

# Reinforcement Learning

HSE, autumn - winter 2022

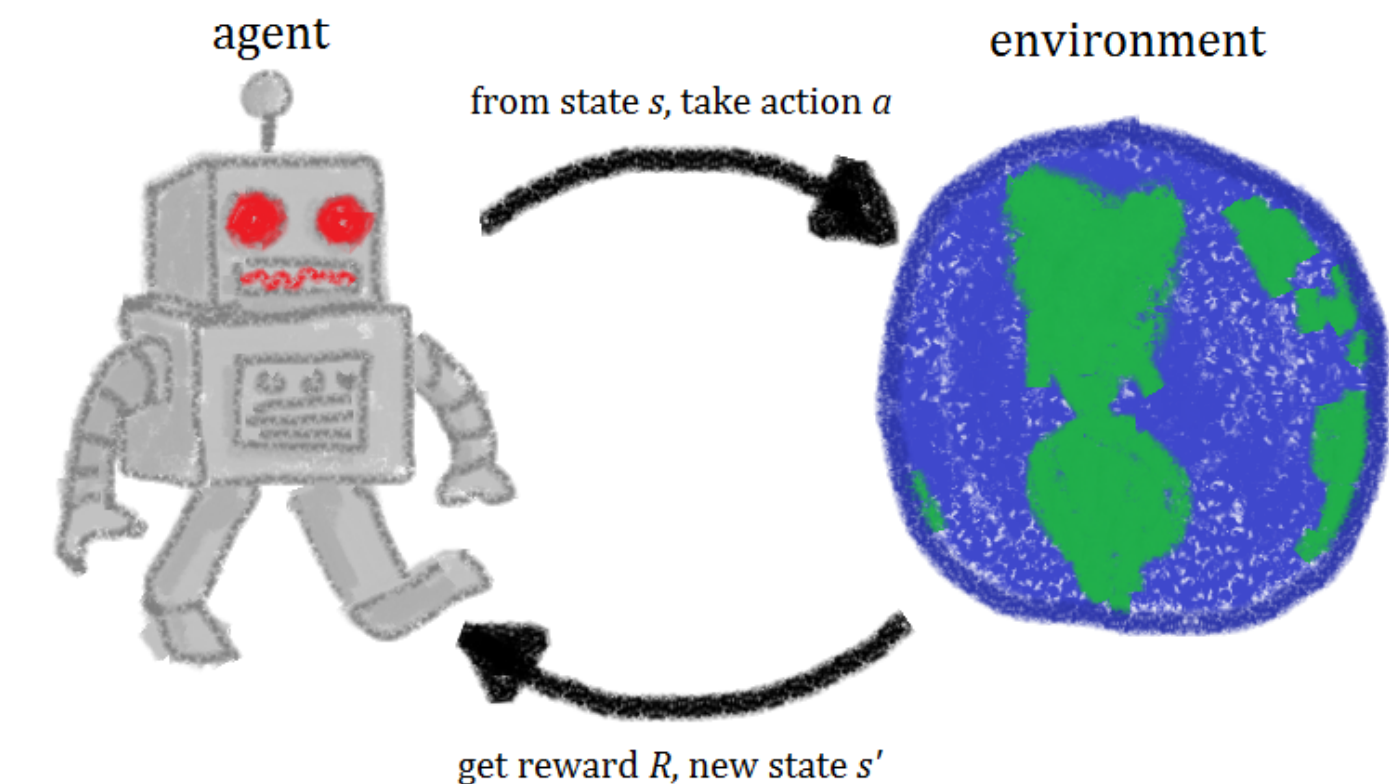
## Lecture 2: Dynamic Programming



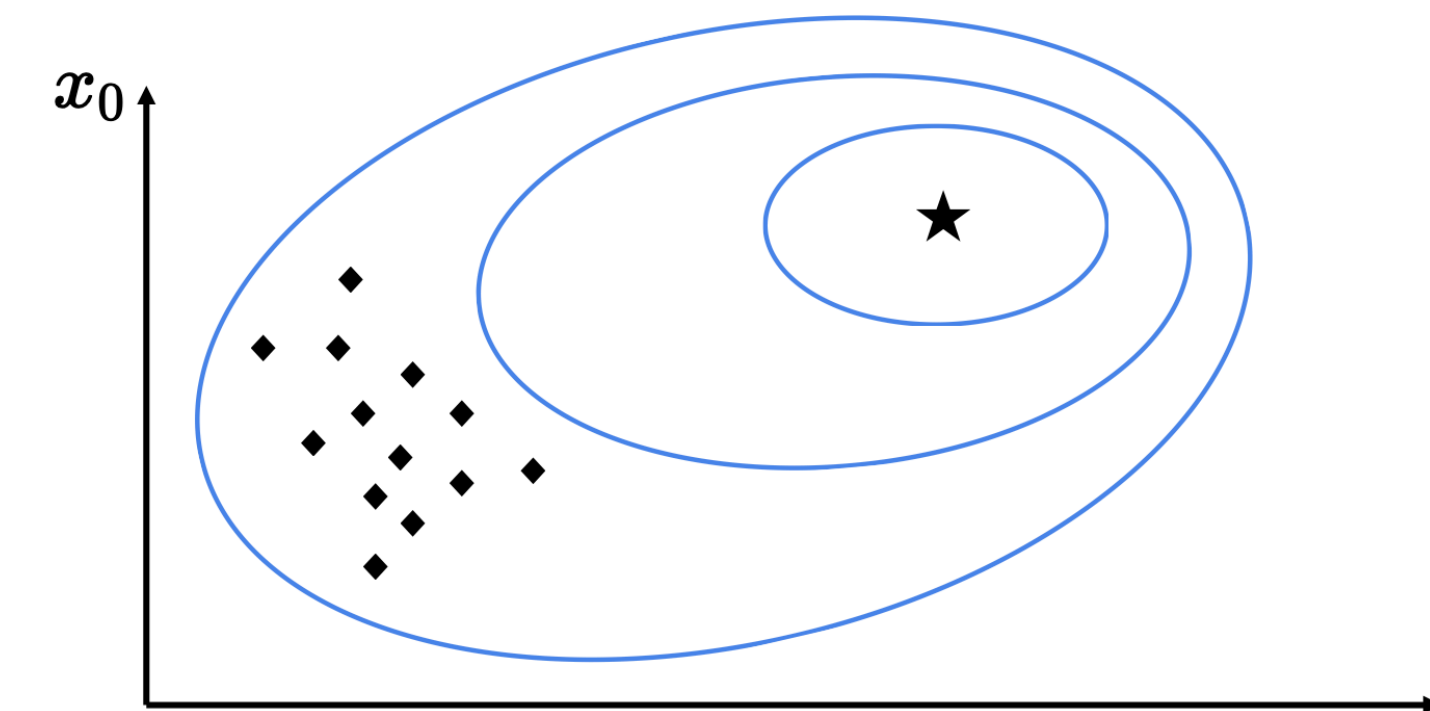
**Sergei Laktionov**  
**[slaktionov@hse.ru](mailto:slaktionov@hse.ru)**  
**[LinkedIn](#)**

# Previously on RL Course

- RL formalism: state (observation), action, reward, MDP, cumulative reward etc.
- (Deep) cross-entropy method:
  - Intuitive idea, rich theoretical background, satisfactory results
  - Zeroth-order **black-box** optimization technique
    - No information regarding the environment rewards and dynamics



Source



Source

However if we really know something about the environment...

