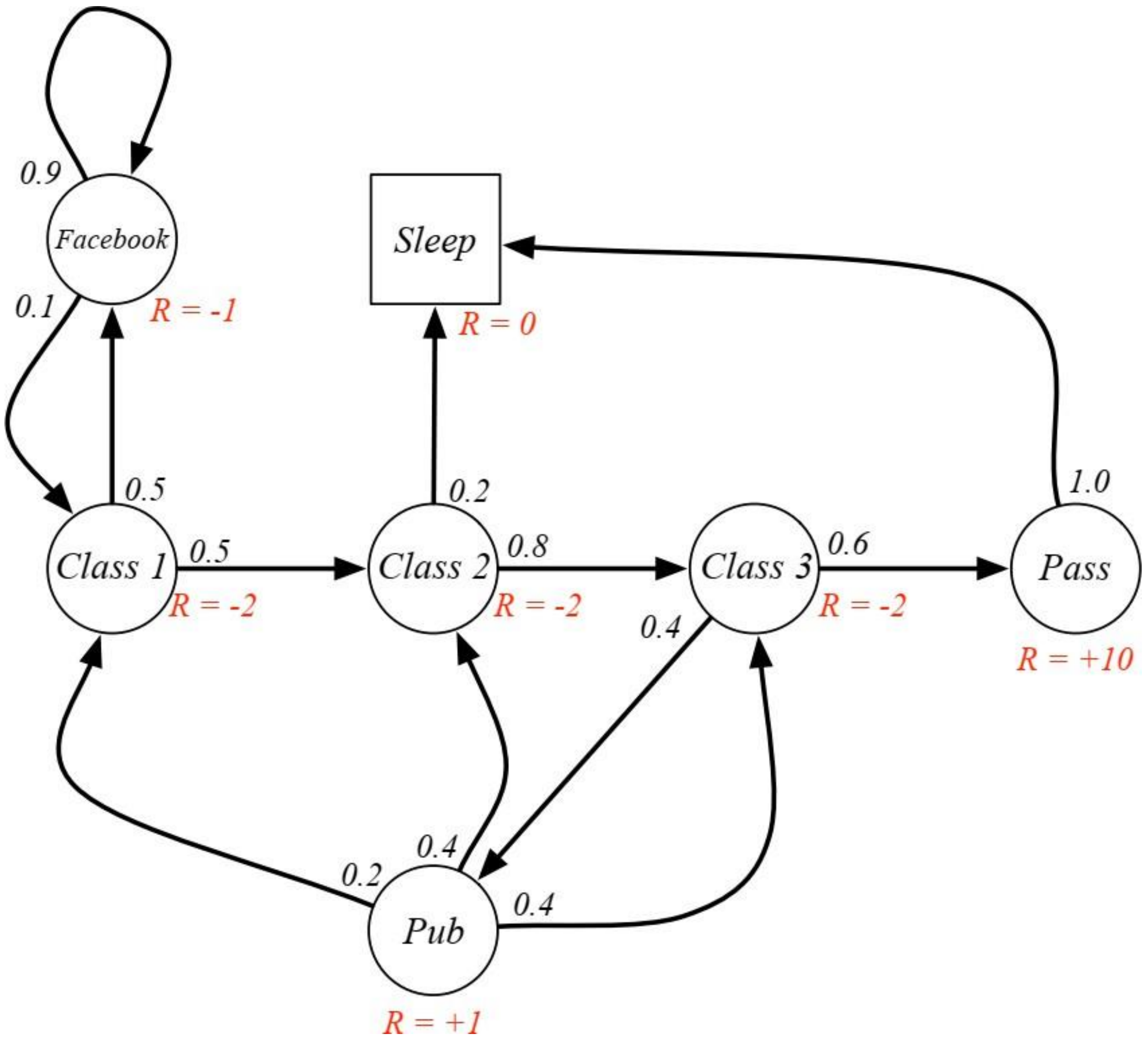
Markov Reward Process

A Markov Reward Process (MRP) is a Markov chain with reward values

Definition (Markov Reward Process)

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

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



Example – Student MRP

Return

Definition (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 value of receiving reward R after $k+1$ timesteps is $\gamma^k R$
- ✓ γ values immediate reward Vs delayed reward
  - ✓ $\gamma \approx 0$ leads to "myopic" evaluation
  - ✓ $\gamma \approx 1$ leads to "far-sighted" evaluation

On the discount term

- ✓ Mathematically convenient to discount rewards
- ✓ Avoids infinite returns in cyclic Markov processes
- ✓ Uncertainty about the future may not be fully represented
- ✓ Application dependent
  - ✓ If the reward is financial, immediate rewards may earn more interest than delayed rewards
  - ✓ Biological plausibility (animal 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

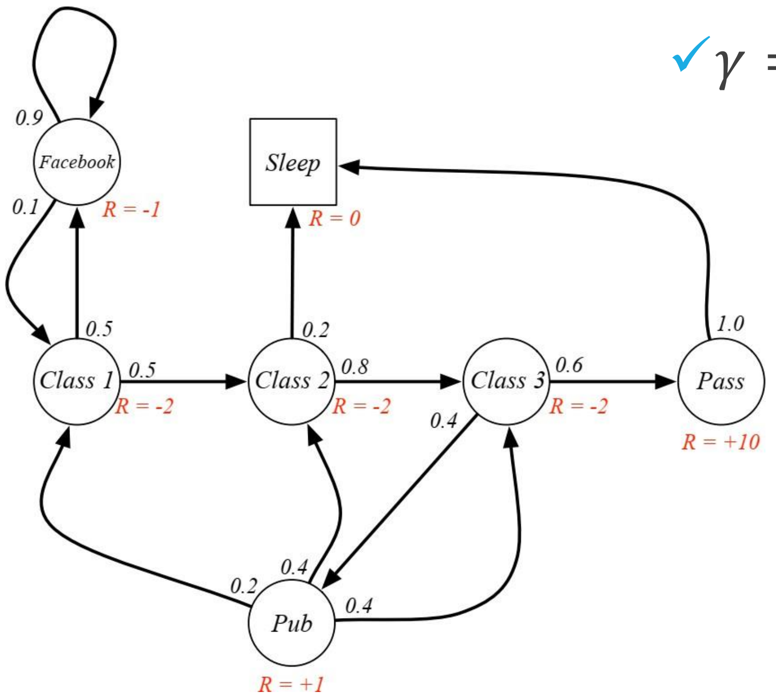
Measures the long-term value of being in a certain state s

Definition (Value Function)

The state-value function $v(s)$ of a Markov Reward Process is the expected return starting from state s

