Reinforcement Learning - Lecture 2

Profa. Dra. Esther Luna Colombini
esther@ic.unicamp.br

Prof. Dr. Alexandre Simoes
alexandre.simoes@unesp.br

LaRoCS – Laboratory of Robotics and Cognitive Systems

UNICAMP

Advanced Institute for Artificial Intelligence

Summary

- Why MDPs?
- Markov Processes
- Markov Reward Processes
  - Discounted Rewards
- Markov Decision Processes
  - How the Markov Decision Process is defined

Disclaimer

- These slides were built upon David Silver's Lecture notes on Reinforcement Learning

Why MDPs?

- **Markov Decision Processes** formally describe an environment for reinforcement learning
- The environment is fully observable
  - The current state completely characterizes the process
- Almost all RL problems can be formalized as MDPs
  - Optimal control primarily deals with continuous MDPs
  - Partially observable problems can be converted into MDPs
  - Bandits are MDPs with one state

Markov Property

- The **Markov property** refers to the **memoryless** property of a stochastic process
  - A stochastic process has the Markov property if
    - the conditional probability distribution $\mathbb{P}[S_{t+1}|S_t]$ of future states S_{t+1} of the process (conditional on both past and present states) depends only upon the **present state,** not on the sequence of events that preceded it
  - The future is independent of the past given the present
  - 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 statistic of the future
  - **Stationary Assumption**
    - Transition probabilities are independent of time (t)
      - What if they are not?
  - **A state S_t is Markov** if and only if

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

State Transition Matrix

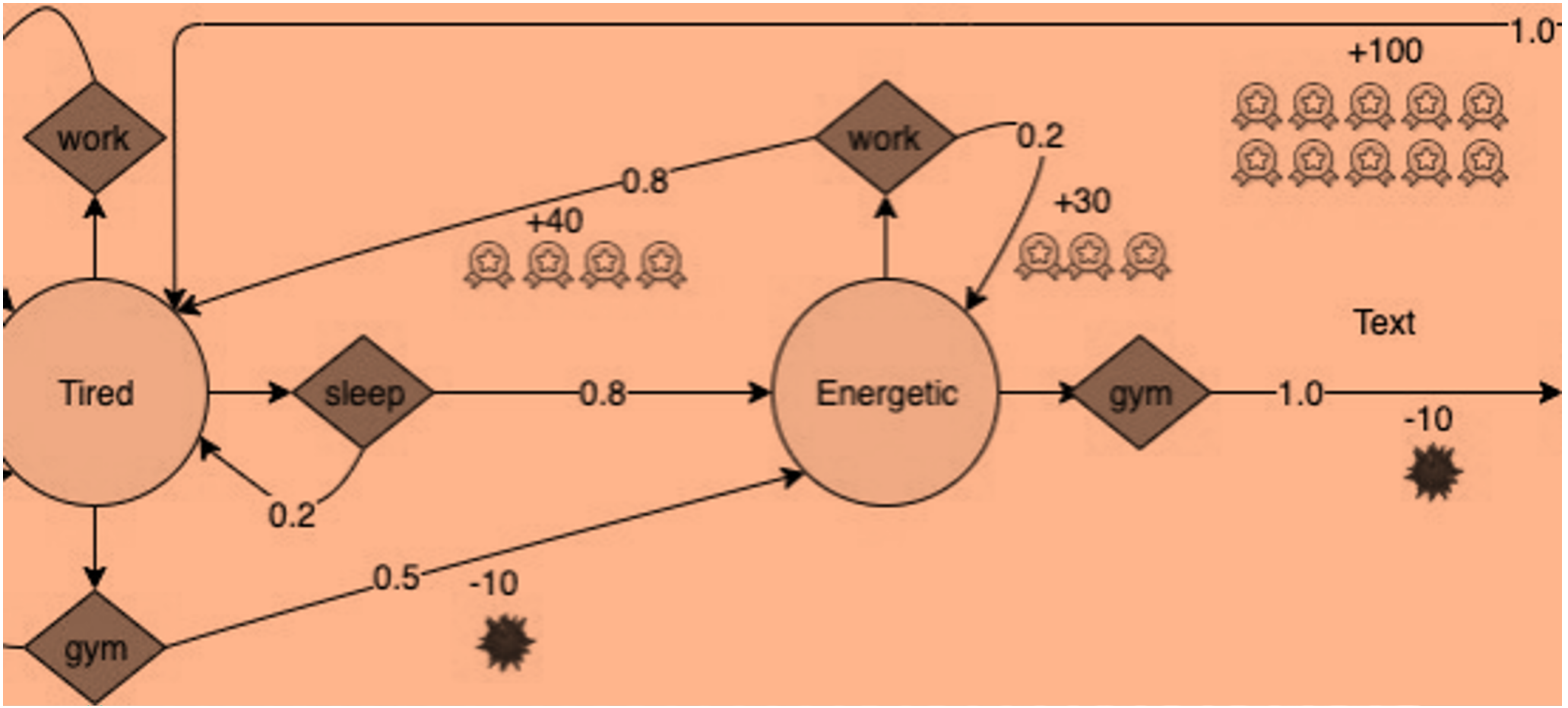
- For a Markov state s and successor state s', the **state transition probability** is defined by

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

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

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

With $\sum_{i=1}^{n} \mathcal{P}_{mi} = 1$, for all m in [1,n]



Markov Processes

Markov Process

- A Markov process is a memoryless **random process**, i.e. a sequence of random states $S_1, S_2, \ldots$ that are Markov

- A **Markov Process** (or Markov Chain) is a tuple $\langle S, \mathcal{P} \rangle$
  - S is a finite set of states
  - $\mathcal{P}$ s a state transition probability matrix

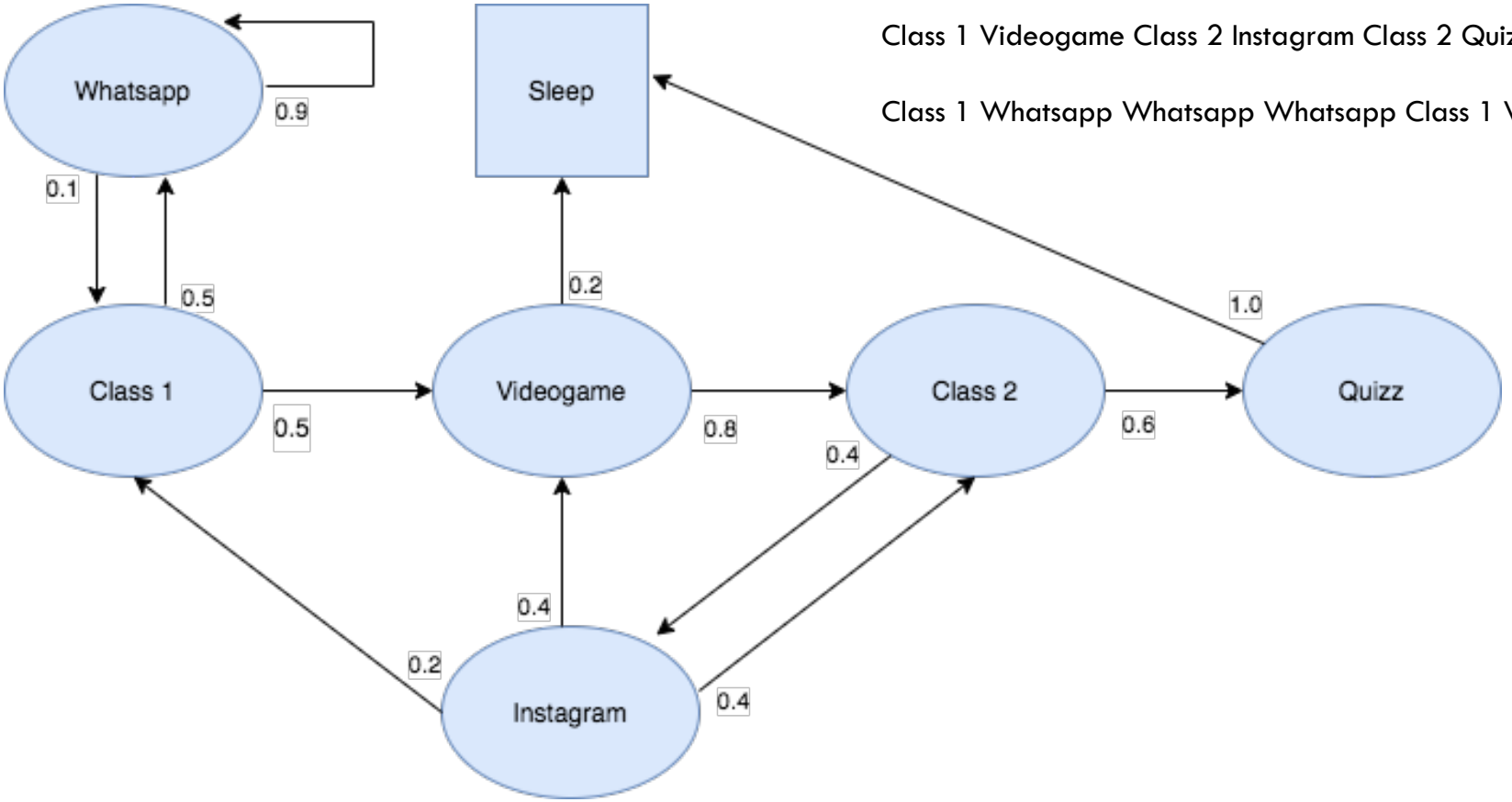
$$\mathcal{P}_{ss'} = \mathbb{P}[S_{t+1} = s' | S_t = s]$$

