RL@Coursera
week 2

Global optimality:
discounted return

Yandex
Data Factory

LAMBDA

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 return:

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

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 return:

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

end of episode

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 behavior!

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

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?

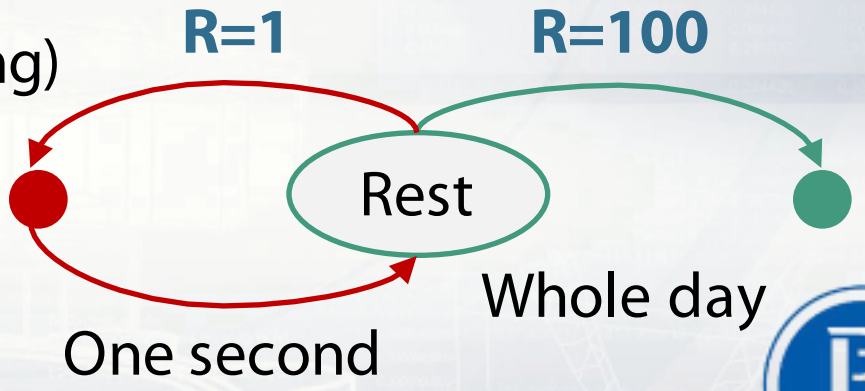
E.g.: cleaning robot

- **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!



Reward discounting

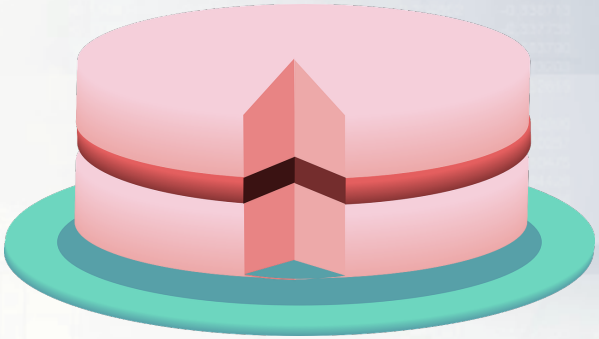
Get rid of infinite sum by discounting $0 \leq \gamma \leq 1$

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

discount factor

The same cake compared to today's one worth

- γ times less tomorrow
- γ^2 times less the day after tomorrow



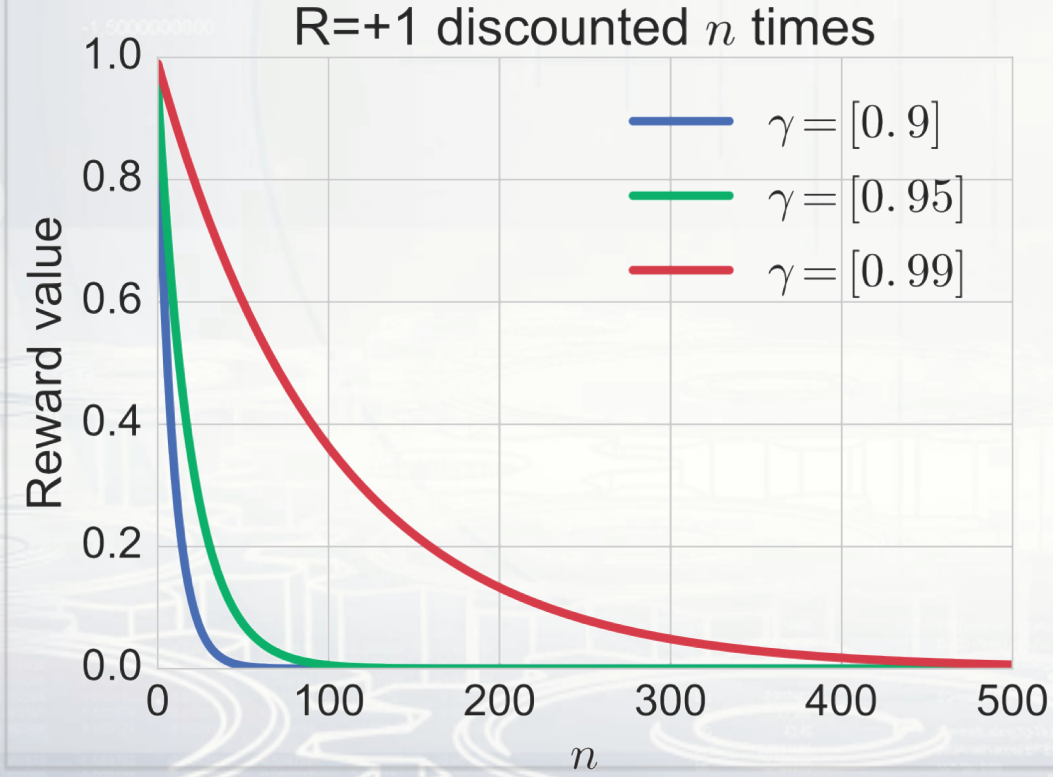
γ will eat it day by day

Discounting makes sums finite

Maximal return for **R = +1**

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

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



Any discounting changes optimization task and its solution!

Discounting is inherent to humans

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

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

Mathematical convenience

$$G_t \triangleq R_t + \gamma(R_{t+1} + \gamma R_{t+2} + \dots)$$

$$= R_t + \gamma G_{t+1}$$

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: another view

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

Discounting: another view

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

Action indirectly affects future rewards

But how long does this effect lasts?

