# Lecture 9: Value based methods

**Radoslav Neychev**

Harbour.Space University
18.07.2019, Barcelona, Spain

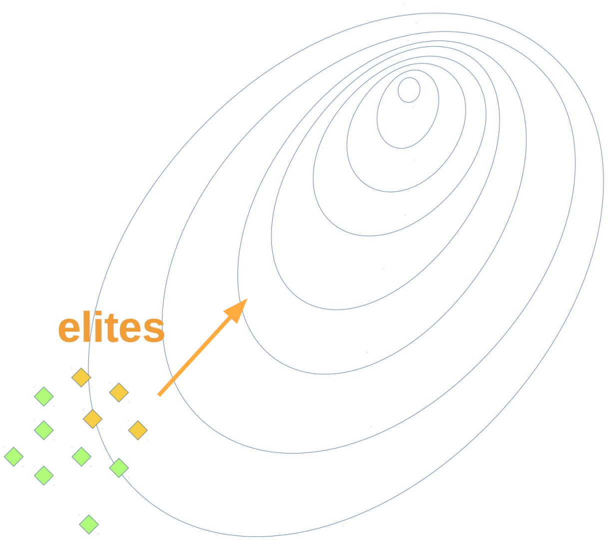
# References

These slides are almost the exact copy of Practical RL course week 2 slides by Shvechikov Pavel.
Special thanks to YSDA team for making them publicly available.

Original slides link: week02_value_based

# 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



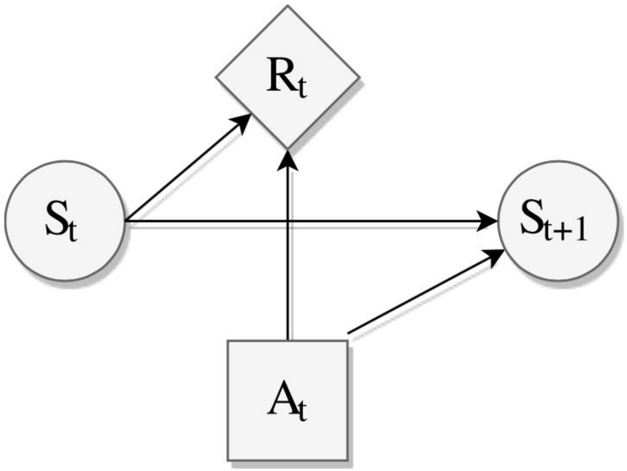

Improve on the CEM → dive into the black box

# 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. $\mathcal{P} : \mathcal{S} \times \mathcal{A} \mapsto \triangle(\mathcal{S})$ – state-transition function, giving us $p(s_{t+1} \mid s_t, a_t)$
4. $\mathcal{R} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ – reward function, giving us $\mathbb{E}_R \left[ R(s_t, a_t) \mid s_t, a_t \right]$.




**Markov property**

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

(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:**

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

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**:

end of an episode

$$\boxed{G_t} \triangleq \boxed{R_t} + R_{t+1} + R_{t+2} + \ldots + R_{\boxed{T}}$$

immediate reward

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!

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

# E.g.: cleaning robot

- **S**tates – dust sensors, air
- **A**ctions – cleaning / rest / conditioning on or off
- **R = 100** for long tedious floor cleaning task done
- **R = 1** for turning air conditioning on-off
- Episode ends each day

What could go wrong with such a design?

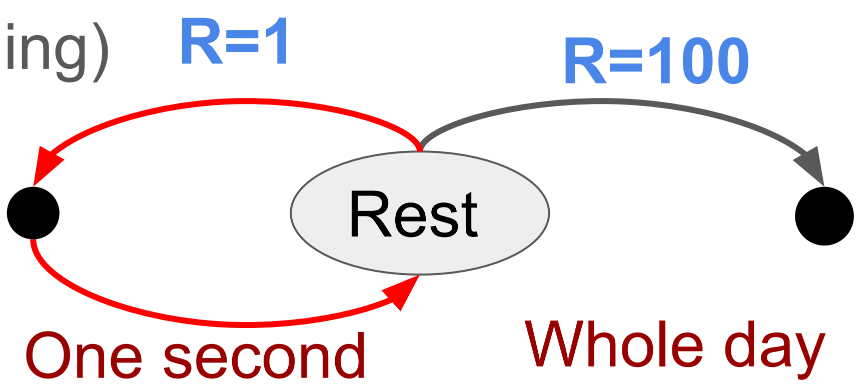
# E.g.: cleaning robot

- **S**tates – dust sensors, air
- **A**ctions – cleaning / rest / conditioning on or off
- **R = 100** for long tedious floor cleaning task done
- **R = 1** for turning air conditioning on-off
- Episode ends each day

What could go wrong with such a design?

Reward(air) < Reward(cleaning)
Time(air) << Time(cleaning)

Positive feedback loop!




