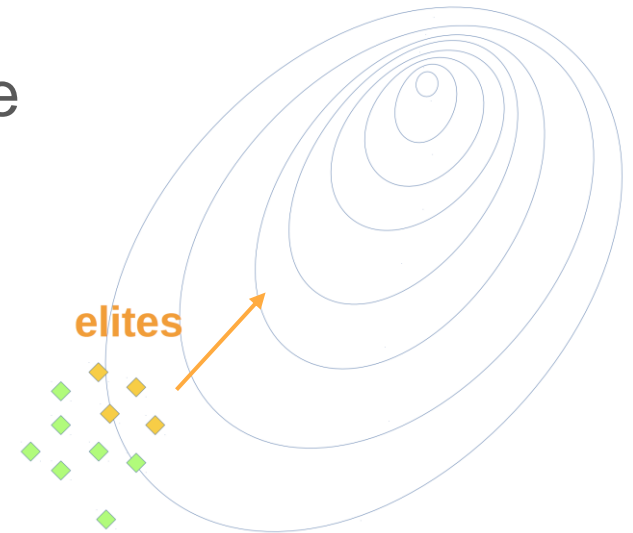


# Practical RL – Week 2

Shvechikov Pavel

# Previously in the course

- The MDP formalism
  - State, Action, Reward, next State
- Cross-Entropy Method (CEM)
  - easy to implement, good results
  - rich theoretical background
  - black box
    - no knowledge of environment
    - no knowledge of intermediate rewards



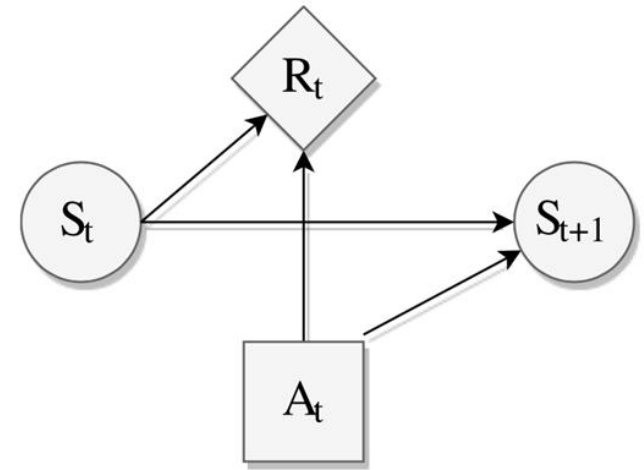
Improve on the CEM → dive into the black box

# Given dynamics, how to find an optimal policy?

## Definition of Markov Decision Process

MDP is a tuple  $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$ , where

- ①  $\mathcal{S}$  – set of states of the world
- ②  $\mathcal{A}$  – set of actions
- ③  $\mathcal{P} : \mathcal{S} \times \mathcal{A} \mapsto \Delta(\mathcal{S})$  – state-transition function, giving us  $p(s_{t+1} | s_t, a_t)$
- ④  $\mathcal{R} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$  – reward function, giving us  $\mathbb{E}_R [ R(s_t, a_t) | s_t, a_t ]$ .



## Markov property

$$p(r_t, s_{t+1} | s_0, a_0, r_0, \dots, s_t, a_t) = p(r_t, s_{t+1} | s_t, a_t)$$

(next state, expected reward) depend on (previous state, action)

# Goal: solve an MDP by finding **an** optimal policy

1. What is the objective?
  - a. Reward: discounting and design
  - b. Expected objective: state- and action-value function
2. How to evaluate the objective?
  - a. Bellman **expectation** equations
3. How to improve the objective?
  - a. Bellman **optimality** equations
4. Combine evaluation and improvement:
  - a. Generalized Policy Iteration

# Explaining goals to agent through reward

## Reward hypothesis (R.Sutton)

Goals and purposes can be thought of as the maximization of the expected value of the cumulative sum of a received scalar signal

# Explaining goals to agent through reward

## Reward hypothesis (R.Sutton)

Goals and purposes can be thought of as the maximization of the expected value of the cumulative sum of a received scalar signal

Cumulative reward is called a **return**:

$$G_t \triangleq R_t + R_{t+1} + R_{t+2} + \dots + R_T$$

E.g.: reward in **chess** – value of taken opponent's piece

# Explaining goals to agent through reward

## Reward hypothesis (R.Sutton)

Goals and purposes can be thought of as the maximization of the expected value of the cumulative sum of a received scalar signal

Cumulative reward is called a return:

$$G_t \triangleq R_t + R_{t+1} + R_{t+2} + \dots + R_T$$

Diagram illustrating the components of the return equation:

- A blue arrow points from the text "Cumulative reward" to the box containing  $G_t$ .
- A green arrow points from the text "immediate reward" to the box containing  $R_t$ .
- A red arrow points from the text "end of an episode" to the box containing  $R_T$ .

E.g.: reward in **chess** – value of taken opponent's piece

**E.g.:** data center non-stop cooling system

- **S**tates – temperature measurements
- **A**ctions – different fans speed
- **R = 0** for exceeding temperature thresholds
- **R = +1** for each second system is cool

What could go wrong with such a design?



## E.g.: data center non-stop cooling system

- **S**tates – temperature measurements
- **A**ctions – different fans speed
- **R = 0** for exceeding temperature thresholds
- **R = +1** for each second system is cool

What could go wrong with such a design?

Infinite return for **non optimal** behaviour!

$$G_t = 1 + 1 + 0 + 1 + 1 + 0 + \dots = \sum_{t=1}^{\infty} R_t = \infty$$

E.g.: moving to destination



- **State** – position, velocities of joints
- **Actions** – actuator forces to joints
- **$R = \max(0, d(x, B) - d(x', B))$**

What could go wrong with such a design?

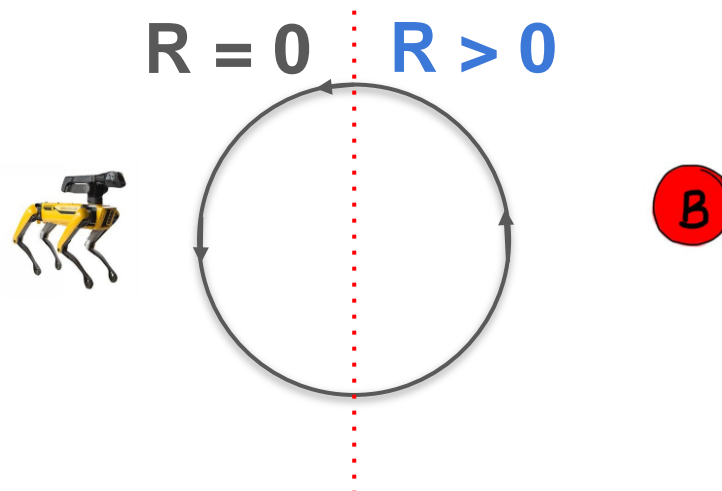
E.g.: moving to destination



- **State** – position, velocities of joints
- **Actions** – actuator forces to joints
- **$R = \max(0, d(x, B) - d(x', B))$**

What could go wrong with such a design?