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

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

Discounting: another view

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

Action indirectly affects future rewards

But how long does this effect last?

$$G_0 = R_0 + \gamma R_1 + \gamma^2 R_2 + \cdots + \gamma^T R_T$$

$$= (1-\gamma) R_0$$

"End of effect" probability

"Effect continuation" probability

$$+(1-\gamma)\gamma(R_0 + R_1)$$

$$+(1-\gamma)\gamma^2(R_0 + R_1 + R_2)$$

$$\cdots$$

$$+\gamma^T \cdot \sum_{t=0}^{T} R_t$$

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 think about cleaning the floor!

Reward design – don't shift, reward for WHAT

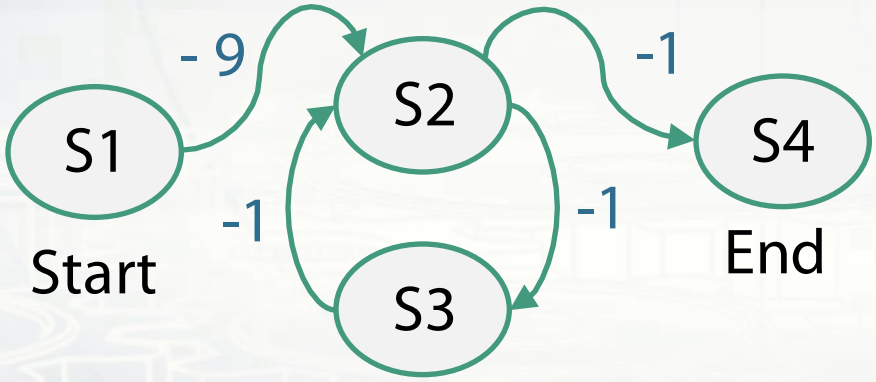
E.g.: chess – value of taken opponent's piece

Problem: agent will not have a desire to win!

E.g.: cleaning robot, **+100** (cleaning), **+0.1** (on-off)

Problem: agent will not think about cleaning the floor!

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



Reward design – don't shift, reward for WHAT

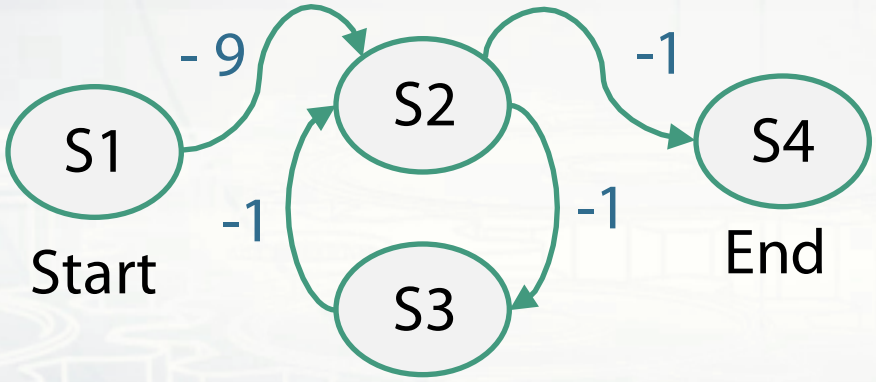
E.g.: chess – value of taken opponent's piece

Problem: agent will not have a desire to win!

E.g.: cleaning robot, **+100** (cleaning), **+0.1** (on-off)

Problem: agent will not think about 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 an optimal policy?

- Reward **scaling** – division by nonzero constant
  - May be useful in practice 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 an optimal policy?

- Reward **scaling** – division by nonzero constant
  - May be useful in practice for approximate methods
- Reward **shaping** – we could add to all rewards $R(s, a, s')$ in MDP values of potential-based shaping function $F(s, a, s')$ without changing an optimal policy:

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

definition

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).

Global optimality: state and action value functions

How to find an optimal policy?

Dynamic programming!

Method to solve a complex problem by

- breaking it into small pieces
- until no more unsolved pieces
  - solve a single piece
  - reuse the solution to solve another piece

DP equations lies at the heart of RL

It is essential to deeply understand them.

How to find an optimal policy?

We know! Maximize cumulative discounted return!

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

How to find an optimal policy?

We know! Maximize cumulative discounted return!

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

But policy and / or environment could be random!

Let get rid of randomness by taking expectation!

State-value function $v(s)$

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

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

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

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

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

Intuition: value of following policy π from state s

State-value function $v(s)$

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

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

stochasticity in policy & environment

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

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

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

Intuition: value of following policy π from state s

State-value function $v(s)$

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

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

stochasticity in policy & environment

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

Environment stochasticity

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

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

Policy stochasticity

Intuition: value of following policy π from state s

State-value function $v(s)$

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

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

stochasticity in policy & environment

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

Environment stochasticity

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

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

Policy stochasticity

By definition

Intuition: value of following policy π from state s

Bellman expectation equation for $v(s)$

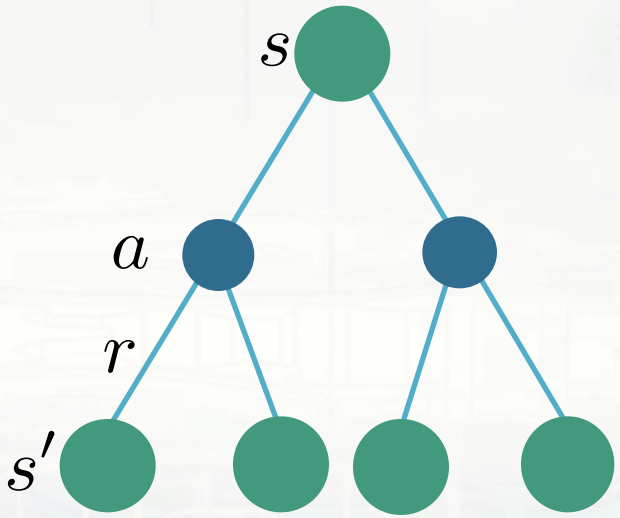
Recursive definition of $v(s)$ is an important concept in RL

