Reinforcement learning in robotics

Karel Zimmermann

http://cmp.felk.cvut.cz/~zimmerk/

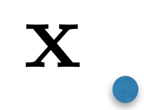
Vision for Robotics and Autonomous Systems
https://cyber.felk.cvut.cz/vras/

Center for Machine Perception
https://cmp.felk.cvut.cz

Department for Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague

Tasks often formalised as MDP

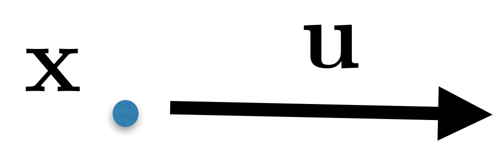
States: $\mathbf{x} \in \mathcal{R}^n$



Czech Technical University in Prague
Faculty of Electrical Engineering, Department of Cybernetics

Both tasks formalised as reinforcement learning problems

States: $\mathbf{x} \in \mathcal{R}^n$



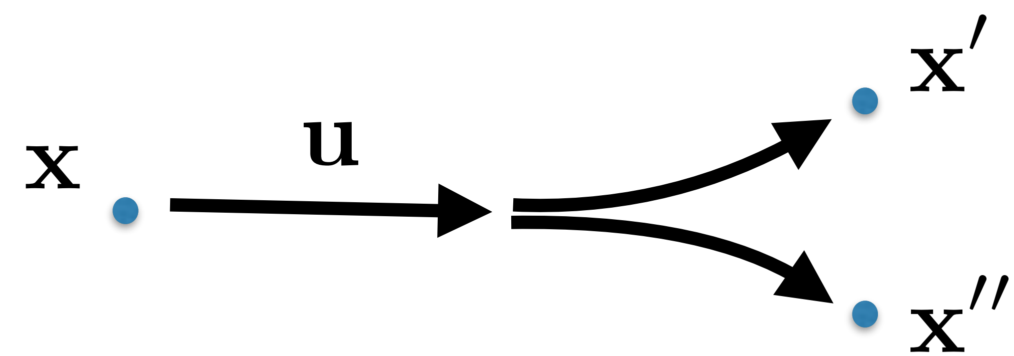
Actions: $\mathbf{u} \in \mathcal{R}^m$

Both tasks formalised as reinforcement learning problems

States: $\mathbf{x} \in \mathcal{R}^n$

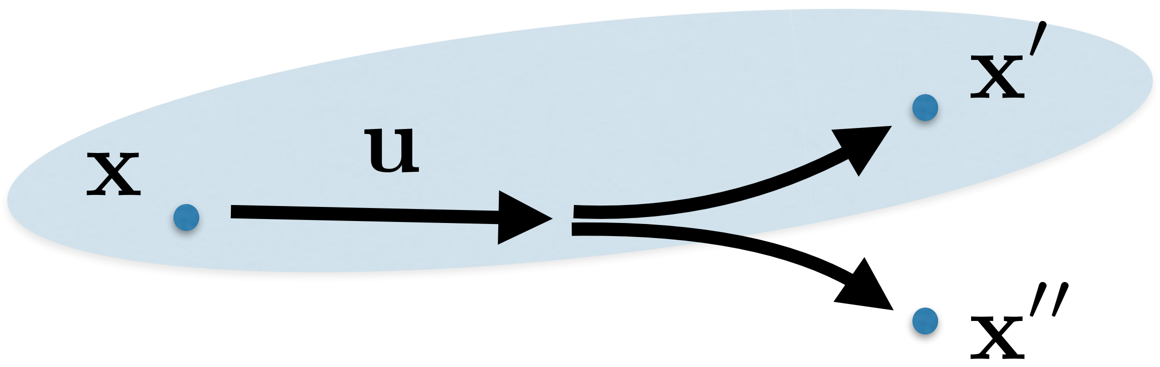
Actions: $\mathbf{u} \in \mathcal{R}^m$

Model: $p(\mathbf{x}' | \mathbf{x}, \mathbf{u})$



Czech Technical University in Prague
Faculty of Electrical Engineering, Department of Cybernetics

Both tasks formalised as reinforcement learning problems



States: $\mathbf{x} \in \mathcal{R}^n$

Actions: $\mathbf{u} \in \mathcal{R}^m$

Model: $p(\mathbf{x}'|\mathbf{x}, \mathbf{u})$

Rewards: $r(\mathbf{x}, \mathbf{u}, \mathbf{x}') \in \mathcal{R}$

Both tasks formalised as reinforcement learning problems

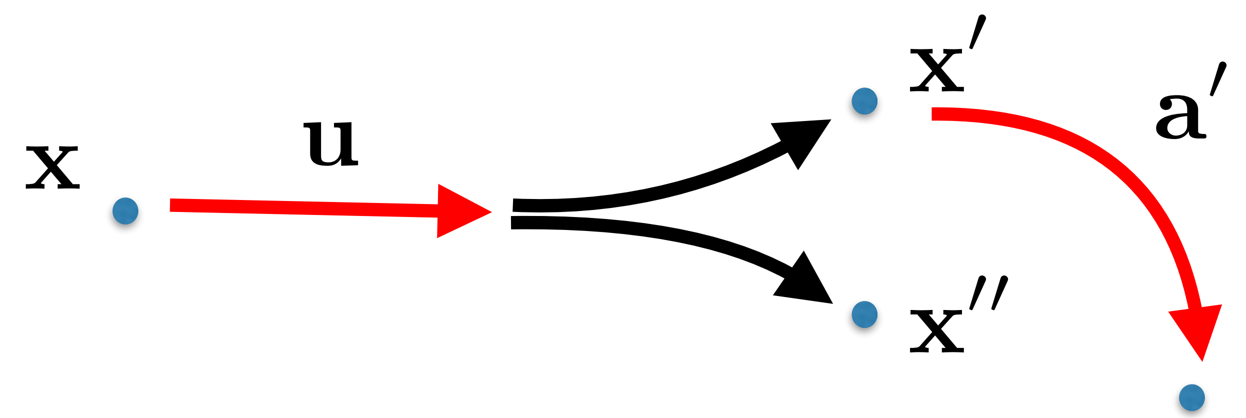
States: $\mathbf{x} \in \mathcal{R}^n$

Actions: $\mathbf{u} \in \mathcal{R}^m$

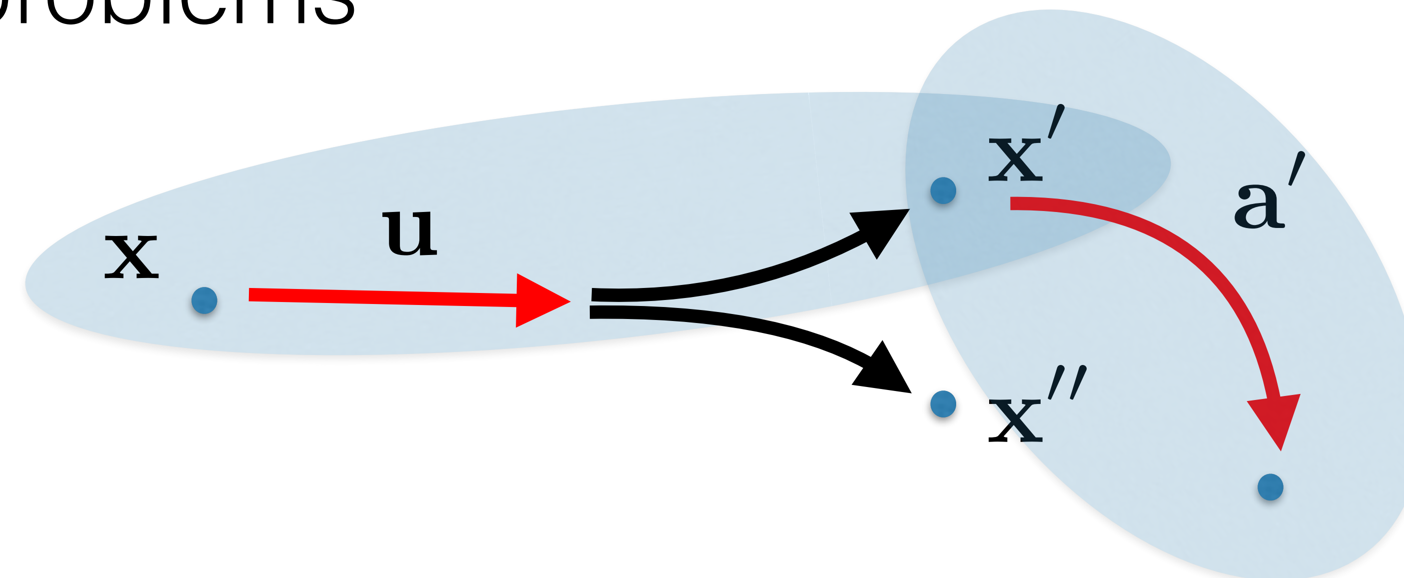
Model: $p(\mathbf{x}'|\mathbf{x}, \mathbf{u})$