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

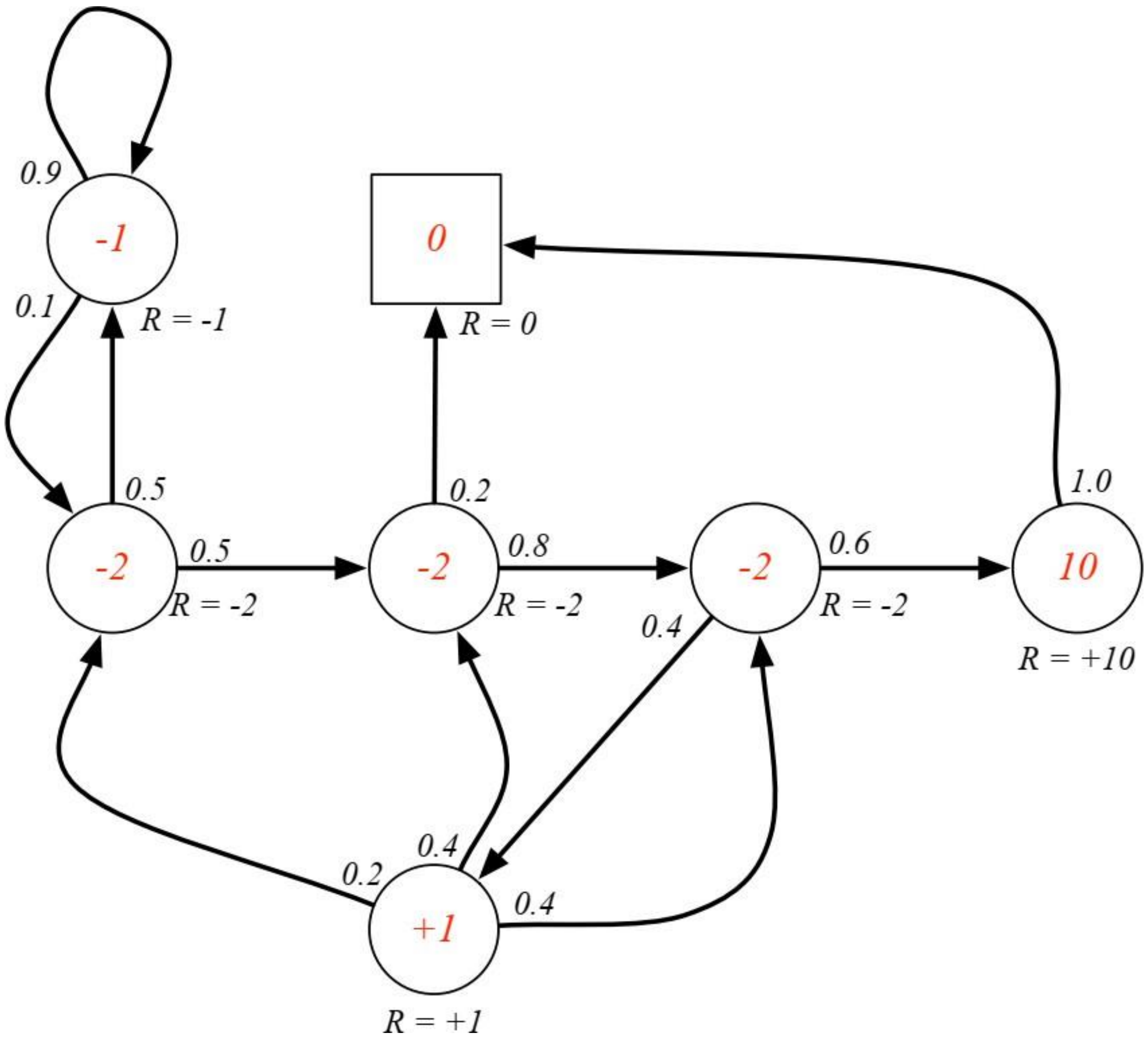
Example – Student MRP Returns



Sample returns for student Markov Reward Process

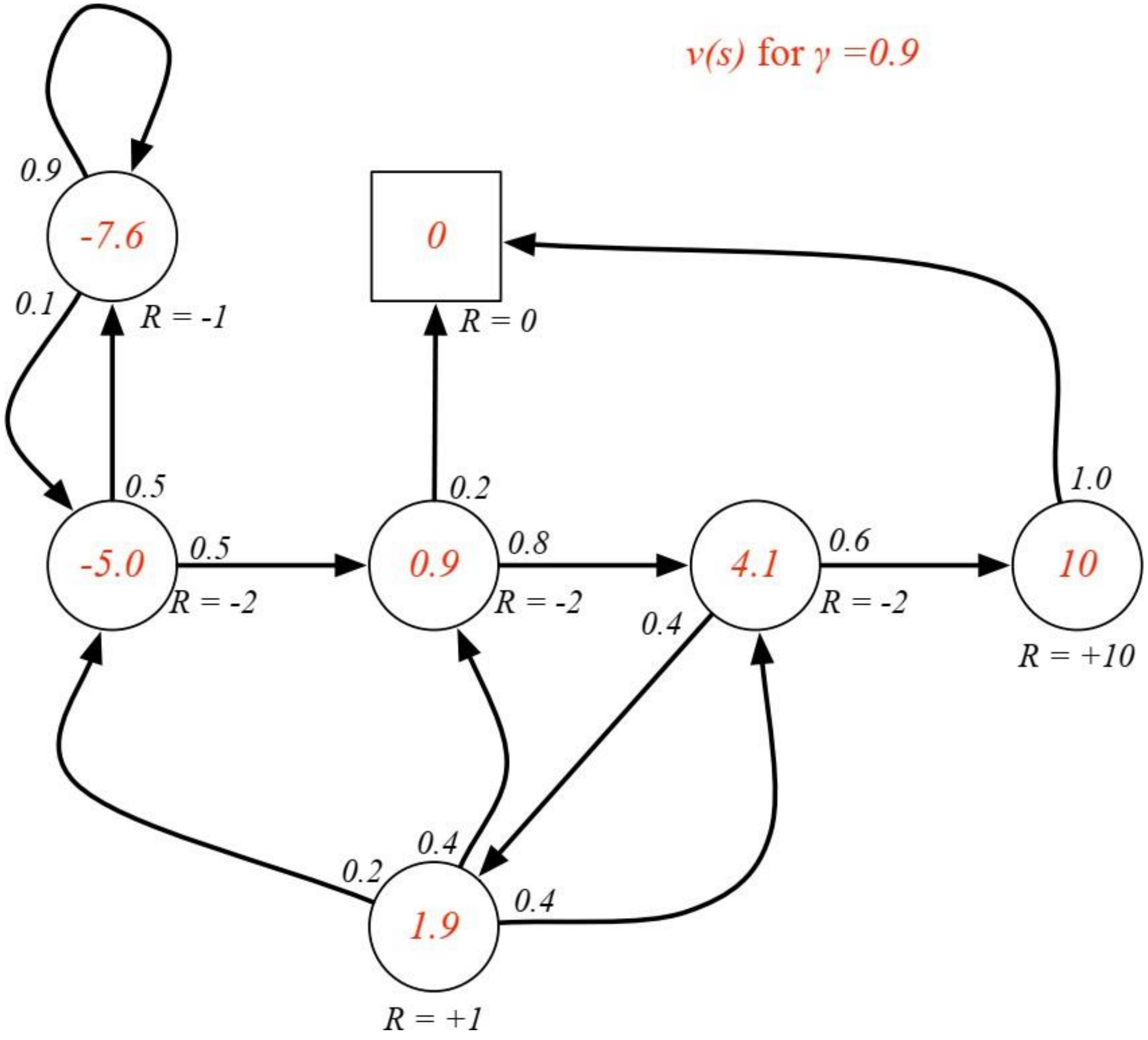
- Starting from $S_1 = C1$
- $\gamma = \frac{1}{2}$