OpenAI blog post about faulty rewards: https://openai.com/blog/faulty-reward-functions/

# Reward discounting

# Reward discounting

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

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

discount factor

The same cake compared to today's one worth

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

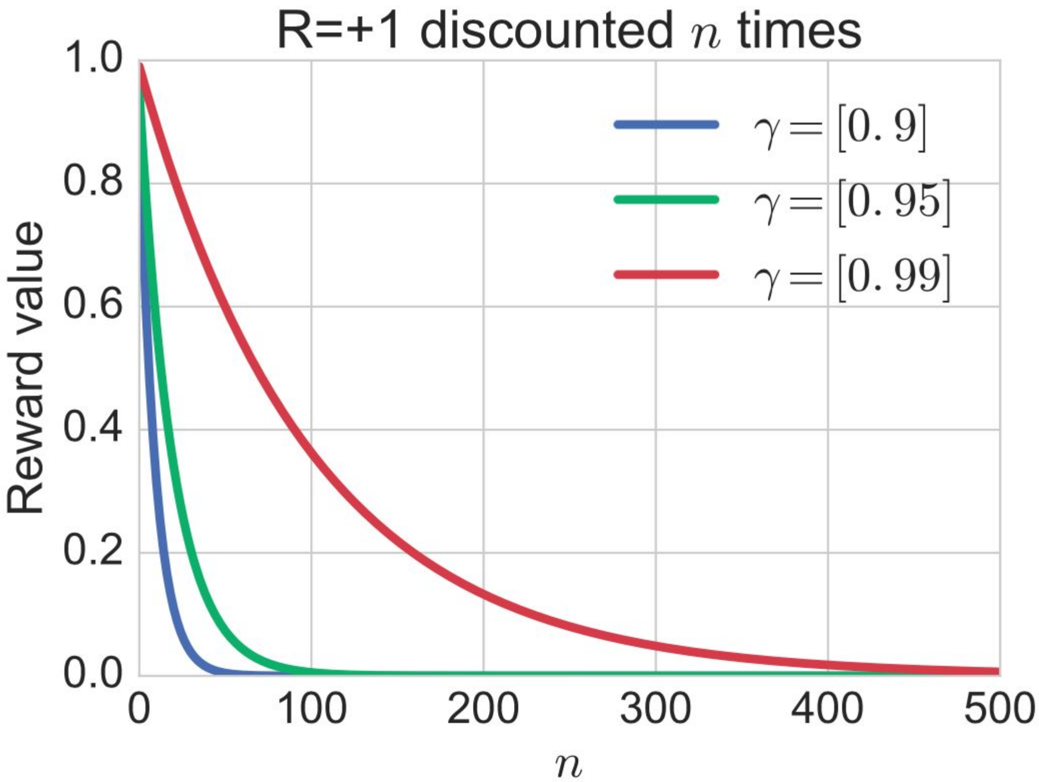
$\gamma$ will eat it day by day

# 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 |



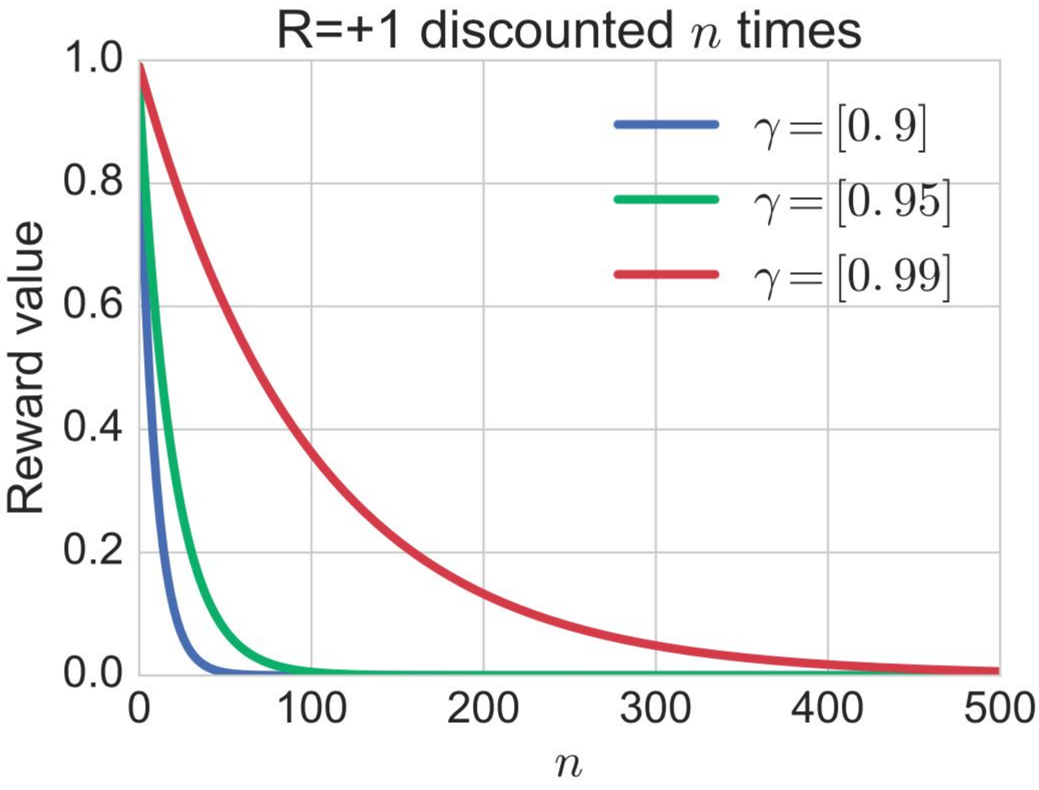
# Discounting makes sums finite

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 $f(t) = \dfrac{1}{1 + \beta t}$

Laibson, D. (1997). Golden eggs and hyperbolic discounting. The Quarterly Journal of Economics, 112(2), 443-478.

# Discounting is inherent to humans

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

Mathematical convenience

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

Remember this one!
We will need it later

Laibson, D. (1997). Golden eggs and hyperbolic discounting. The Quarterly Journal of Economics, 112(2), 443-478.

# 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?

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

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?