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

Bellman expectation equation for $v(s)$

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

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

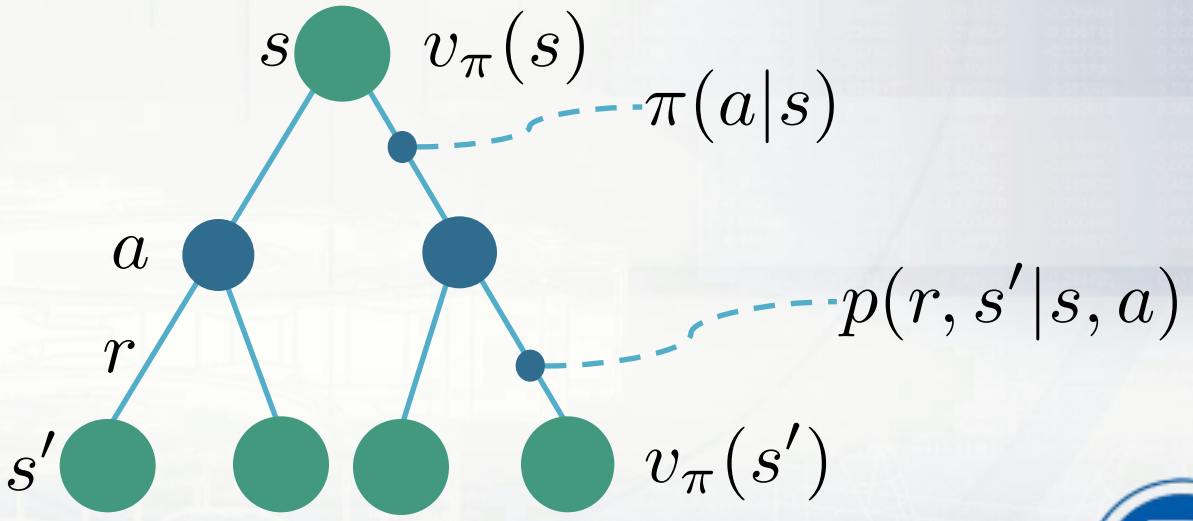


Backup diagram

Bellman expectation equation for $v(s)$

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

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



Backup diagram

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

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

Intuition: value of following policy π **after** committing action **a** in state **s**

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

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

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

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

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

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

Intuition: value of following policy π **after** committing action **a** in state **s**

No policy stochasticity at first step

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

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

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

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

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

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

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

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

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

$$= \sum_a \pi(a|s) q_\pi(s,a)$$

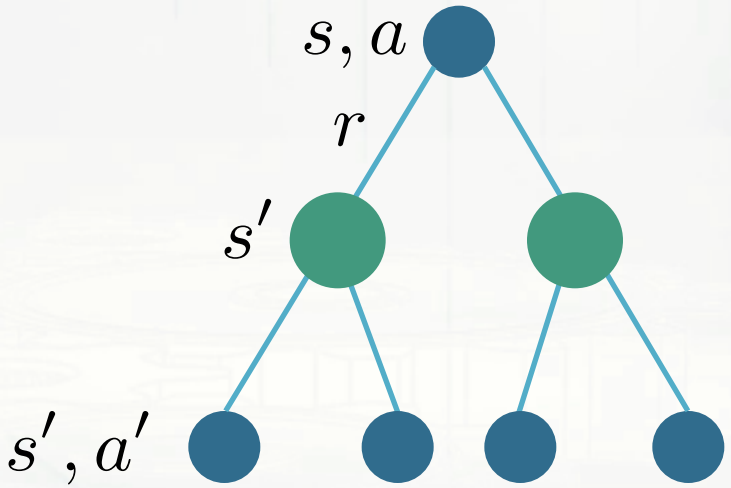
So, we could now write $q(s,a)$ in terms of $q(s,a)$!