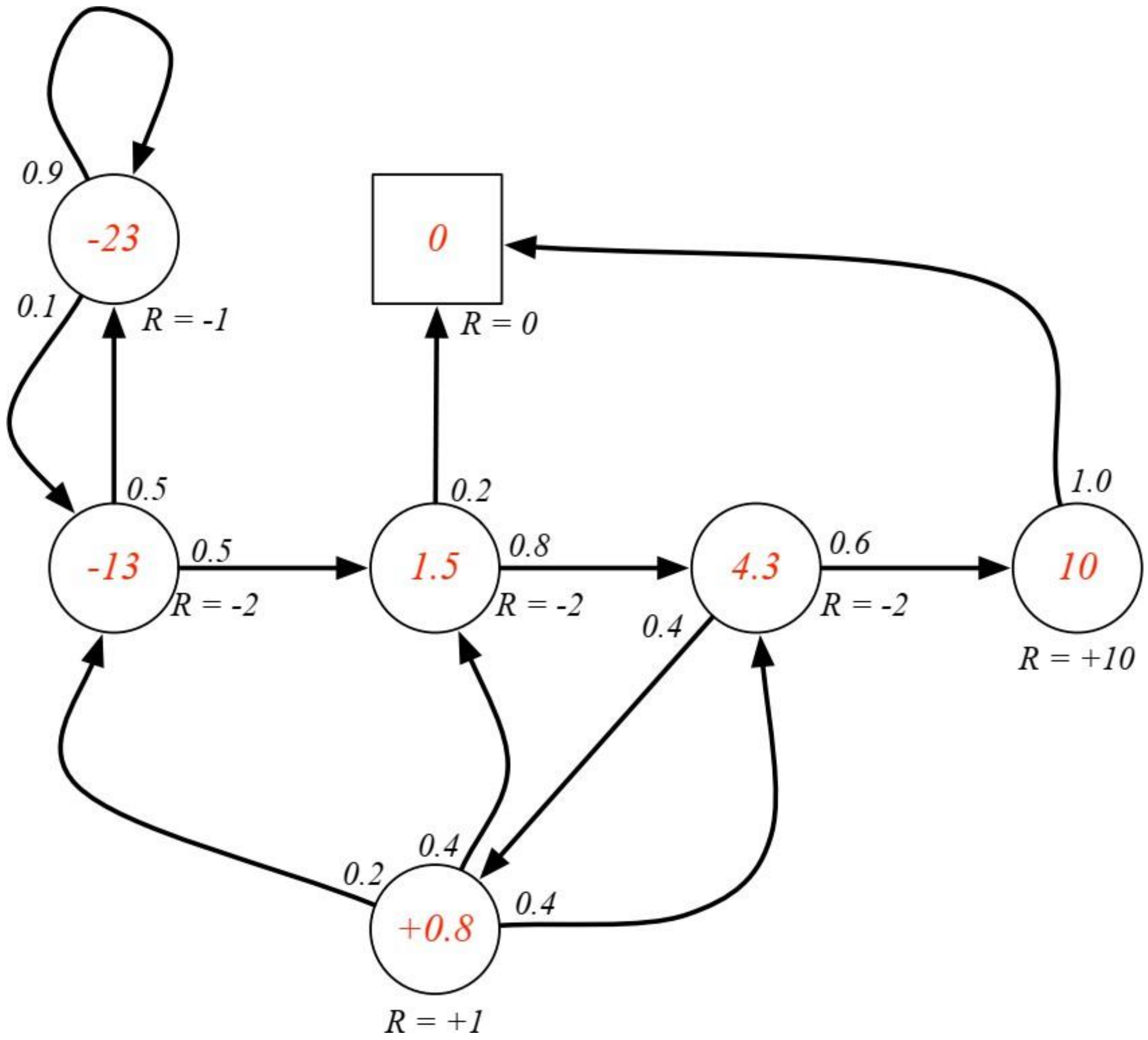
- C1 C2 C3 Pass Sleep
  - $v(C1) = -2 - 2 * \frac{1}{2} - 2 * \frac{1}{4} + 10 * \frac{1}{8} = -2.25$
- C1 FB FB C1 C2 Sleep
  - ?
- ...



Example – Student MRP State-Value Function ($\gamma = 0$)

Example – Student MRP State-Value Function ($\gamma = 0.9$)





Example – Student MRP State-Value Function ($\gamma = 1$)

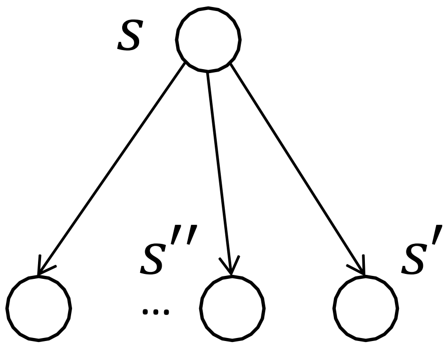
Bellman Equation for MRPs

- The value function $v(S_t)$ can be decomposed into two parts
  - Immediate reward R_{t+1}
  - Discounted value of successor state $\gamma v(S_{t+1})$

$$
\begin{aligned}
v(s) = \mathbb{E}[G_t | S_t = s] &= \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,|S_t \right] \\
&= \mathbb{E}[R_{t+1} + \textstyle\sum_{k=1}^{\infty} \gamma^k R_{t+k+1} \,|S_t] \\
&= \mathbb{E}\left[R_{t+1} + \gamma\left(R_{t+2} + \textstyle\sum_{k=2}^{\infty} \gamma^k R_{t+k+1}\right)|S_t\right] \\
&= \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t] \\
&= \mathbb{E}[R_{t+1} + \gamma v(S_{t+1}) | S_t]
\end{aligned}
$$

Bellman Equation for MRPs – Which future state?

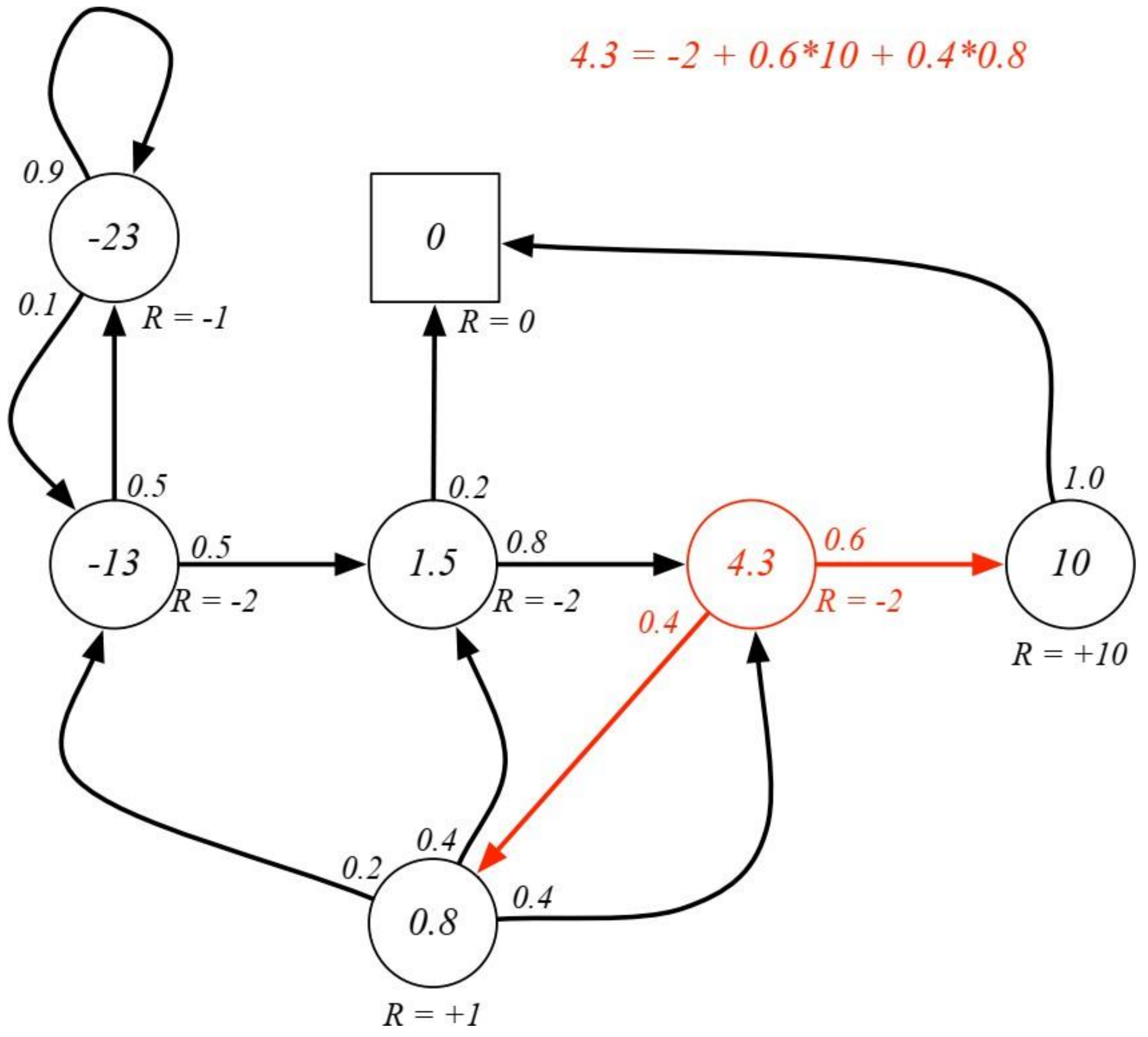
$$v(s) = \mathbb{E}[R_{t+1} + \gamma v(S_{t+1}) | S_t = s]$$



$$v(s) = \overbrace{\mathbb{E}[R_{t+1} | S_t = s]}^{\text{Reward function } \mathcal{R}_s} + \gamma \mathbb{E}[v(S_{t+1}) | S_t = s]$$