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

“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

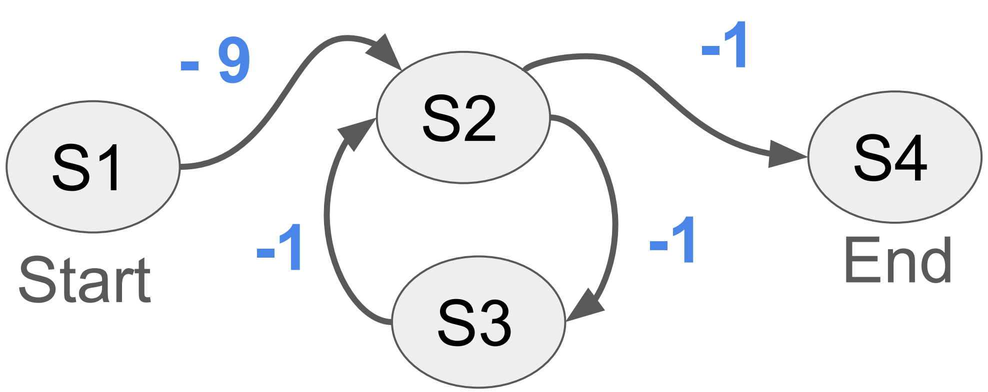
- E.g.: chess – value of taken opponent's piece
  - Problem: agent will not have a desire to win!
- E.g.: cleaning robot, **+100** (cleaning), **+0.1** (on-off)
  - Problem: agent will not bother cleaning the floor!

# 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.: cleaning robot, **+100** (cleaning), **+0.1** (on-off)
  - Problem: agent will not bother cleaning the floor!

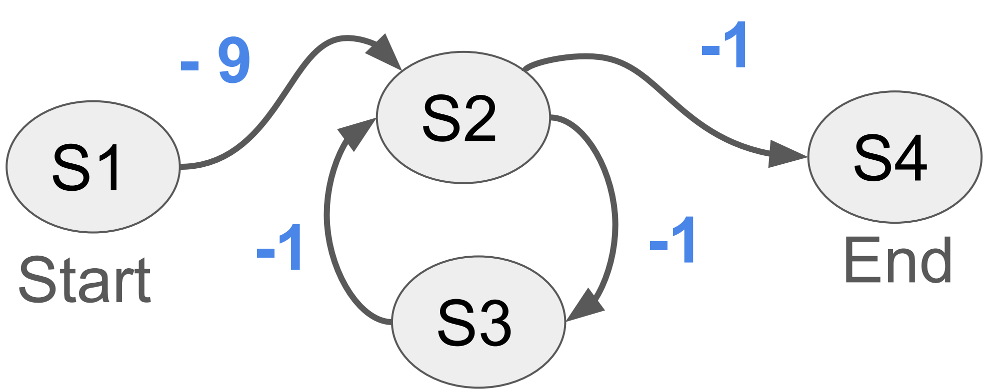
**Take away**: reward only for WHAT, but never for HOW



# 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.: cleaning robot, **+100** (cleaning), **+0.1** (on-off)
  - Problem: agent will not bother cleaning the floor!

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




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

# 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

Ng, A. Y., Harada, D., & Russell, S. (1999, June). Policy invariance under reward transformations: Theory and application to reward shaping. In ICML (Vol. 99, pp. 278-287).

# 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** – we could add to all rewards in MDP  values of potential-based shaping function F(s, a, s') without changing an optimal policy:

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

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

Ng, A. Y., Harada, D., & Russell, S. (1999, June). Policy invariance under reward transformations: Theory and application to reward shaping. In ICML (Vol. 99, pp. 278-287).

# Expected objective

# Optimal policy maximizes expected return

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

$$
p_\theta(\tau) = p(s_0) \prod_{t=0}^{T-1} \pi_\theta(a_t|s_t)\; p(s_{t+1}|s_t, a_t)
$$

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

## **State-value** function v(s)

v(s) is expected return conditional on state:

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

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

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

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

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

Environment stochasticity

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

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

Policy stochasticity

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

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

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

Environment stochasticity

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

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

Policy stochasticity

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

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**

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

## **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**

No policy stochasticity at first step

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

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

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

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

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)

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

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

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

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

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

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

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

Recursive definition of v(s) is an important concept in RL

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

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

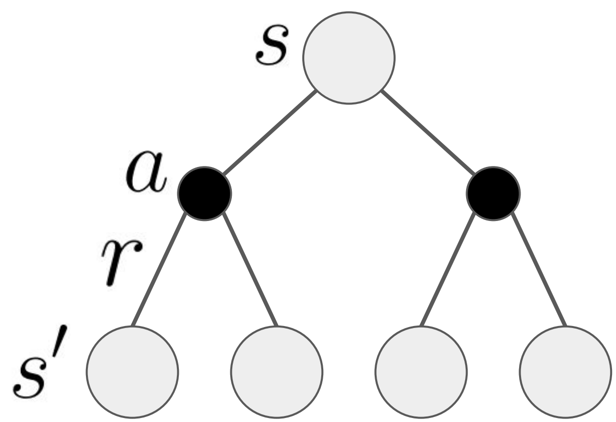
# Bellman expectation equation for **v(s)**