**Positive  
feedback  
loop!**



# Reward discounting

# Reward discounting

Get rid of infinite sum by **discounting**  $0 \leq \gamma < 1$

$$G_t \triangleq R_t + \underbrace{\gamma}_{\text{discount factor}} R_{t+1} + \gamma^2 R_{t+2} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

The same cake compared to today's one worth

- $\gamma$  times less tomorrow
- $\gamma^2$  times less the day after tomorrow



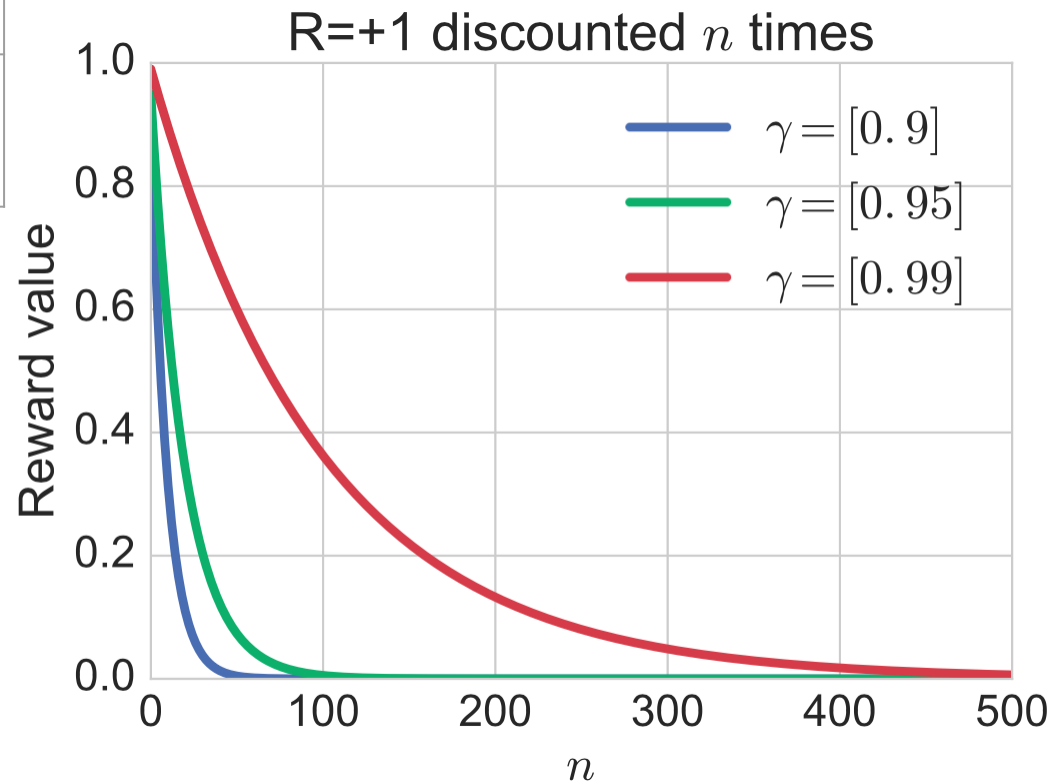
$\gamma$  will eat it day by day

# Discounting makes sums finite

Maximal return for **R = +1**

$$G_0 = \sum_{k=0}^{\infty} \gamma^k = \frac{1}{1 - \gamma}$$

$\gamma$	0.9	0.95	0.99
$\frac{1}{1-\gamma}$	10	20	100



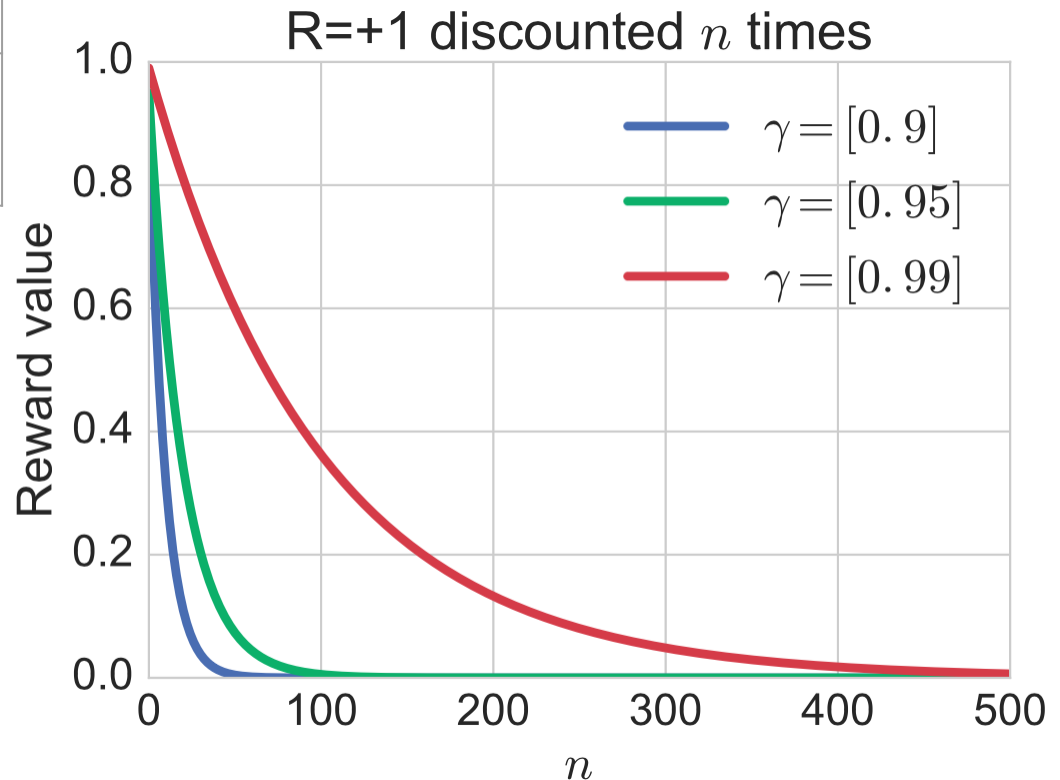
# Discounting makes sums finite

Maximal return for **R = +1**

$$G_0 = \sum_{k=0}^{\infty} \gamma^k = \frac{1}{1 - \gamma}$$

$\gamma$	0.9	0.95	0.99
$\frac{1}{1-\gamma}$	10	20	100

Any **discounting**  
**changes** optimisation  
**task** and its solution!



# Discounting is inherent to humans

- Quasi-hyperbolic  $f(t) = \beta\gamma^t$
- Hyperbolic discounting  $f(t) = \frac{1}{1 + \beta t}$



# Discounting is inherent to humans

- Quasi-hyperbolic  $f(t) = \beta\gamma^t$
- Hyperbolic discounting  $f(t) = \frac{1}{1 + \beta t}$

## Mathematical convenience

$$\begin{aligned} G_t &= R_t + \gamma(R_{t+1} + \gamma R_{t+2} + \dots) \\ &= \boxed{R_t + \gamma G_{t+1}} \end{aligned}$$



Remember this one!  
We will need it later

Discounting is a stationary end-of-effect model

Any action affects (1) immediate reward (2) next state

# Discounting is a stationary end-of-effect model

Any action affects (1) immediate reward (2) next state

Action indirectly affects future rewards 

But how long does this effect lasts?

$$\begin{aligned} G_0 &= R_0 + \gamma R_1 + \gamma^2 R_2 + \dots + \gamma^T R_T \\ &= (1 - \gamma) R_0 \\ &\quad + (1 - \gamma) \gamma (R_0 + R_1) \\ &\quad + (1 - \gamma) \gamma^2 (R_0 + R_1 + R_2) \\ &\quad \dots \\ &\quad + \gamma^T \cdot \sum_{t=0}^T R_t \end{aligned}$$

G is expected return under stationary end-of-effect model

# Discounting is a stationary end-of-effect model

Any action affects (1) immediate reward (2) next state

Action indirectly affects future rewards

But how long does this effect lasts?

$$\begin{aligned} G_0 &= R_0 + \gamma R_1 + \gamma^2 R_2 + \dots + \gamma^T R_T \\ &= \boxed{(1 - \gamma)} R_0 + \boxed{(1 - \gamma)} \boxed{\gamma} (R_0 + R_1) \\ &\quad + \boxed{(1 - \gamma)} \boxed{\gamma^2} (R_0 + R_1 + R_2) \\ &\quad \dots \\ &\quad + \gamma^T \cdot \sum_{t=0}^T R_t \end{aligned}$$