# Background

1. Sutton & Barto, Chapter 3 + 4
2. RL for Finance Book, Chapter 4 + 5
3. Practical RL course by YSDA, week 2
4. Past iteration course, lecture 2
5. DeepMind course, lectures 3 + 4

# Markov Decision Process

MDP is a 4-tuple  $(\mathcal{S}, \mathcal{A}, p, r)$ :

1.  $\mathcal{A}$  is an action space
2.  $\mathcal{S}$  is a state space
3.  $p(s' | s, a) = \mathbb{P}(S_{t+1} = s' | S_t = s, A_t = a)$  is a state-transition function
4.  $r : \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$  is a reward function giving an expected reward:  
 $r(s, a) = \mathbb{E}[R | s, a]$

**Markov property:**

**(the future is independent of the past given the present)**

$$p(R_t, S_{t+1} | S_t, A_t, R_{t-1}, S_{t-1}, A_{t-1}, \dots) = p(R_t, S_{t+1} | S_t, A_t)$$

# Abuse of Notation

1.  $R_t, S_t, A_t$  are random variables;
2.  $r, s, a$  are realisation of random variables;
3.  $r(s, a)$  is a function which maps state and action realisations into conditional reward expectation.

# Equivalent Definition of MDP

MDP is a 3-tuple  $(\mathcal{S}, \mathcal{A}, p)$ :

1.  $\mathcal{A}$  is an action space
2.  $\mathcal{S}$  is a state space
3.  $p(r, s' | s, a) = \mathbb{P}(R_t = r, S_{t+1} = s' | S_t = s, A_t = a)$  is the joint probability of a reward  $r$  and next state  $s'$ , given a state  $s$  and action  $a$

**Markov property:**

**(the future is independent of the past given the present)**

$$p(R_t, S_{t+1} | S_t, A_t, R_{t-1}, S_{t-1}, A_{t-1}, \dots) = p(R_t, S_{t+1} | S_t, A_t)$$

# Equivalent Definition of MDP

MDP is a 3-tuple  $(\mathcal{S}, \mathcal{A}, p)$ :

1.  $\mathcal{A}$  is an action space
2.  $\mathcal{S}$  is a state space
3.  $p(r, s' | s, a) = \mathbb{P}(R_t = r, S_{t+1} = s' | S_t = s, A_t = a)$  is the joint probability of a reward  $r$  and next state  $s'$ , given a state  $s$  and action  $a$

$$p(s' | s, a) = \sum_r p(s', r | s, a)$$

$$r(s, a) = \sum_r r \sum_{s'} p(s', r | s, a)$$

# Relations

$$1. \quad p(s' | s, a) = \sum_r p(s', r | s, a)$$

$$2. \quad r(s, a) = \sum_r r \sum_{s'} p(s', r | s, a)$$

$$3. \quad r(s, a, s') = \mathbb{E}[R_t | S_t = s, A_t = a, S_{t+1} = s'] = \sum_r r p(r | s, a, s') =$$

$$= \sum_r r \frac{p(r, s', s, a)}{p(s', s, a)} = \sum_r r \frac{p(r, s' | s, a)}{p(s' | s, a)}$$



# Returns and Episodes

Let  $T$  is a final time step. If  $T < \infty$  then environment is called *episodic*.

**Cumulative reward** is called a **return or reward-to-go**. Note that in general it is a random variable.

The diagram shows the equation for the cumulative reward  $G_t$  at time step  $t$ . The equation is  $G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \dots = \sum_{k=t}^T \gamma^{k-t} R_k$ . Annotations include: a blue arrow pointing to  $G_t$  labeled 'Cumulative reward'; a green arrow pointing to  $R_t$  labeled 'Immediate reward'; an orange arrow pointing to  $\gamma$  labeled 'Discount factor'; and a red arrow pointing to the upper limit  $T$  of the summation labeled 'Episode length'.

$$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \dots = \sum_{k=t}^T \gamma^{k-t} R_k$$

Immediate reward

Discount factor

Episode length

Reminder: either  $T = \infty$  or  $\gamma = 1$

# Returns and Episodes

Let  $T$  is a final time step. If  $T < \infty$  then environment is called *episodic*.

**Cumulative reward** is called a **return or reward-to-go**. Note that in general it is a random variable.

The diagram shows the equation for the cumulative reward  $G_t$  with several annotations. A blue arrow points from the text 'Cumulative reward' to the  $G_t$  term, which is enclosed in a blue box. A green arrow points from the text 'Immediate reward' to the  $R_t$  term, which is enclosed in a green box. An orange arrow points from the text 'Discount factor' to the  $\gamma$  term, which is enclosed in an orange box. A red arrow points from the text 'Episode length' to the  $T$  term in the summation, which is enclosed in a red box. The equation is: 
$$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \dots = \sum_{k=t}^T \gamma^{k-t} R_k$$

Important note:  $G_t = R_t + \gamma(R_{t+1} + \gamma R_{t+2} + \dots) = R_t + \gamma G_{t+1}$

# Objective Function

Agent interacts with the environment following some policy  $\pi(a | s)$ .

As the result a trajectory  $\tau$  is generated:

$$\tau = (s_0, a_0, s_1, \dots, a_{T-1}, s_T), a_t \sim \pi(\cdot | s_t)$$

$$J(\pi) = \mathbb{E}_{\pi}[G_0] = \mathbb{E}_{p(\tau|\pi)}[G_0] =$$

$$\mathbb{E}_{s_0 \sim p(s_0)}[\mathbb{E}_{a_0 \sim \pi(\cdot|s_0)}[R_0 + \mathbb{E}_{s_1 \sim p(\cdot|s_0, a_0)}[\mathbb{E}_{a_1 \sim \pi(\cdot|s_1)}[\gamma R_1 + \dots]]]] \rightarrow \max_{\pi}$$

# State-value Function

Estimate how good it is for an agent to be in a given state  $s$  and follow a policy  $\pi$ :

$$V_{\pi}(s) = \mathbb{E}_{\pi}[G_t | S_t = s] = \mathbb{E}_{\pi}\left[\sum_{k=0}^T \gamma^k R_{t+k} | S_t = s\right]$$

If  $s$  is a terminal state then  $V_{\pi}(s) = 0$ .