Rewards: $r(\mathbf{x}, \mathbf{u}, \mathbf{x}') \in \mathcal{R}$

Policy: $\pi(\mathbf{u}|\mathbf{x})$



Both tasks formalised as reinforcement learning problems



States: $\mathbf{x} \in \mathcal{R}^n$

Actions: $\mathbf{u} \in \mathcal{R}^m$

Model: $p(\mathbf{x}'|\mathbf{x}, \mathbf{u})$

Rewards: $r(\mathbf{x}, \mathbf{u}, \mathbf{x}') \in \mathcal{R}$

Policy: $\pi(\mathbf{u}|\mathbf{x})$

Goal: $\pi^* = \arg\max_{\pi} J_\pi$ (e.g. $J_\pi = \mathtt{E}\left[\sum_{t=0}^{T} r_t\right]$)

Challenges in real tasks

States: $\mathbf{x} \in \mathcal{R}^n$ incomplete, noisy

Actions: $\mathbf{u} \in \mathcal{R}^m$ continuous high-dimensional

Model: $p(\mathbf{x}'|\mathbf{x}, \mathbf{u})$ inaccurate model

Rewards: $r(\mathbf{x}, \mathbf{u}, \mathbf{x}') \in \mathcal{R}$ hard to engineer

Policy: $\pi(\mathbf{u}|\mathbf{x})$ execution endanger the robot

Goal: $\pi^* = \arg\max_{\pi} J_\pi$ (e.g. $J_\pi = \mathtt{E}\left[\sum_{t=0}^{T} r_t\right]$)

Challenges in real tasks

- Can I learn something without the model $p(\mathbf{x}'|\mathbf{x}, \mathbf{u})$ just from interactions?

Taxonomy of policy search methods

- Direct policy search (primal task)

  e.g. gradient ascent for $\pi^* = \arg\max_{\pi} J_\pi$

grey zone

Episodic REPS [Peters, 2010]

PILCO [Deisenroth, ICML 2011]

Actor-critic (e.g. DPG [Silver,JMLR 2014])

Deep Q-learning (e.g. [Mnih,Nature 2015])

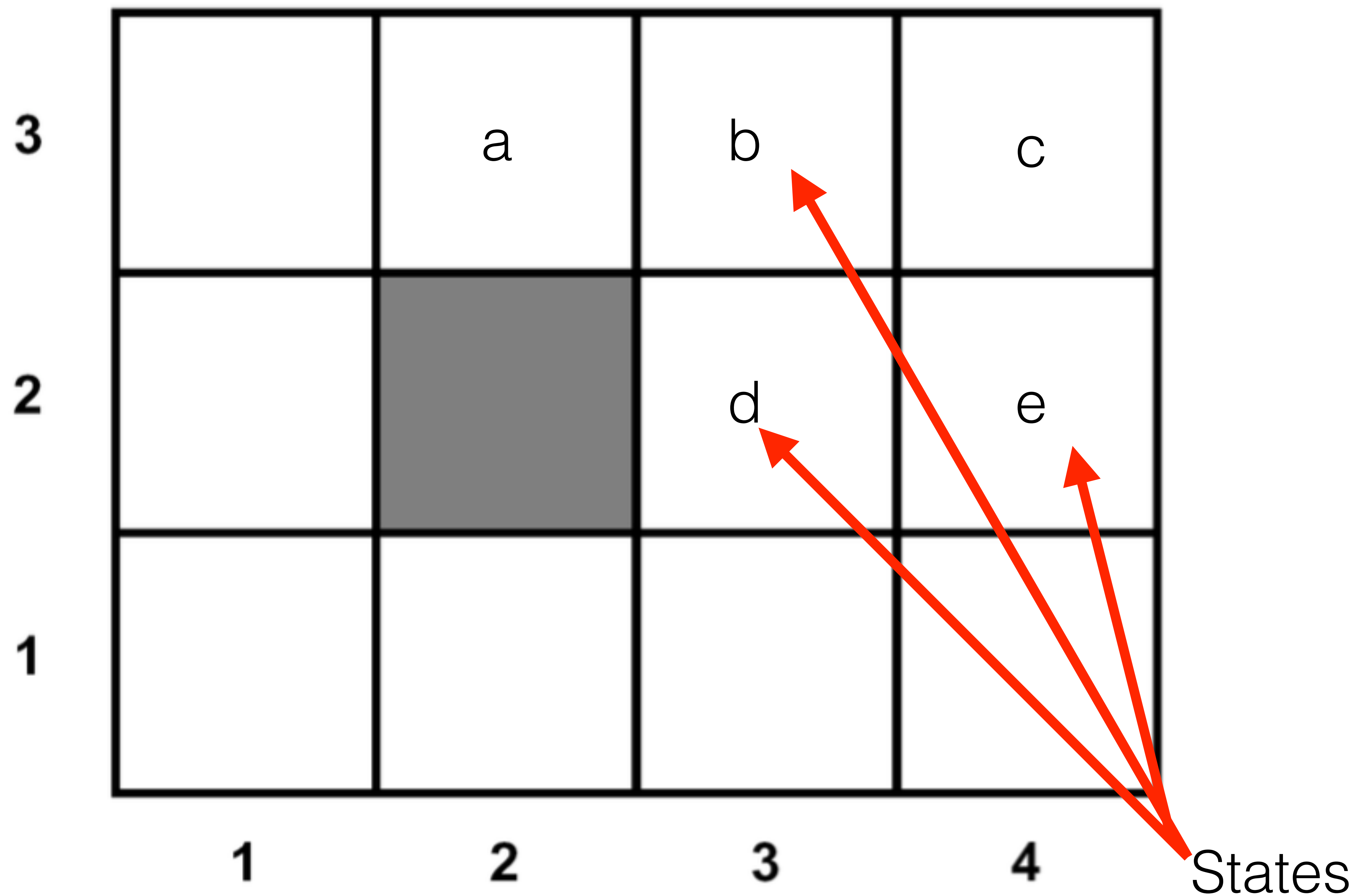
- Value-based methods (dual function [Kober, 2013])

  e.g. search for $Q(\mathbf{x},\mathbf{a}) = r(\mathbf{x},\mathbf{a},\mathbf{x}') + \gamma \max_{\mathbf{a}'} Q(\mathbf{x}',\mathbf{a}')$