The expected state-value of being in any state reachable from s

$$v(s) = \mathcal{R}_s + \gamma \sum_{s'} P_{ss'} v(s')$$



Example – Bellman Equation for Student MRP (γ=1)

Bellman Equation – Matrix Form

Considering n available states

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

$$v = \begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix} \quad \mathcal{R} = \begin{bmatrix} \mathcal{R}_1 \\ \vdots \\ \mathcal{R}_n \end{bmatrix} \quad \boldsymbol{P} = \begin{bmatrix} P_{11} & \cdots & P_{1n} \\ \vdots & \ddots & \vdots \\ P_{n1} & \cdots & P_{nn} \end{bmatrix}$$

Provides us with a nice linear system

Solving the linear Bellman Equation

$$v = \mathcal{R} + \gamma \boldsymbol{P} v = (\boldsymbol{I} - \gamma \boldsymbol{P})^{-1} \mathcal{R}$$

- ✓ Computational complexity is $\mathrm{O}(\mathrm{n}^3)$
  - ✓ Direct solution only feasible for small MRPs
- ✓ Iterative methods for large MRPs
  - ✓ Dynamic programming
  - ✓ Monte-Carlo evaluation
  - ✓ Temporal-Difference learning

Markov Decision Processes

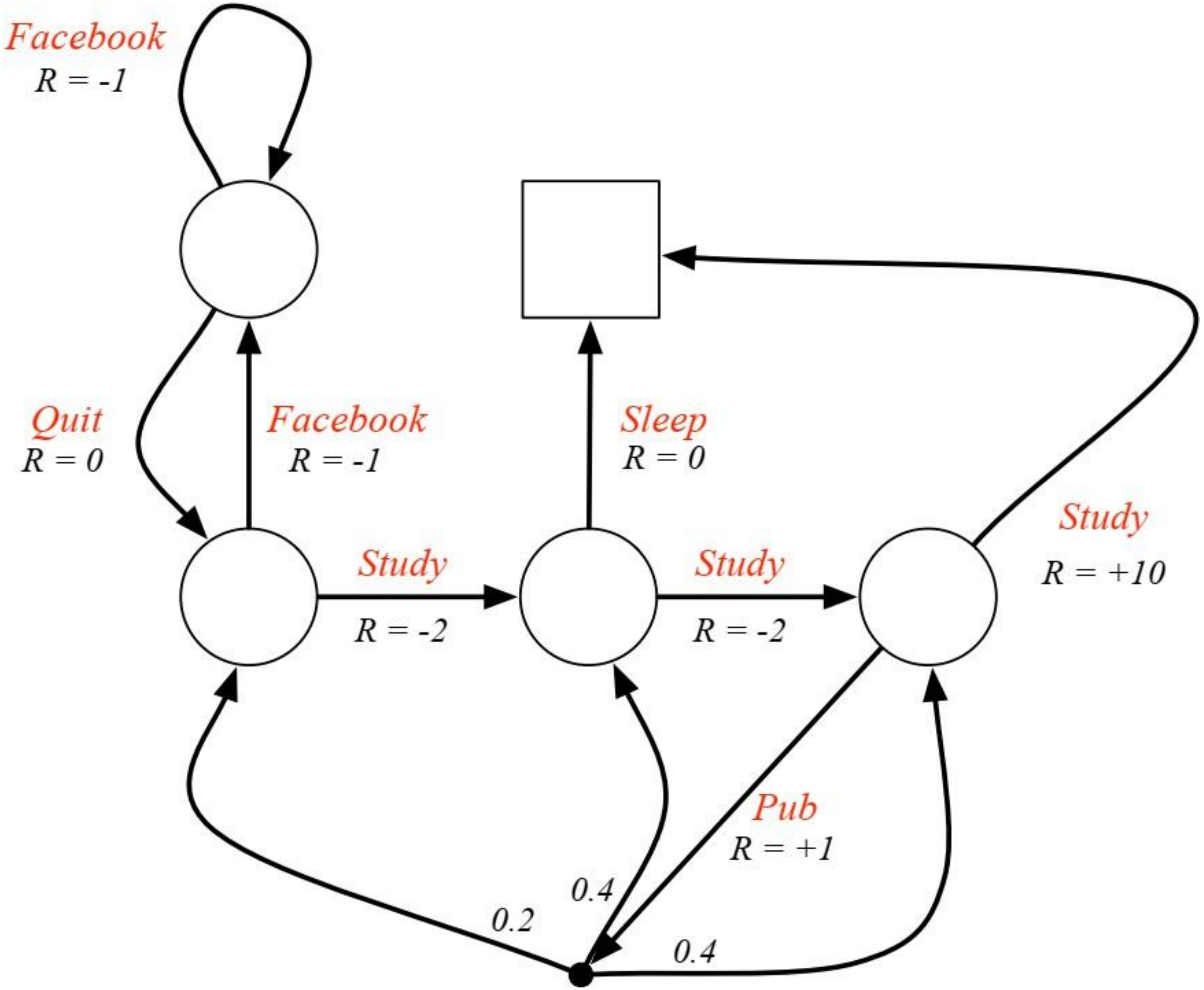
Markov Decision Process

A Markov Decision Process (MDP) is a Markov reward process with actions. It is an environment in which all states are Markov

Definition (Markov Decision Process)

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

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



Example – Student Markov Decision Process

Policy - Definition

Definition (Policy)

A policy π is a distribution over actions a given states s

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

- ✓ Define the behavior of an agent
- ✓ MDP policies depend only on the current state (Markovian)
- ✓ Policies are stationary (time-independent): $A_t \sim \pi(\cdot\,|s), \forall t > 0$

Under Policy

Given an MDP $\langle \mathcal{S}, \mathcal{A}, \boldsymbol{P}, \mathcal{R}, \gamma \rangle$ and a policy π

✓The state sequence $S_1, S_2, \ldots$ is a Markov process $\langle \mathcal{S}, \boldsymbol{P}^{\pi} \rangle$ (under policy)

✓The state and reward sequence $S_1, R_2, S_2 \ldots$ is a Markov reward process $\langle \mathcal{S}, \boldsymbol{P}^{\pi}, \mathcal{R}^{\boldsymbol{\pi}} \rangle$ (under policy), such that

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

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

Value Function (with policy)

Definition (Value Function)