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

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

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

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

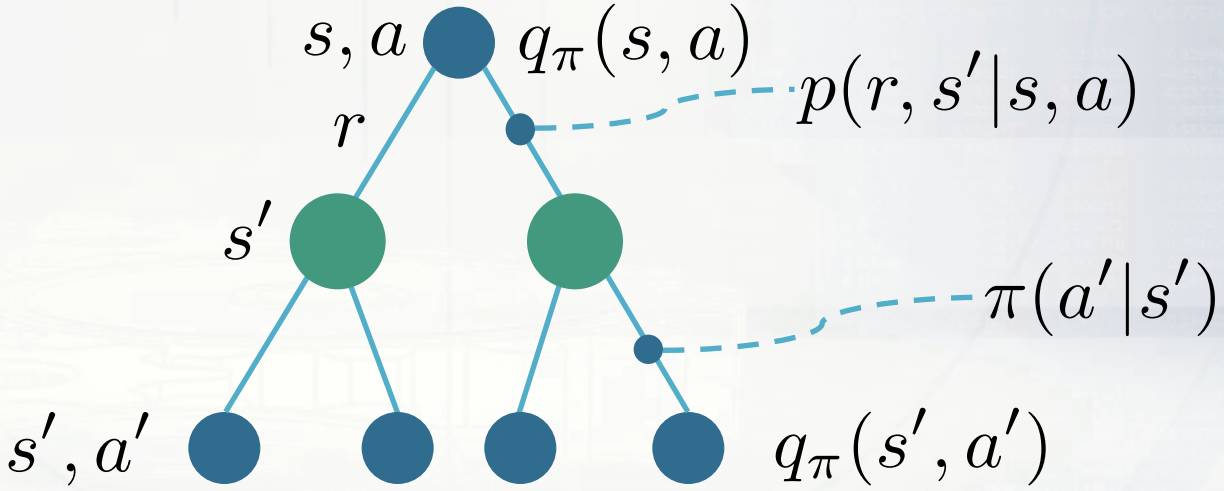


Backup diagram for $q(s,a)$

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

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

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



Backup diagram for $q(s,a)$

What are we going to do with value function?

Bellman expectation equations → assess policy performance.

We know

- what is return,
- what is value- and action-value functions.

But we want to find optimal policy!

That is – to know optimal actions in each possible state.

But how to know which policy is better?

How to compare them?