# State-value Function

Law of iterated expectation

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

$$= \sum_a \pi(a | s) \sum_{r, s'} \mathbb{P}(R_t = r, S_{t+1} = s' | S_t = s, A_t = a, S_{t-1} = s'', A_{t-1} = a'', R_{t-1} = r'', \dots) [r + \gamma \mathbb{E}_{\pi}[G_{t+1} | S_{t+1} = s']] =$$

Definition of reward-state transition function

$$= \sum_a \pi(a | s) \sum_{r, s'} \mathbb{P}(R_t = r, S_{t+1} = s' | S_t = s, A_t = a) [\dots] =$$

Definition of  $V^{\pi}(s')$

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

Markov property

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

Policy stochasticity

Environment stochasticity

# Action-value Function

Estimate how good it is for the agent to take an action  $a$  being in a given state  $s$  and follow a policy  $\pi$ :

$$Q_{\pi}(s, a) = \mathbb{E}_{\pi}[G_t | S_t = s, A_t = a] = \mathbb{E}_{\pi}\left[\sum_{k=0}^T \gamma^k R_{t+k} | S_t = s, A_t = a\right]$$

If  $s$  is a terminal state then  $\forall a \in \mathcal{A} \ Q_{\pi}(s, a) = 0$ .

# Action-value Function

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

# Relationship between $V_\pi$ and $Q_\pi$

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

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



# Relationship between $V_\pi$ and $Q_\pi$

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

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

$$V_\pi(s) = \sum_a \pi(a | s) Q_\pi(s, a) = \mathbb{E}_{a \sim \pi(\cdot | s)} Q_\pi(s, a)$$

# Relationship between $V_\pi$ and $Q_\pi$

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

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

$$V_\pi(s) = \sum_a \pi(a | s) Q_\pi(s, a) = \mathbb{E}_{a \sim \pi(\cdot | s)} Q_\pi(s, a)$$

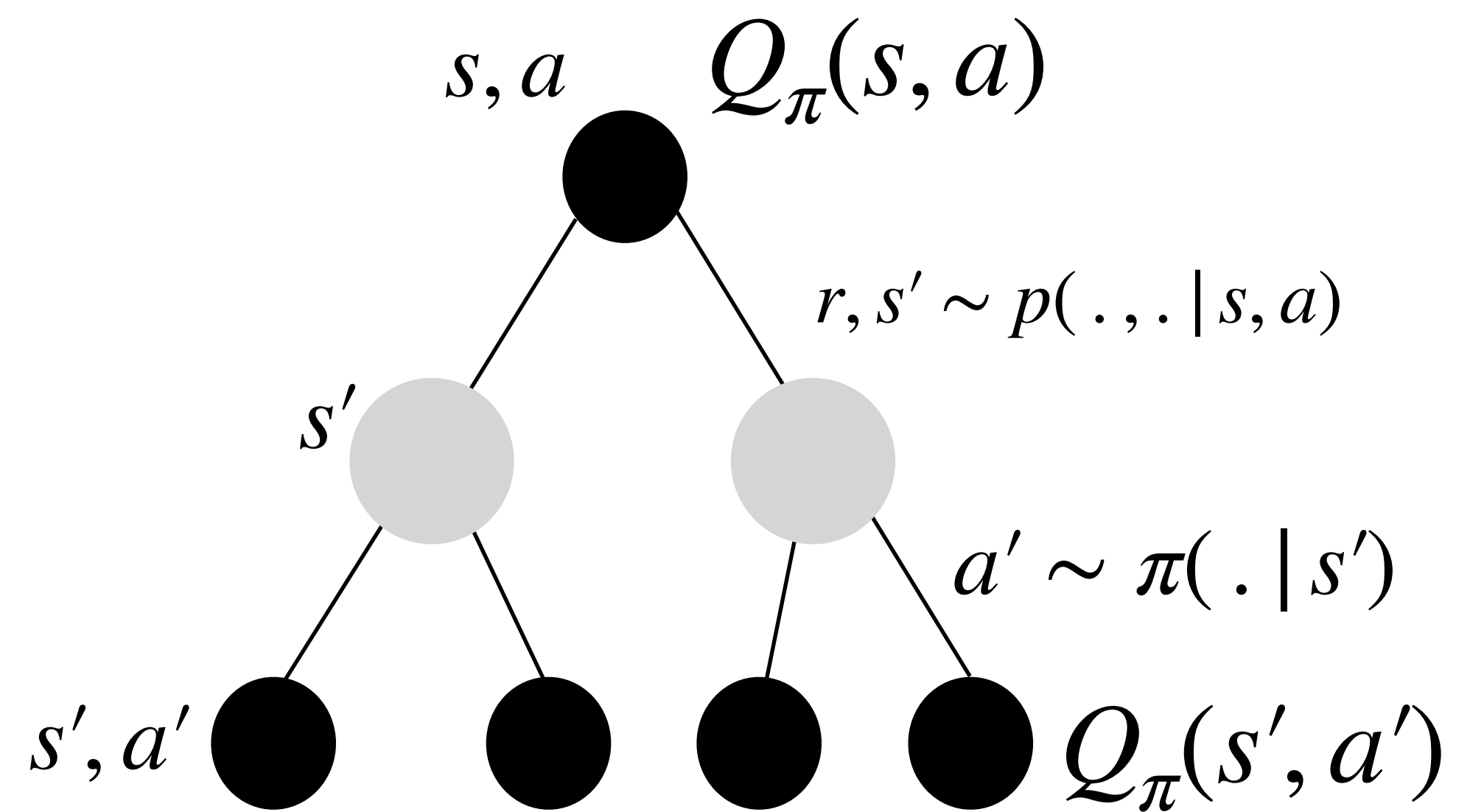
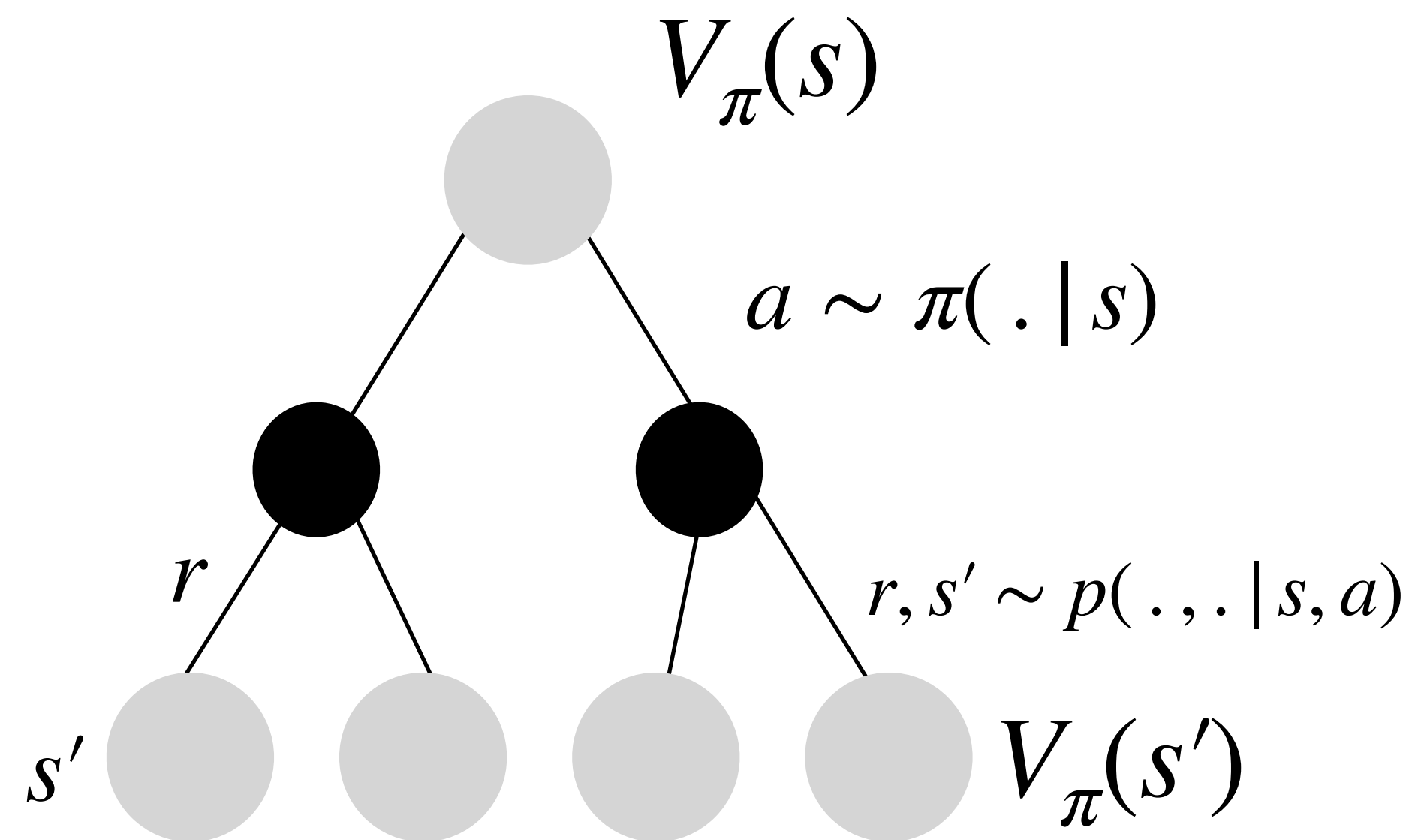
$$Q_\pi(s, a) = \sum_{r,s'} p(r, s' | s, a) [r + \gamma \sum_{a'} \pi(a' | s') Q_\pi(s', a')]$$