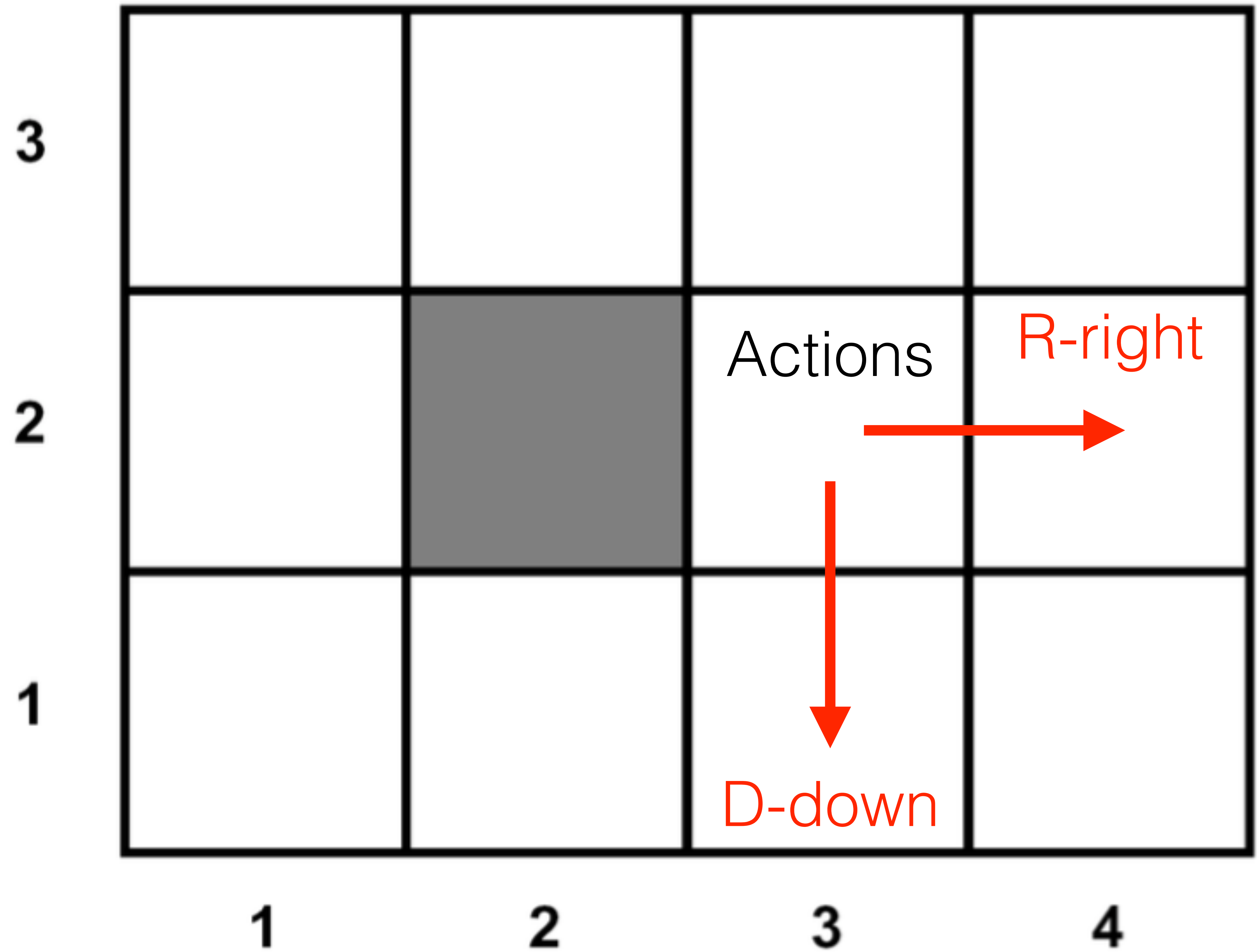
$$\pi^* = \arg\max_{a} Q(\mathbf{x},\mathbf{a})$$

Czech Technical University in Prague
Faculty of Electrical Engineering, Department of Cybernetics