Recursive definition of v(s) is an important concept in RL

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

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

Backup diagram

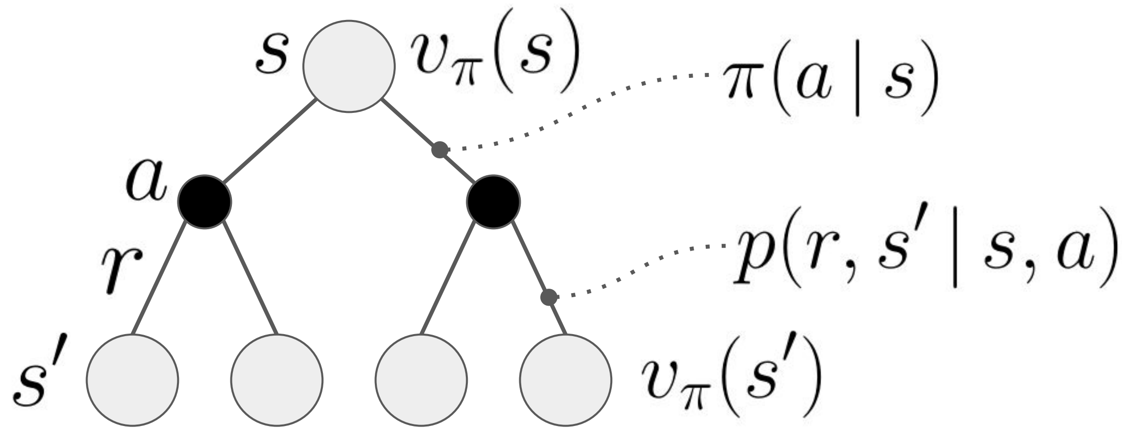


# Bellman expectation equation for **v(s)**

Recursive definition of v(s) is an important concept in RL

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

Backup diagram



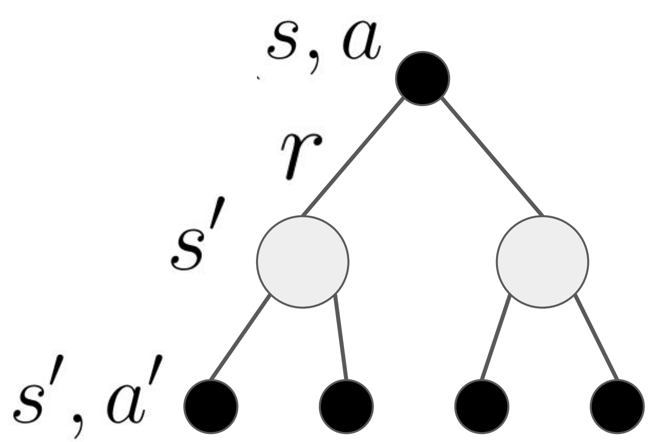
Bellman expectation equation for **q(s,a)**

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

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

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

Backup diagram for q(s, a)

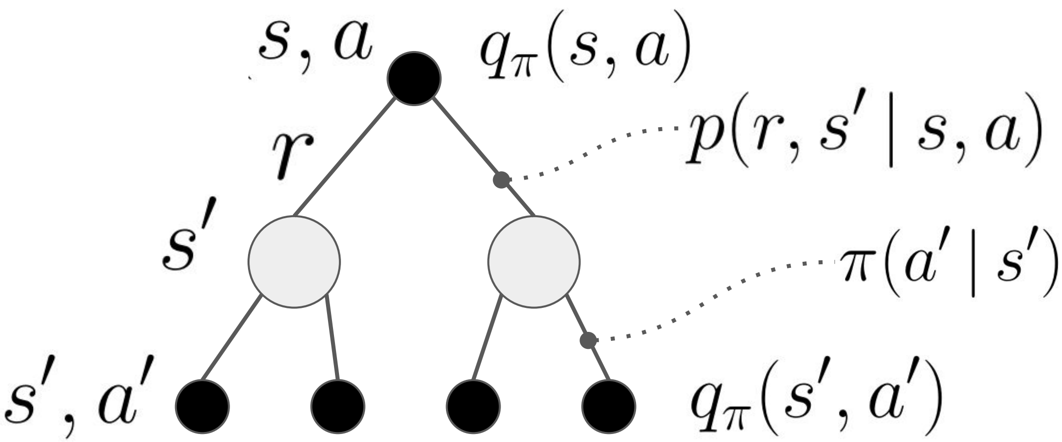


# Bellman expectation equation for **q(s,a)**

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

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

Backup diagram for q(s, a)



