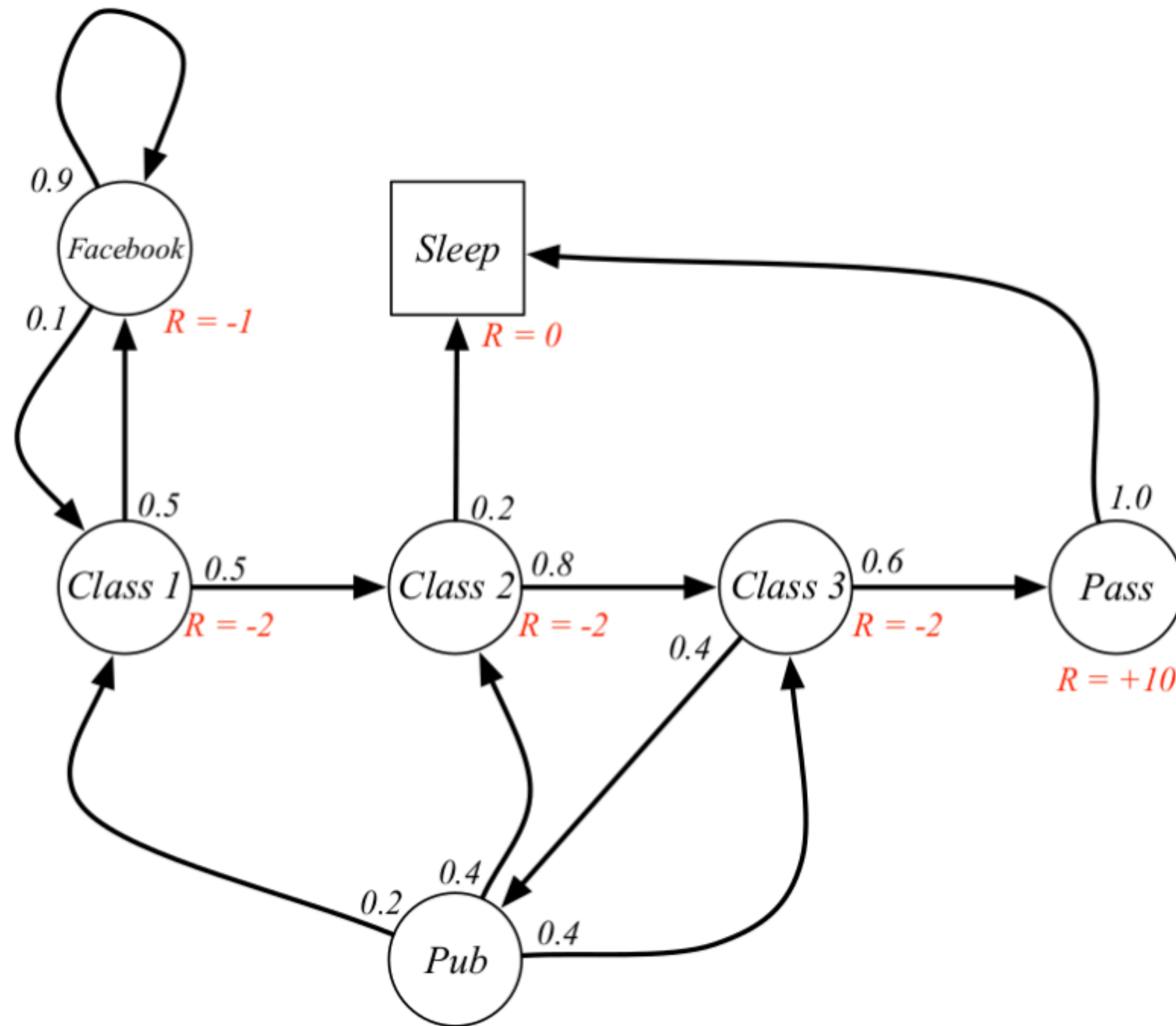


# How can you learn to value future actions?

# Readings for today

- Sutton, R. S., & Barto, A. G. (2020). Chapter 1: Introduction. In Reinforcement learning: An introduction (2nd edition). MIT press.
- Sutton, R. S., & Barto, A. G. (2020). Chapter 2: Multi-armed bandits. In Reinforcement learning: An introduction (2nd edition). MIT press.