“End of effect” probability

“Effect continuation” probability

G is expected return under stationary end-of-effect model

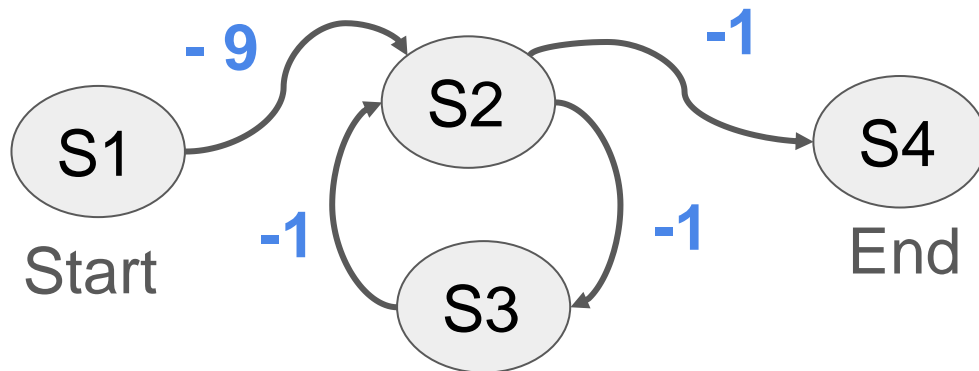
# Reward design – don't shift, reward for WHAT

- E.g.: chess – value of taken opponent's piece
  - Problem: agent will not have a desire to win!
- E.g.: moving to destination
  - Problem: agent will not bother about the goal!

# Reward design – don't shift, reward for WHAT

- **E.g.:** chess – value of taken opponent's piece
  - **Problem:** agent will not have a desire to win!
- **E.g.:** moving to destination
  - **Problem:** agent will not bother about the goal!

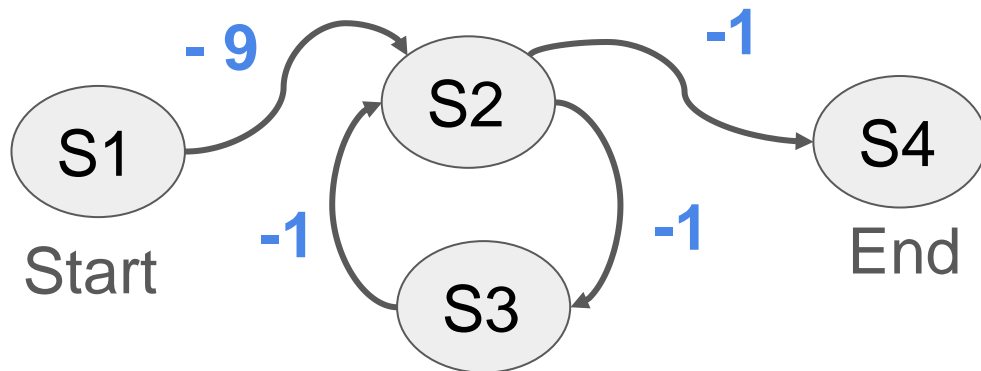
**Take away:** reward only for **WHAT**, but never for **HOW**



# Reward design – don't shift, reward for WHAT

- **E.g.:** chess – value of taken opponent's piece
  - **Problem:** agent will not have a desire to win!
- **E.g.:** moving to destination
  - **Problem:** agent will not bother about the goal!

**Take away:** reward only for **WHAT**, but never for **HOW**



**Take away:** do not **subtract** mean from rewards

# Faulty reward functions

- Reward for ball possession in soccer
  - Vibrating near the ball
- Cyclic behaviours





# Reward design – scaling, shaping

## What transformations do not change optimal policy?

- Reward **scaling** – division by positive constant
  - May be useful in practise for approximate methods

# Reward design – scaling, shaping

## What transformations do not change optimal policy?

- Reward **scaling** – division by positive constant
  - May be useful in practise for approximate methods
- Reward **shaping** – add a **potential-based shaping function**  $F(s, a, s')$ :

$$R'(s, a, s') = R(s, a, s') + F(s, a, s')$$

$$F(s, a, s') = \gamma \Phi(s') - \Phi(s)$$

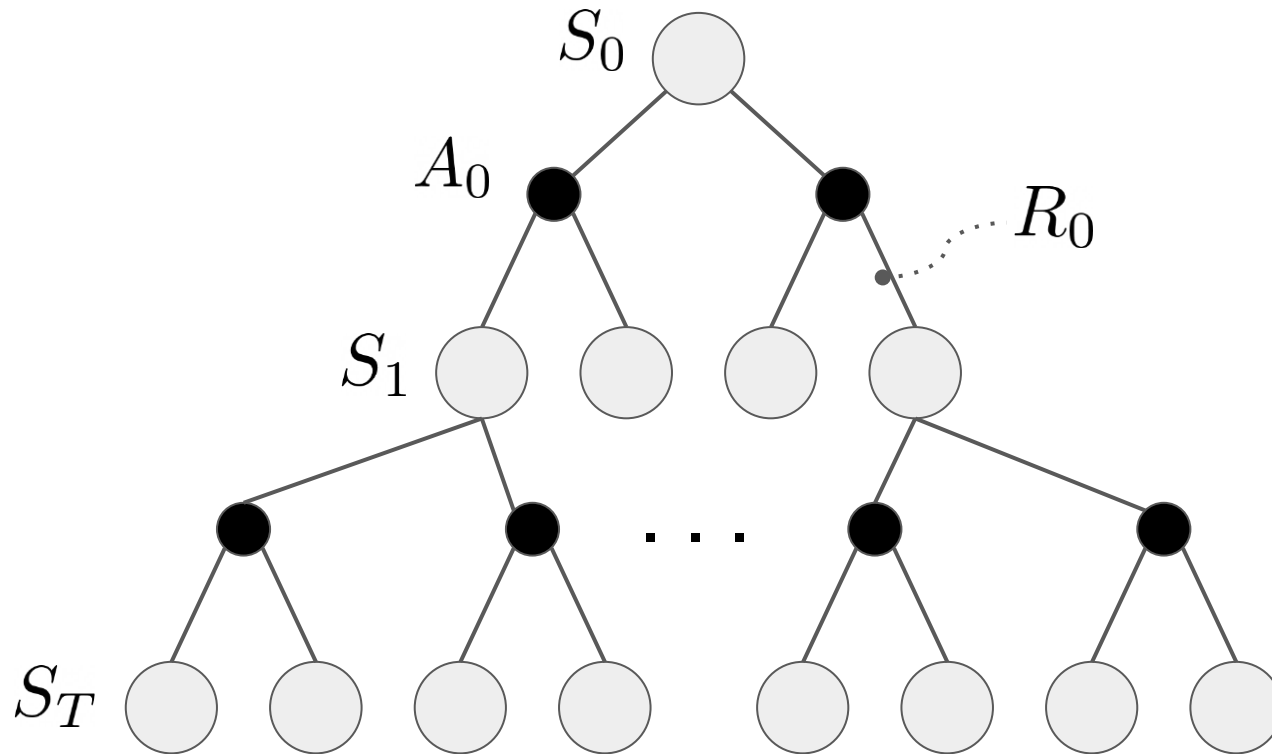
**Intuition:** when no discounting  $F$  adds as much as it subtracts from the total return