# Bellman **Expectations** Equations

- $V_{\pi}(s) = \sum_a \pi(a | s) \sum_{r, s'} p(r, s' | s, a) [r + \gamma V_{\pi}(s')] = \mathbb{E}_{\pi}[R_t + \gamma V_{\pi}(S_{t+1}) | S_t = s]$
- $Q_{\pi}(s, a) = \sum_{r, s'} p(r, s' | s, a) [r + \gamma \sum_{a'} \pi(a' | s') Q_{\pi}(s', a')] = r(s, a) + \gamma \mathbb{E}_{\pi} Q(s', a')$

# Bellman **Expectations** Equations

- $V_{\pi}(s) = \sum_a \pi(a | s) \sum_{r, s'} p(r, s' | s, a) [r + \gamma V_{\pi}(s')] = \mathbb{E}_{\pi}[R_t + \gamma V_{\pi}(S_{t+1}) | S_t = s]$
- $Q_{\pi}(s, a) = \sum_{r, s'} p(r, s' | s, a) [r + \gamma \sum_{a'} \pi(a' | s') Q_{\pi}(s', a')] = r(s, a) + \gamma \mathbb{E}_{\pi} Q(s', a')$



# Bellman **Optimality** Equations

Recall that our goal is to find a policy that achieves maximum expected cumulative reward.

Define a partial order on a set of possible policies:

$$\pi \geq \pi' \iff V_{\pi}(s) \geq V_{\pi'}(s) \forall s \in \mathcal{S}$$

Let  $\pi^*$  be all optimal policies and define optimal value functions:

$$V^*(s) = \max_{\pi} V_{\pi}(s)$$

$$\pi^*(a | s) = \mathbb{I}[a = \operatorname{argmax}_{a'} Q^*(s, a')]$$

$$Q^*(s, a) = \max_{\pi} Q_{\pi}(s, a)$$

# Bellman **Optimality** Equations

$$V^*(s) = \max_{\pi} V_{\pi}(s)$$

$$Q^*(s, a) = \max_{\pi} Q_{\pi}(s, a)$$



$$V^*(s) = \max_a Q^*(s, a)$$

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

# Bellman **Optimality** Equations

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

$$Q^*(s, a) = \mathbb{E}[R_t + \gamma V^*(S_{t+1}) | S_t = s, A_t = a] = \sum_{s', r} p(s', r | s, a) [r + \gamma \max_{a'} Q^*(s', a')]$$