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

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

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

Best policy π_* is better or equal to any other policy

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

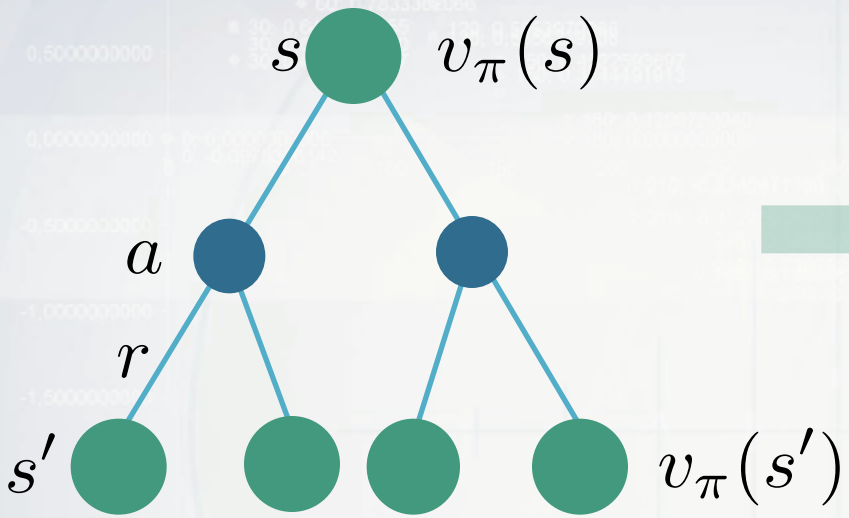
$$v_*(s) = \max_\pi v_\pi(s)$$

Use optimal policy from s

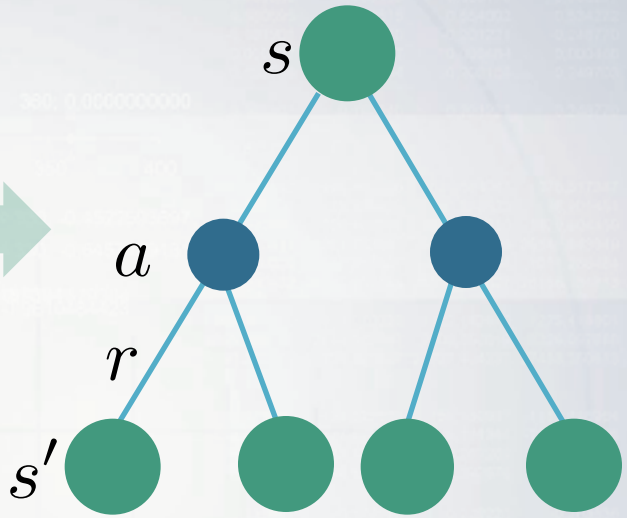
$$q_*(s,a) = \max_\pi q_\pi(s,a)$$

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