Expected objective

Optimal policy maximizes **expected** return

$$\begin{aligned}\mathbb{E}[G_0] &= \mathbb{E}[R_0 + \gamma R_1 + \dots + \gamma^T R_T] \\&= \mathbb{E}_{E, \pi_\theta}[G_0] \\&= \mathbb{E}_{\pi_\theta}[G_0] \\&= \mathbb{E}[G_0 \mid \pi_\theta] \\&= \mathbb{E}_{\substack{s_{0:T} \\ a_{0:T}}}[G_0] \\&= \mathbb{E}_{s_0} \left[ \mathbb{E}_{a_0|s_0} \left[ R_0 + \mathbb{E}_{s_1|s_0, a_0} \left[ \mathbb{E}_{a_1|s_1} [\gamma R_1 + \dots] \right] \right] \right] \\&= \sum_{t=0}^T \mathbb{E}_{(s_t, a_t) \sim p_\theta} [\gamma^t R_t] \\&= \mathbb{E}_{\tau \sim p_\theta(\tau)} [G(\tau)] \qquad \tau \triangleq (s_0, a_0, s_1, \dots, a_{T-1}, s_T) \\&\qquad p_\theta(\tau) = p(s_0) \prod_{t=0}^{T-1} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t)\end{aligned}$$

# Backup Tree: how to find an optimal policy?



# **State-** and **Action-**value functions

# State-value function $v(s)$

$v(s)$  is expected **return** conditional on state:

$$\begin{aligned} v_{\pi}(s) &\triangleq \mathbb{E}_{\pi} [G_t \mid S_t = s] \\ &= \mathbb{E}_{\pi} [R_t + \gamma G_{t+1} \mid S_t = s] \\ &= \sum_a \pi(a \mid s) \sum_{r, s'} p(r, s' \mid s, a) \left[ r + \gamma \mathbb{E}_{\pi} [G_{t+1} \mid S_{t+1} = s'] \right] \\ &= \sum_a \pi(a \mid s) \sum_{r, s'} p(r, s' \mid s, a) [r + \gamma v_{\pi}(s')] \end{aligned}$$

**Intuition:** value of following policy  $\pi$  from state  $s$

# State-value function $v(s)$

$v(s)$  is expected **return** conditional on state:

— stochasticity in policy & environment

$$\begin{aligned} v_{\pi}(s) &\triangleq \mathbb{E}_{\pi}[G_t \mid S_t = s] \\ &= \mathbb{E}_{\pi}[R_t + \gamma G_{t+1} \mid S_t = s] \\ &= \sum_a \pi(a \mid s) \sum_{r, s'} p(r, s' \mid s, a) \left[ r + \gamma \mathbb{E}_{\pi}[G_{t+1} \mid S_{t+1} = s'] \right] \\ &= \sum_a \pi(a \mid s) \sum_{r, s'} p(r, s' \mid s, a) [r + \gamma v_{\pi}(s')] \end{aligned}$$

**Intuition:** value of following policy  $\pi$  from state  $s$



# State-value function $v(s)$

$v(s)$  is expected **return** conditional on state:

stochasticity in policy & environment

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

Environment  
stochasticity

$$= \mathbb{E}_{\pi} [R_t + \gamma G_{t+1} \mid S_t = s]$$

$$= \sum_a \pi(a \mid s) \sum_{r, s'} p(r, s' \mid s, a) \left[ r + \gamma \mathbb{E}_{\pi} [G_{t+1} \mid S_{t+1} = s'] \right]$$

Policy  
stochasticity

$$= \sum_a \pi(a \mid s) \sum_{r, s'} p(r, s' \mid s, a) [r + \gamma v_{\pi}(s')]$$

**Intuition:** value of following policy  $\pi$  from state  $s$

# State-value function $v(s)$

$v(s)$  is expected **return** conditional on state:

stochasticity in policy & environment

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

Environment  
stochasticity

$$= \mathbb{E}_{\pi} [R_t + \gamma G_{t+1} | S_t = s]$$

$$= \sum_a \pi(a | s) \sum_{r, s'} p(r, s' | s, a) \left[ r + \gamma \mathbb{E}_{\pi} [G_{t+1} | S_{t+1} = s'] \right]$$

Policy  
stochasticity

$$= \sum_a \pi(a | s) \sum_{r, s'} p(r, s' | s, a) [r + \gamma v_{\pi}(s')]$$

By definition

**Intuition:** value of following policy  $\pi$  from state  $s$

# Action-value function $q(s, a)$

Is expected **return** conditional on state and action:

**Intuition:** value of following policy  $\pi$  after committing action **a** in state **s**

$$\begin{aligned} q_{\pi}(s, a) &= \mathbb{E}_{\pi} [G_t \mid S_t = s, A_t = a] \\ &= \mathbb{E}_{\pi} [R_t + \gamma G_{t+1} \mid S_t = s, A_t = a] \\ &= \sum_{r, s'} p(r, s' \mid s, a) \left[ r + \gamma \mathbb{E}_{\pi} [G_{t+1} \mid S_{t+1} = s'] \right] \\ &= \sum_{r, s'} p(r, s' \mid s, a) [r + \gamma v_{\pi}(s')] \end{aligned}$$

# Action-value function $q(s, a)$

Is expected **return** conditional on state and action:

**Intuition:** value of following policy  $\pi$  after committing action **a** in state **s**

$$\begin{aligned} q_{\pi}(s, a) &= \mathbb{E}_{\pi} [G_t \mid S_t = s, A_t = a] \\ &= \mathbb{E}_{\pi} [R_t + \gamma G_{t+1} \mid S_t = s, A_t = a] \\ &= \sum_{r, s'} p(r, s' \mid s, a) \left[ r + \gamma \mathbb{E}_{\pi} [G_{t+1} \mid S_{t+1} = s'] \right] \\ &= \sum_{r, s'} p(r, s' \mid s, a) [r + \gamma v_{\pi}(s')] \end{aligned}$$

No policy stochasticity at first step

## Relations between $v(s)$ and $q(s,a)$

We already know how to write  $q(s,a)$  in terms of  $v(s)$

$$q_{\pi}(s, a) = \sum_{r, s'} p(r, s' | s, a) [r + \gamma v_{\pi}(s')]$$

What about  $v(s)$  in terms of  $q(s,a)$ ?

## Relations between $v(s)$ and $q(s,a)$

We already know how to write  $q(s,a)$  in terms of  $v(s)$

$$q_{\pi}(s, a) = \sum_{r, s'} p(r, s' | s, a) [r + \gamma v_{\pi}(s')]$$

What about  $v(s)$  in terms of  $q(s,a)$ ?

$$\begin{aligned} v_{\pi}(s) &= \sum_a \pi(a | s) \sum_{r, s'} p(r, s' | s, a) [r + \gamma v_{\pi}(s')] \\ &= \sum_a \pi(a | s) q_{\pi}(s, a) \end{aligned}$$

So, we could now write  $q(s, a)$  in terms of  $q(s,a)$ !

$$q_{\pi}(s, a) = \sum_{r, s'} p(r, s' | s, a) \left[ r + \gamma \sum_{a'} \pi(a' | s') q_{\pi}(s', a') \right]$$

Bellman expectation equations

# Bellman **expectation** equations

**For  $v(s)$ :**

$$\begin{aligned} v_{\pi}(s) &= \sum_a \pi(a | s) \sum_{r, s'} p(r, s' | s, a) [r + \gamma v_{\pi}(s')] \\ &= \mathbb{E}_{\pi} [R_t + \gamma v_{\pi}(S_{t+1}) | S_t = s] \end{aligned}$$

**For  $q(s, a)$ :**

$$\begin{aligned} q_{\pi}(s, a) &= \sum_{r, s'} p(r, s' | s, a) [r + \gamma v_{\pi}(s')] \\ &= \sum_{r, s'} p(r, s' | s, a) \left[ r + \gamma \sum_{a'} \pi(a' | s') q_{\pi}(s', a') \right] \end{aligned}$$



# What do we gonna do with value functions?

Already know

- Return, value- and action-value functions
- Bellman equations – assess policy performance

Optimal policy makes

- best actions in each possible state

But how to know which policy **is better**?