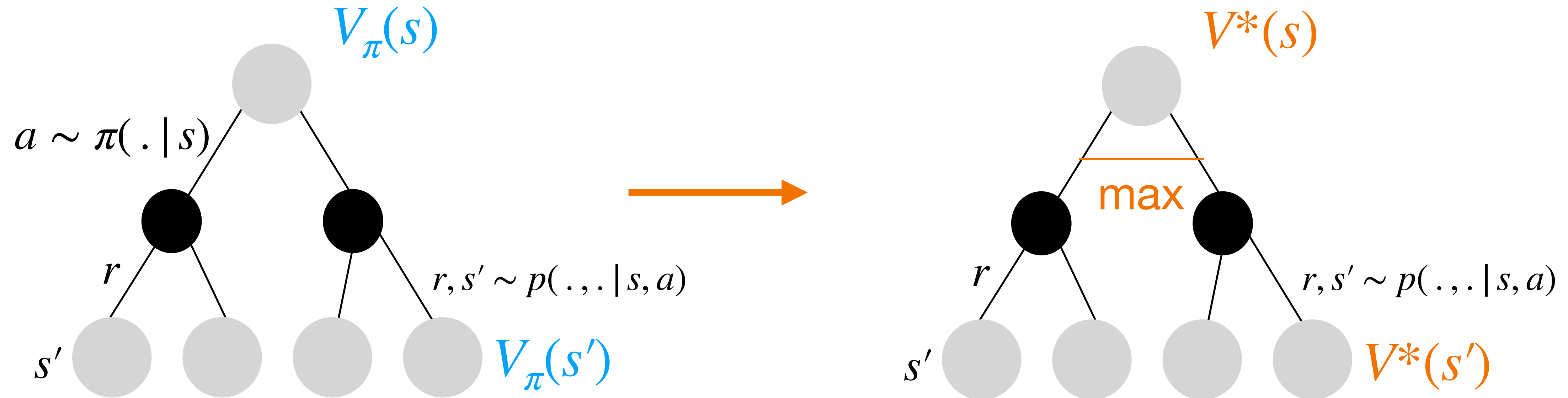
# Bellman **Optimality** Equations

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

$$Q^*(s, a) = \mathbb{E}[R_t + \gamma V^*(S_{t+1}) | S_t = s, A_t = a] = \sum_{s', r} p(s', r | s, a) [r + \gamma \max_{a'} Q^*(s', a')]$$