# 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)

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

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

Use optimal policy from s

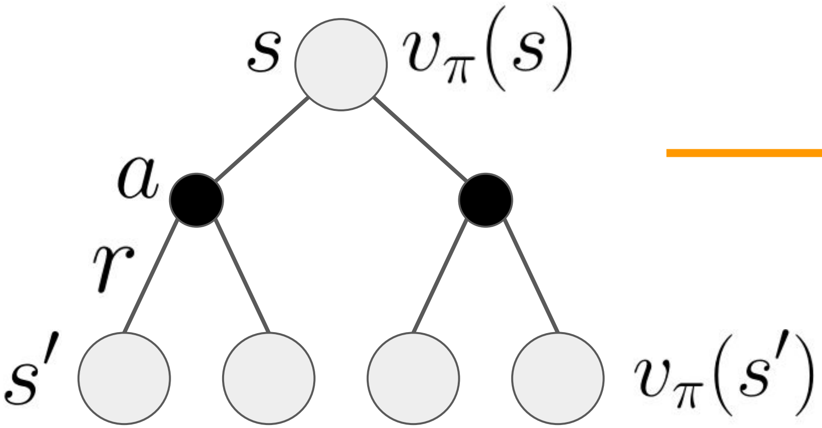
$$\boxed{v_*(s) = \max_\pi v_\pi(s)}$$

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

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

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

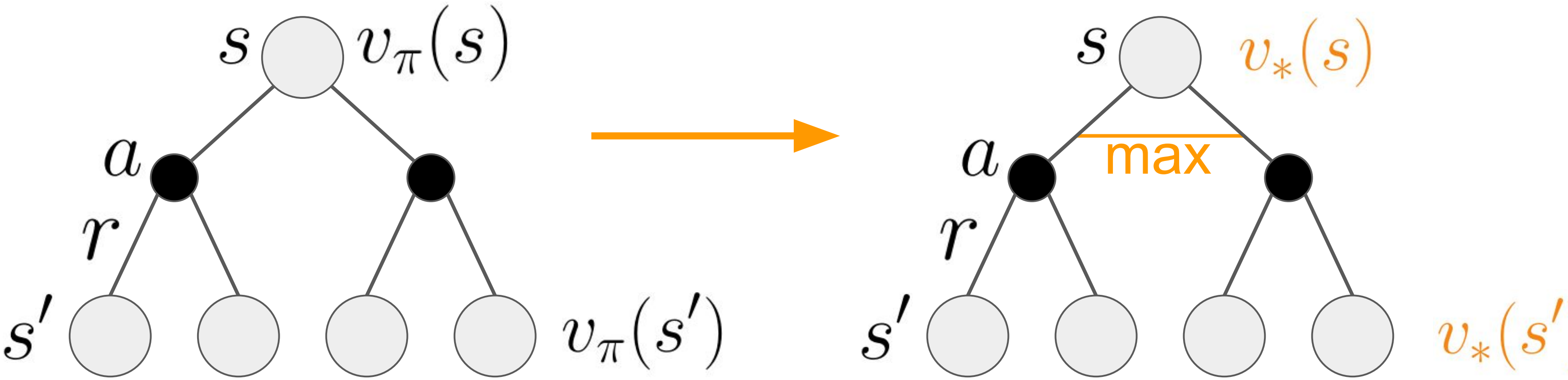
# Bellman optimality equation for **v(s)**




Bellman expectation equation for v(s)

Bellman optimality equation for $v_*(s)$

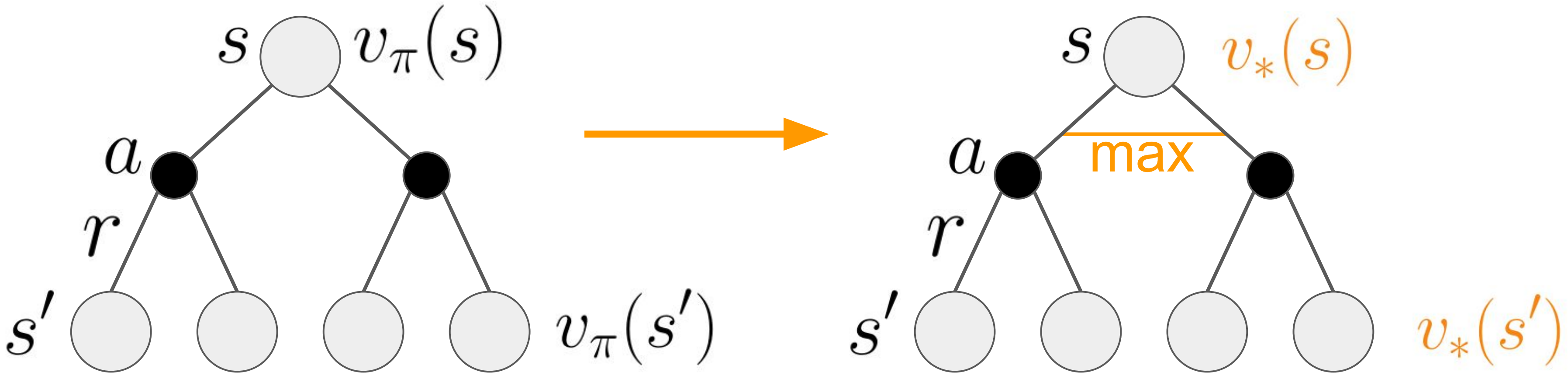
# Bellman optimality equation for **v(s)**




Bellman expectation equation for v(s)

Bellman optimality equation for $v_*(s)$

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

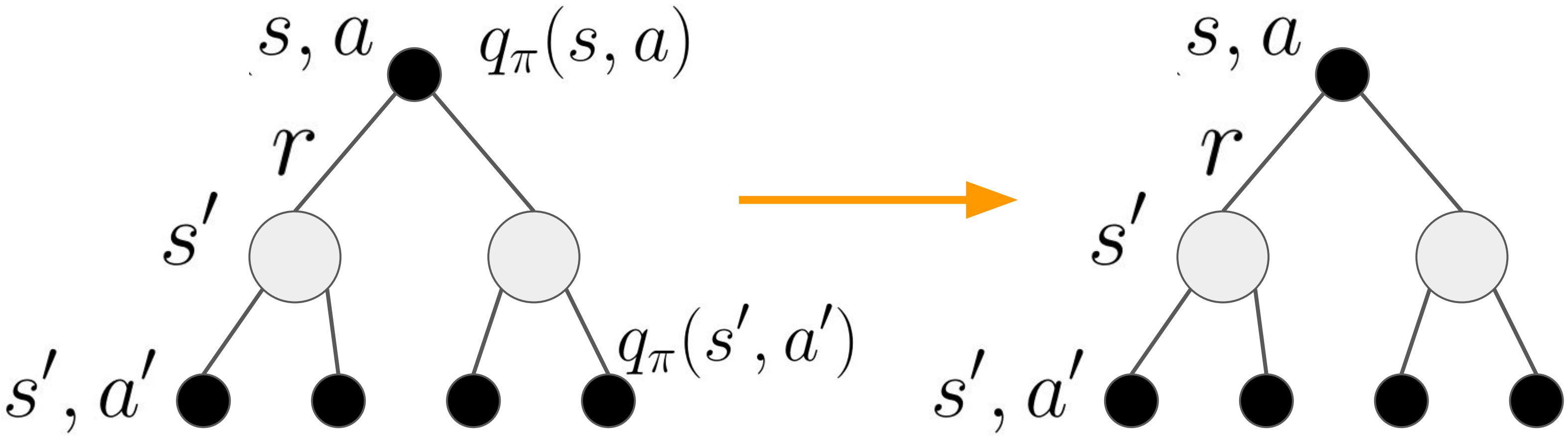



Bellman expectation equation for v(s)