How to compare them?

Bellman optimality equations


Optimal policy is the one with the largest  $v(s)$


We could compare policies on the basis of  $v(s)$

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

Best policy  $\pi_*$  is better or equal to any other policy

Use optimal policy from  $s$

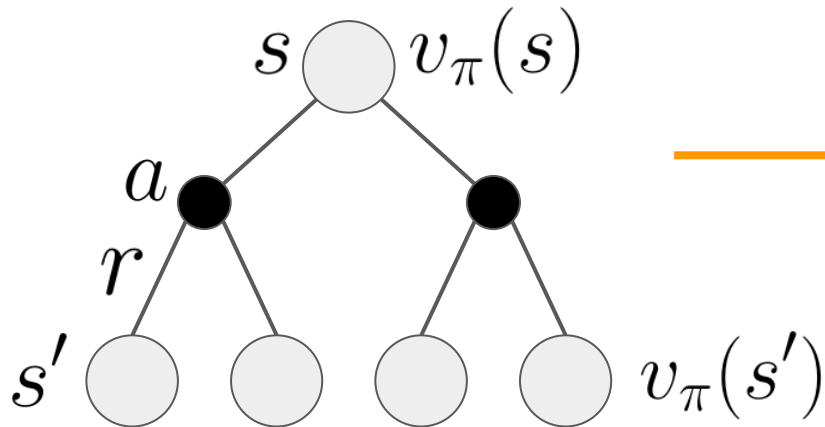

$$v_*(s) = \max_{\pi} v_\pi(s)$$


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

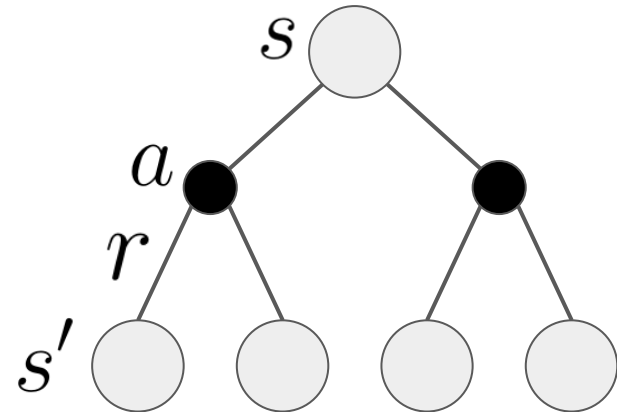
In any finite MDP there is  
always **at least one**  
deterministic optimal policy