Bellman optimality equation for $v(s)$

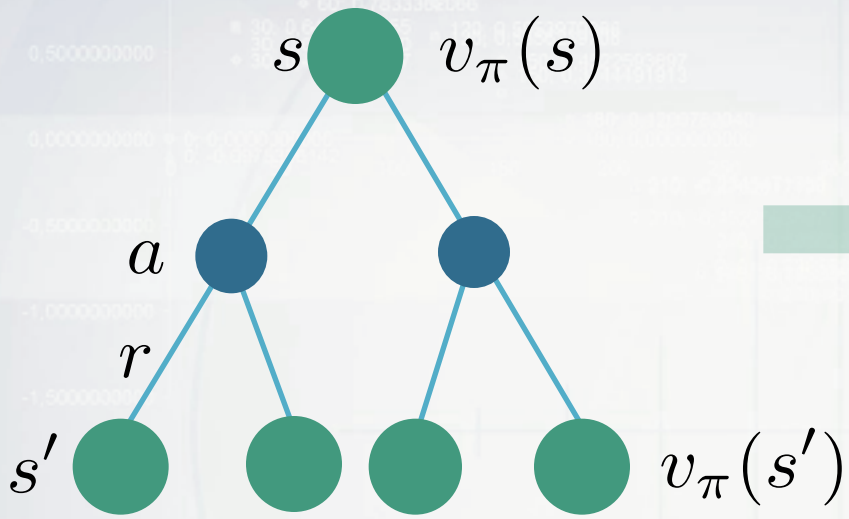


Bellman **expectation** equation for $v(s)$

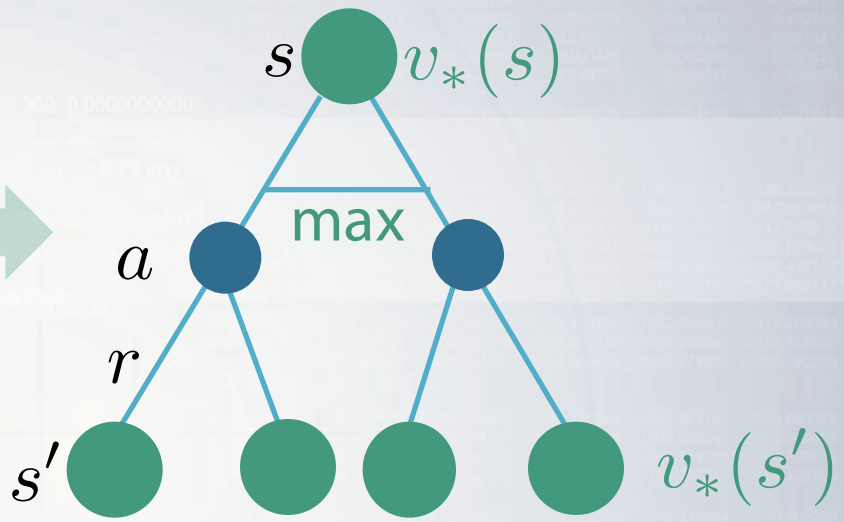


Bellman **optimality** equation for $v_*(s)$

Bellman optimality equation for $v(s)$

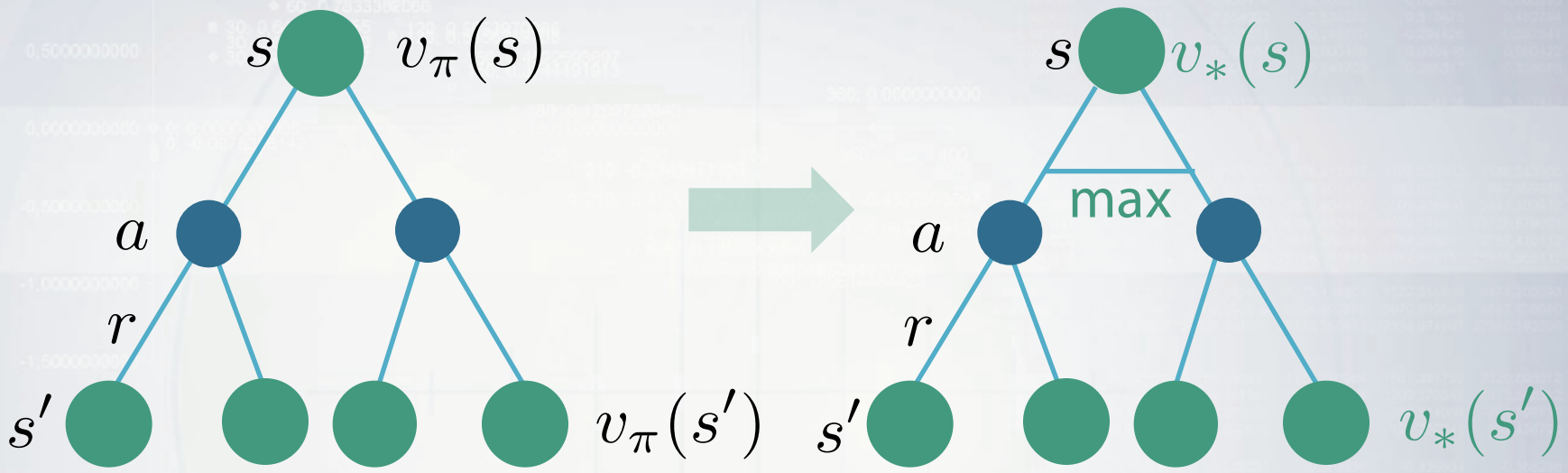


Bellman **expectation** equation for $v(s)$



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

Bellman optimality equation for $v(s)$



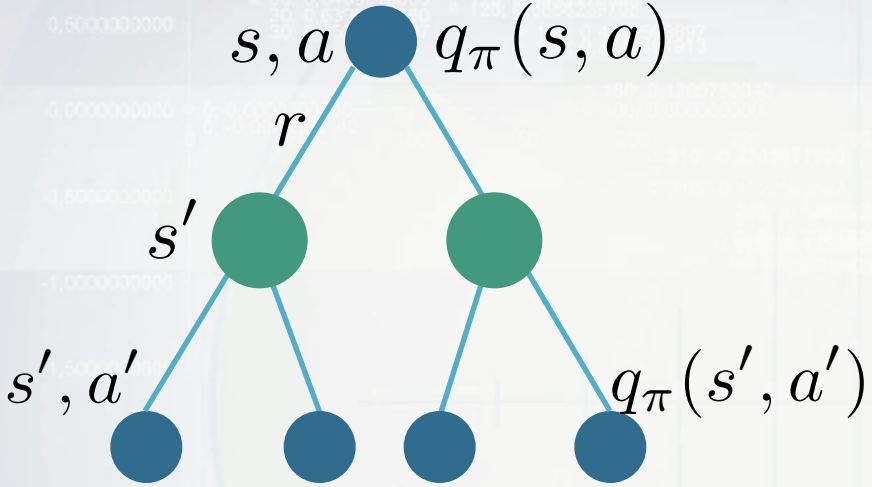
Bellman **expectation** equation for $v(s)$