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

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

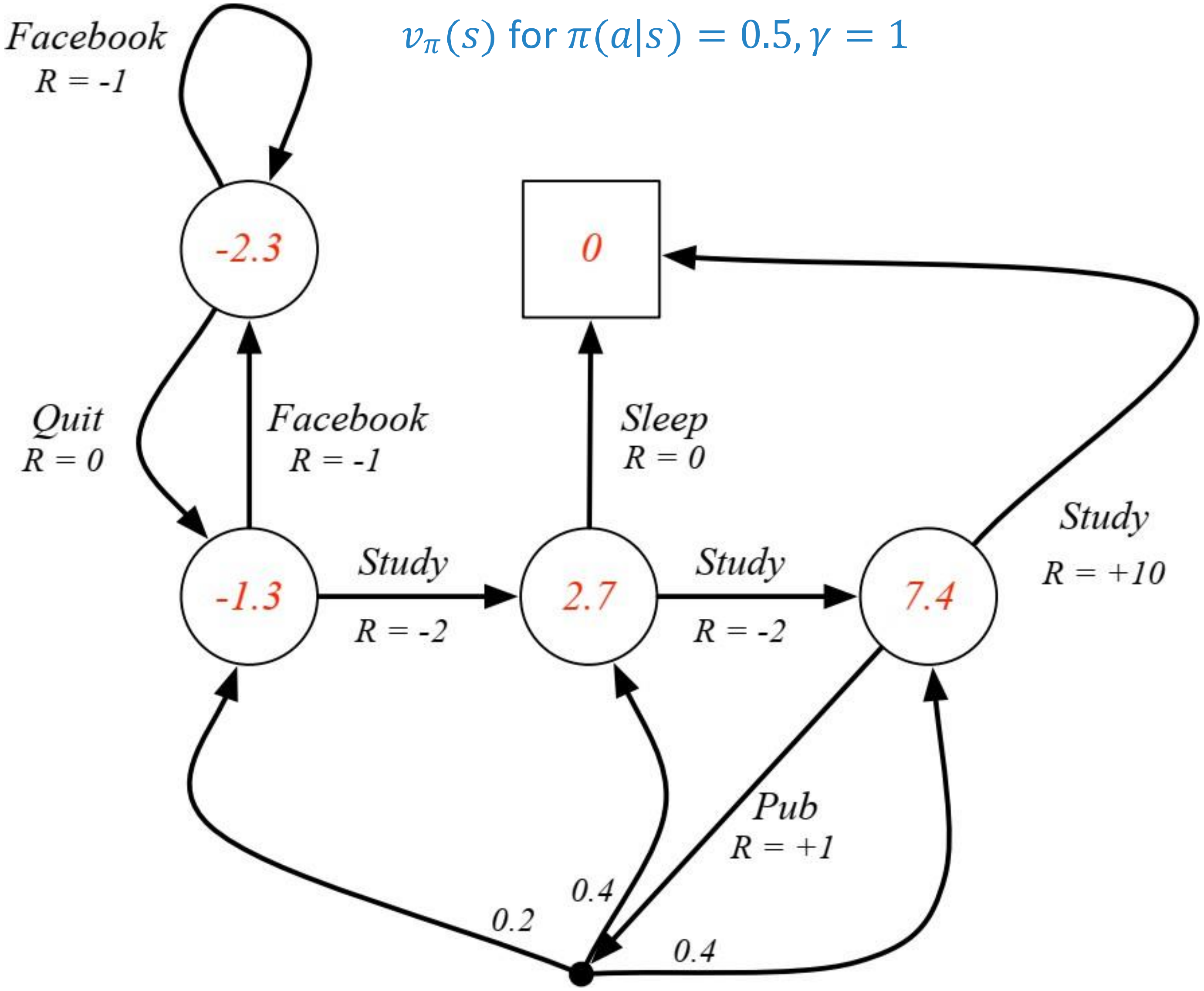
Definition (Action-Value Function)

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

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

Example – Student State-Value Function

$v_\pi(s)$ for $\pi(a|s) = 0.5, \gamma = 1$



Bellman Expectation Equation – Value and Action-Value Functions

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

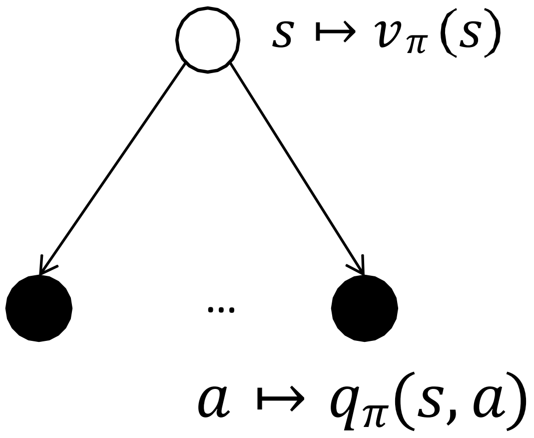
$$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s]$$

Similarly, we can decompose the action-value function

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

Both come from the recursive nature of return G_t

Bellman Expectation for v_π (I)

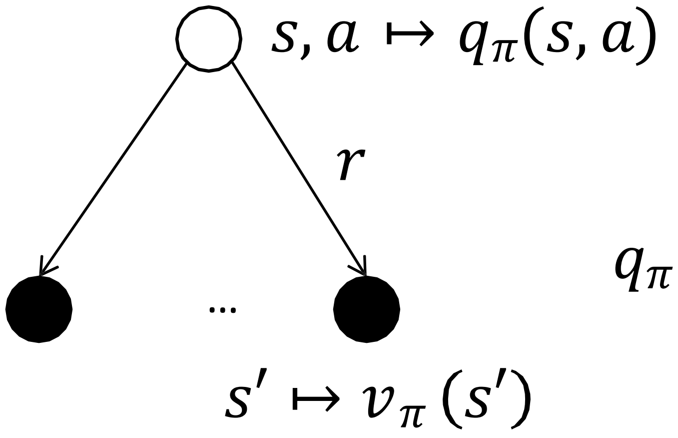


$$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s]$$

Expectation with respect to the actions that can be taken starting from s

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

Bellman Expectation for q_π (I)

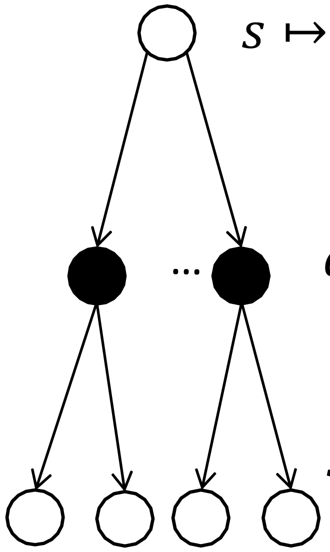


$$q_\pi(s,a) = \mathbb{E}_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a]$$

Expectation with respect to the states reachable from s having taken action a

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

Bellman Expectation for v_π (II)

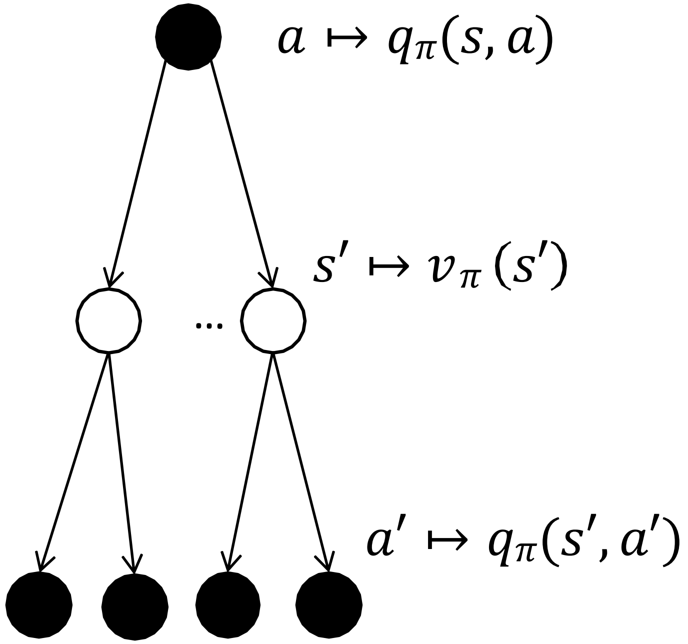


$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s,a)$$

The expected return of being in a state reachable from s through action a and then continue following policy

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

Bellman Expectation for q_π (II)



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

The expected return of any action a' taken from states reachable from s through action a (and then follow policy)