- Let's sample episodes from the Markov Chain starting from S_1 = Class 1

Class 1 Videogame Class 2 Quizz Sleep

Class 1 Videogame Class 2 Instagram Class 2 Quizz Sleep

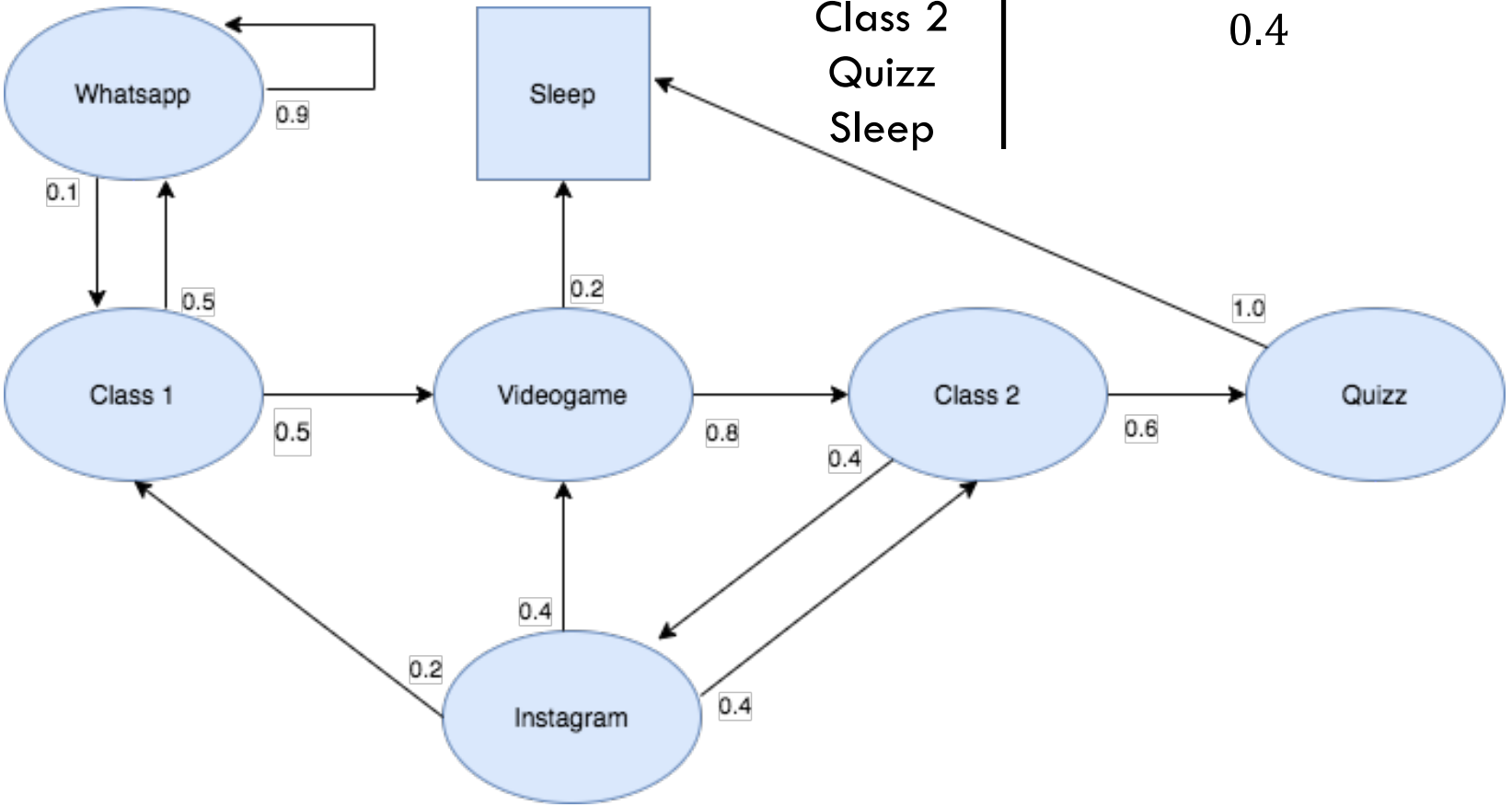
Class 1 Whatsapp Whatsapp Whatsapp Class 1 Videogame Sleep



Markov Process

Studying at home Markov Chain

$$\mathcal{P} = \begin{array}{c} \\ \text{Whatsapp} \\ \text{Class 1} \\ \text{Instagram} \\ \text{Videogame} \\ \text{Class 2} \\ \text{Quizz} \\ \text{Sleep} \end{array} \begin{array}{c} \begin{array}{ccccccc} \text{Whatsapp} & \text{Class 1} & \text{Instagram} & \text{Videogame} & \text{Class 2} & \text{Quizz} & \text{Sleep} \end{array} \\ \begin{bmatrix} 0.9 & 0.1 & & & & & \\ 0.5 & & & 0.5 & & & \\ & 0.2 & & 0.4 & 0.4 & & \\ & & & & 0.8 & & 0.2 \\ & & 0.4 & & & 0.6 & \\ & & & & & & 1 \\ & & & & & & \end{bmatrix} \end{array}$$



Markov Reward Process

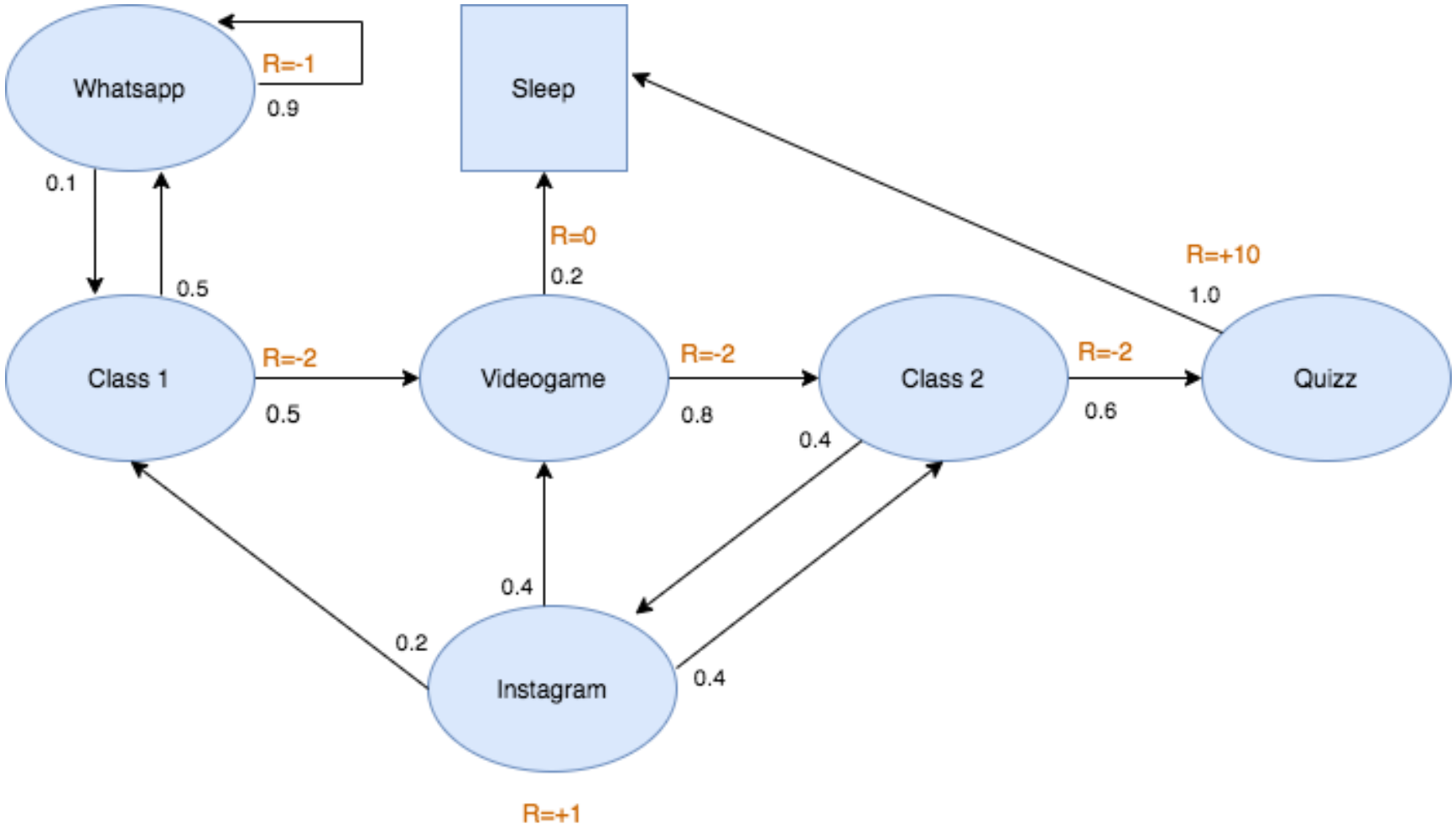
- A Markov reward process is a Markov chain with values
- A **Markov Reward Process** (or MRP) is a tuple $\langle S, \mathcal{P}, \mathcal{R}, \gamma \rangle$
  - S is a finite set of states
  - $\mathcal{P}$ s a state transition probability matrix

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

  - $\mathcal{R}$ is a Reward function $\mathcal{R}_S = \mathbb{E}[R_{t+1} | S_t = s]$
  - γ is the discount factor, $\gamma \in [0,1]$
- No actions

Markov Reward Process

Studying at home Markov Reward Process



- The **return** G_t is discounted sum of rewards from time step t to horizon

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

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

Return - why the discount?

- Most Markov reward and decision processes are discounted because:
  - 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
  - If episode lengths are always finite, can use $\gamma = 1$

The Value Function

- The state **Value Function** $v(s)$ is the long-term quality of state s, i.e., is the expected return starting from state s

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

The Value Function

- Let's compute the stata value for each sample drawn from our **MRP**
- Start from S_1 = Class 1 and consider $\gamma = 1/2$
  - $G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

  - Class 1 Videogame Class 2 Quizz Sleep
    - $v_1 = -2 + \left(\frac{1}{2} * -2\right) + \left(\frac{1}{4} * -2\right) + \left(\frac{1}{8} * 10\right)$=-2.25
  - Class 1 Videogame Class 2 Instagram Class 2 Quizz Sleep
    - $v_1 = -2 + \left(\frac{1}{2} * -2\right) + \left(\frac{1}{4} * 1\right) + \left(\frac{1}{8} * -2\right) + \left(\frac{1}{16} * -2\right) + \left(\frac{1}{32} * 10\right)$=-2.8125
  - Class 1 Whatsapp Whatsapp Whatsapp Class 1 Videogame Sleep
    - $v_1 = -2 + \left(\frac{1}{2} * -1\right) + \left(\frac{1}{4} * -1\right) + \left(\frac{1}{8} * -1\right) + \left(\frac{1}{16} * -2\right) + \left(\frac{1}{32} * 0\right)$=-3