Commit action  $a$ , and **afterwards** use optimal policy

# Bellman **optimality** equation for $v(s)$

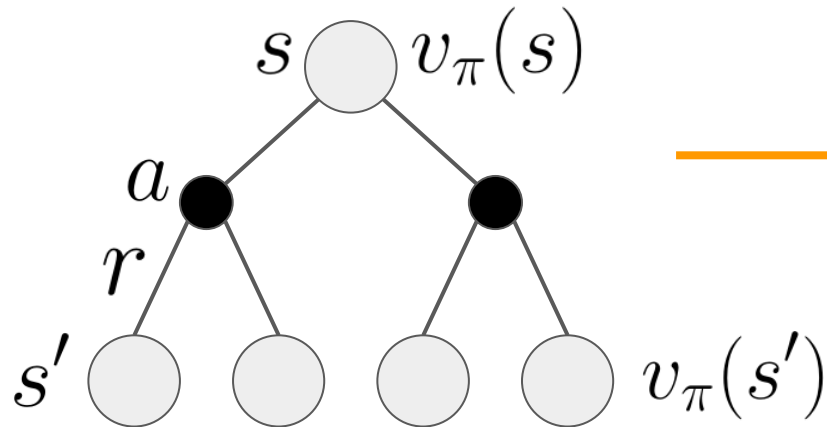


Bellman **expectation**  
equation for  $v(s)$

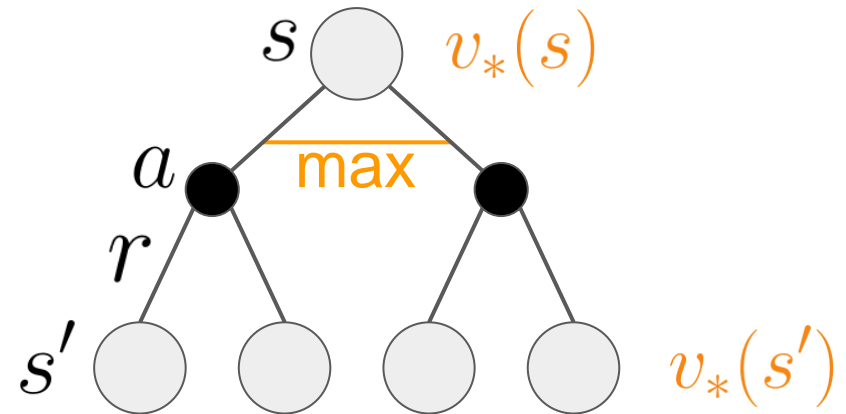


Bellman **optimality**  
equation for  $v_*(s)$

# Bellman **optimality** equation for $v(s)$

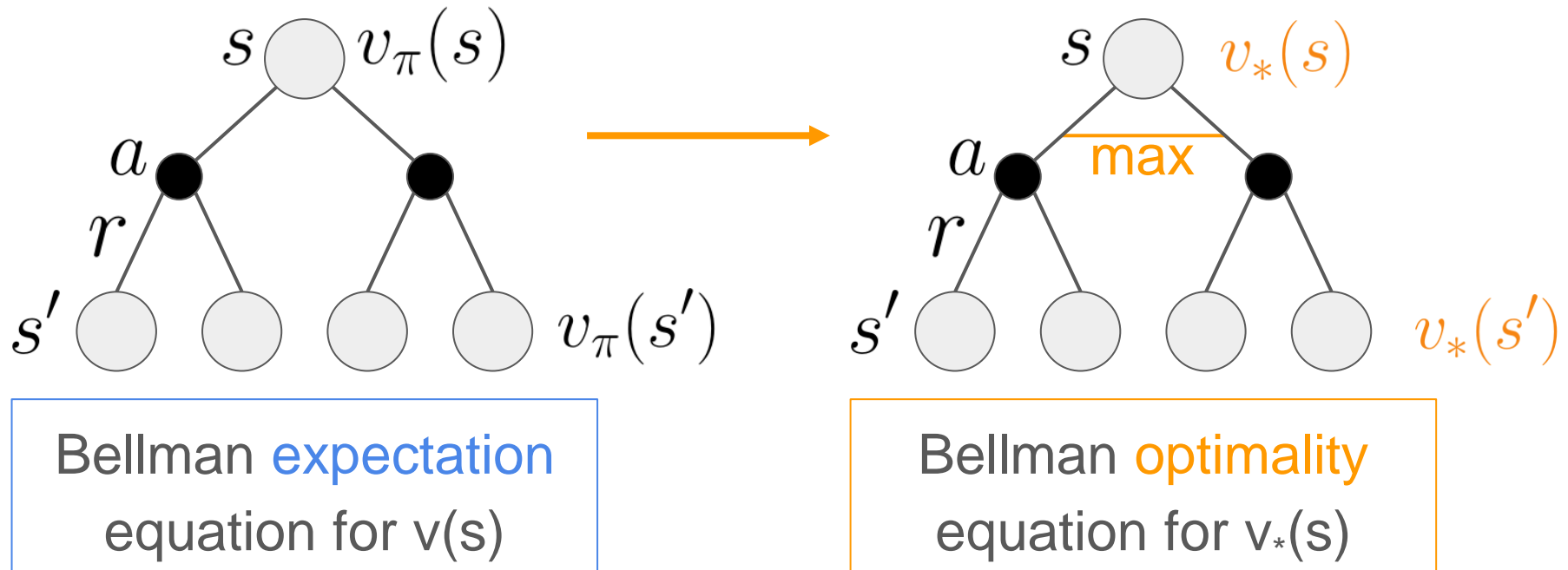


Bellman **expectation**  
equation for  $v(s)$



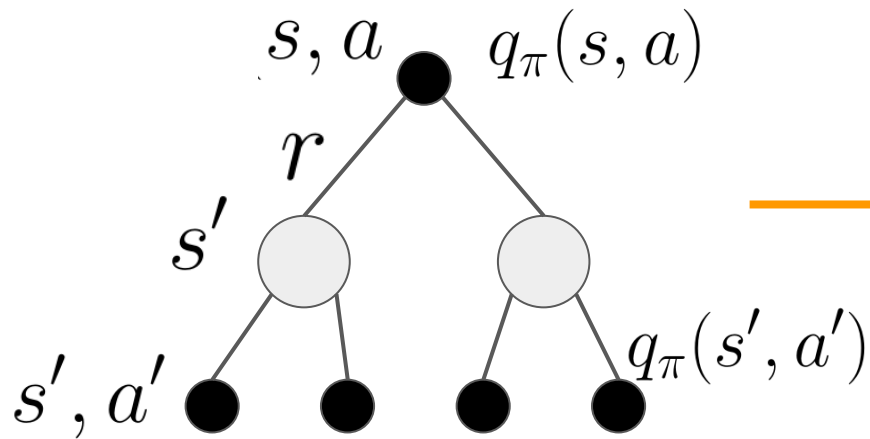
Bellman **optimality**  
equation for  $v_*(s)$

# Bellman **optimality** equation for $v(s)$

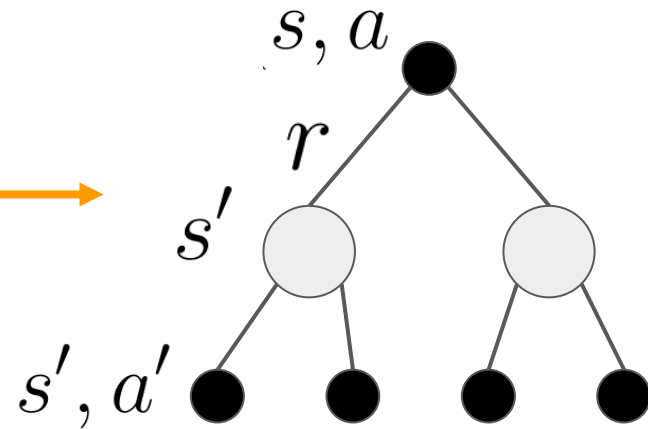


$$\begin{aligned} v_*(s) &= \max_a \sum_{r, s'} p(r, s' | s, a) [r + \gamma v_*(s')] \\ &= \max_a \mathbb{E} [R_t + \gamma v_*(S_{t+1}) | S_t = s, A_t = a] \end{aligned}$$