# The state-action problem



What is the best way to strategically ***learn to*** shift from one state to another?

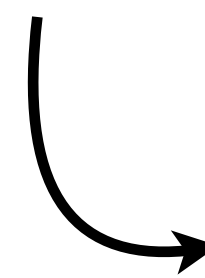
# The Bellman equation

What is the *optimal* path through potential states that has the highest value?

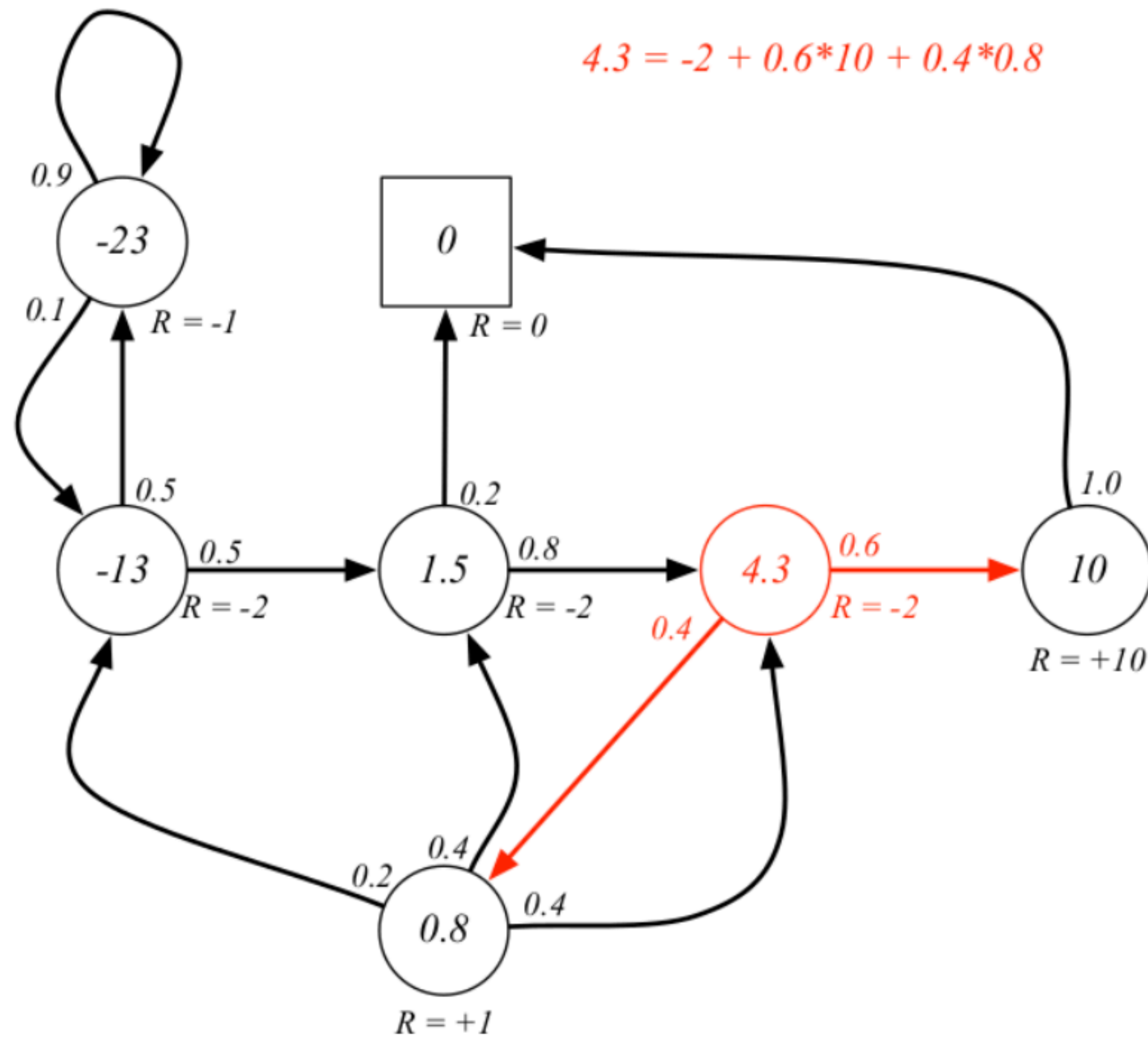