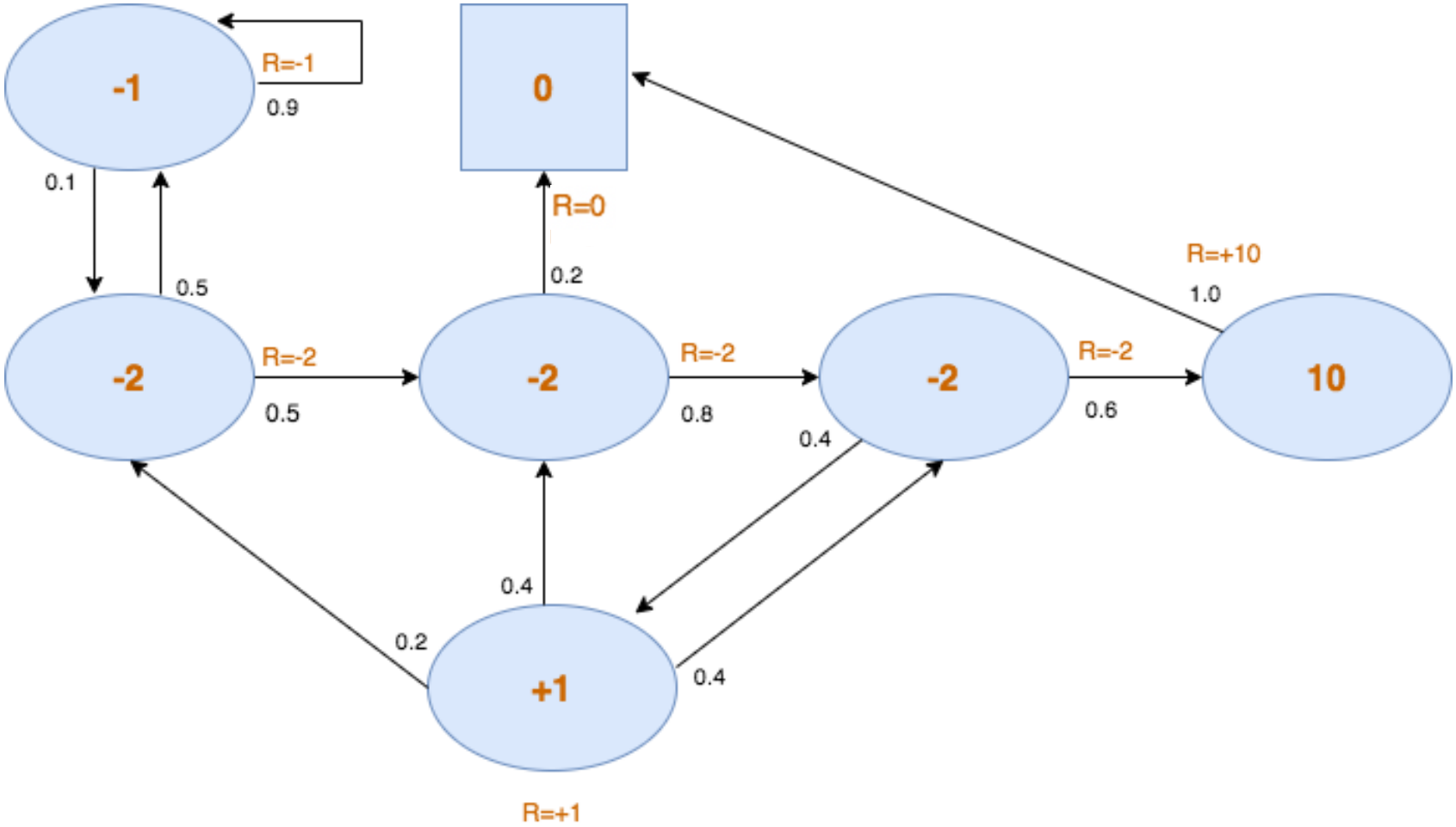
The Value Function

- Do you think we have defined a good reward function?
  - Class 1 Videogame Class 2 Quizz Sleep
    - $v_1 = -2 + \left(\frac{1}{2} * -2\right) + \left(\frac{1}{4} * -2\right) + \left(\frac{1}{8} * 10\right)$=-2.25
  - Class 1 Videogame Class 2 Instagram Class 2 Quizz Sleep
    - $v_1 = -2 + \left(\frac{1}{2} * -2\right) + \left(\frac{1}{4} * 1\right) + \left(\frac{1}{8} * -2\right) + \left(\frac{1}{16} * -2\right) + \left(\frac{1}{32} * 10\right)$=-2.8125
  - Class 1 Whatsapp Whatsapp Whatsapp Class 1 Videogame Sleep
    - $v_1 = -2 + \left(\frac{1}{2} * -1\right) + \left(\frac{1}{4} * -1\right) + \left(\frac{1}{8} * -1\right) + \left(\frac{1}{16} * -2\right) + \left(\frac{1}{32} * 0\right)$=-3