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

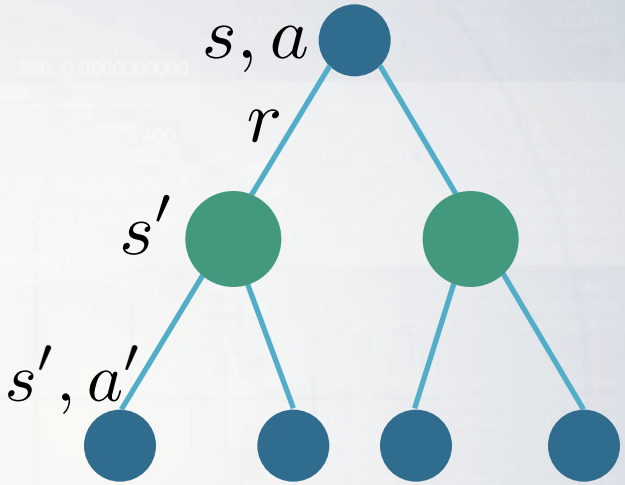
$$v_*(s) = \max_a \sum_{r,s'} p(r, s'|s, a)[r + \gamma v_*(s')]$$

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

Bellman optimality equation for $v(s)$

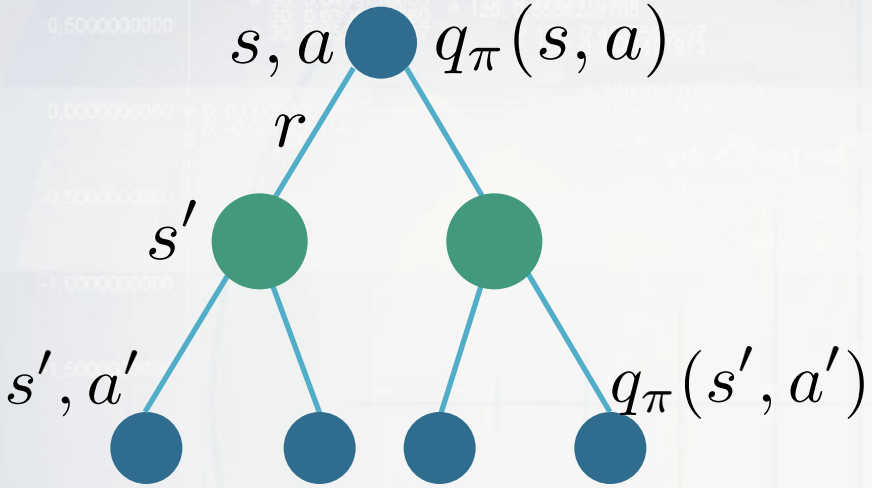


Bellman **expectation** equation for $q(s, a)$

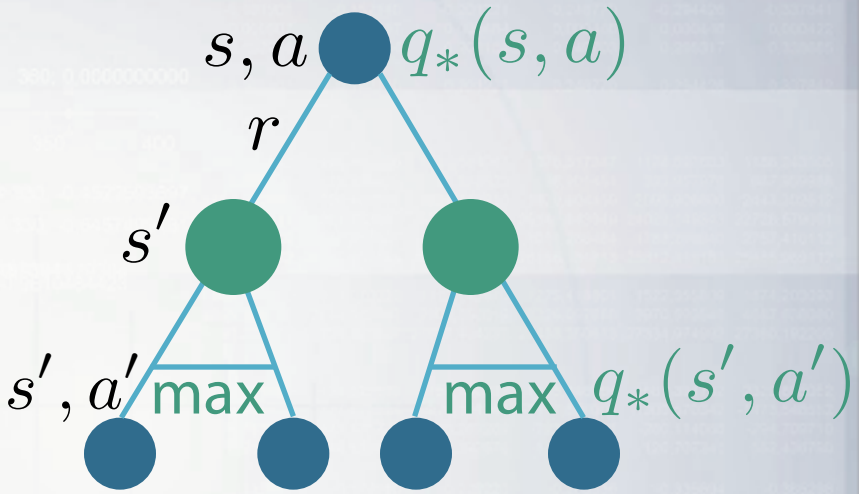


Bellman **optimality** equation for $q_*(s, a)$

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

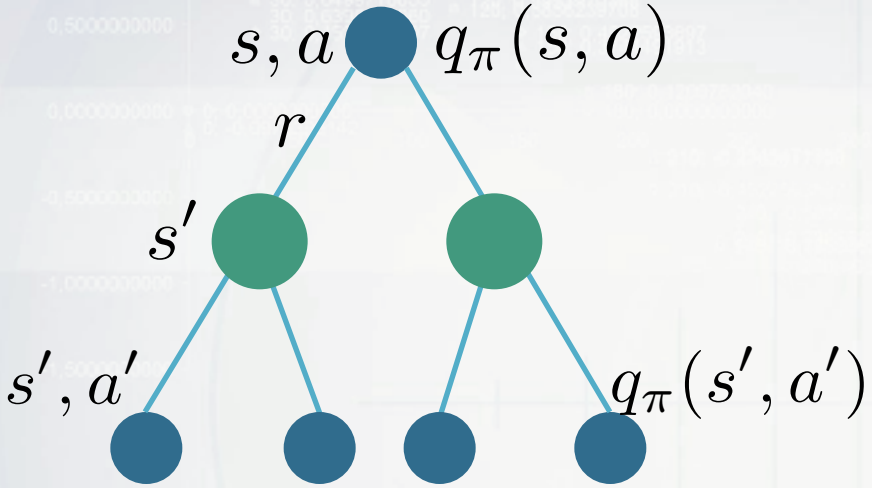


Bellman **expectation** equation for $q(s,a)$

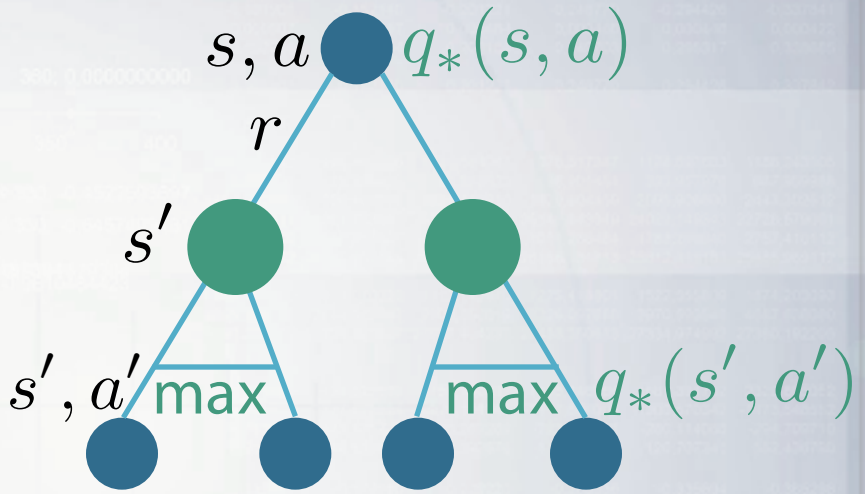


Bellman **optimality** equation for $q_*(s,a)$

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



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



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

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

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

Well, that was tough. What's next?

Now we are equipped with heavy artillery of

- Bellman **expectation** equation for $v(s)$ and $q(s, a)$
- Bellman **optimality** equation for $v_*(s)$ and $q_*(s, a)$

That will be our toolkit for finding optimal policy

using dynamic programming!

Generalized policy iteration: Policy: evaluation & improvement

Model-based setup with value-based approach

How to find optimal policy?

model & value based

Model-based setup – model of the world is known, i.e

- $p(r, s'|s, a)$ for all r, s', s, a is known

Value based approach

1. Build or estimate a value
2. Extract a policy from the value

Policy evaluation: motivation

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

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

- predict value function for a particular policy.

Bellman expectation equation

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

is basically a system of linear equations where

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

Policy evaluation: algorithm

Input π, the policy to be evaluated

Initialize an array $v(s) = 0$, for all $s \in S$

Repeat

$\Delta \leftarrow 0$

For each $s \in S$:

Bellman expectation equation for $v(s)$

$v_{old}(s) \leftarrow v(s)$

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

$\Delta \leftarrow \max(\Delta, |v_{old}(s) - v(s)|)$

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

Output $v \approx v_\pi$

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. $v(s)$!

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

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. $v(s)$!

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

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 a new policy after improvement

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

is the same as the old one

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

then it is optimal!

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

Policy improvement: convergence

If a new policy after improvement

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

is the same as the old one

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

then it is optimal!