Bellman optimality equation for $v_*(s)$

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

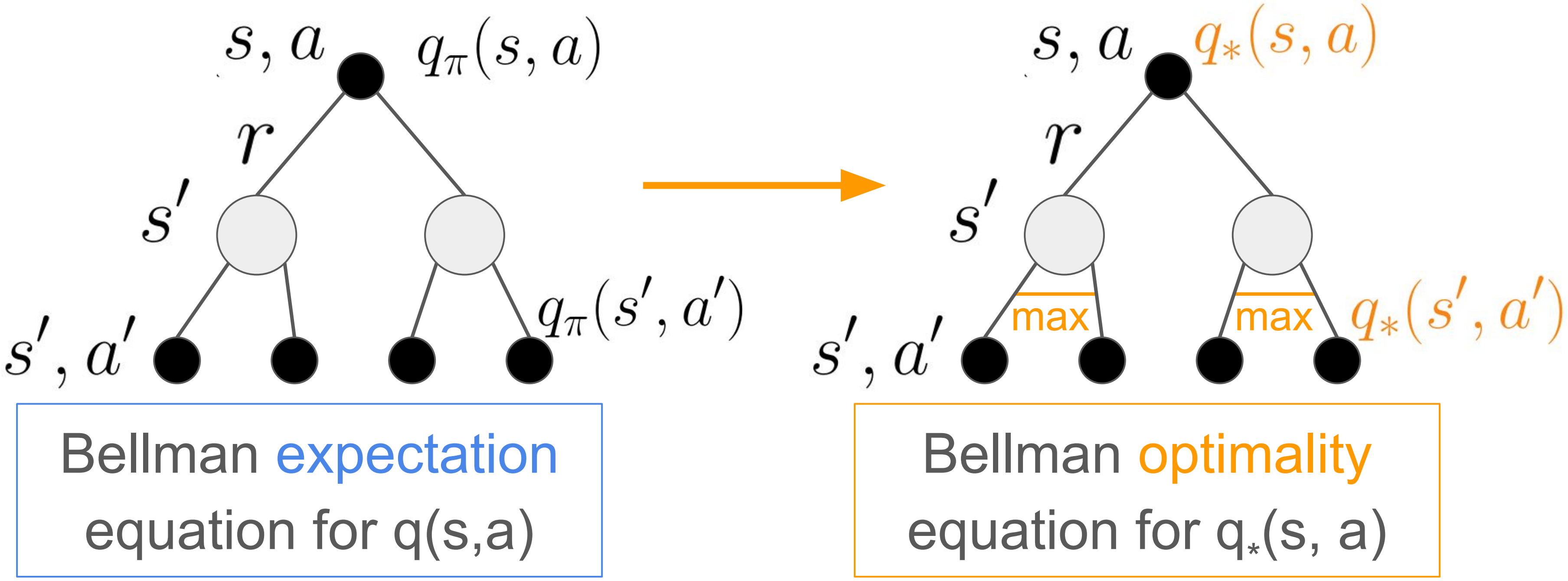
# Bellman optimality equation for **q(s,a)**



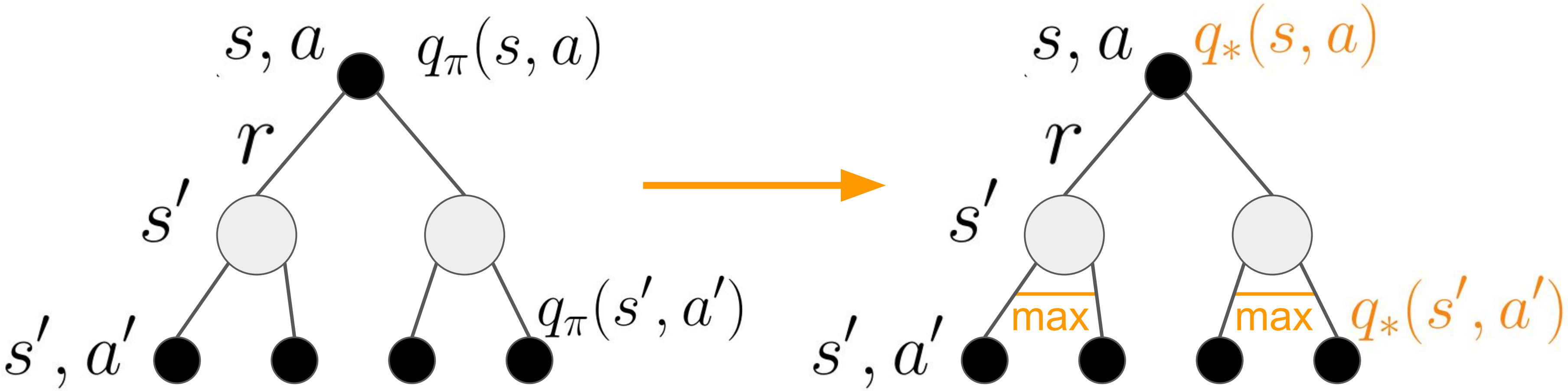

Bellman expectation equation for q(s,a)

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

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



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




Bellman expectation equation for q(s,a)

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

$$q_*(s, a) = \mathbb{E}\left[R_t + \gamma \max_{a'} q_*(S_{t+1}, a') \,\middle|\, S_t = s, A_t = a\right]$$

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

# Bellman equations: operator view

# Bellman equations: operator view

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

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

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

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

# Bellman equations: operator view

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

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

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

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

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

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

Bellman optimality equation for **$q_*$(s,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: $[\mathcal{T}v](s) = \max_a \mathbb{E}_{r,s'|s,a}\left[\, r + \gamma v(s') \,\right]$

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

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

Hence, taking max over s:

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

**G**eneralized **P**olicy **I**teration:

1. Policy Evaluation
2. Policy Improvement

# Policy evaluation

If you can't measure it, you can't improve it.
*Peter Drucker*

# Policy evaluation: motivation

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

- predict value function for a particular policy.