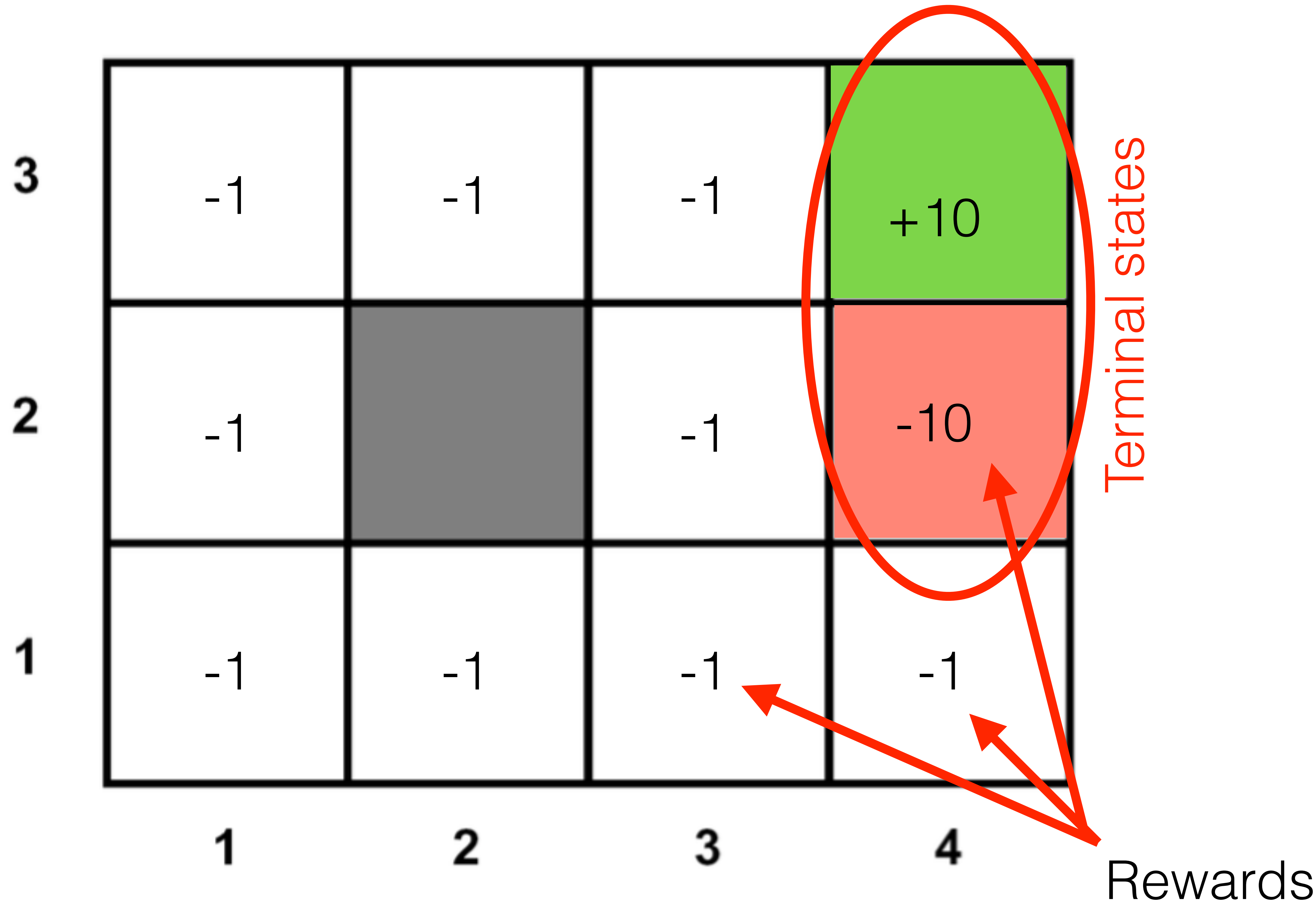
Value-based methods: Q-learning

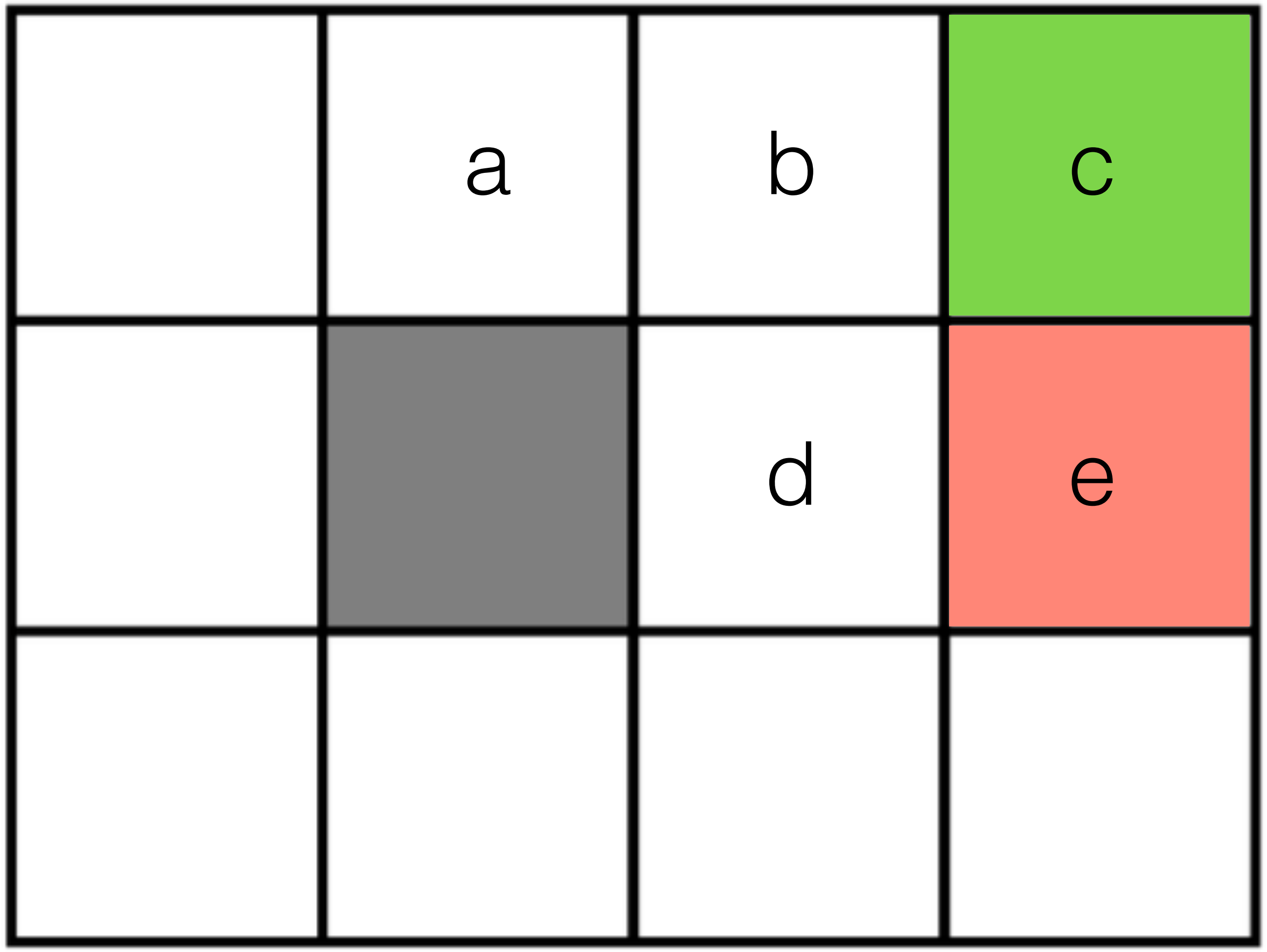


Value-based methods: Q-learning



Value-based methods: Q-learning





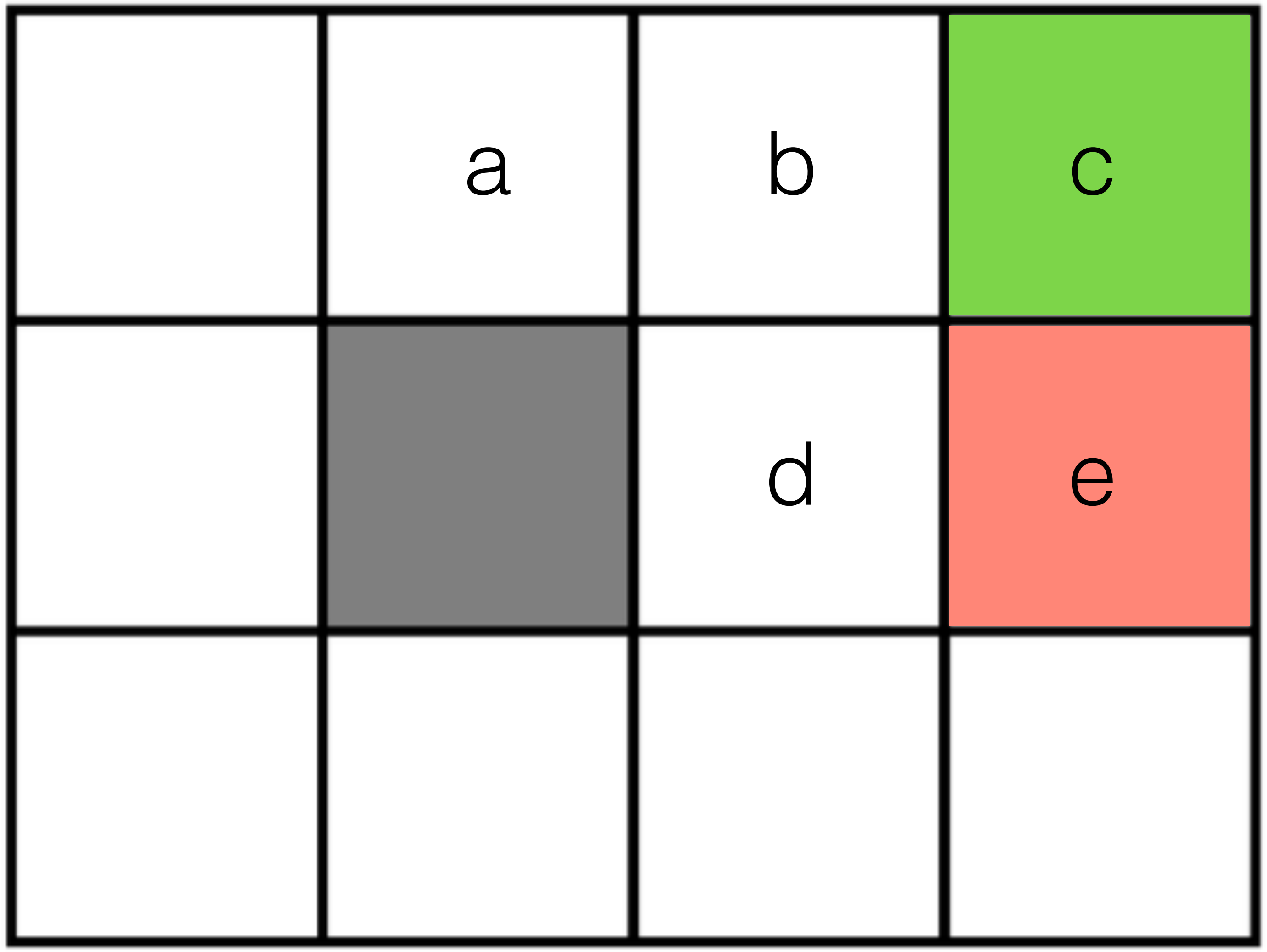
State-action value function

$$Q(\mathbf{x}, \mathbf{u}) : X \times U \rightarrow \mathbb{R}$$

The best sum of rewards I can get, when following action u in state x and then controlling optimally