State-Value Function for Student MRP

Studying at home MRP

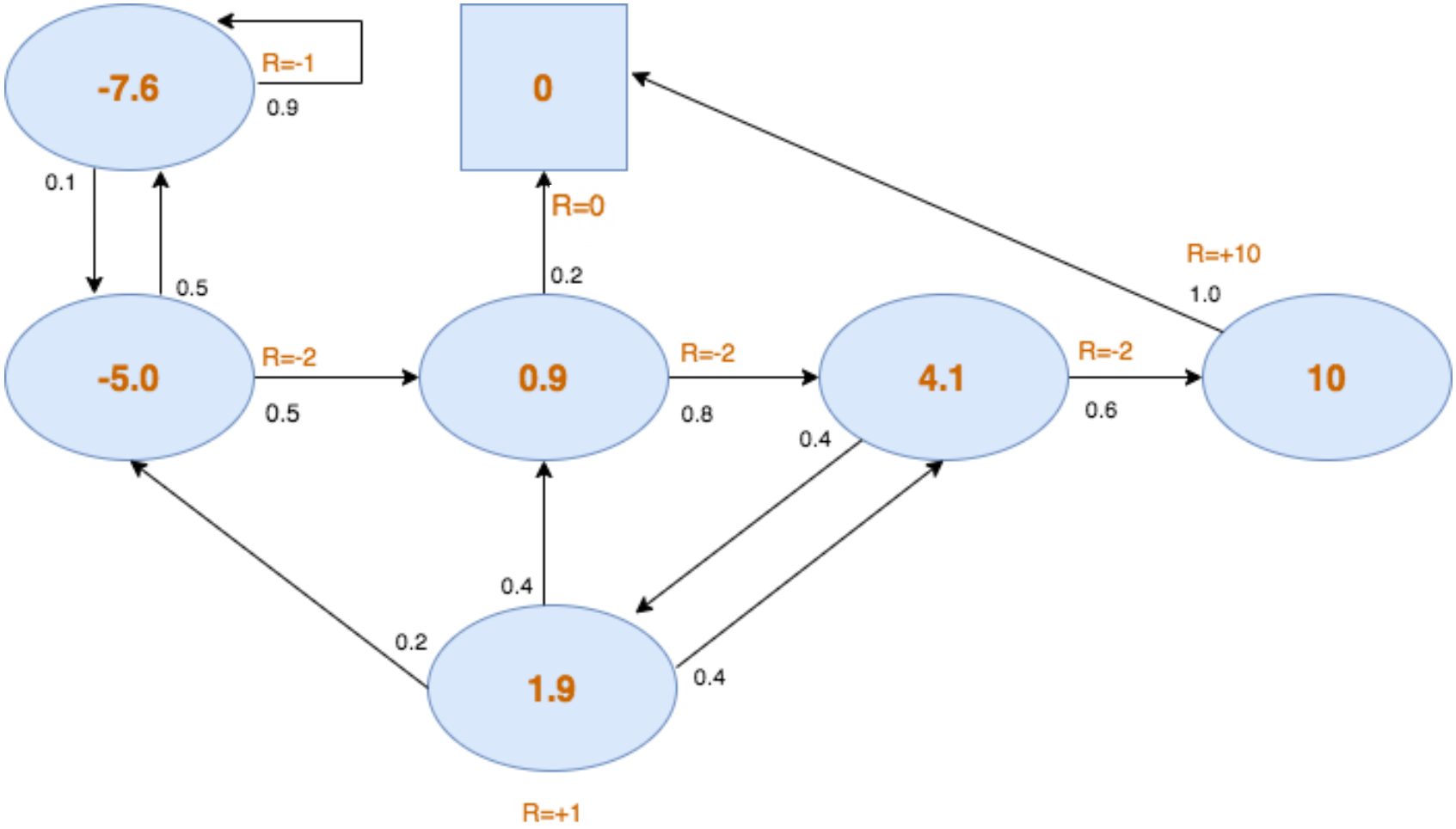
$v(s)$ for $\gamma = 0$



State-Value Function for Student MRP

Studying at home MRP

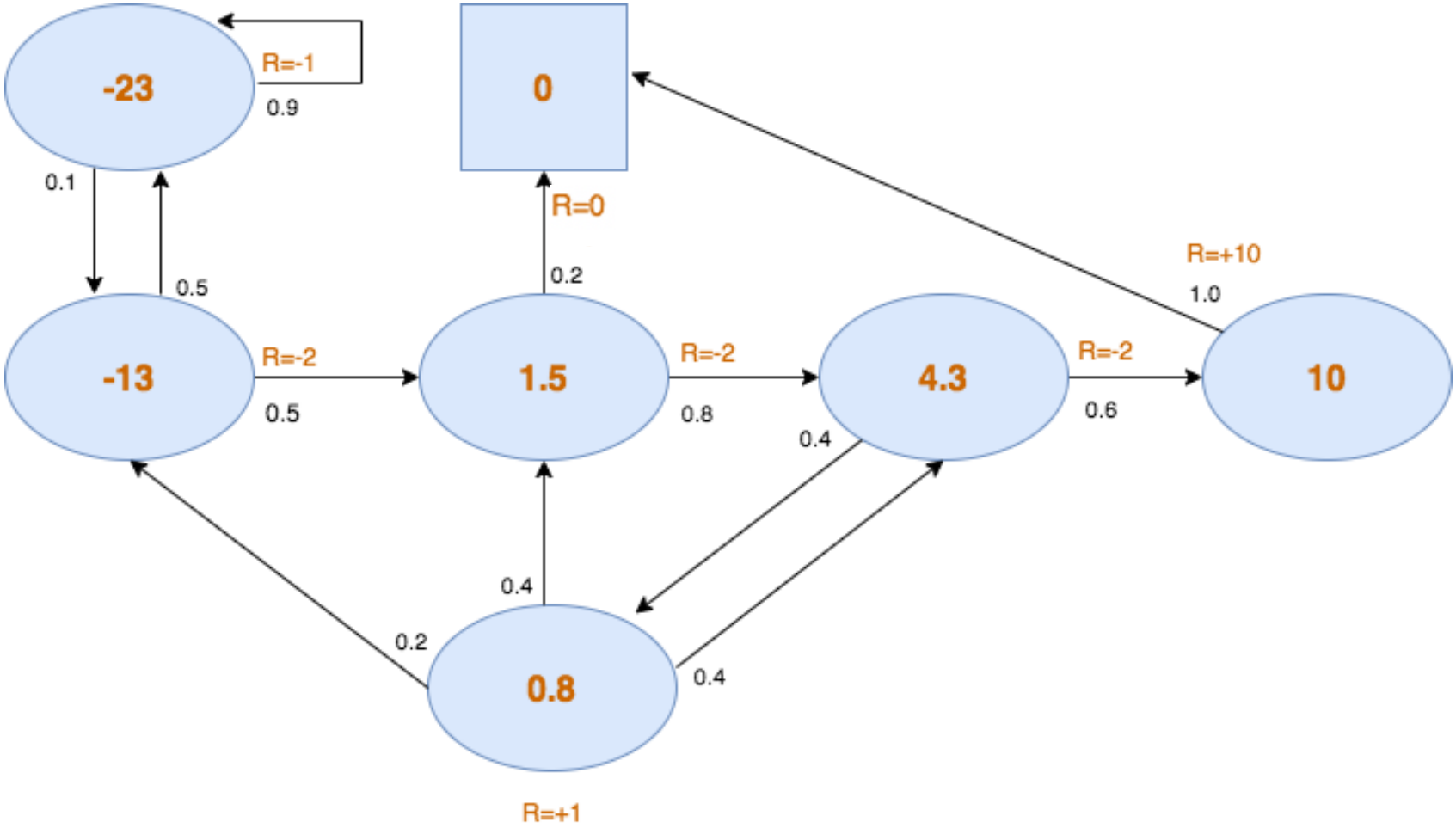
$v(s)$ for $\gamma = 0.9$



State-Value Function for Student MRP

Studying at home MRP

$v(s)$ for $\gamma = 1$



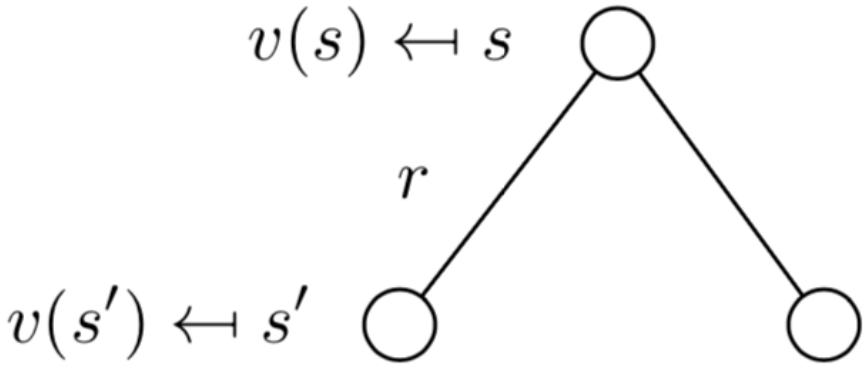
Bellman Equation for MRPs

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

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

Bellman Equation for MRPs

- Let's have a look at the one-step backup diagram
  - $v(s) = \mathbb{E}[R_{t+1} + \gamma v(S_{t+1}) | S_t = s]$



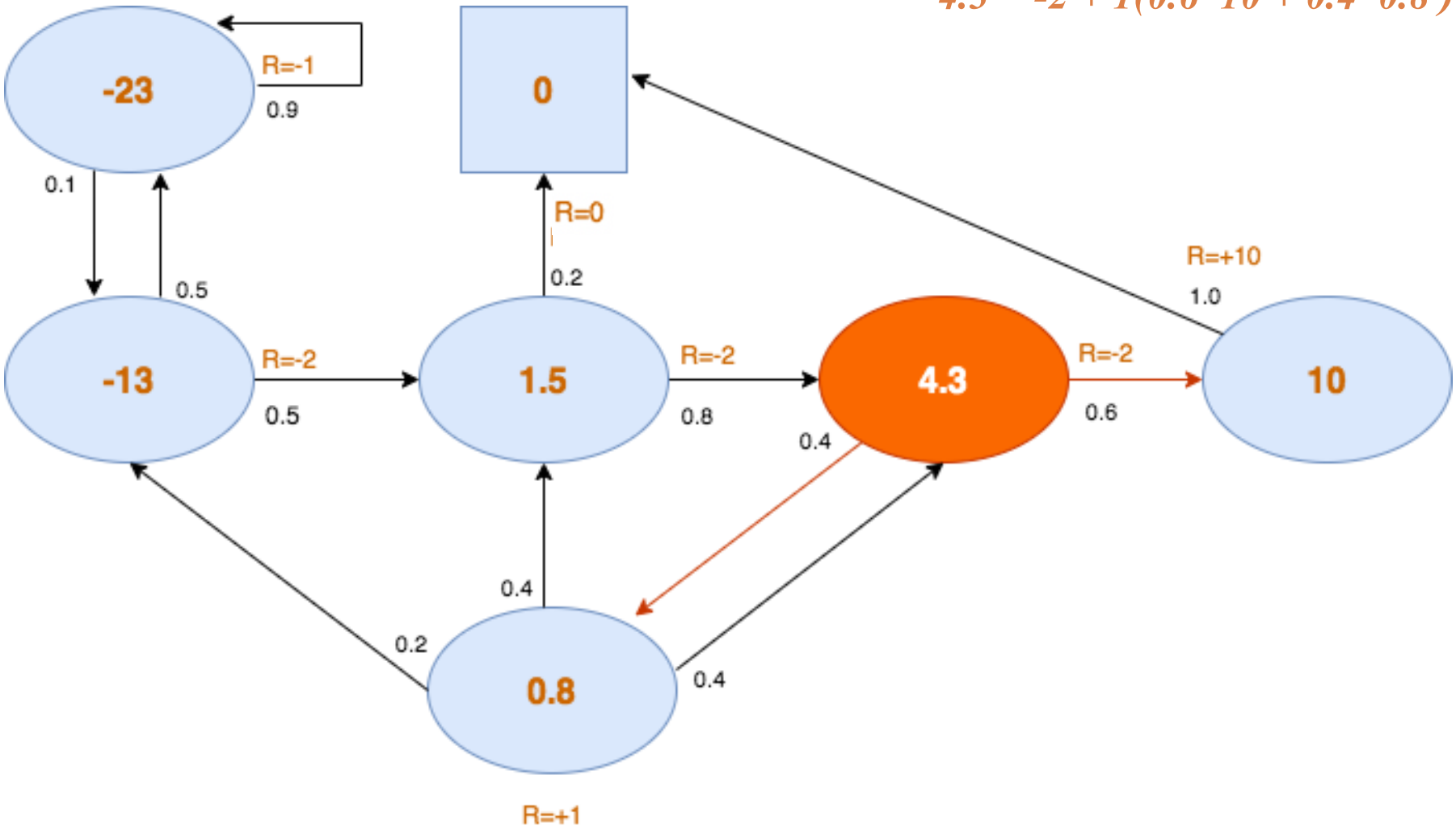
  - $v(s) = R_S + \gamma \sum_{s' \in S} \mathcal{P}_{ss'} v(s')$

State-Value Function for Student MRP

Studying at home MRP

$\gamma = 1$

$4.3 = -2 + 1(0.6*10 + 0.4*0.8)$



Bellman Equation in Matrix Form

- The Bellman equation can be expressed concisely using matrices,
  - $\mathrm{v} = \mathrm{R} + \gamma \mathrm{P}\mathrm{v}$
- where v is a column vector with one entry per state

$$\begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix} = \begin{bmatrix} R_1 \\ \vdots \\ R_n \end{bmatrix} + \gamma \begin{bmatrix} \mathcal{P}_{11} \cdots \mathcal{P}_{1n} \\ \vdots \quad \vdots \\ \mathcal{P}_{n1} \cdots \mathcal{P}_{nn} \end{bmatrix} \begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix}$$

Solving the Bellman Equation

- The Bellman equation is a linear equation It can be solved directly:
  - $v = R + \gamma P v$
  - $(I - \gamma P)v = R$
  - $v = (I - \gamma P)^{-1} R$

- 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

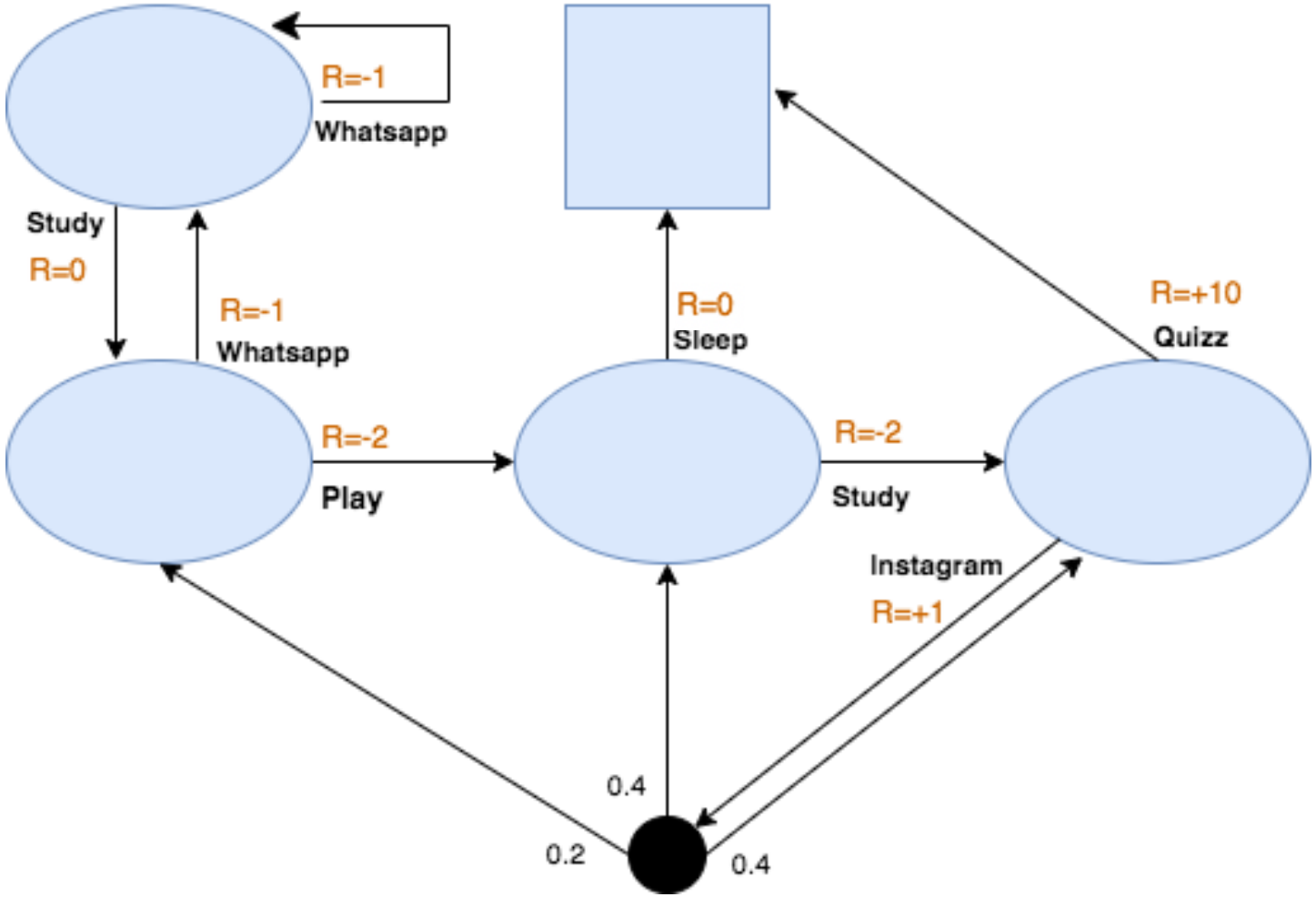
Markov Decision Process

- A Markov decision process is a Markov chain with values

- A **Markov Decision Process** (or MDP) is a tuple $\langle S, \mathcal{P}, A, \mathcal{R}, \gamma \rangle$
  - S is a finite set of states
  - A is a finite set of actions
  - $\mathcal{P}$ s a state transition probability matrix
    $$\mathcal{P}_{ss'}^{a} = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$$
  - $\mathcal{R}$ is a Reward function $\mathcal{R}_S = \mathbb{E}[R_{t+1} | S_t = s]$
  - γ is the discount factor, $\gamma \in [0,1]$