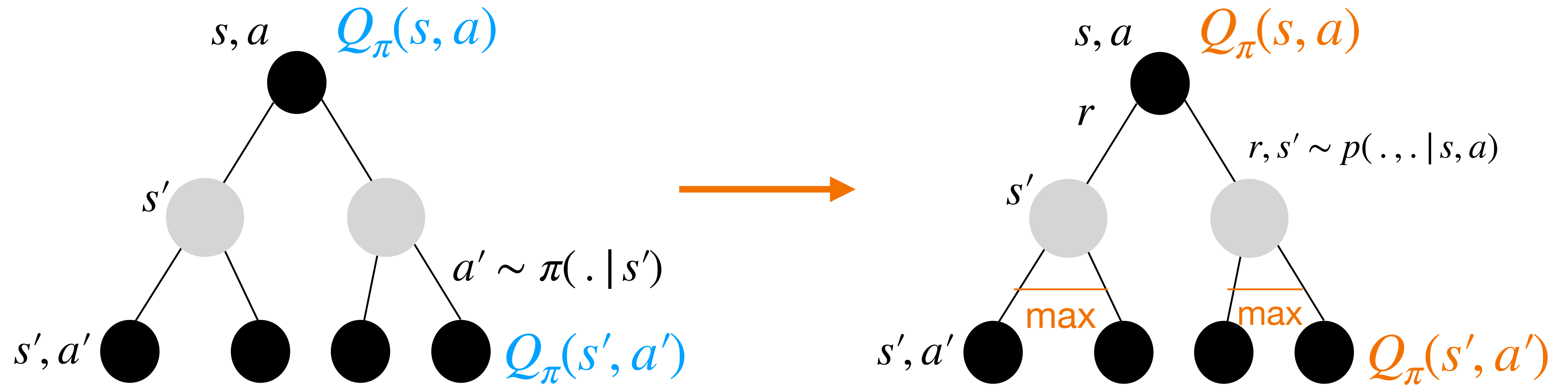


# Intuition



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

# Intuition

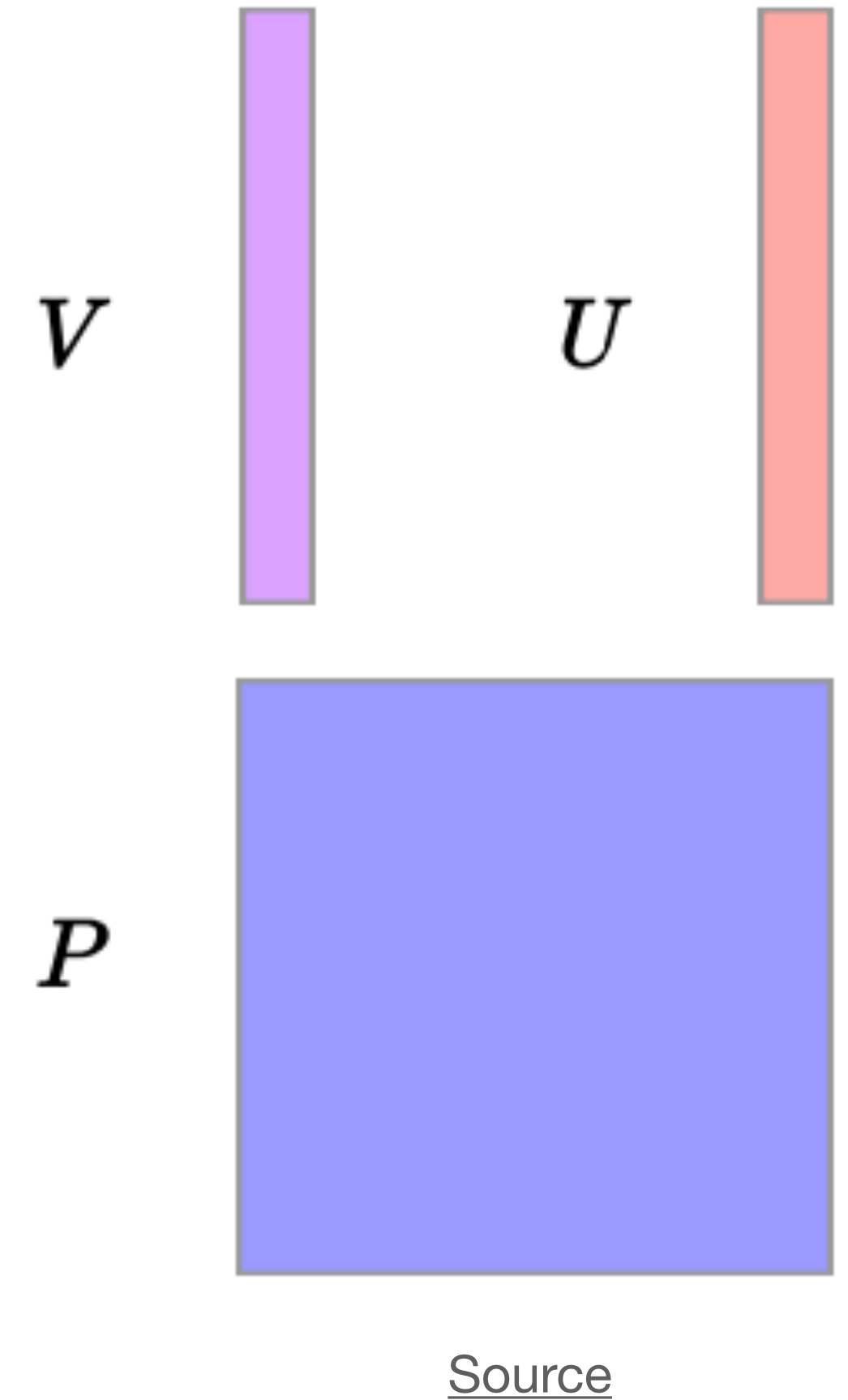


$$Q^*(s, a) = \mathbb{E}[R_t + \gamma V^*(S_{t+1}) | S_t = s, A_t = a] = \sum_{s', r} p(s', r | s, a) \left[ r + \gamma \max_{a'} Q^*(s', a') \right]$$

# Linear Equations

$V_{\pi}(s) = \sum_a \pi(a | s) \sum_{r,s'} p(r, s' | s, a) [r + \gamma V_{\pi}(s')]$  is a system of  $|S|$  linear equations:

$$V = U + \gamma P V \rightarrow (I - \gamma P) V = U \rightarrow V = (I - \gamma P)^{-1} U$$



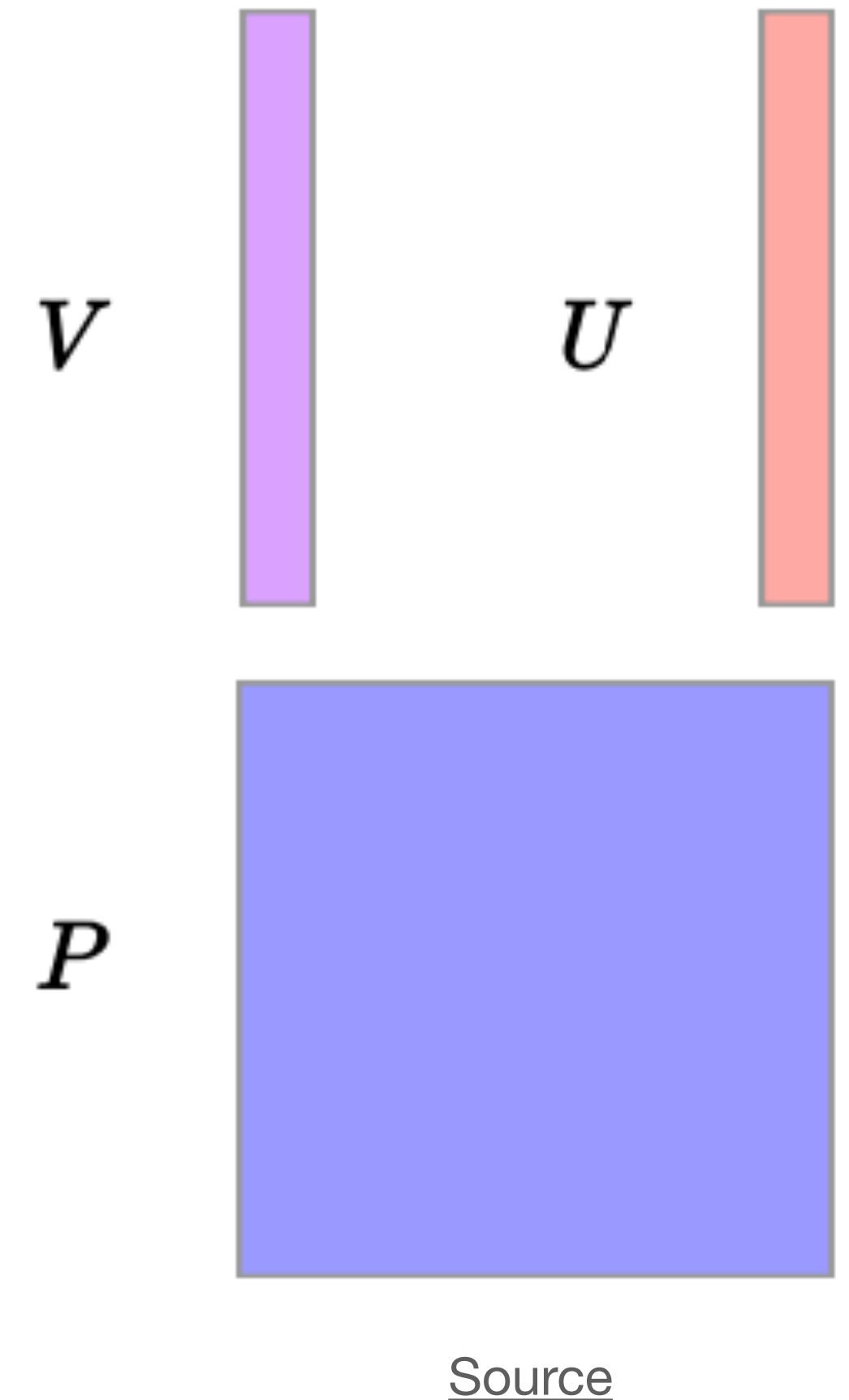
# Linear Equations

$V_{\pi}(s) = \sum_a \pi(a | s) \sum_{r,s'} p(r, s' | s, a) [r + \gamma V_{\pi}(s')]$  is a system of  $|S|$  linear equations:

$$V = U + \gamma P V \rightarrow (I - \gamma P) V = U \rightarrow V = (I - \gamma P)^{-1} U$$

General case:  $\mathcal{O}(|S|^3)$

Maybe we can try iterative solution methods?