- Search for the Q, which satisfies Bellman equation

$$Q(\mathbf{x}, \mathbf{u}) = r(\mathbf{x}, \mathbf{u}, \mathbf{x}') + \max_{\mathbf{u}'} Q(\mathbf{x}', \mathbf{u}')$$



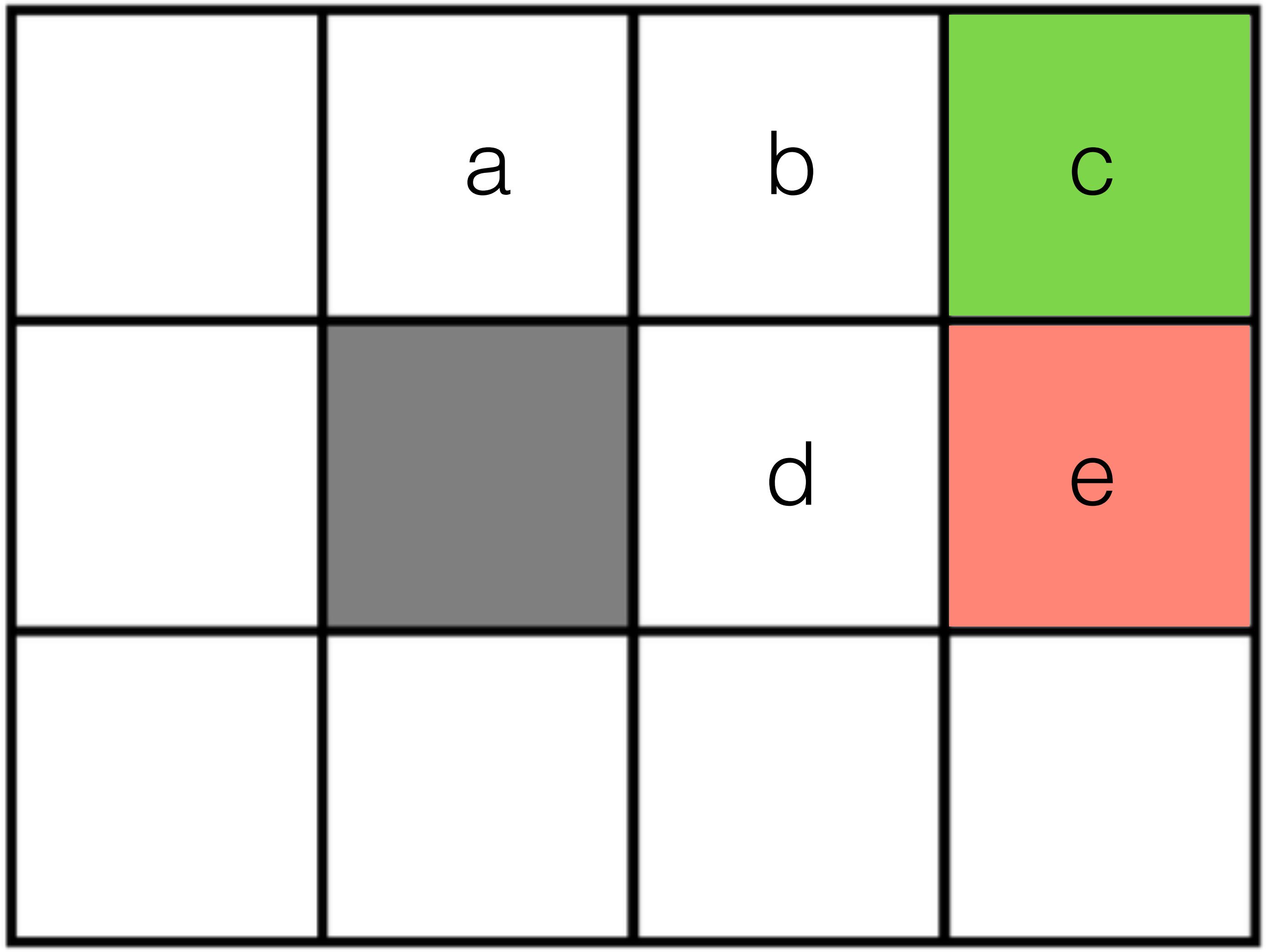
State-action value function

$$Q(\mathbf{x}, \mathbf{u}) : X \times U \rightarrow \mathbb{R}$$

The best sum of rewards I can get, when following action u in state x and then controlling optimally

- Search for the Q, which satisfies Bellman equation
$$Q(\mathbf{x}, \mathbf{u}) = r(\mathbf{x}, \mathbf{u}, \mathbf{x}') + \max_{\mathbf{u}'} Q(\mathbf{x}', \mathbf{u}')$$
- Once we find it, we can control optimally as follows:

$$\pi^*(\mathbf{x}) = \arg\max_{\mathbf{u}} Q(\mathbf{x}, \mathbf{u}) = \arg\max_{\pi} J_\pi$$



State-action value function

$$Q(\mathbf{x},\mathbf{u}) : X \times U \rightarrow \mathbb{R}$$

The best sum of rewards I can get, when following action u in state x and then controlling optimally

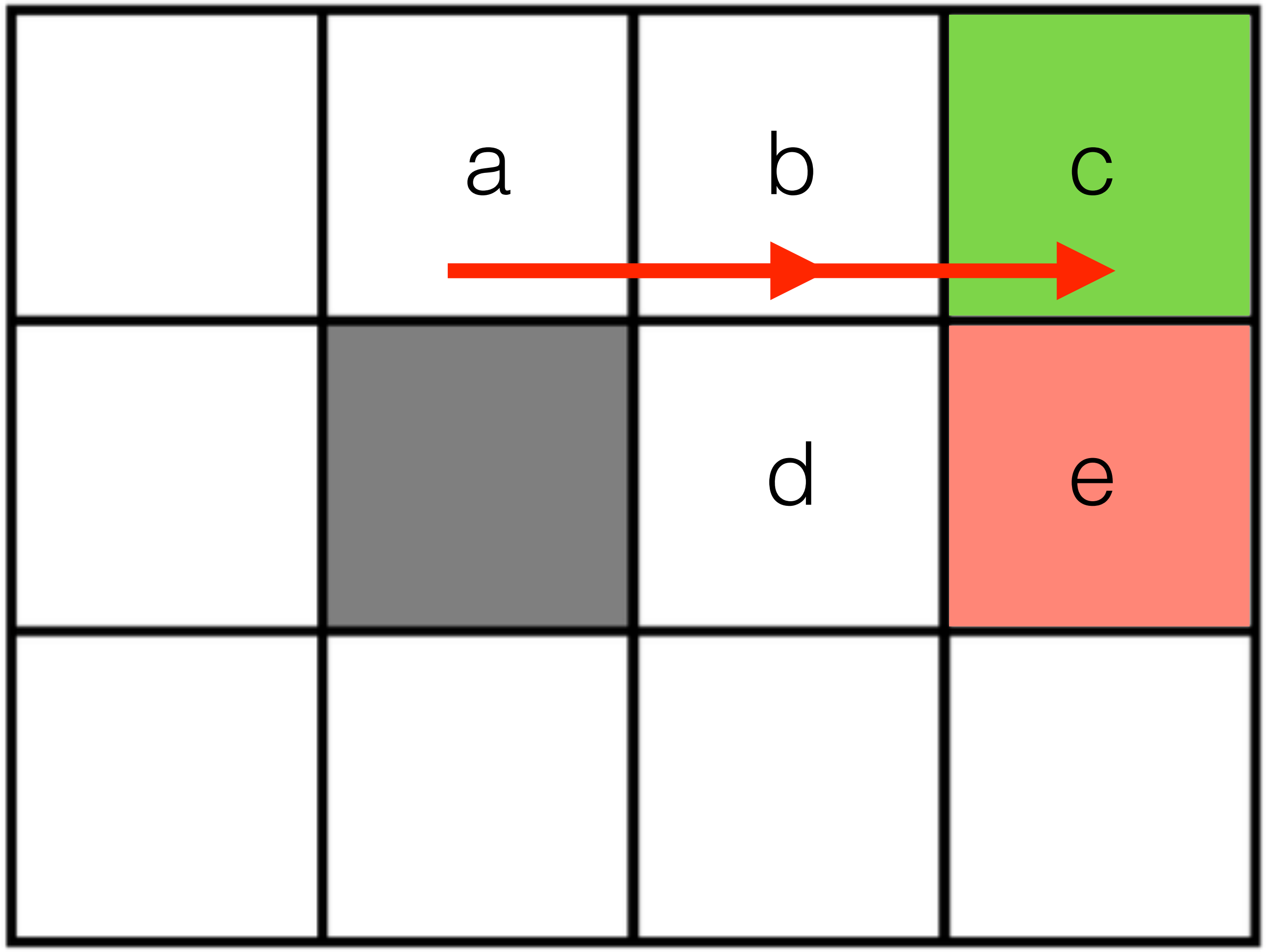
- Search for the Q, which satisfies Bellman equation