$$v(s) = \mathbb{E}(G_t | S_t = s)$$

Bellman equation

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


$$G_{t+1} \rightarrow v(S_{t+1})$$

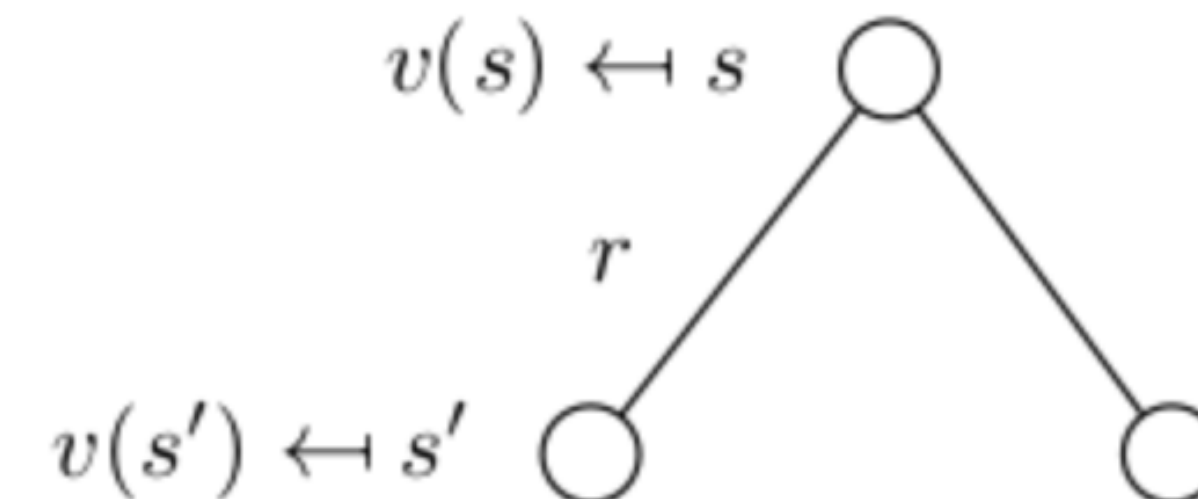
# The Bellman equation



The value both depends on the reward and the transition probability .