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

Bellman Expectation Equation – Matrix Form

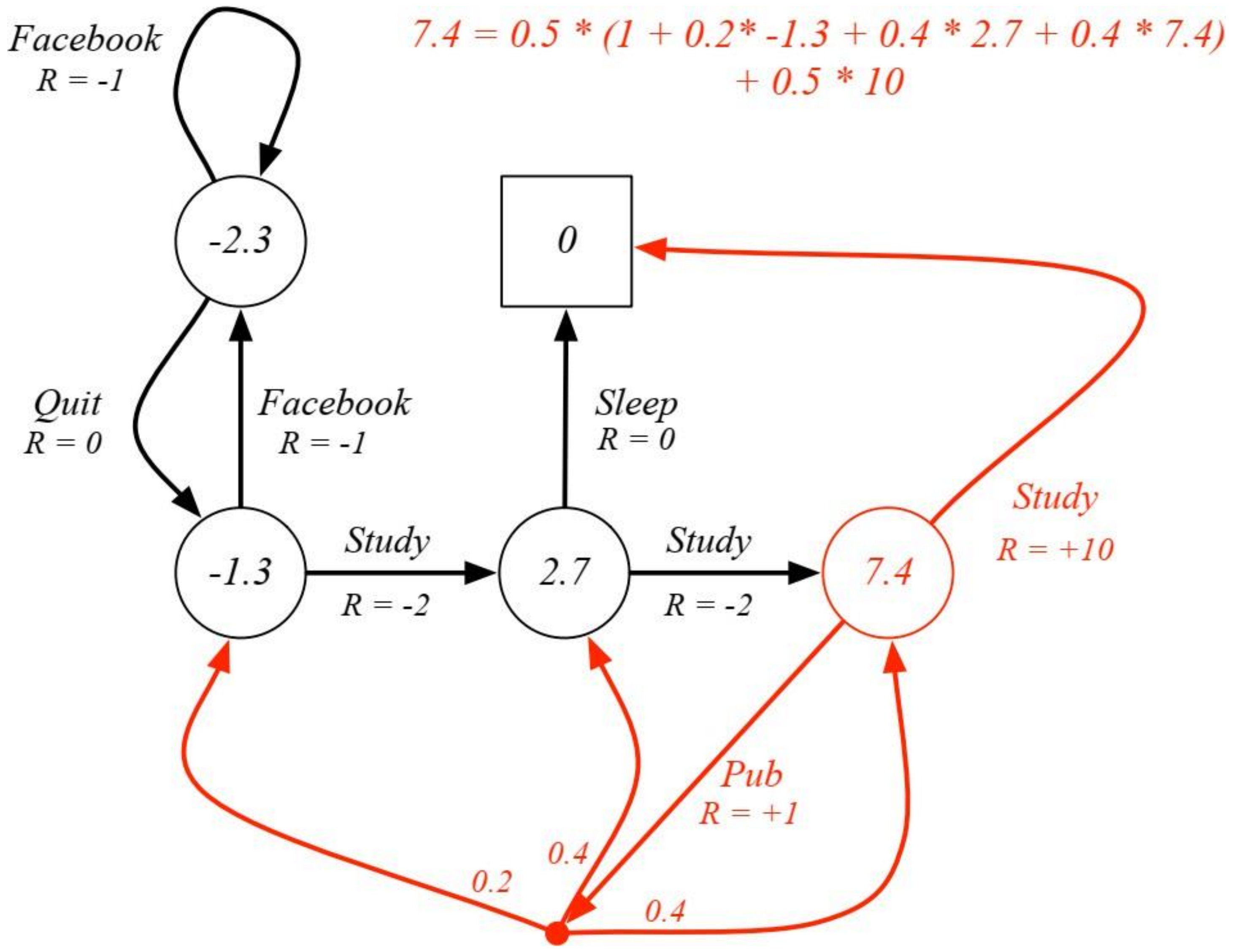
Again a linear system

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

With direct solution

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

Example – Bellman Expectation in Student MDP



Optimal Value Function

Definition (Optimal State/Action Functions)

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

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

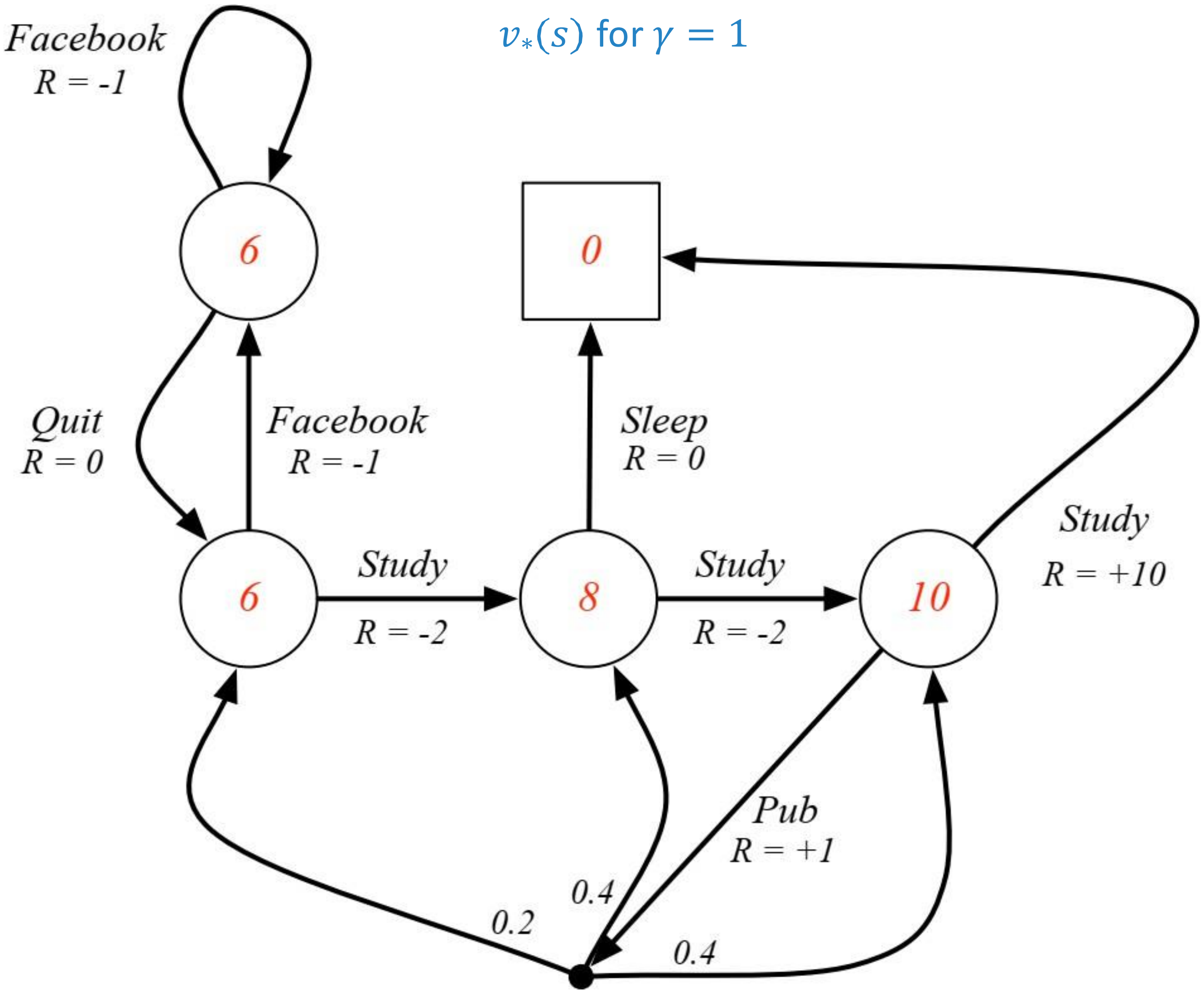
The optimal action-value function $q_*(s,a)$ is the maximum action-value function over all policies