$$Q(\mathbf{x},\mathbf{u}) = r(\mathbf{x},\mathbf{u},\mathbf{x}') + \max_{\mathbf{u}'} Q(\mathbf{x}',\mathbf{u}')$$

- Once we find it, we can control optimally as follows:

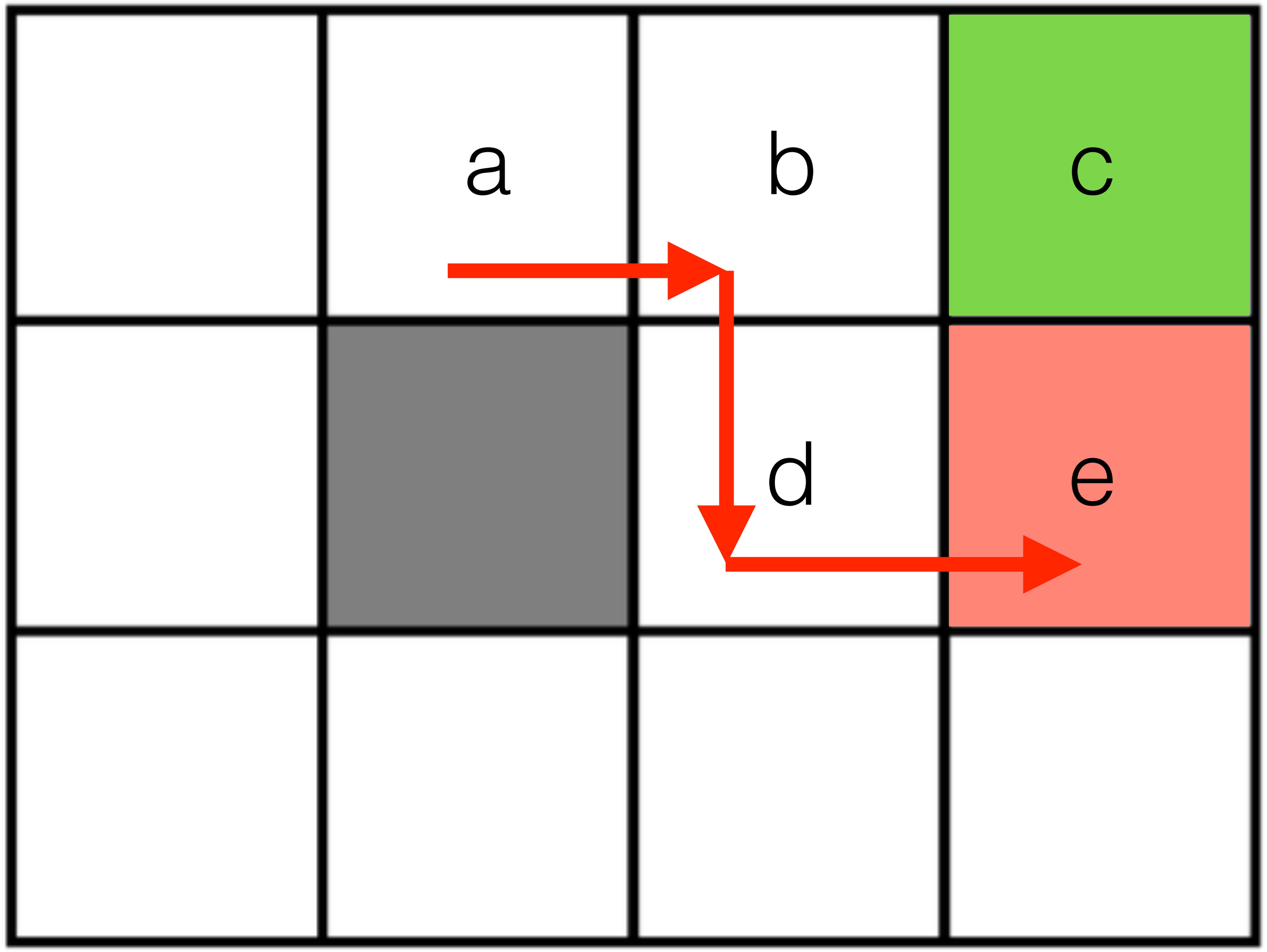
$$\pi^*(\mathbf{x}) = \arg\max_{\mathbf{u}} Q(\mathbf{x},\mathbf{u}) = \arg\max_{\pi} J_\pi$$

- Search without model is based on collecting trajectories



$$\tau_1 :$$
$$(a, R, \text{-}1),\ (b, R, \text{-}1),\ (c, R, 10)$$

Q	R - right	D - down
a	?	?
b	?	?
c	?	?
d	?	?
e	?	?