Markov Decision Process

Studying at home MDP



Policy

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

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

- 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),

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

Policy

- Given an MDP $\langle S, \mathcal{P}, A, \mathcal{R}, \gamma \rangle$ and a policy π
- The state sequence S_1, S_2 ...is a Markov Process $\langle S, \mathcal{P}^{\pi} \rangle$
- The state and reward sequence S_1, R_1, S_2, R_2 ... is a Markov Reward Process $\langle S, \mathcal{P}^{\pi}, R^{\pi}, \gamma \rangle$
- where

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

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

Value Function with Policy

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

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

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

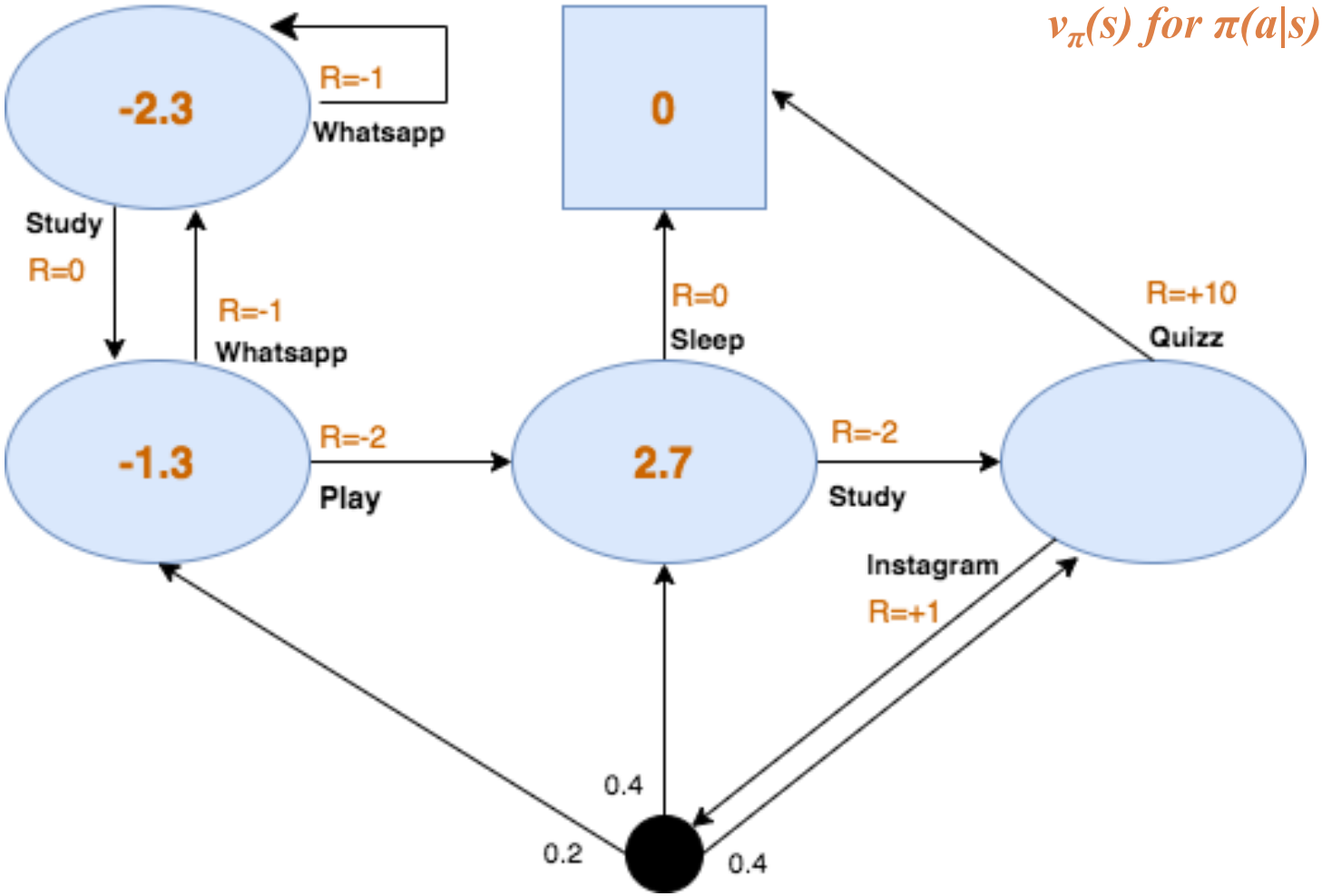
$$q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$$

State Value Function with Policy

Studying at home MDO

$\gamma = 1$

$v_\pi(s)$ *for* $\pi(a|s)=0.5$



Bellman Expectation Equation

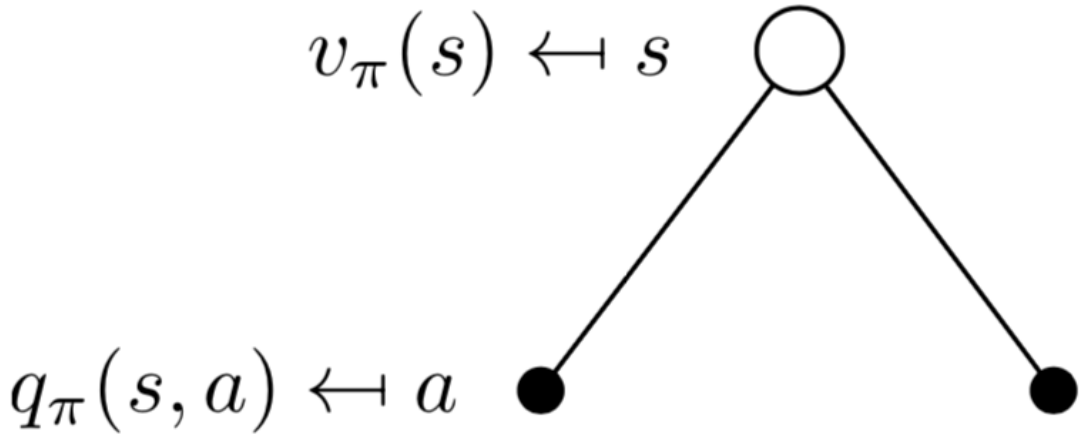
- The state-value function can again be decomposed into immediate reward plus discounted value of successor state,

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

The action-value function can similarly be decomposed,

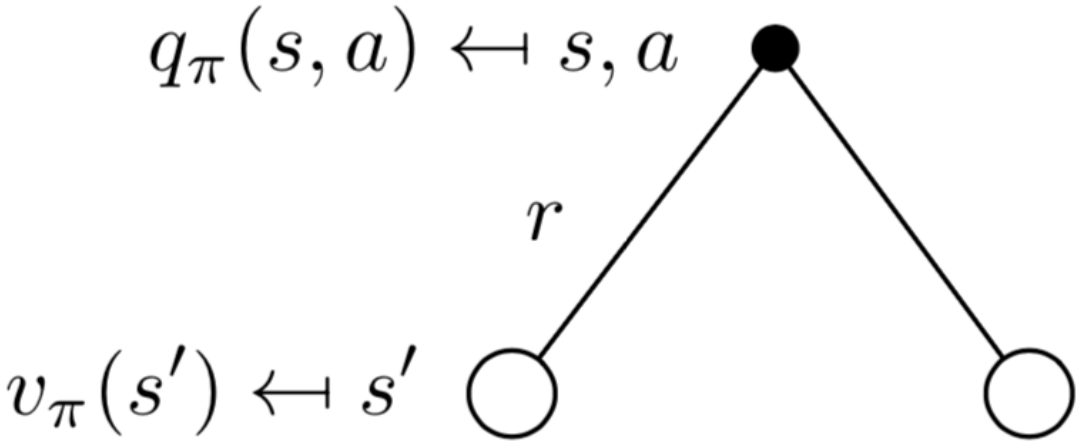
$$q_\pi(s, a) = \mathbb{E}_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a]$$

Bellman Expectation Equation for V^{π}



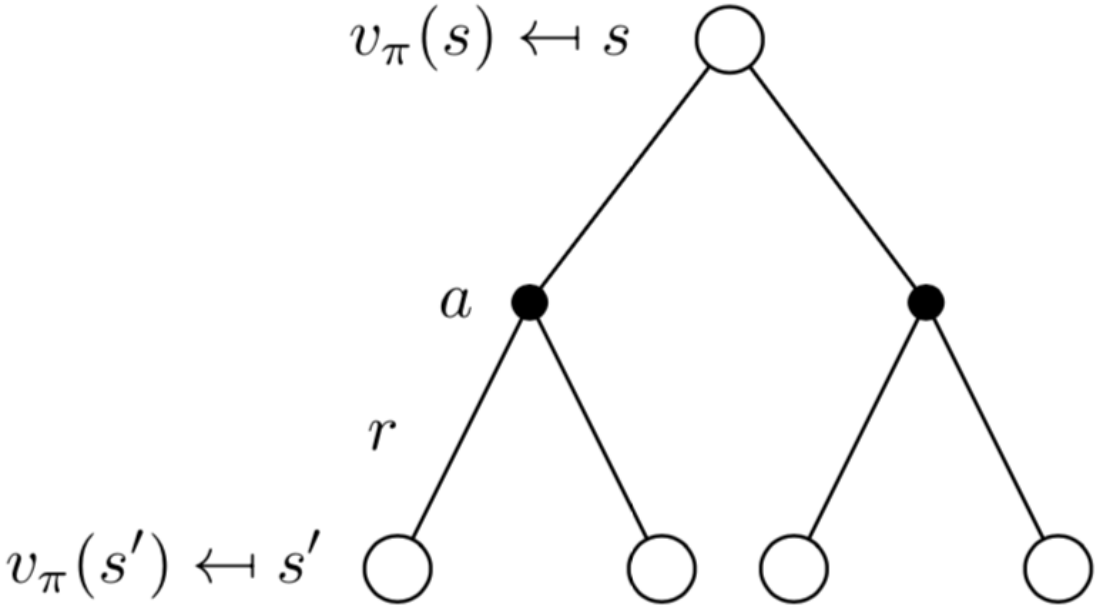
$$v_\pi(s) = \sum_{a \in A} \pi(a|s) q_\pi(s,a)$$