Bellman expectation equation

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

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 $s \in \mathcal{S}$:
        $v \leftarrow V(s)$
        $\boxed{V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]}$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
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. v(s)!

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

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. v(s)!

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

This procedure is guaranteed to produce a better policy!

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

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

meaning that $\quad \pi' \geq \pi$

# Policy improvement: convergence

If new policy after improvement

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

is the same as the old one

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

then it is optimal, since it satisfies:

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

# Policy improvement: convergence

If new policy after improvement

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

is the same as the old one

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

then it is optimal, since it satisfies:

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

Bellman optimality equation

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

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

$$\pi_*(s) \leftarrow \arg\max_a q_*(s, a)$$

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?

$$\pi_*(s) \leftarrow \arg\max_a q_*(s, a)$$

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

$$\pi_*(s) \leftarrow \arg\max_a \overbrace{\sum_{r, s'} \underline{p(r, s' \mid s, a)} \left[r + \gamma v_*(s')\right]}^{q_*(s, a)}$$

Unknown model dynamics → unable to recover optimal policy from v*

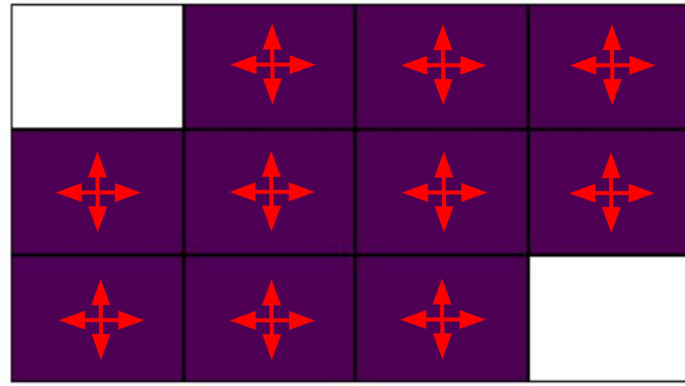
# Precise evaluation is excessive

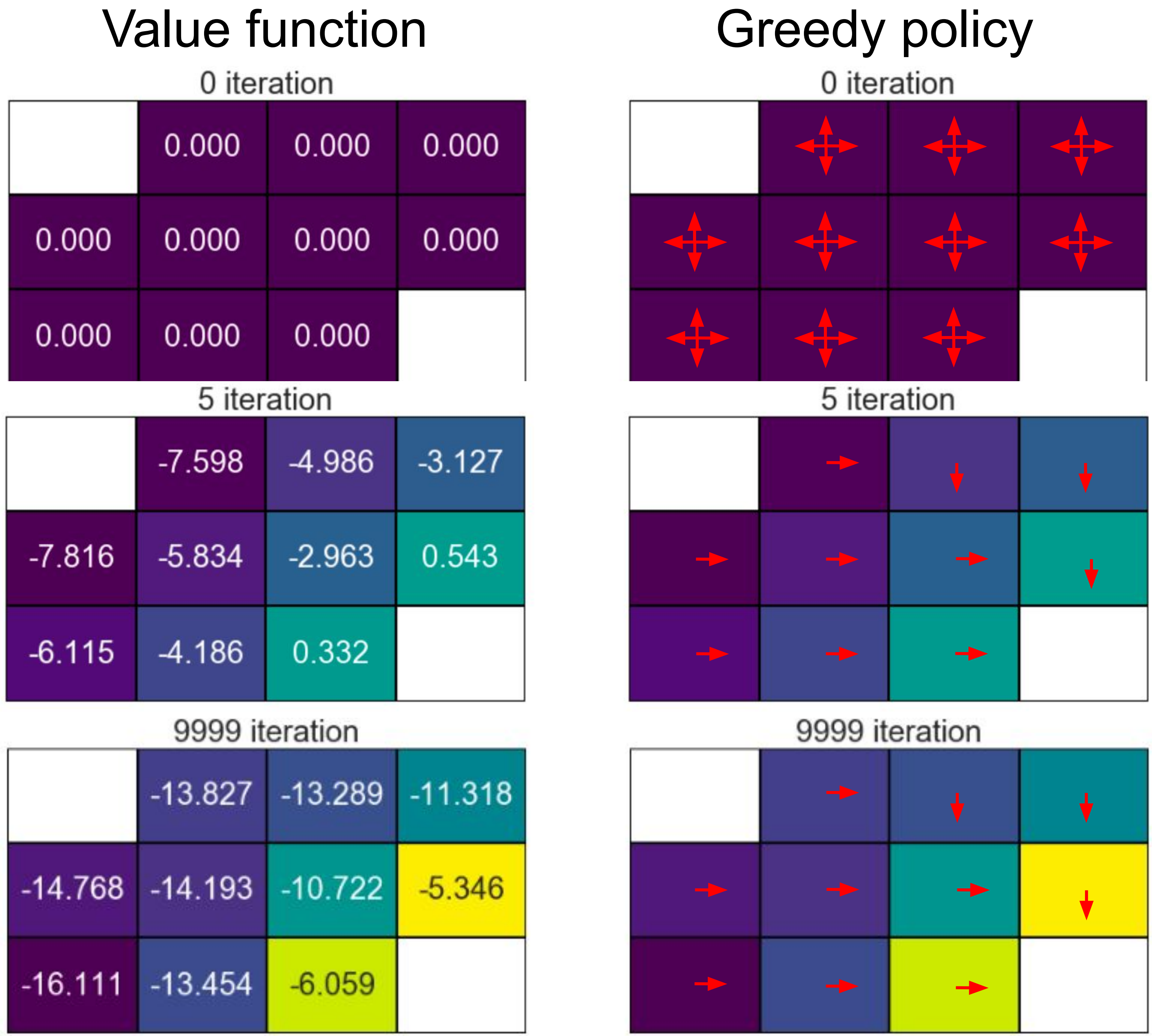
## Value function

0 iteration


## Greedy policy

0 iteration




Value function
Greedy policy
0 iteration
0.000
0.000
0.000
0.000
0.000
0.000
0.000
0.000
0.000
0.000
0 iteration