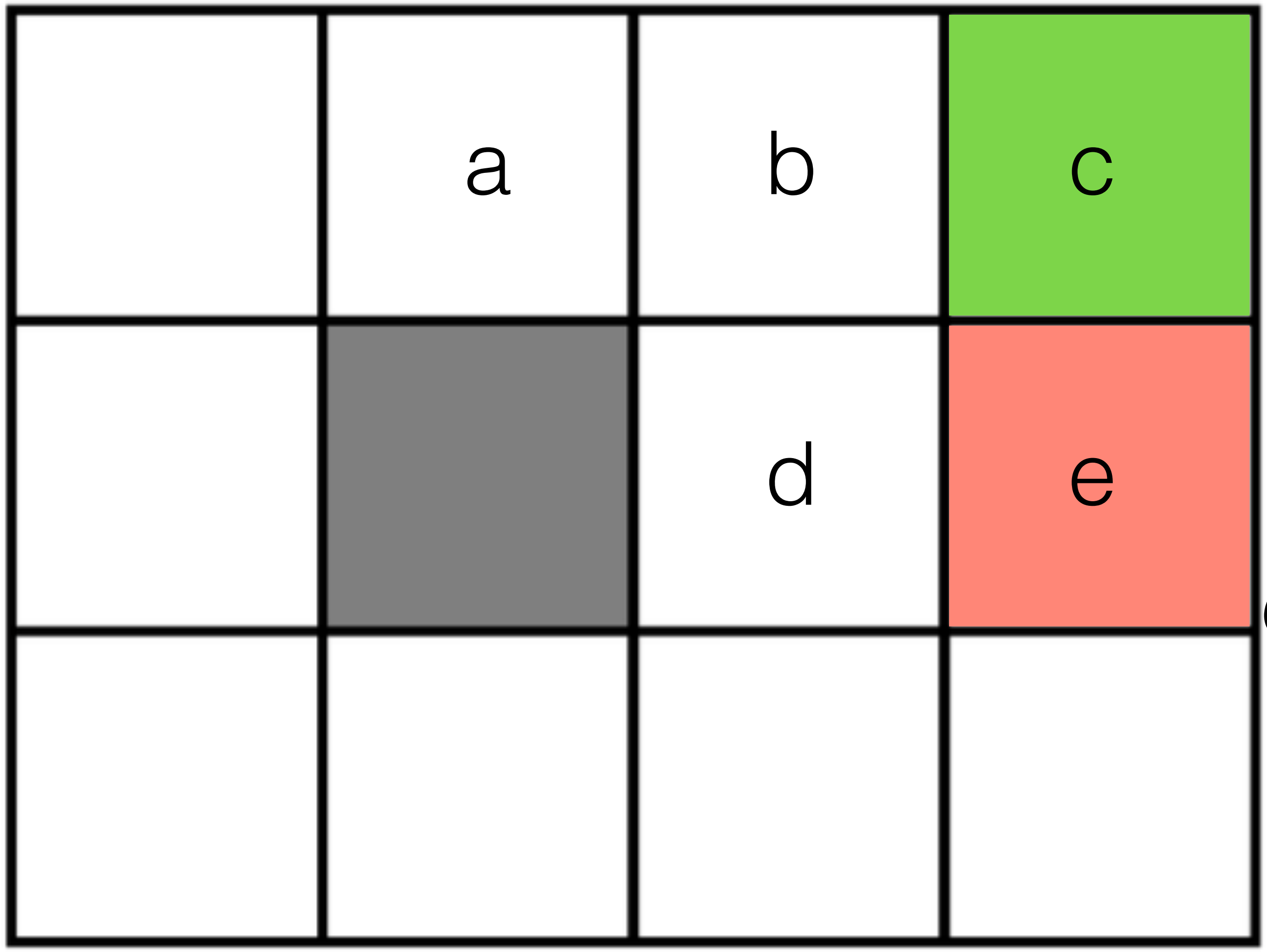


τ_2 :

$$(a, R, \text{-}1),\ (b, D, \text{-}1),$$
$$(d, R, \text{-}1),\ (e, R, \text{-}10)$$

Q	R - right	D - down
a	?	?
b	?	?
c	?	?
d	?	?
e	?	?

Having a trajectory, each transition gives one equation



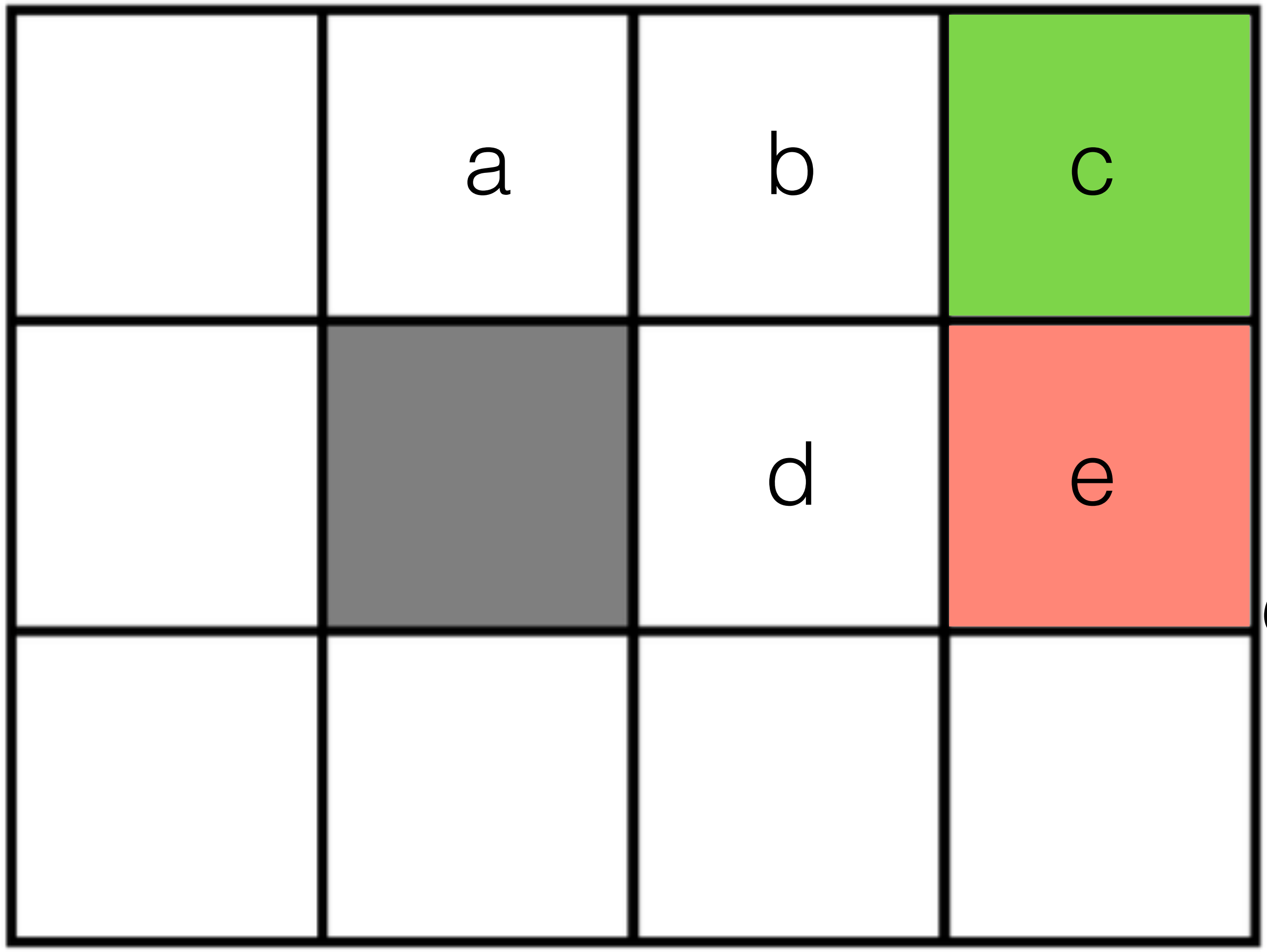
τ_2 :

$(a, R, \text{-}1)$, $(b, D, \text{-}1)$,
$(d, R, \text{-}1)$, $(e, R, \text{-}10)$

$$Q(\mathrm{b}, \mathrm{D}) = r(\mathrm{b}) + \max_{\mathbf{u}} Q(\mathrm{d}, \mathbf{u})$$

Q	R - right	D - down
a	?	?
b	?	?
c	?	?
d	?	?
e	?	?

Having a trajectory, each transition gives one equation



τ_2 :