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

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



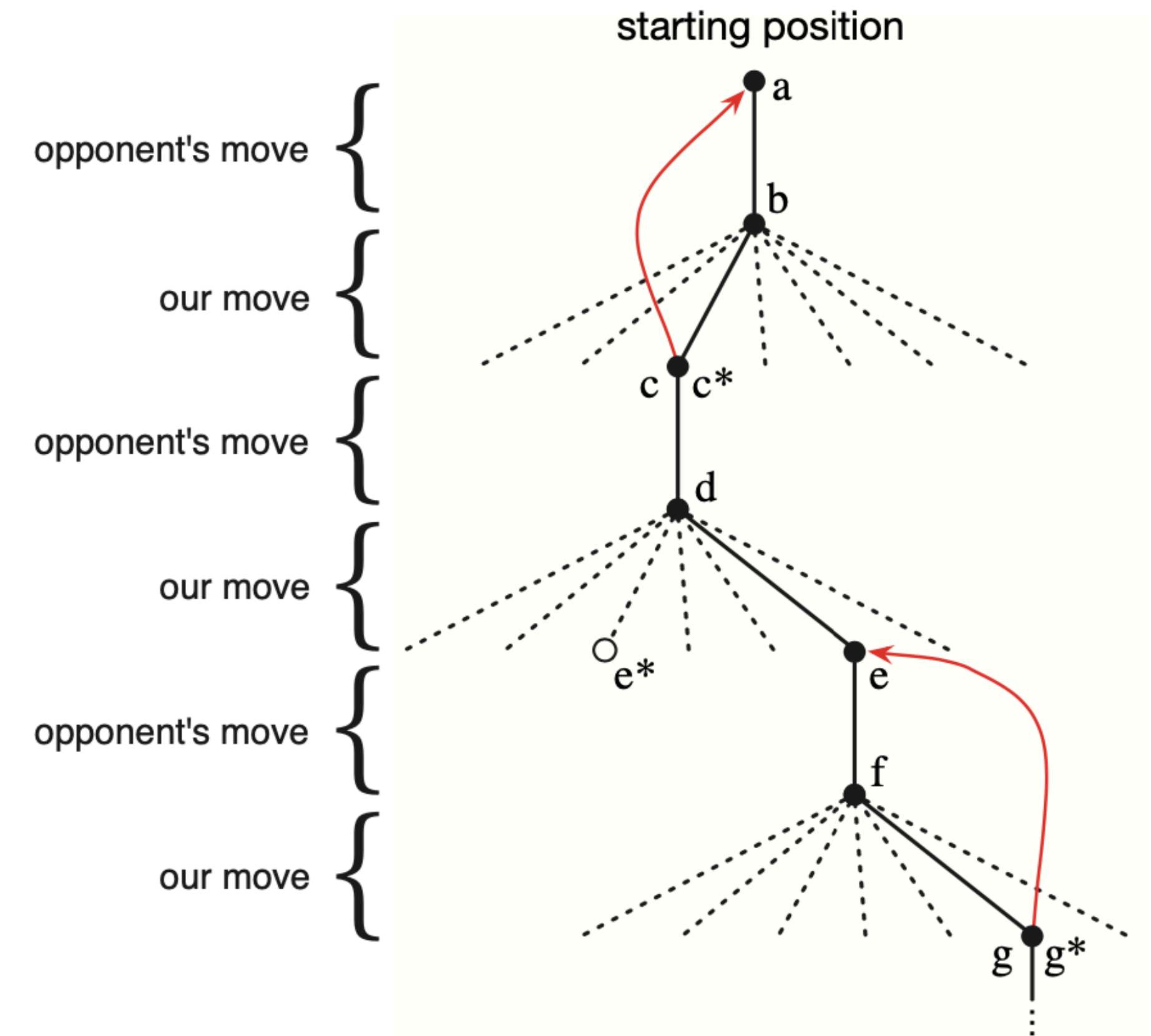
# Temporal difference update

How do you update your state value from one move to the next?

$$V(S_{t+1}) \leftarrow V(S_t) + \alpha[V(S_{t+1}) - V(S_t)]$$

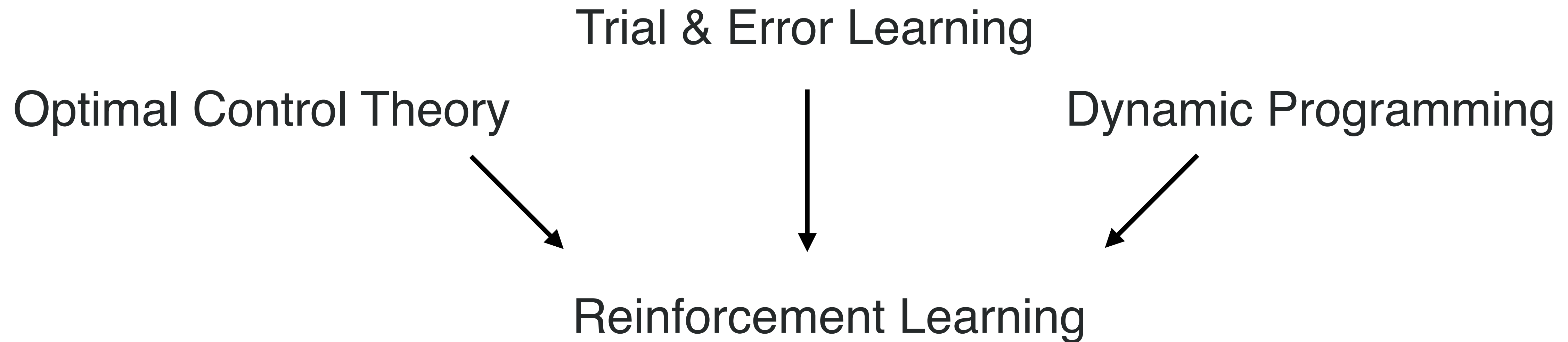
error

learning rate



This works in the forward looking framing of the Bellman equation.