# Bellman **optimality** equation for $q(s,a)$

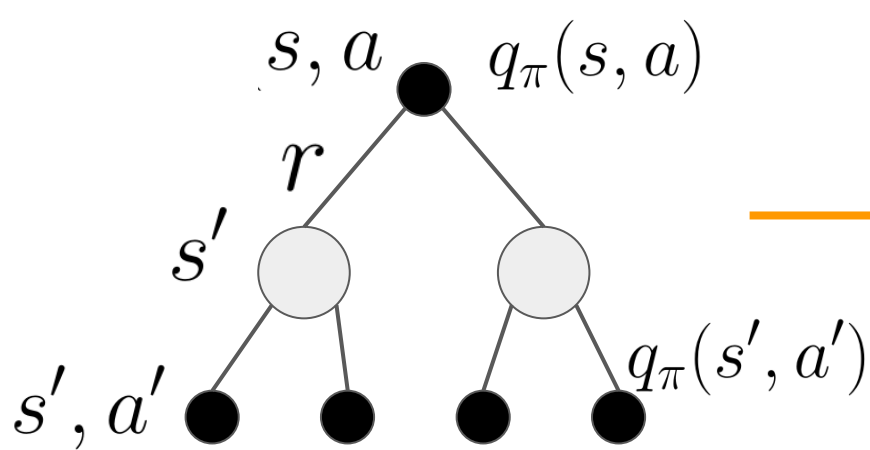


Bellman **expectation**  
equation for  $q(s,a)$

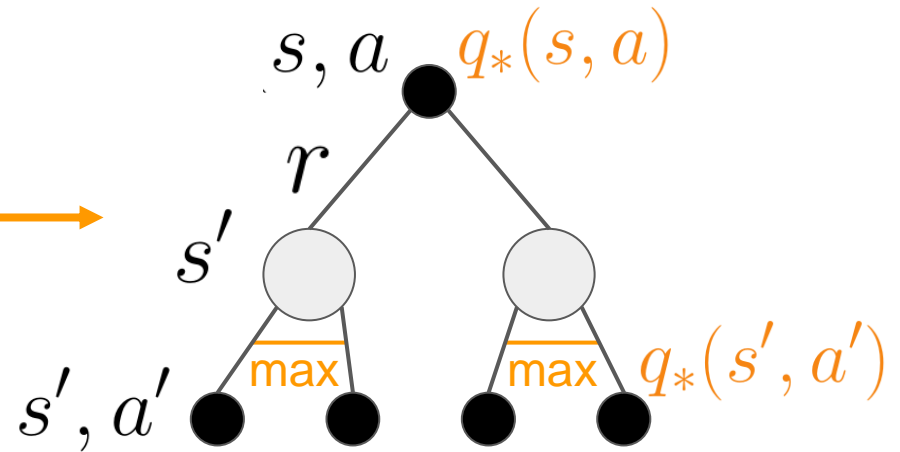


Bellman **optimality**  
equation for  $q_*(s, a)$

# Bellman **optimality** equation for $q(s,a)$



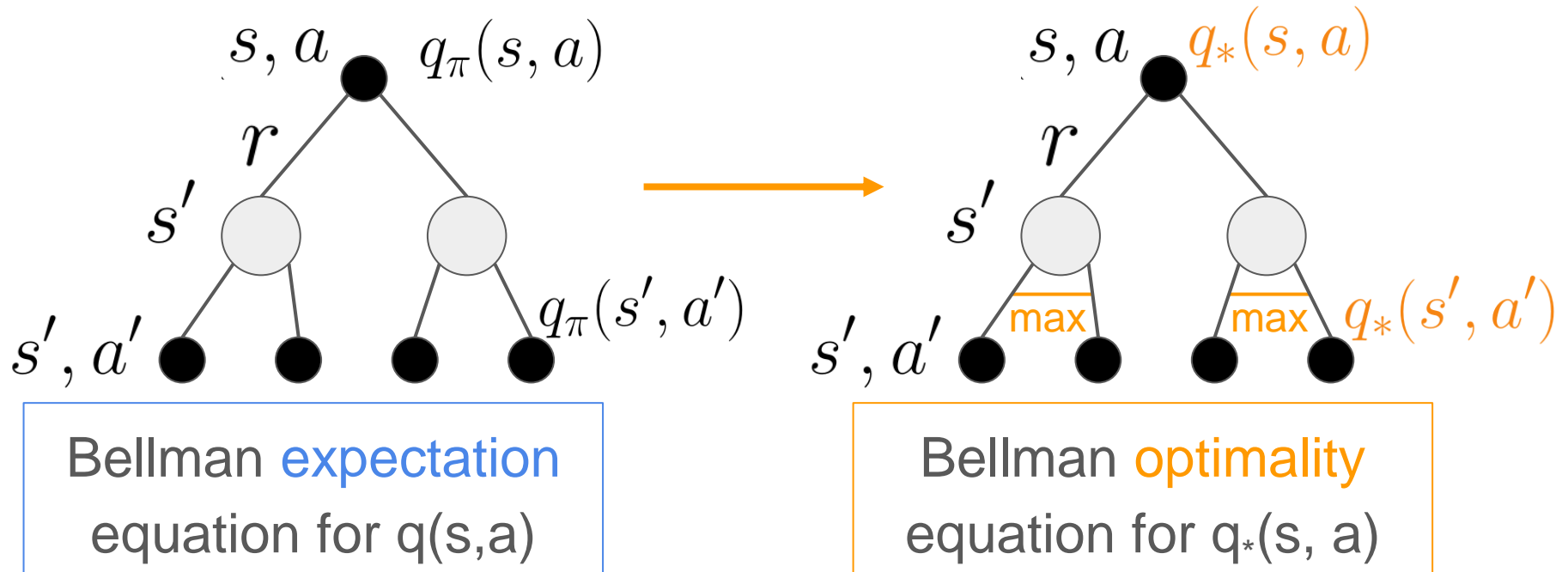
Bellman **expectation**  
equation for  $q(s,a)$



Bellman **optimality**  
equation for  $q_*(s, a)$



# Bellman **optimality** equation for $q(s,a)$



$$\begin{aligned} q_*(s, a) &= \mathbb{E} \left[ R_t + \gamma \max_{a'} q_*(S_{t+1}, a') \mid S_t = s, A_t = a \right] \\ &= \sum_{r, s'} p(r, s' \mid s, a) \left[ r + \gamma \max_{a'} q_*(s', a') \right] \end{aligned}$$

## Bellman equations: operator view

# Bellman equations: operator view

$$[\mathcal{T}^\pi V](s) = \mathbb{E}_{r,s'|s,a=\pi(s)} [r + \gamma V(s')] ]$$

$$[\mathcal{T}^\pi Q](s, a) = \mathbb{E}_{r,s'|s,a} [r + \gamma \mathbb{E}_{a' \sim \pi(s')} [Q(s', a')]] ]$$

$$[\mathcal{T}V](s) = \max_a \mathbb{E}_{r,s'|s,a} [r + \gamma V(s')] ]$$

$$[\mathcal{T}Q](s, a) = \mathbb{E}_{r,s'|s,a} \left[ r + \gamma \max_{a'} Q(s', a') \right]$$

# Bellman equations: operator view

Bellman **expectation** equation for  $\mathbf{v}(\mathbf{s})$

$$[\mathcal{T}^\pi V](s) = \mathbb{E}_{r,s'|s,a=\pi(s)} [r + \gamma V(s')] ]$$

Bellman **expectation** equation for  $\mathbf{q}(\mathbf{s},\mathbf{a})$

$$[\mathcal{T}^\pi Q](s, a) = \mathbb{E}_{r,s'|s,a} [r + \gamma \mathbb{E}_{a' \sim \pi(s')} [Q(s', a')]] ]$$

Bellman **optimality** equation for  $\mathbf{v}_*(\mathbf{s})$

$$[\mathcal{T}V](s) = \max_a \mathbb{E}_{r,s'|s,a} [r + \gamma V(s')] ]$$

Bellman **optimality** equation for  $\mathbf{q}_*(\mathbf{s},\mathbf{a})$

$$[\mathcal{T}Q](s, a) = \mathbb{E}_{r,s'|s,a} \left[ r + \gamma \max_{a'} Q(s', a') \right]$$

# Bellman equations are **contractions**

Contraction:  $d(\mathcal{T}(v), \mathcal{T}(u)) \leq \gamma d(v, u), \quad 0 \leq \gamma < 1$

Operator:  $[\mathcal{T}v](s) = \max_a \mathbb{E}_{r, s'|s, a} [r + \gamma v(s')]$

Denote  $a^* = \arg \max_a \mathbb{E}_{r, s'|s, a} [r + \gamma v(s')]$ , then, for any s:

$$\begin{aligned} [\mathcal{T}v](s) - [\mathcal{T}u](s) &\leq r(s, a^*) + \gamma \mathbb{E}_{s'|s, a^*} [v(s')] - r(s, a^*) - \gamma \mathbb{E}_{s'|s, a^*} [u(s')] \\ &= \gamma \mathbb{E}_{s'|s, a^*} [v(s') - u(s')] \\ &\leq \gamma \mathbb{E}_{s'|s, a^*} [|v(s') - u(s')|] \\ &\leq \gamma \max_{s'} |v(s') - u(s')| \\ &= \gamma \|v - u\|_\infty \end{aligned}$$

Hence, taking max over s:

$$\|\mathcal{T}v - \mathcal{T}u\|_\infty \leq \gamma \|v - u\|_\infty$$

# Generalized Policy Iteration:

1. Policy Evaluation
2. Policy Improvement

# **Policy evaluation**

# Policy evaluation: motivation

Policy evaluation is also called **prediction problem**:

- predict value function for a particular policy.

Bellman **expectation** equation

$$\begin{aligned} v_{\pi}(s) &= \sum_a \pi(a | s) \sum_{r, s'} p(r, s' | s, a) [r + \gamma v_{\pi}(s')] \\ &= \mathbb{E}_{\pi} [R_t + \gamma v_{\pi}(S_{t+1}) | S_t = s] \end{aligned}$$

is basically a system of linear equations where

- # of unknowns = # of equations = # of states



# Policy evaluation: algorithm

Input  $\pi$ , the policy to be evaluated

Initialize an array  $V(s) = 0$ , for all  $s \in \mathcal{S}^+$

Repeat

$\Delta \leftarrow 0$

For each  $s \in \mathcal{S}$ :

$v \leftarrow V(s)$

$V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) [r + \gamma V(s')]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

until  $\Delta < \theta$  (a small positive number)

Output  $V \approx v_\pi$

Bellman **expectation**  
equation for  $v(s)$

# Policy improvement

# Policy improvement: an idea

Once we know what is  $v(s)$  for a particular policy

We could improve it by acting greedily w.r.t.  $q(s, a)$ !

$$\pi'(s) \leftarrow \underset{a}{\operatorname{arg\,max}} \sum_{r, s'} \overbrace{p(r, s' \mid s, a) [r + \gamma v_{\pi}(s')]}^{q_{\pi}(s, a)}$$

This procedure is guaranteed to produce a better policy!