Bellman Expectation Equation for Q^{π}



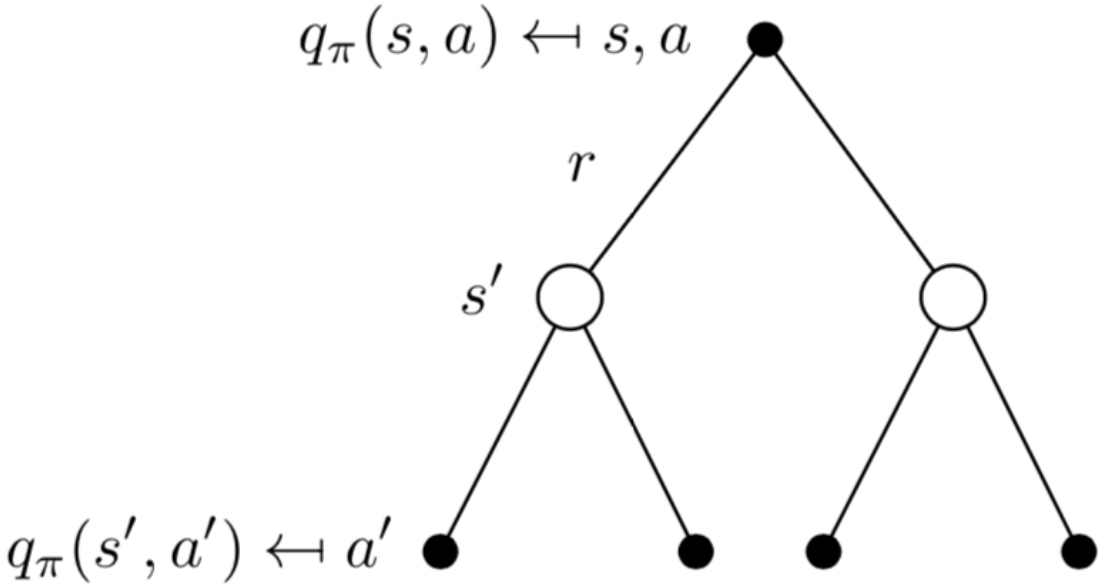
$$q_\pi(s,a) = R_s^a + \gamma \sum_{s' \in S} \mathcal{P}_{ss'}^a v_\pi(s')$$

Bellman Expectation Equation for V^{π}



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

Bellman Expectation Equation for Q^{π}



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

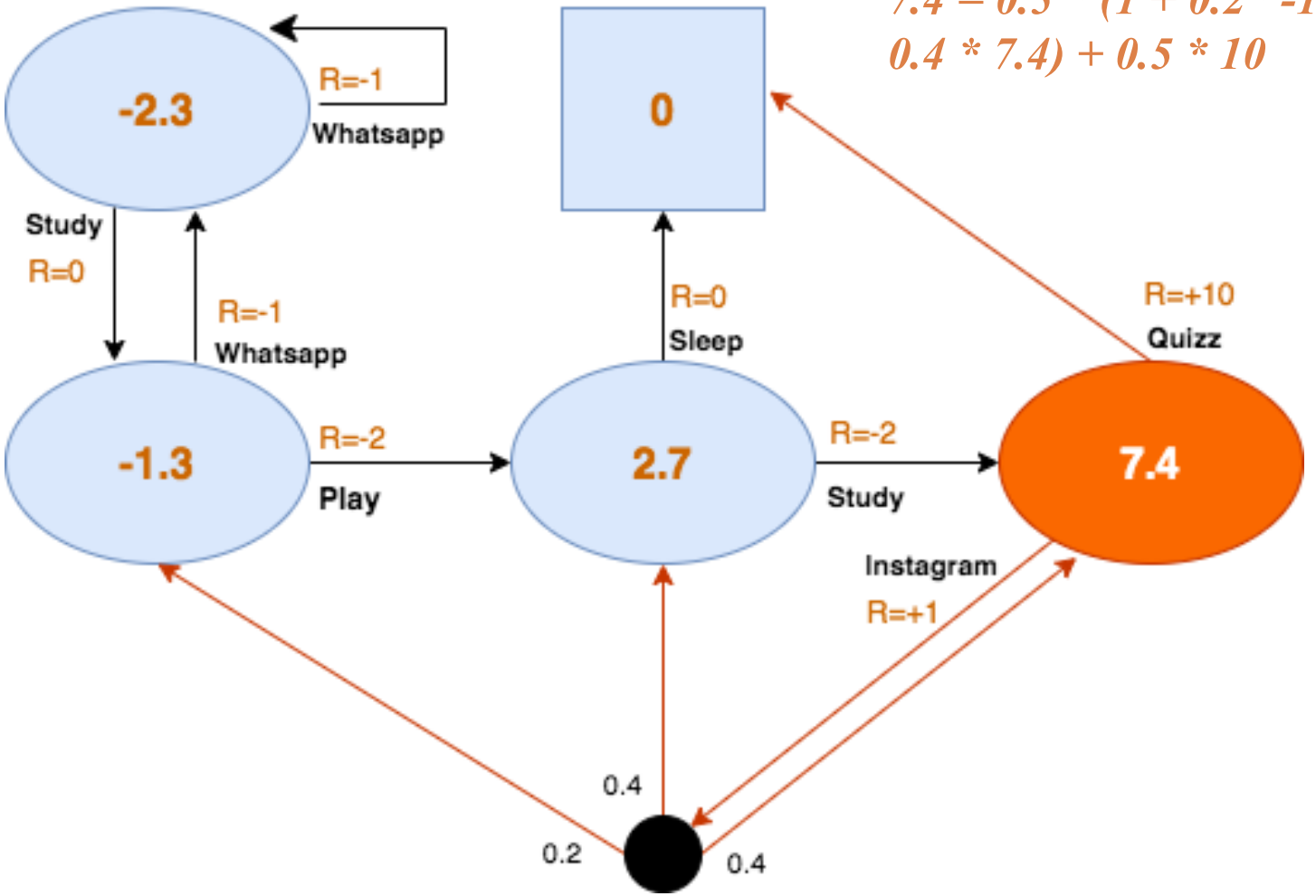
Markov Decision Process

Studying at home MDP

$\gamma = 1$

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

$7.4 = 0.5 * (1 + 0.2* -1.3 + 0.4 * 2.7 + 0.4 * 7.4) + 0.5 * 10$



Bellman Expectation Equation in Matrix Form

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

$$v_\pi(s) = R^\pi + \gamma P^\pi$$

- with direct solution

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

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

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

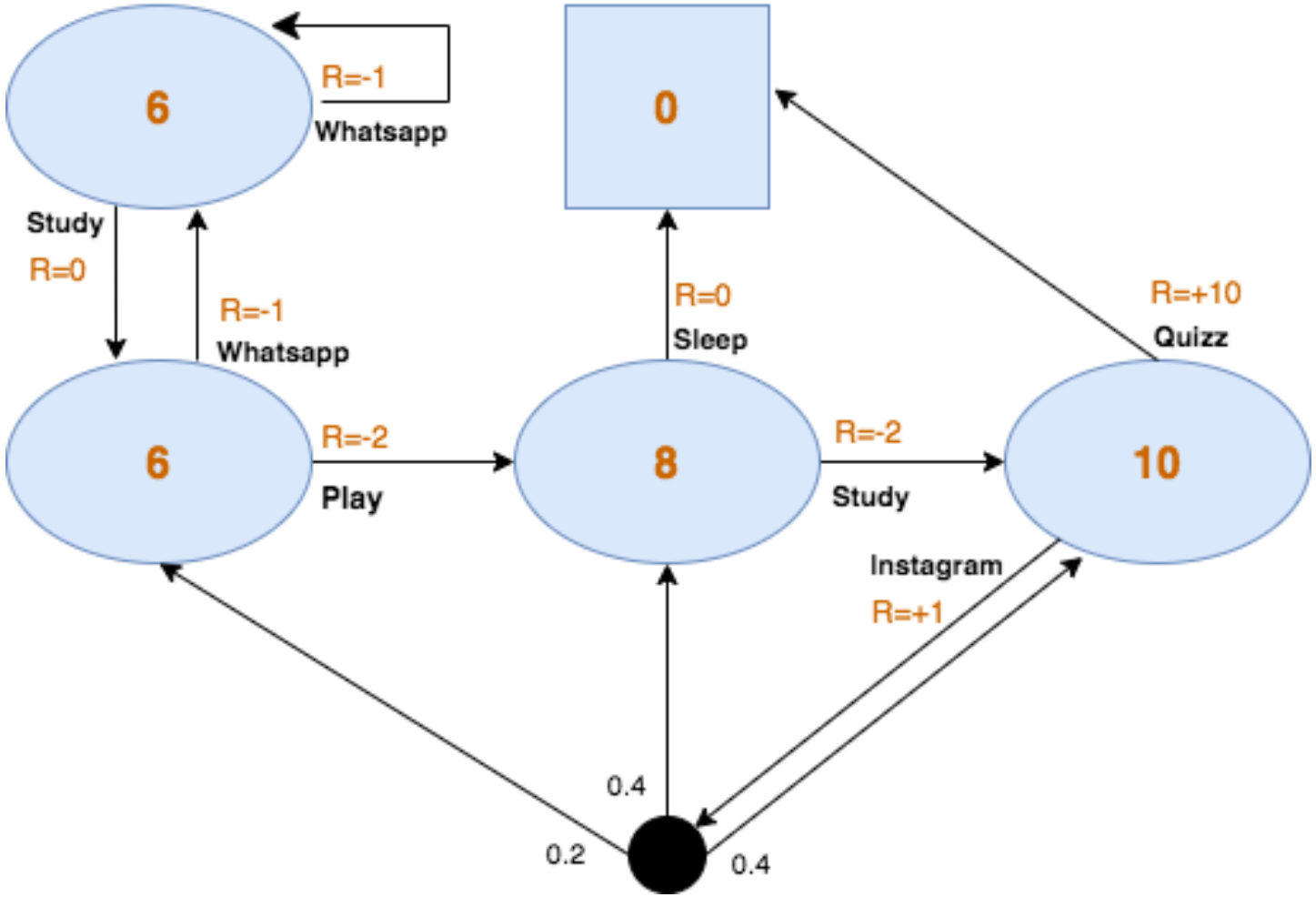
- The optimal value function specifies the best possible performance in the MDP.
- An MDP is "solved" when we know the optimal value function

.