# Reinforcement learning



Learning what to do—how to map situations to actions—so as to maximize a numerical reward signal.



# The $k$ -armed bandit problem



## The Bandit task

- There are  $k$  many actions (arms),  $a$ , that can be made at any time  $t$
- Each action has returns a reward  $r$  probabilistically, from a (typically) unchanging probability distribution.
- **Goal:** Find the action (arm) that has the highest expected return  $\mathbb{E}[R_t]$

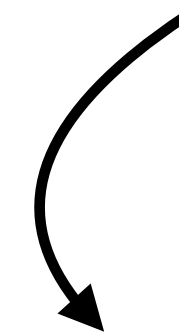


# The $k$ -armed bandit problem



Action value problem

$$q(a) = \mathbb{E}[R_t | A_t = a]$$



$$Q_t(a) = \frac{\sum_{i=1}^{t-1} R_i | A_t = a}{\sum_{i=1}^{t-1} A_t = a}$$

# The $\epsilon$ -greedy method

**Action value**

$$Q_t(a) = \frac{\sum_{i=1}^{t-1} R_i | A_t = a}{\sum_{i=1}^{t-1} A_t = a}$$

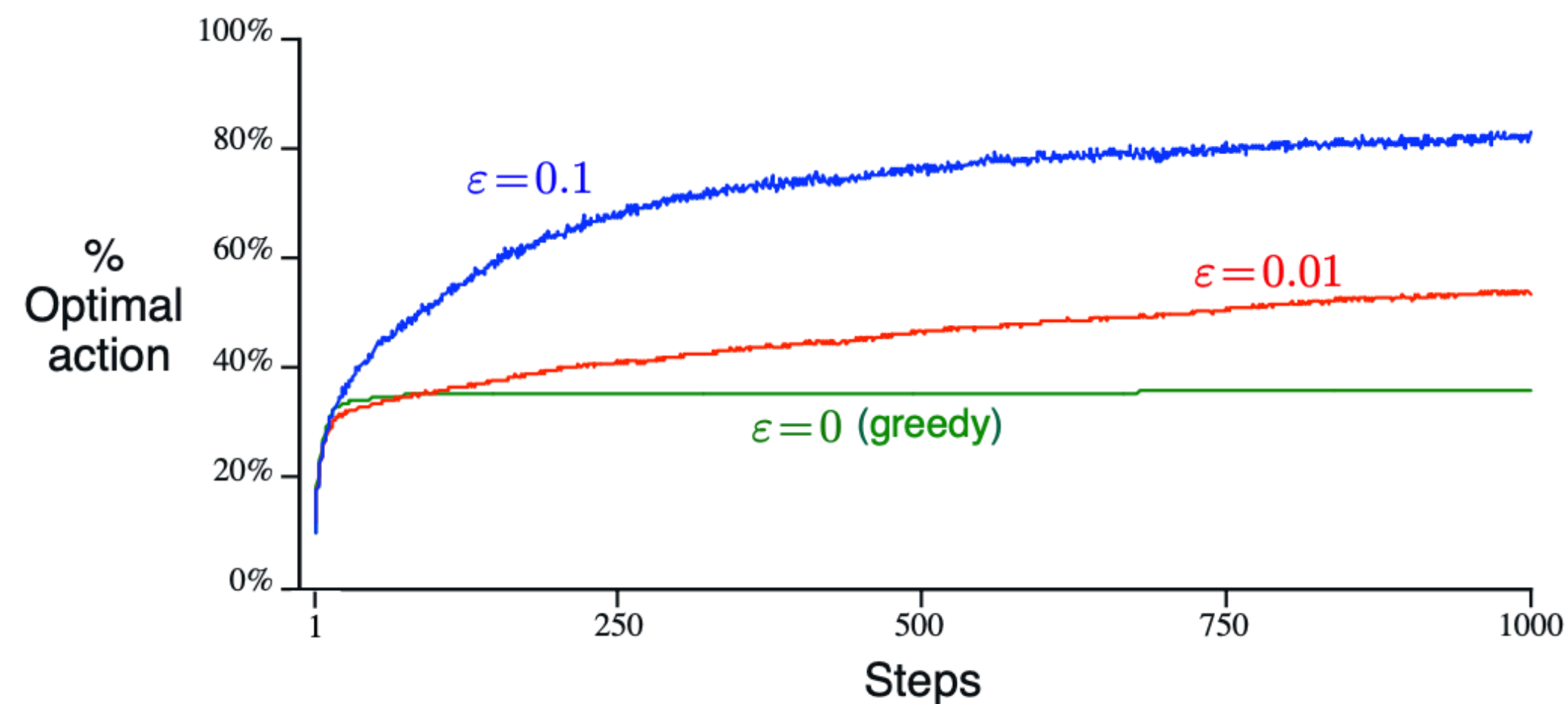
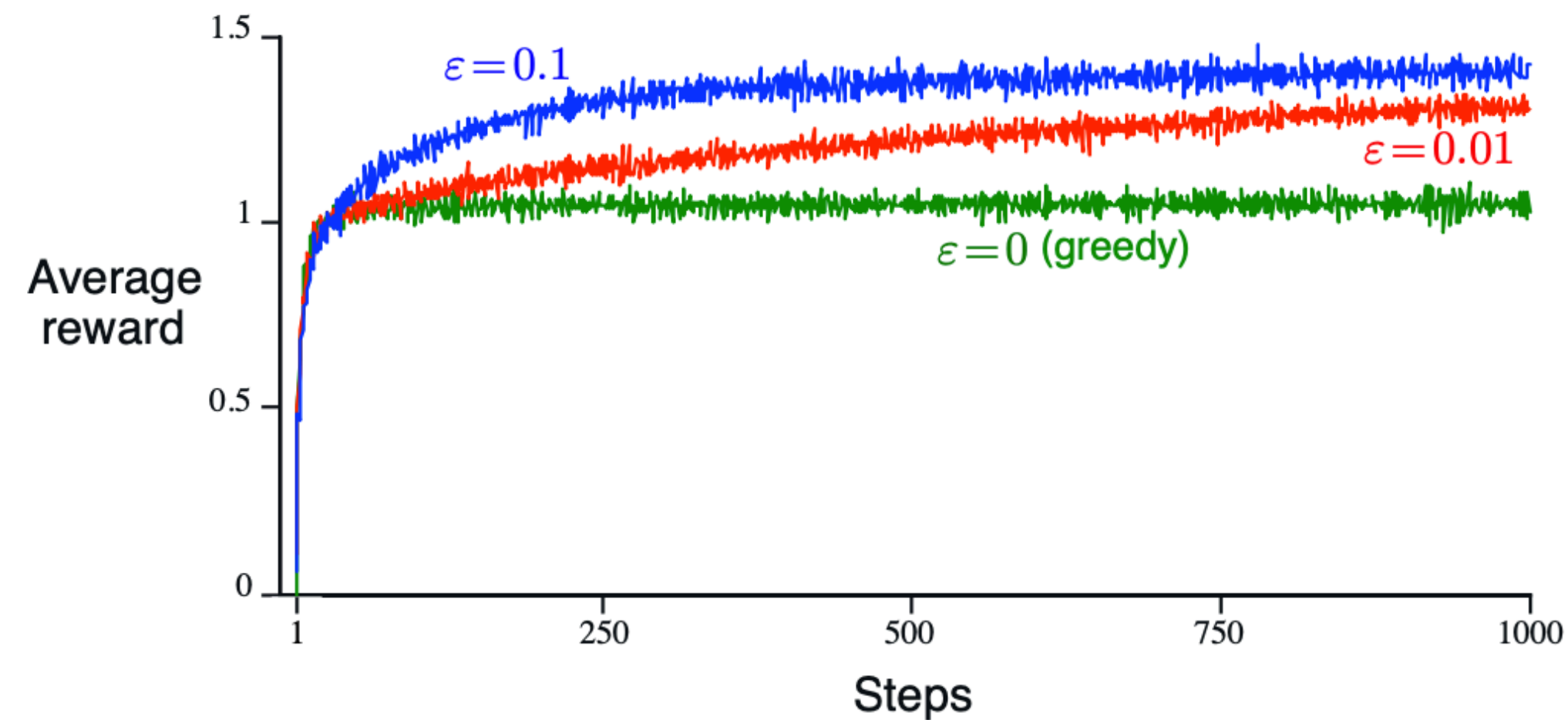
**Best action**

$$A_t = \arg \max_a Q_t(a)$$

**Decision policy**     $\max Q_t(a),$   
any  $a,$

with probability  $1 - \epsilon$   
with probability  $\epsilon$

# The $\epsilon$ -greedy method



Reducing the *greediness* of the algorithm, increases the exploration of the agent, allowing for optimal learning.