# Dynamic Programming

# Problems in RL

1. Estimating  $V^\pi$  or  $Q^\pi$  is called **policy evaluation** or **prediction**
2. Estimating  $V^*$  or  $Q^*$  is called **control** because they can be used for **policy optimisation**

# Bellman Operators

Bellman **expectation** operator for  $V(s)$ :

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

Bellman **expectation** operator for  $Q(s, a)$ :

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

Bellman **optimality** operator for  $V(s)$ :

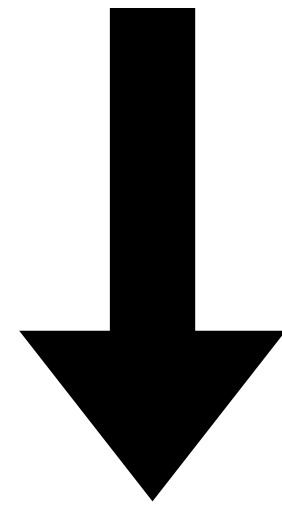
$$[\mathcal{T} V](s) = \max_a \mathbb{E}_{r,s'|s,a} [r + \gamma V(s')]$$

Bellman **optimality** operator for  $Q(s, a)$ :

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

# Policy Evaluation

$$V_{\pi}(s) = \sum_a \pi(a | s) \sum_{r, s'} p(r, s' | s, a) [r + \gamma V_{\pi}(s')] \quad \text{Linear equations}$$



$$V_{k+1}(s) = [\mathcal{T} V_k](s) = \sum_a \pi(a | s) \sum_{r, s'} p(r, s' | s, a) [r + \gamma V_k(s')] \quad \text{Simple iteration method}$$

for fixed policy  $\pi$



# Policy Iteration 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  $s \in \mathcal{S}$ :

$v \leftarrow V(s)$

Bellman expectation operator for  $V^\pi(s)$

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

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

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

Output  $V \approx v_\pi$

# Policy Improvement

Let us act greedily w.r.t.  $Q^\pi(s, a)$ :

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

# Policy Improvement

Let us act greedily w.r.t.  $Q^\pi(s, a)$ :

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

$\pi^*$  is not worse than  $\pi$ :

If  $\forall s \in \mathcal{S} \ Q^\pi(s, \pi'(s)) \geq V^\pi(s)$

then  $\forall s \in \mathcal{S} \ V^{\pi'}(s) \geq V^\pi(s) \iff \pi' \geq \pi$

# Policy Improvement

Let us act greedily w.r.t.  $Q^\pi(s, a)$ :

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

If  $\pi' = \pi \rightarrow V^{\pi'} = V^\pi$  and  $V^\pi$  satisfies Bellman **optimality** equation:

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

# Policy Improvement

Let us act greedily w.r.t.  $Q^\pi(s, a)$ :

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

# Recover Optimal Policy

1. If  $Q^*$  is known:  $\pi(s) = \operatorname{argmax}_a Q^*(s, a)$

2. If  $V^*$  is known:  $\pi(s) = \operatorname{argmax}_a \sum_{r,s'} p(r, s' | s, a) [r + \gamma V^*(s')]$

Unknown in a model-free setting

# Policy Iteration

## 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  $s \in \mathcal{S}$ :

Bellman expectation operator for  $V^\pi(s)$

$v \leftarrow V(s)$

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

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

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

## 3. Policy Improvement

*policy-stable*  $\leftarrow$  true

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

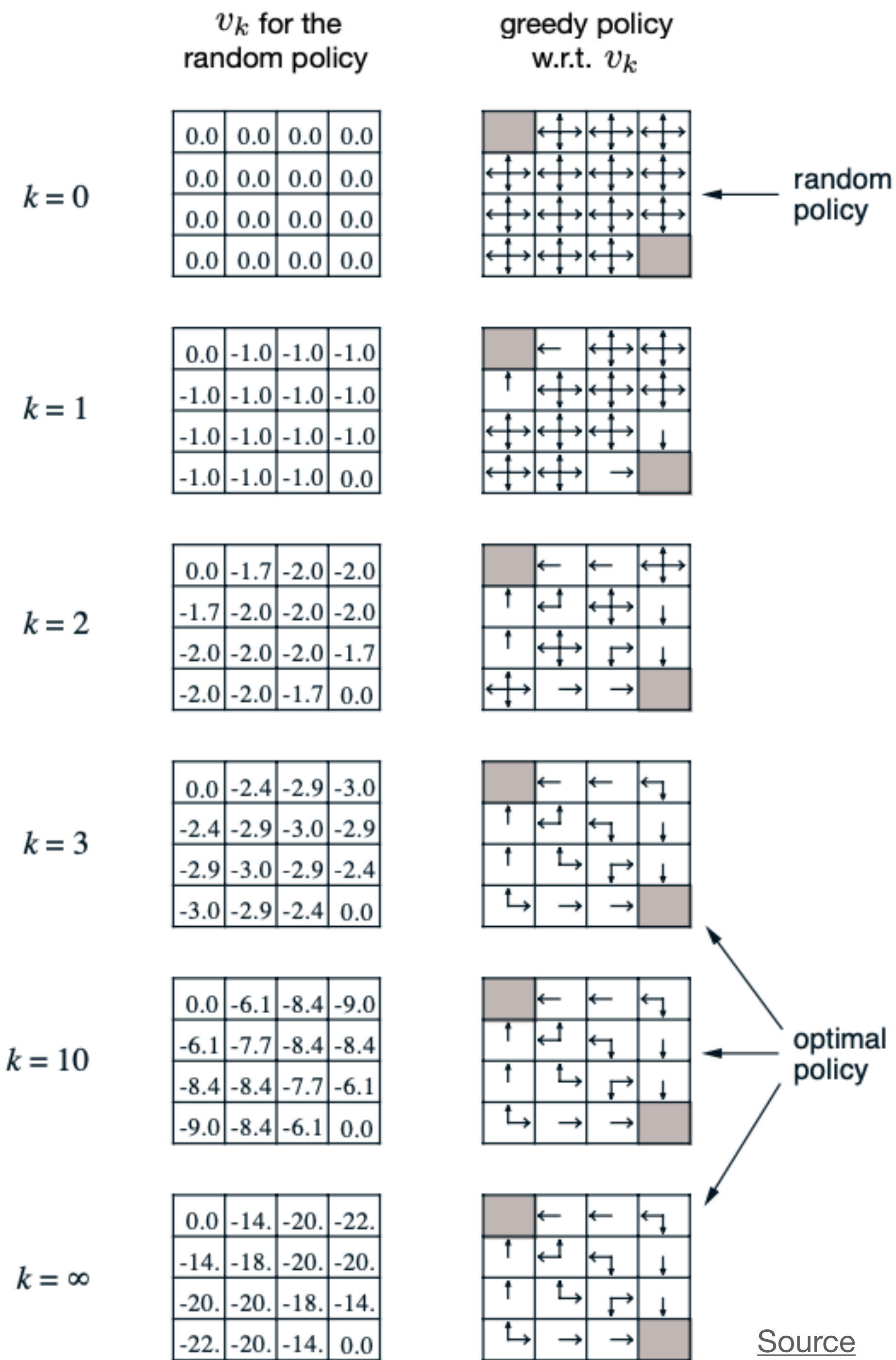
Almost Bellman optimality operator for  $V^\pi(s)$