# Policy improvement: an idea

Once we know what is  $v(s)$  for a particular policy

We could improve it by acting greedily w.r.t.  $q(s, a)$ !

$$\pi'(s) \leftarrow \underset{a}{\operatorname{arg\,max}} \sum_{r, s'} \overbrace{p(r, s' | s, a) [r + \gamma v_{\pi}(s')]}^{q_{\pi}(s, a)}$$

This procedure is guaranteed to produce a better policy!

if  $q_{\pi}(s, \pi'(s)) \geq v_{\pi}(s)$  for all states

then  $v_{\pi'}(s) \geq v_{\pi}(s)$

meaning that  $\pi' \geq \pi$

# Policy improvement: convergence

If new policy after improvement

$$\pi'(s) \leftarrow \arg \max_a \overbrace{\sum_{r, s'} p(r, s' | s, a) [r + \gamma v_\pi(s')]}^{q_\pi(s, a)}$$

is the same as the old one

$$\pi' = \pi \quad \rightarrow \quad v_{\pi'} = v_\pi$$

then it is optimal, since it satisfies:

$$v_{\pi'}(s) = \max_a \sum_{r, s'} p(r, s' | s, a) [r + \gamma v_\pi(s')]$$

# Policy improvement: convergence

If new policy after improvement

$$\pi'(s) \leftarrow \arg \max_a \overbrace{\sum_{r, s'} p(r, s' | s, a) [r + \gamma v_\pi(s')]}^{q_\pi(s, a)}$$

is the same as the old one

$$\pi' = \pi \rightarrow v_{\pi'} = v_\pi$$

then it is optimal, since it satisfies:

$$v_{\pi'}(s) = \max_a \sum_{r, s'} p(r, s' | s, a) [r + \gamma v_\pi(s')]$$

Bellman  
optimality  
equation

# Determining optimal policy from $v_*(s)$ , $q_*(s,a)$

If  $q_*$  is known – how to recover the optimal policy?

$$\pi_*(s) \leftarrow \underset{a}{\text{arg max}} \ q_*(s, a)$$

If  $v_*$  is known – how to recover the optimal policy?

# Determining optimal policy from $v_*(s)$ , $q_*(s,a)$

If  $q_*$  is known – how to recover the optimal policy?

$$\pi_*(s) \leftarrow \underset{a}{\text{arg max}} q_*(s, a)$$

If  $v_*$  is known – how to recover the optimal policy?

$$\pi_*(s) \leftarrow \underset{a}{\text{arg max}} \sum_{r, s'} \overbrace{p(r, s' | s, a) [r + \gamma v_*(s')]}^{q_*(s, a)}$$

Unknown model dynamics → unable to recover optimal policy from  $v_*$



Precise evaluation is excessive











# Value function

0 iteration

	0.000	0.000	0.000
0.000	0.000	0.000	0.000
0.000	0.000	0.000	

# Greedy policy

0 iteration

# Value function

0 iteration

	0.000	0.000	0.000
0.000	0.000	0.000	0.000
0.000	0.000	0.000	

5 iteration











	-7.598	-4.986	-3.127
-7.816	-5.834	-2.963	0.543
-6.115	-4.186	0.332	

9999 iteration

	-13.827	-13.289	-11.318
-14.768	-14.193	-10.722	-5.346
-16.111	-13.454	-6.059	

# Greedy policy

0 iteration

5 iteration

9999 iteration

# Roadmap

Now we know what is

- Policy evaluation (based on Bellman **expectation** eq)
- Policy improvement (based on Bellman **optimality** eq)

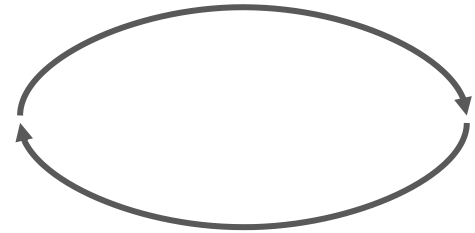
The finishing touches:

how to combine them to obtain optimal policy?

# **Generalized Policy Iteration**

# The idea of policy and value iterations

Policy evaluation



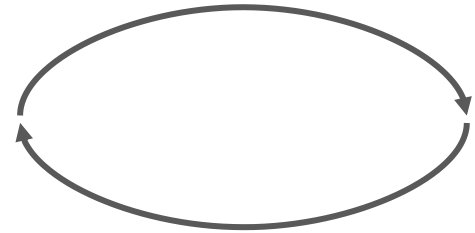
Policy improvement

# The idea of policy and value iterations

## Generalized policy iteration

1. Evaluate given policy
2. Improve policy by acting greedily w.r.t. to its value function

Policy evaluation



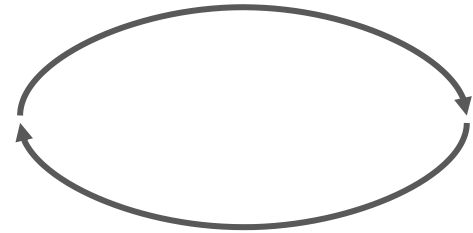
Policy improvement

# The idea of policy and value iterations

## Generalized policy iteration

1. Evaluate given policy
2. Improve policy by acting greedily w.r.t. to its value function

Policy evaluation



Policy improvement

Robustness:

- No dependence on initialization
- No need in complete policy evaluation (states / converg.)
- No need in exhaustive update (states)
  - Example of update robustness:
    - Update only one state at a time
    - in a random direction
    - that is correct only in a expectation

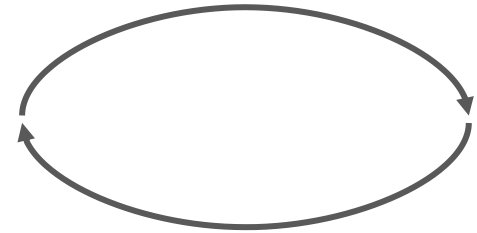


# The idea of policy and value iterations

## Generalized policy iteration

1. Evaluate given policy
2. Improve policy by acting greedily w.r.t. to its value function

Policy evaluation



Policy improvement

## Policy iteration

1. Evaluate policy until convergence (with some tolerance)
2. Improve policy

## Value iteration

1. Evaluate policy only with single iteration
2. Improve policy

# Policy iteration

# Policy iteration: scheme

## 1. Initialization

$V(s) \in \mathbb{R}$  and  $\pi(s) \in \mathcal{A}(s)$  arbitrarily for all  $s \in \mathcal{S}$

## 2. Policy Evaluation

Repeat

$\Delta \leftarrow 0$

For each  $s \in \mathcal{S}$ :

$v \leftarrow V(s)$

$V(s) \leftarrow \sum_{s',r} p(s',r|s, \pi(s)) [r + \gamma V(s')]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

until  $\Delta < \theta$  (a small positive number)

Bellman expectation  
equation for  $v(s)$

## 3. Policy Improvement

$\text{policy-stable} \leftarrow \text{true}$

For each  $s \in \mathcal{S}$ :

$\text{old-action} \leftarrow \pi(s)$

$\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s',r|s, a) [r + \gamma V(s')]$

If  $\text{old-action} \neq \pi(s)$ , then  $\text{policy-stable} \leftarrow \text{false}$

If  $\text{policy-stable}$ , then stop and return  $V \approx v_*$  and  $\pi \approx \pi_*$ ; else go to 2

$q(s,a)$

Value iteration

# Value iteration

Initialize array  $V$  arbitrarily (e.g.,  $V(s) = 0$  for all  $s \in \mathcal{S}^+$ )

Repeat

$$\Delta \leftarrow 0$$

For each  $s \in \mathcal{S}$ :

$$v \leftarrow V(s)$$

$$V(s) \leftarrow \max_a \sum_{s',r} p(s', r | s, a) [r + \gamma V(s')]$$

$$\Delta \leftarrow \max(\Delta, |v - V(s)|)$$

until  $\Delta < \theta$  (a small positive number)

Bellman **optimality**  
equation for  $v(s)$

Output a deterministic policy,  $\pi \approx \pi_*$ , such that

$$\pi(s) = \operatorname{argmax}_a \sum_{s',r} p(s', r | s, a) [r + \gamma V(s')]$$

# Value iteration (VI) vs. Policy iteration (PI)

- VI is **faster** per iteration –  $O(|A||S|^2)$
- VI requires **many** iterations
- PI is **slower** per iteration –  $O(|A||S|^2 + |S|^3)$
- PI requires **few** iterations

**No silver bullet** → experiment with # of steps spent in policy evaluation phase to find the best