Bellman optimality equation

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

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

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

$$\pi_*(s) \leftarrow \underset{a}{\operatorname{argmax}}\ q_*(s,a)$$

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

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

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

$$\pi_*(s) \leftarrow \underset{a}{\operatorname{argmax}}\ q_*(s,a)$$

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

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

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

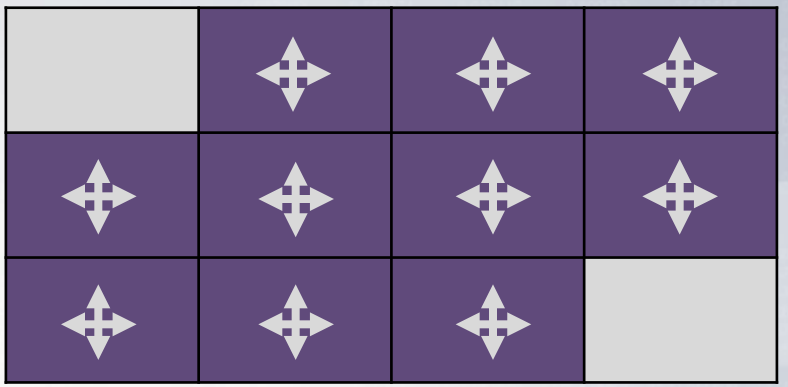
Precise evaluation is not needed

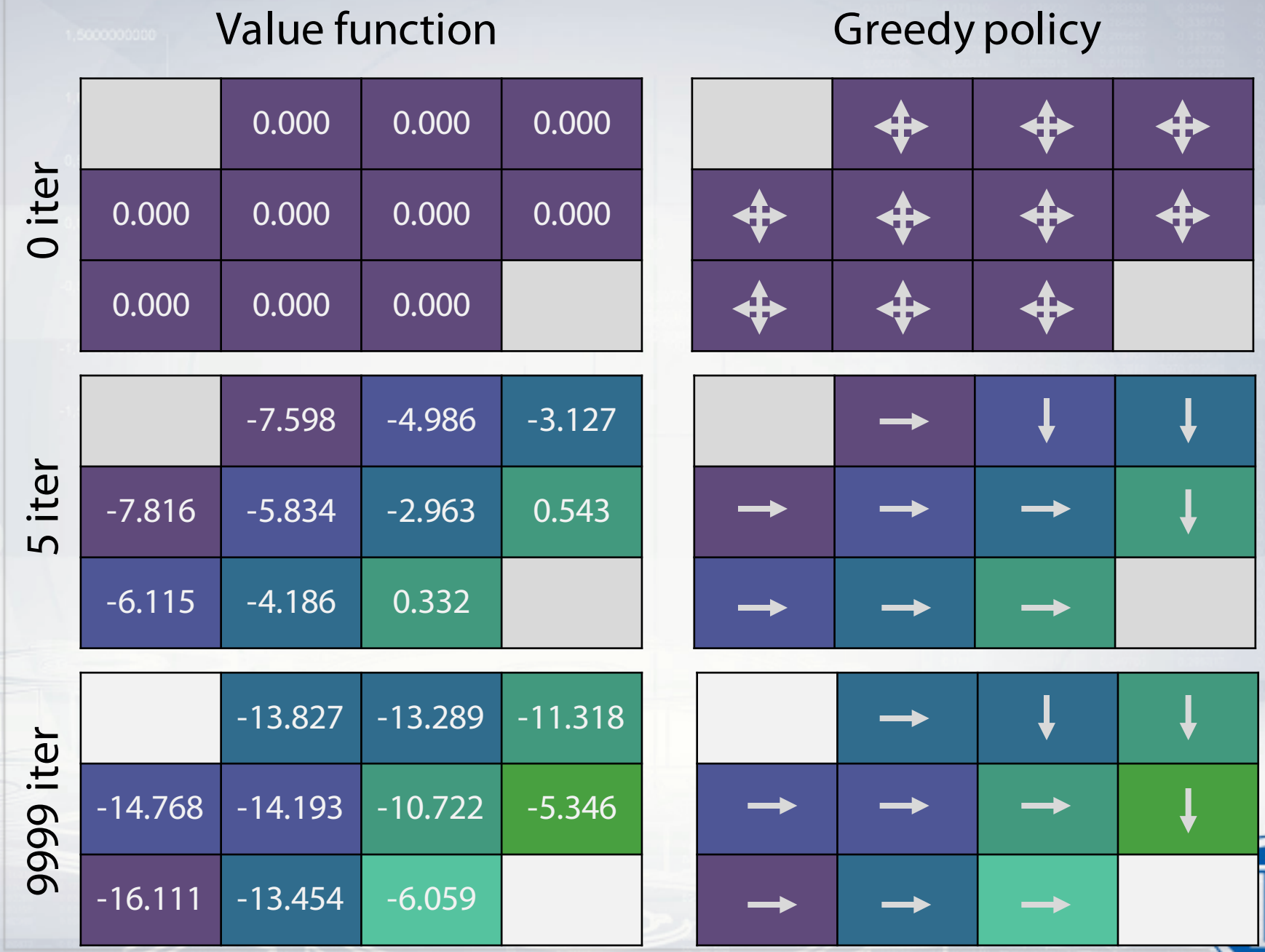
Value function

Greedy policy

0 iter

	0.000	0.000	0.000
0.000	0.000	0.000	0.000
0.000	0.000	0.000	




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

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 an optimal policy?

Generalized policy iteration: Policy and value 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 it's value function

The idea of policy and value iterations

Generalized policy iteration

1. Evaluate given policy
2. Improve policy by acting greedily w.r.t. to it's 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: scheme

1. Initialize $v(s)$ and $\pi(s)$ arbitrarily for all $s \in \mathcal{S}$
2. Perform policy evaluation (without internal initialization)
3. Policy Improvement

`policy_stable` $\leftarrow true$

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

$q(s,a)$

`old_action` $\leftarrow \pi(s)$

$$\pi(s) \leftarrow \text{argmax}_a \sum_{s',r} p(s',r|s,a)[r + \gamma v(s')]$$

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

Value iteration

Initialize array v arbitrarily (e.g., $v(s) = 0$ for all $s \in S$)

Repeat

$\Delta \leftarrow 0$

For each $s \in S$

$v_{old} \leftarrow v(s)$

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

Bellman optimality equation for $v(s)$

$\Delta \leftarrow \max(\Delta, |v_{old}(s) - v(s)|)$

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

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

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

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

- VI is faster per cycle – $O(|A||S|^2)$
- VI requires many cycles

- PI is slower per cycle – $O(|A||S|^2 + |S|^3)$
- PI requires few cycles

No silver bullet → experiment with # of steps spent in policy evaluation phase to find the best algorithm for the task at hand