## A simple bandit algorithm

Initialize, for  $a = 1$  to  $k$ :

$$Q(a) \leftarrow 0$$

$$N(a) \leftarrow 0$$

Loop forever:

$$A \leftarrow \begin{cases} \operatorname{argmax}_a Q(a) & \text{with probability } 1 - \epsilon \quad (\text{breaking ties randomly}) \\ \text{a random action} & \text{with probability } \epsilon \end{cases}$$

$$R \leftarrow \text{bandit}(A)$$

$$N(A) \leftarrow N(A) + 1$$

$$Q(A) \leftarrow Q(A) + \frac{1}{N(A)} [R - Q(A)]$$

# Nonstationary problems

**Incremental update rule**      $NewEstimate \leftarrow OldEstimate + StepSize \left[ Target - OldEstimate \right]$

$$\begin{aligned} Q_{n+1} &= \frac{1}{n} \sum_{i=1}^n R_i \\ &= \frac{1}{n} \left( R_n + \sum_{i=1}^{n-1} R_i \right) \\ &= \frac{1}{n} \left( R_n + (n-1) \frac{1}{n-1} \sum_{i=1}^{n-1} R_i \right) \quad \longrightarrow \quad Q_{n+1} \doteq Q_n + \alpha \left[ R_n - Q_n \right] \\ &= \frac{1}{n} \left( R_n + (n-1) Q_n \right) \\ &= \frac{1}{n} \left( R_n + n Q_n - Q_n \right) \\ &= Q_n + \frac{1}{n} \left[ R_n - Q_n \right], \end{aligned}$$

# The reinforcement learning structure

$$NewEstimate \leftarrow OldEstimate + StepSize [Target - OldEstimate]$$

$$Q_{n+1} \doteq Q_n + \alpha [R_n - Q_n]$$

Reward prediction error (RPE)

learning rate



# The reinforcement learning structure

$$NewEstimate \leftarrow OldEstimate + StepSize \left[ Target - OldEstimate \right]$$

$$\begin{aligned} Q_{n+1} &= Q_n + \alpha [R_n - Q_n] \\ &= \alpha R_n + (1 - \alpha) Q_n \\ &= \alpha R_n + (1 - \alpha) [\alpha R_{n-1} + (1 - \alpha) Q_{n-1}] \\ &= \alpha R_n + (1 - \alpha) \alpha R_{n-1} + (1 - \alpha)^2 Q_{n-1} \\ &= \alpha R_n + (1 - \alpha) \alpha R_{n-1} + (1 - \alpha)^2 \alpha R_{n-2} + \\ &\quad \dots + (1 - \alpha)^{n-1} \alpha R_1 + (1 - \alpha)^n Q_1 \\ &= (1 - \alpha)^n Q_1 + \sum_{i=1}^n \alpha (1 - \alpha)^{n-i} R_i. \end{aligned}$$

Assuming  $\alpha \in (0, 1]$ , then  $Q_t$  is simply the weighted average of past rewards and the initial estimate  $Q_1$

# Take home message

- Reinforcement learning approximates the Bellman equation to learn optimal action values from post-action feedback.
- Simple action value policies are learned as the aggregate sum of previous rewards.
- Introducing randomness to the selection of actions reduces the ‘greediness’ of an agent, allowing for greater learning/flexibility.



# Break out group discussion

- The basic RL models that we went over are principled off of decision problems that follow the form of the k-armed bandit task (in fact, the algorithms we discussed were explicitly designed to work in bandit environments). ***Does the bandit task reasonably capture the properties of real life value-based decisions?*** Justify your answer.