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

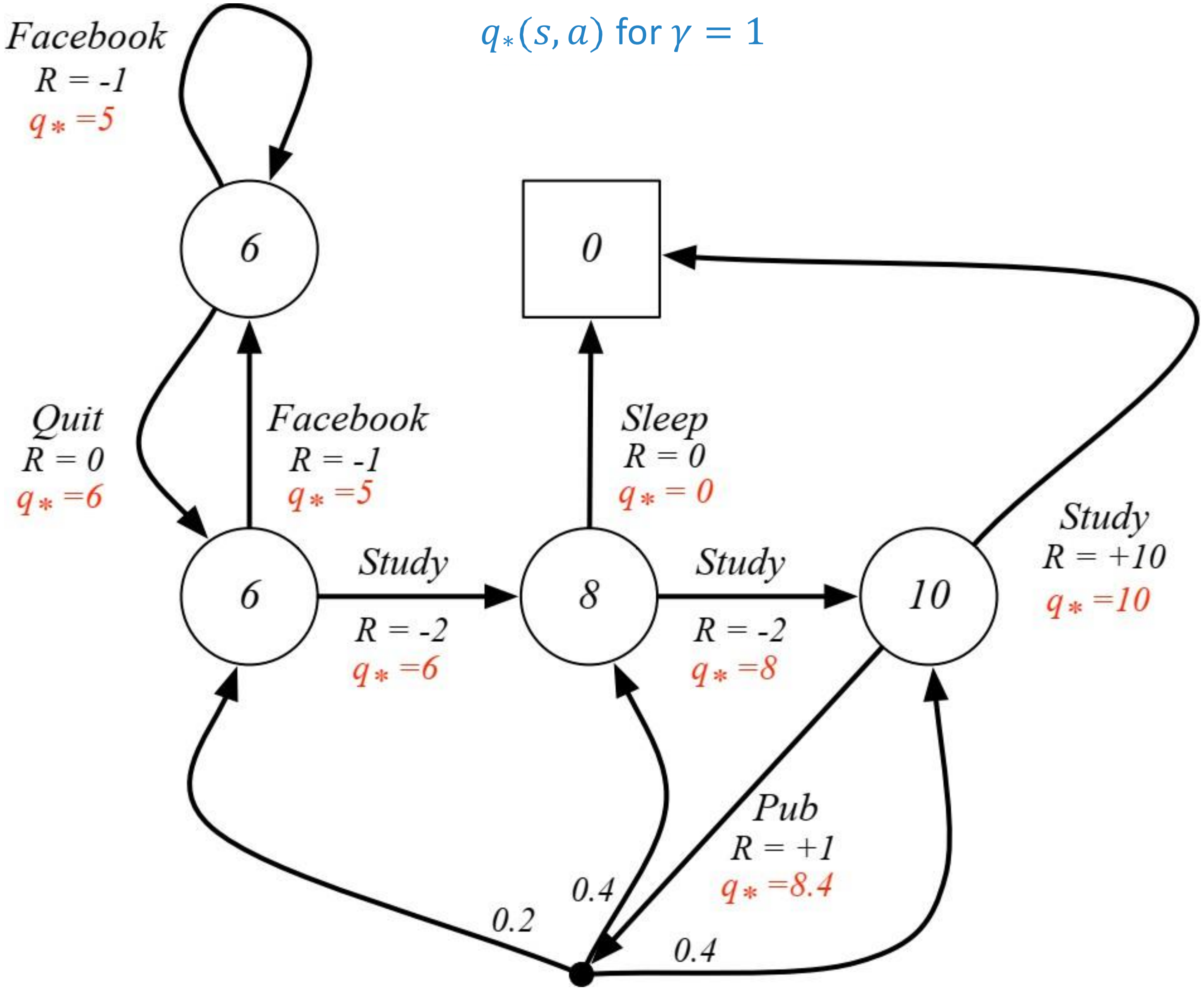
- ✓ The optimal value function determines the best possible performance in the MDP
- ✓ An MDP is solved when we know the optimal value function