Markov Decision Process

Studying at home MDP

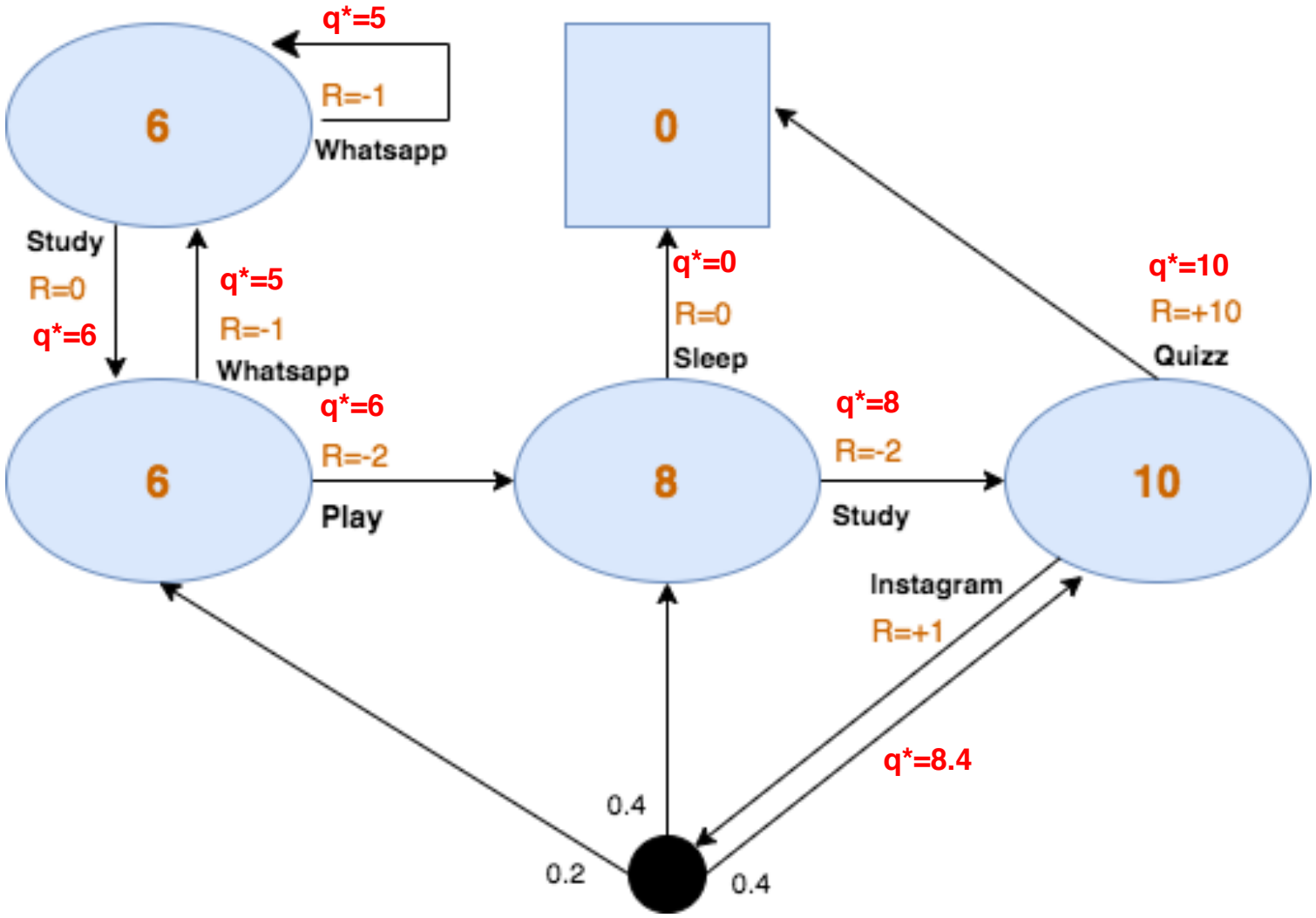
$v_*(s)$ *for* $\gamma = 1$



Markov Decision Process

Studying at home MDP

$q_*(s,a)$ *for* $\gamma = 1$



Optimal Policy

- Define a partial ordering over policies

$$\pi \geq \pi' \text{ for all } v_{\pi}(s) \geq v_{\pi'}(s), \forall s$$

- 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,

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

Optimal Policy for our MDP

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

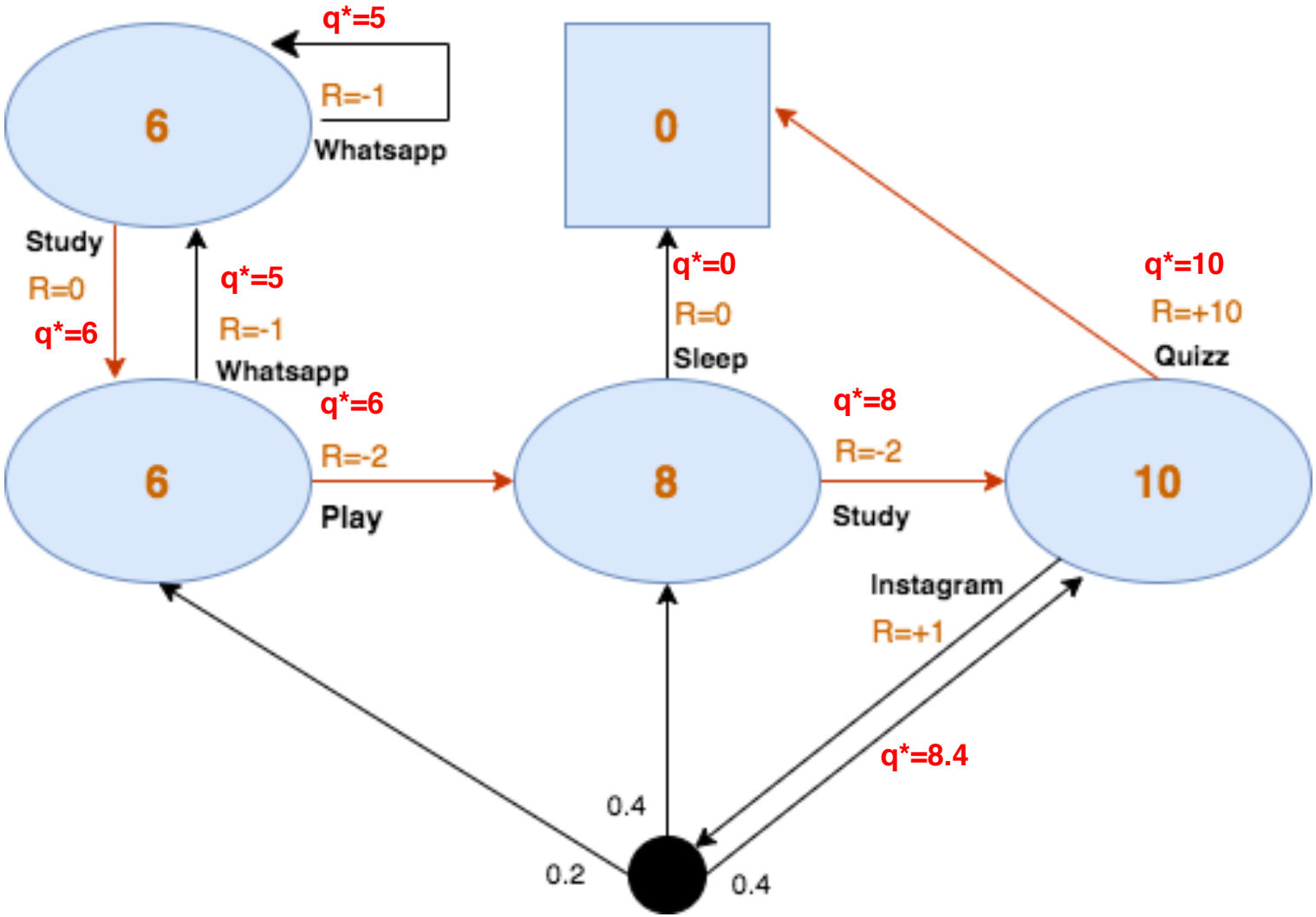
$$\pi_*(a|s) = \begin{cases} 1 \;\; if \; a = \underset{a \in A}{\operatorname{argmax}}\, q_*(s,a) \\ 0 \qquad\qquad\qquad\qquad \text{otherwise} \end{cases}$$

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

Optimal Policy for our MDP

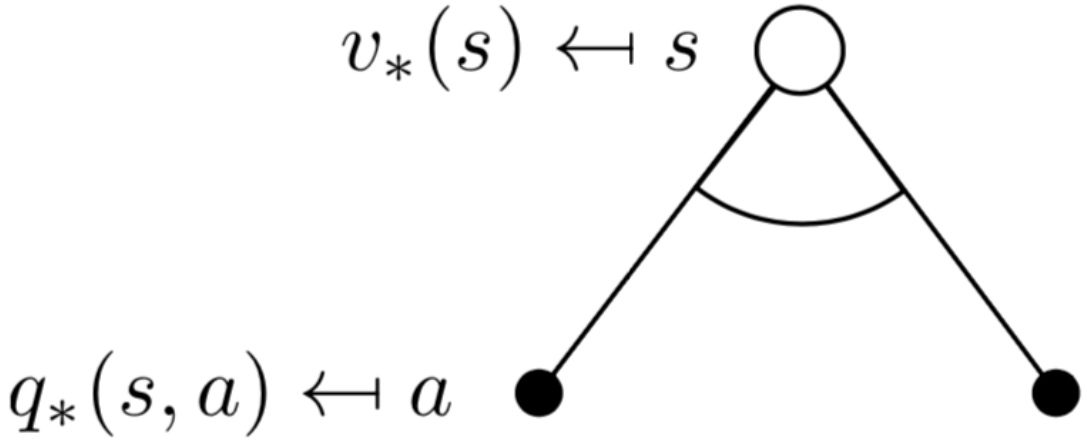
Studying at home MDP

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



Bellman Optimality Equation for v_*

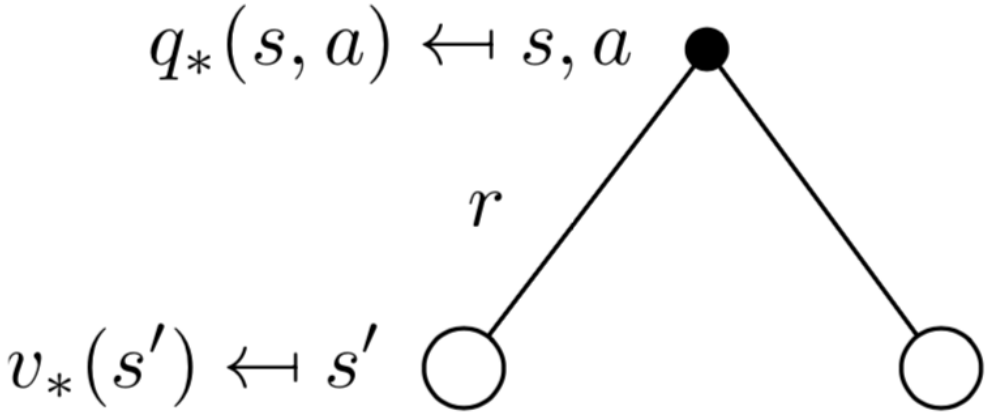
- The optimal value functions are recursively related by the Bellman optimality equations:



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

Bellman Optimality Equation for q_*

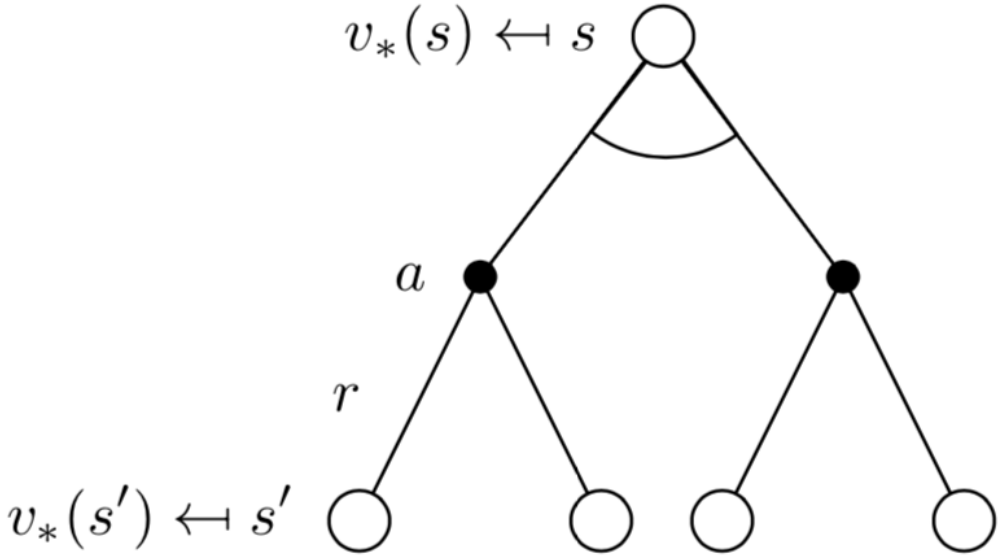
- The optimal state value functions are recursively related by the Bellman optimality equations:



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

Bellman Optimality Equation for v_*

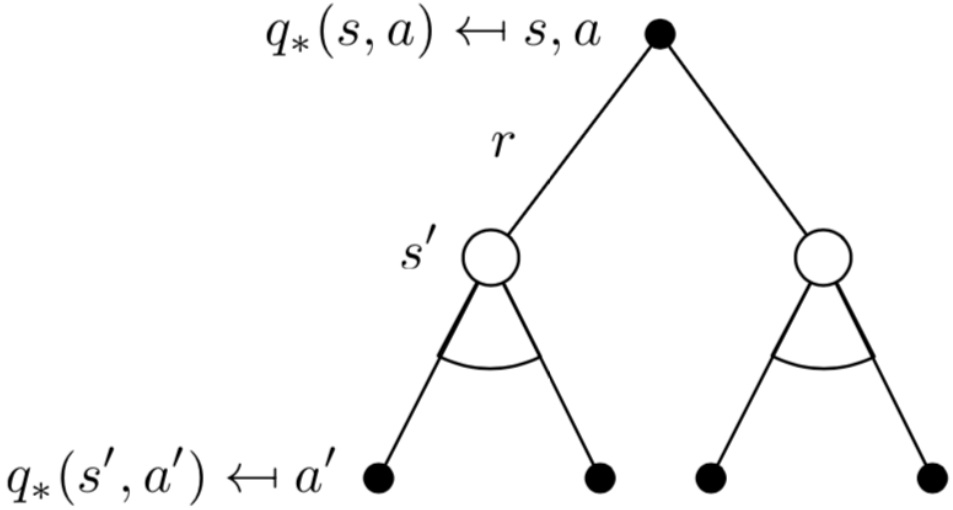
- The optimal value functions are recursively related by the Bellman optimality equations:



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

Bellman Optimality Equation for q_*

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

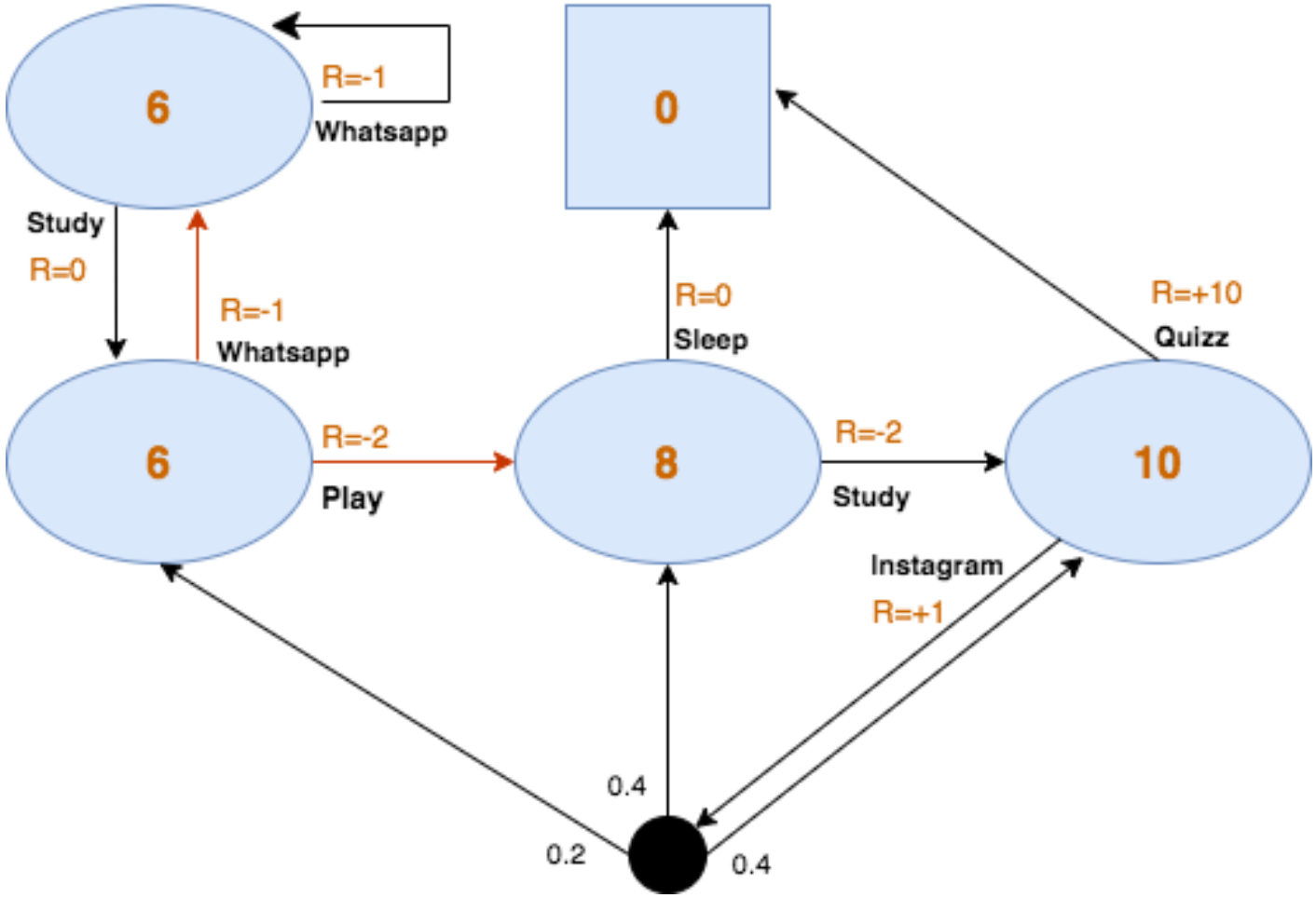


$$q_*(s,a) = R_s^a + \gamma \sum_{s' \in S} \mathcal{P}_{ss'}^a \max_{a'} q_*(s',a')$$

Markov Decision Process

Studying at home Markov Chain

$$6 = max\ \{-2 + 8, -1 + 6\}$$



Optimal Policy

- Bellman Optimality Equation is non-linear
- No closed form solution (in general)
- Many iterative solution methods
  - Value Iteration
  - Policy Iteration
  - Q-learning
  - Sarsa

Other types of MDPs

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

- What happens if we do not have full MDP?
  - That is, we need to learn the associated rewards
  - Well .. We know about states and actions
  - We do not know about the system model (transition function) or the reward function
  - We can learn from experience and carry out actions to generate such experiences.
- This is the main goal of reinforcement learning...

Summary

- What is the Markov property?
- Why do we need the Markov Decision Process?
- When do we prefer immediate rewards?
- What is the use of the discount factor?
- Why do we use the Bellman function?
- How would you derive the Bellman equation for a q function?
- How are the value function and Q function related?

- How could we solve iteratively an MDP?
  - Next class

Lecture 2

- Reading:
  - RUSSELL, S. NORVIG, P. Artificial Intelligence. 3a edição. Chapter 21.
  - BARTO, A., SUTTON, R. Reinforcement Learning: An Introduction. Second Edition. Freely Available at: https://drive.google.com/file/d/1xeUDVGWGUUv1-ccUMAZHJLej2C7aAFWY/view?usp=sharing

References

Lecture 2

- BARTO, A., SUTTON, R. Reinforcement Learning: An Introduction. Second Edition.
- MURPHY, R. R. Introduction to AI robotics. MIT Press, 2002.
- Lex Fridman, MIT Deep Learning Course, MIT, 2019.
- DUDEK, G.; JENKIN, M. Computational Principles of mobile robotics. Cambridge Press, 2000.
- ROMERO, R. A. F.; PRESTES, E.; OSÓRIO, F.; WOLF, D. (Orgs) Robótica móvel. LTC, 2014.
- BROOKS, R. Intelligence without representation. Artificial Intelligence, 47:139-159, 1991.
- RUSSEL, S. NORVIG, P. Artificial Intelligence: a modern approach. Prentice Hall, 2002.
- BRATKO, I. PROLOG: programming for artificial intelligence. Addison Wesley, 2nd edition, 1990.

This material is part of the Machine Learning Course

By Esther Colombini and Alexandre Simões