*old-action*  $\leftarrow \pi(s)$

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

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

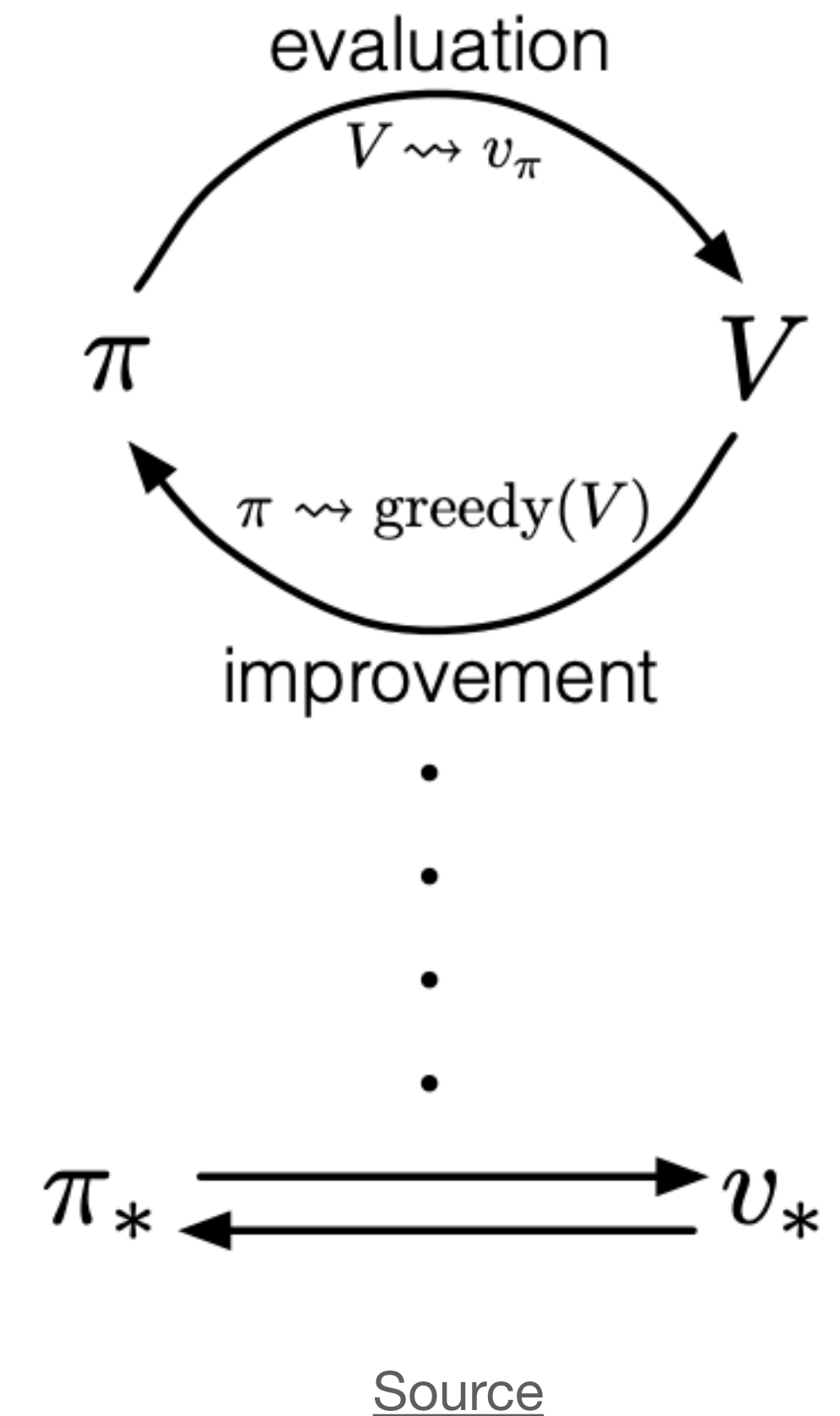




# Policy Iteration

Policy iteration consists of two processes, each completing before the other begins: one making the value function consistent with the current policy (**policy evaluation**), and the other making the policy greedy with respect to the current value function (**policy improvement**)

$$\pi_0 \xrightarrow{\text{E}} v_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} v_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \dots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} v_*$$



# Value Iteration

One drawback to policy iteration is that each of its iterations involves policy evaluation, which may itself be a protracted iterative computation requiring multiple sweeps through the state set. If policy evaluation is done iteratively, then convergence exactly to  $V^\pi$  occurs only in the limit.

# Value Iteration

In fact, the policy evaluation step of policy iteration can be truncated in several ways without losing the convergence guarantees of policy iteration. One important special case is when policy evaluation is stopped after just one sweep (one update of each state). This algorithm is called value iteration.

$$V_{k+1}(s) = \max_a \sum_{r,s'} p(r, s' | s, a) [r + \gamma V_k(s')]$$

# 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)

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

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

# Synchronous Dynamic Programming Algorithms

Problem	Bellman Equation	Algorithm
Prediction	Bellman Expectation Equation	Iterative Policy Evaluation
Control	Bellman Expectation Equation + (Greedy) Policy Improvement	Policy Iteration
Control	Bellman Optimality Equation	Value Iteration

Complexity per iteration:

$$O(|A||S|^2 + |S|^3)$$

$$O(|A||S|^2)$$

Source

Drawback: involve operations over the entire state set of the MDP, that is, they require sweeps of the state set.

# Asynchronous Dynamic Programming

DP methods described so far used synchronous updates (all states in parallel)

Asynchronous DP:

- backs up states individually, in any order
- can significantly reduce computation
- guaranteed to converge if all states continue to be selected

# Asynchronous Dynamic Programming

Three simple ideas for asynchronous dynamic programming:

1. In-place dynamic programming: do not copy  $V^\pi$  but make in-place update during iteration method;
2. Prioritised sweeping: select state based on magnitude of Bellman error
3. Real-time dynamic programming: consider only relevant states using samples  $\langle s, a, r, s' \rangle$

These ideas will be extremely useful in a model-free setting!

**Thank you for your attention!**