5 iteration
-7.598
-4.986
-3.127
-7.816
-5.834
-2.963
0.543
-6.115
-4.186
0.332
5 iteration
9999 iteration
-13.827
-13.289
-11.318
-14.768
-14.193
-10.722
-5.346
-16.111
-13.454
-6.059
9999 iteration

# Roadmap

Now we know what is

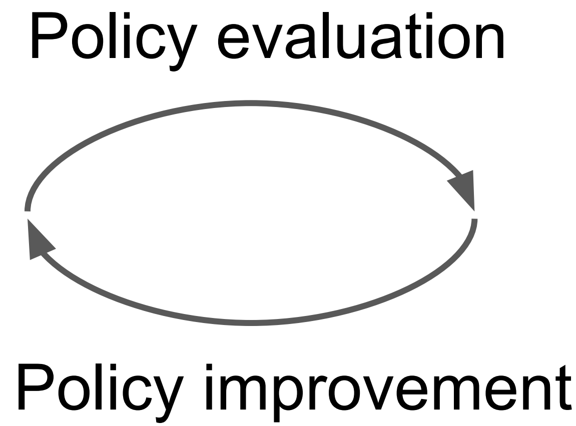
- Policy evaluation (based on Bellman expectation eq)
- Policy improvement (based on Bellman optimality eq)

The finishing touches:

how to combine them to obtain optimal policy?

# **G**eneralized **P**olicy **I**teration

# The idea of policy and value iterations

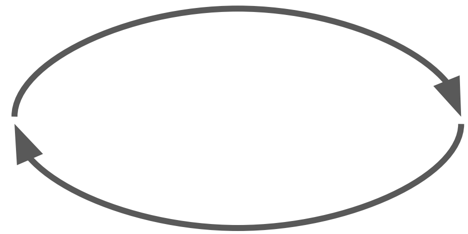


# The idea of policy and value iterations

## Generalized policy iteration

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

Policy evaluation

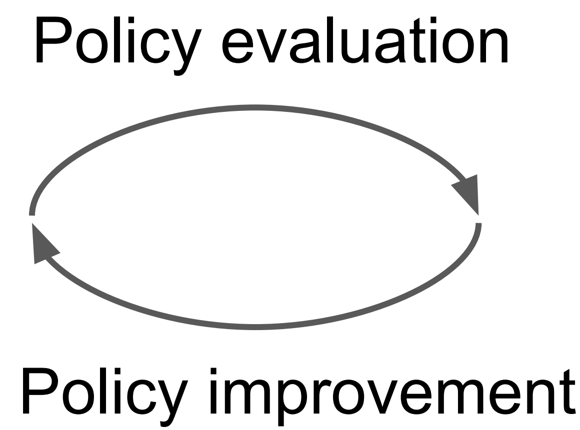


Policy improvement

# 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



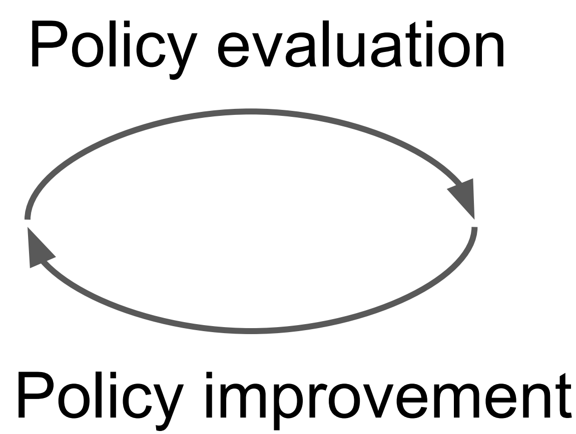

Robustness:

- 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 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 $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $\boxed{V(s) \leftarrow \sum_{s',r} p(s',r|s,\pi(s))\big[r + \gamma V(s')\big]}$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
   *policy-stable* $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   *old-action* $\leftarrow \pi(s)$
   $\pi(s) \leftarrow \operatorname{argmax}_a \boxed{\sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]}$
   If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Bellman expectation equation for v(s)

q(s,a)

# Value iteration

# Value iteration

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

Repeat
$\quad \Delta \leftarrow 0$
$\quad$ For each $s \in \mathcal{S}$:
$\quad\quad v \leftarrow V(s)$
$\quad\quad \boxed{V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]}$
$\quad\quad \Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

Bellman optimality equation for v(s)

Output a deterministic policy, $\pi \approx \pi_*$, such that
$\quad \pi(s) = \arg\max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$

# 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 $\rightarrow$ experiment with # of steps spent in policy evaluation phase to find the best