Example – Optimal Value Function for Student MDP

$v_*(s)$ for $\gamma = 1$



Example – Optimal Action-Value Function for Student MDP



Optimal Policy

Define a partial ordering over policies

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

Theorem

For any Markov Decision Process

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

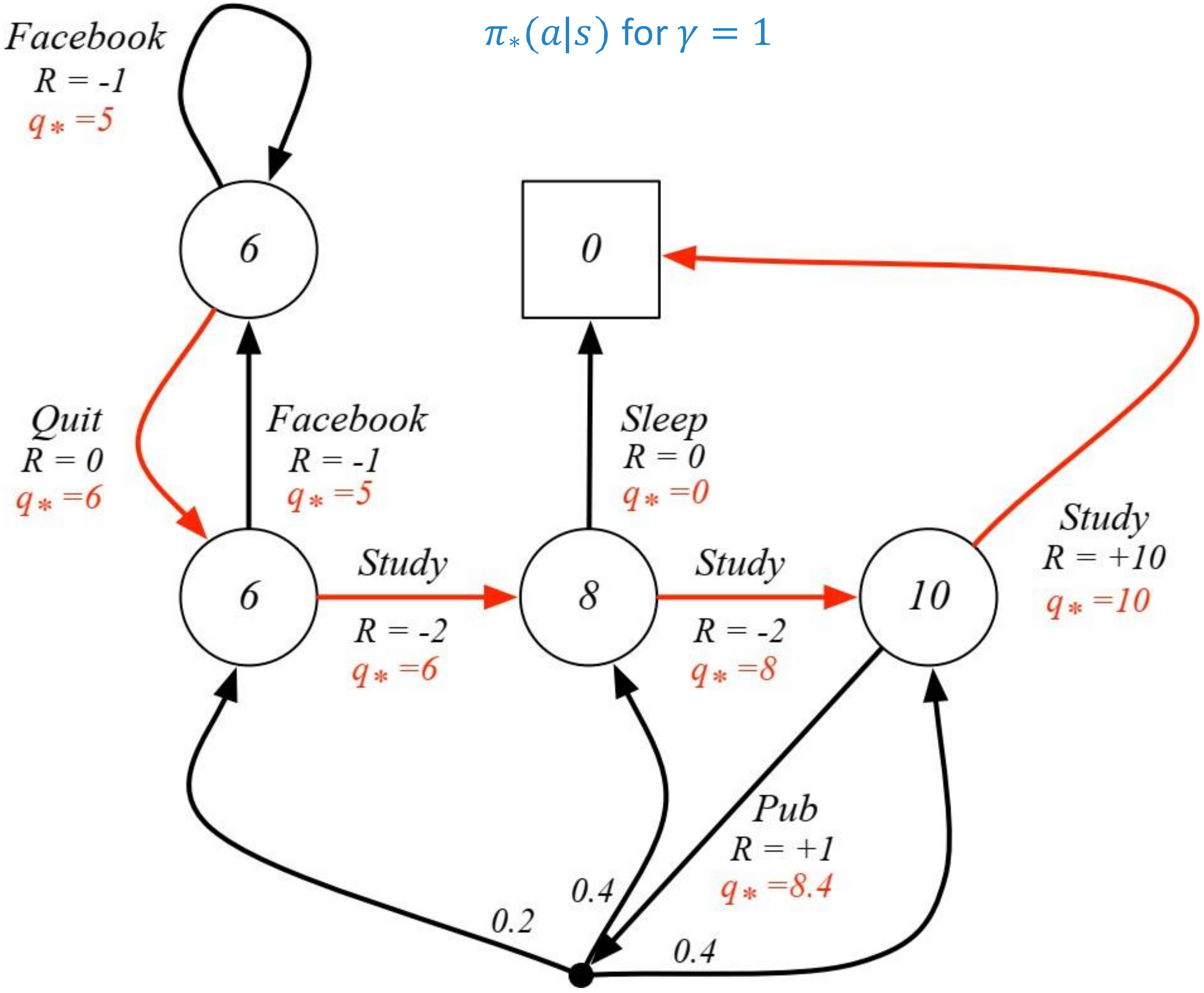
Finding an Optimal Policy

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

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

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

$\pi_*(a|s)$ for $\gamma = 1$



Example – Optimal Policy for Student MDP

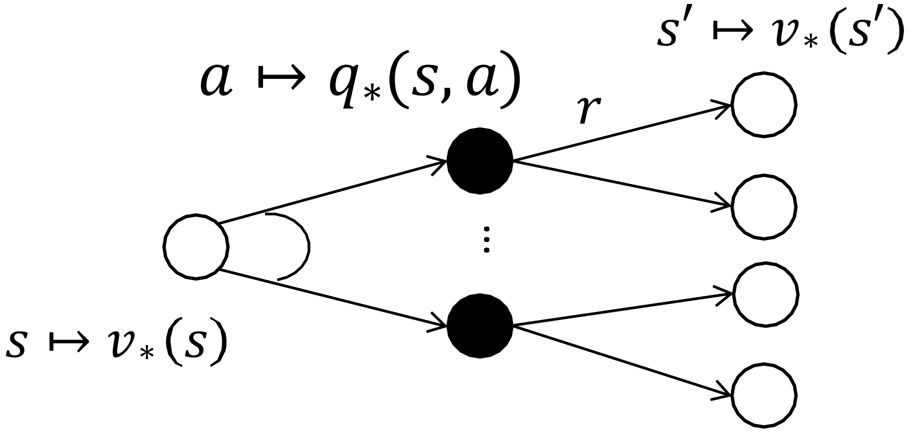
Bellman Optimality Equations

Optimal value functions are recursively related Bellman-style

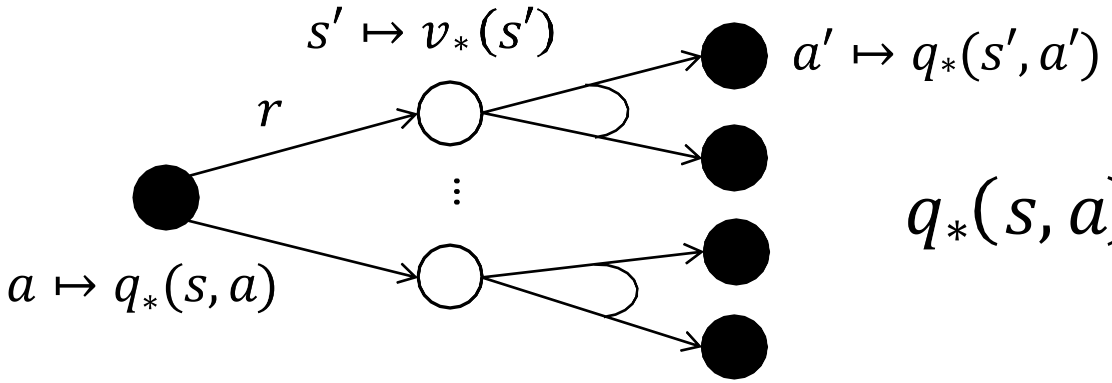
$$v_*(s) = \max_{a \in \mathcal{A}} q_*(s, a)$$

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

Bellman Optimality Equations - v_*, q_*

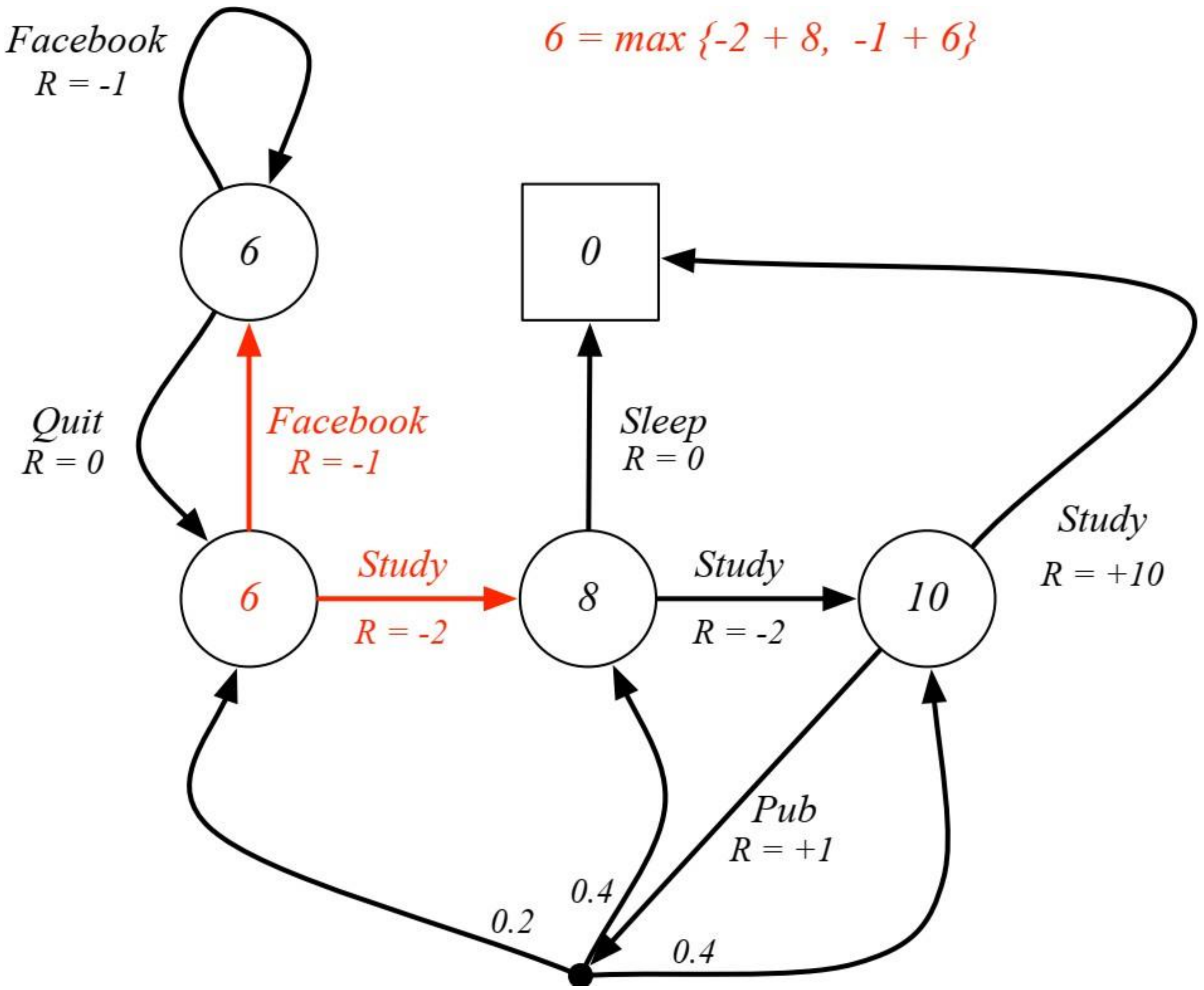


$$v_*(s) = \max_{a\in\mathcal{A}} \mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} P_{ss'}^a \, v_*(s')$$



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

Example – Bellman Optimality Equation



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

MDP Extensions

Partially Observable MDP (POMDP)

✓A Partially Observable Markov Decision Process is an MDP with hidden states

✓A Hidden Markov Model with actions

Definition (POMDP)

A POMDP is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \boldsymbol{P}, \mathcal{R}, \mathcal{Z}, \gamma \rangle$

- $\mathcal{S}$ is a finite set of states
- $\mathcal{A}$ is a finite set of actions a
- $\mathcal{O}$ is a finite set of observations
- $\boldsymbol{P}$ is a state transition matrix, s.t. $P_{ss'}^{a} = P(S_{t+1} = s' \mid S_t = s, A_t = a)$
- $\mathcal{R}$ is a reward function, s.t. $\mathcal{R}_s^a = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$
- $\mathcal{Z}$ is an observation function
- γ is a discount factor, $\gamma \in [0,1]$

Belief States

Definition (History)

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

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

Definition (Belief State)

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

$$b(h) = [P(S_t = s_1 | H_t = h), \dots, P(S_t = s_n | H_t = h)]$$

Wrap-up

Take home messages

- ✓ Markov decision processes are a formalism to describe a fully-observable environment for reinforcement learning
  - ✓ A state-transition system enriched with actions and reward
  - ✓ Leverage Markov assumption to separate future from the past
- ✓ A recursive formulation for value functions
  - ✓ Using Bellman equations
- ✓ Any MDP allows for an optimal policy
  - ✓ Maximisation process on the state-value function
  - ✓ Recursive and nonlinear (no closed form)
- ✓ MPD can be relaxed to infinite and continuous actions/state and partially observable environments (through belief instead of deterministic states)

Next Lecture

Planning by Dynamic Programming

- ✓ A.K.A. solving a known MDP
- ✓ Dynamic programming
  - ✓ A method for solving complex problems by breaking them down into subproblems
- ✓ Policy Evaluation & Iteration
- ✓ Value Evaluation