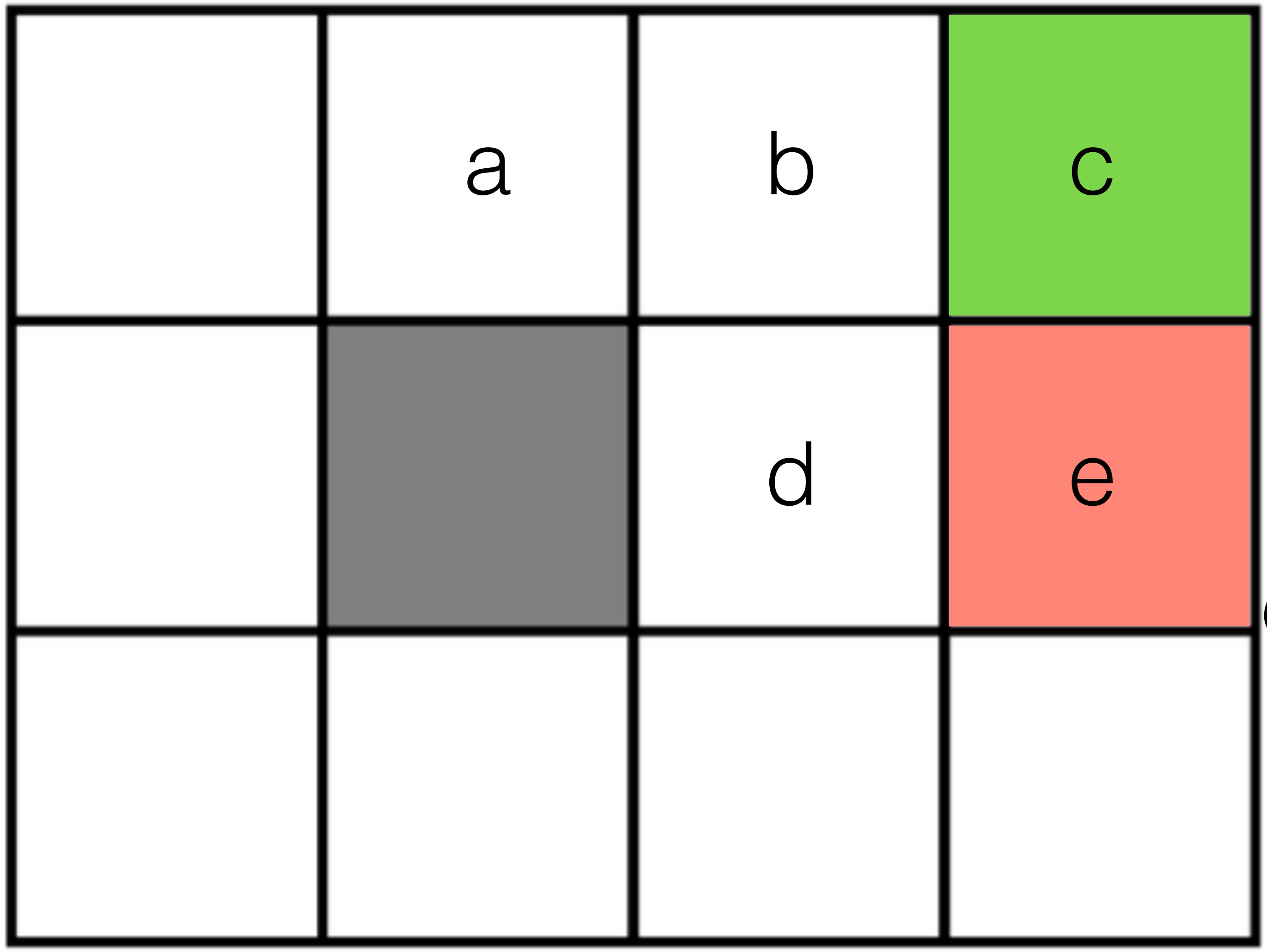
$(a, R, \text{-}1),\ (b, D, \text{-}1),$

$(d, R, \text{-}1),\ (e, R, \text{-}10)$

$$Q(\mathrm{b}, \mathrm{D}) = r(\mathrm{b}) + \max_{\mathbf{u}} Q(\mathrm{d}, \mathbf{u})$$
$$Q(\mathrm{d}, \mathrm{R}) = r(\mathrm{d}) + \max_{\mathbf{u}} Q(\mathrm{e}, \mathbf{u})$$
$$Q(\mathrm{a}, \mathrm{R}) = r(\mathrm{a}) + \max_{\mathbf{u}} Q(\mathrm{b}, \mathbf{u})$$

Q	R - right	D - down
a	?	?
b	?	?
c	?	?
d	?	?
e	?	?

Having a trajectory, each transition gives one equation



$$\tau_2 :$$

$$(a, R, \text{-}1),\ (b, D, \text{-}1),$$

$$(d, R, \text{-}1),\ (e, R, \text{-}10)$$

$$Q(\mathrm{e}, \mathrm{R}) = r(\mathrm{e})$$

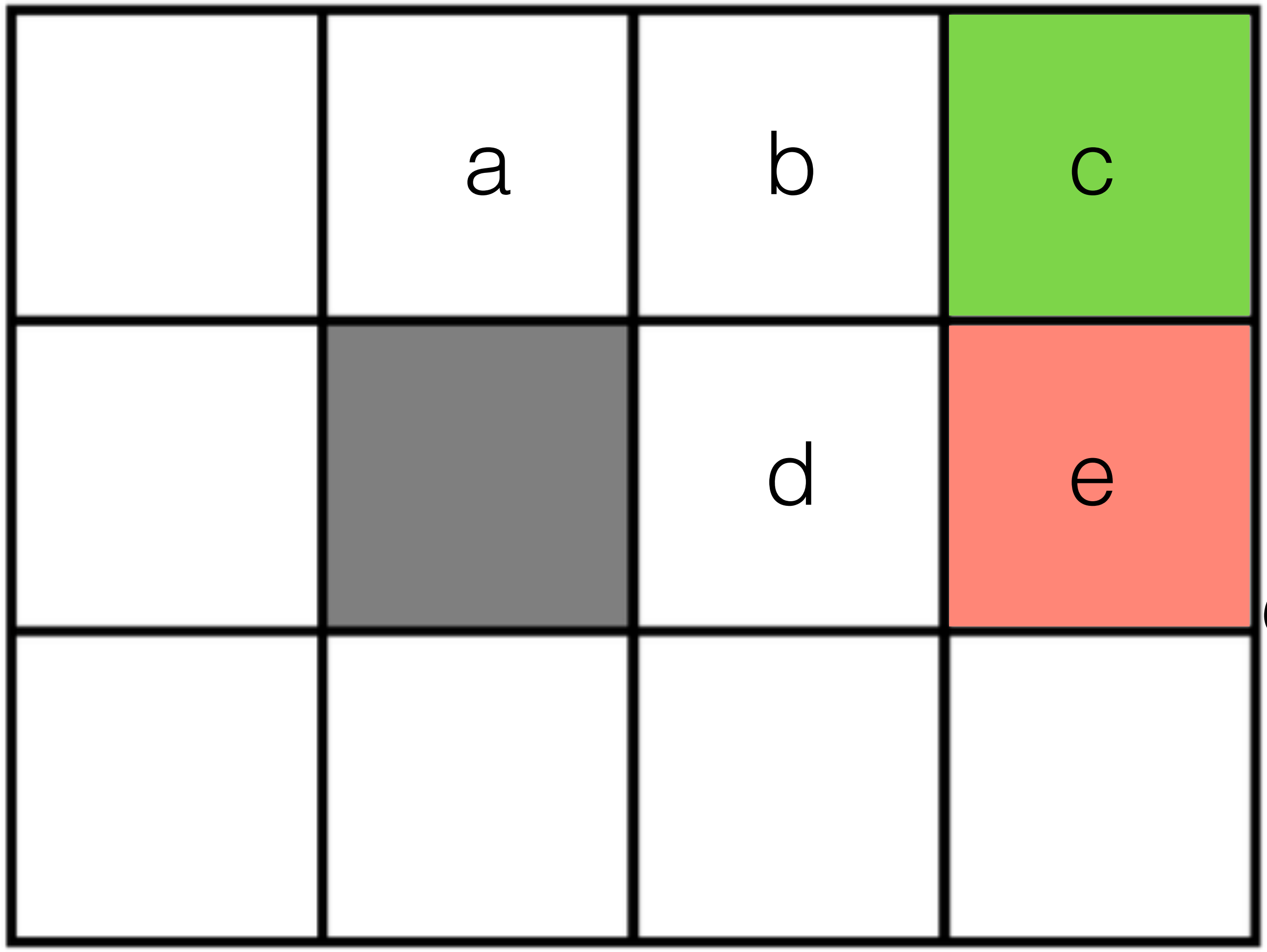
$$Q(\mathrm{b}, \mathrm{D}) = r(\mathrm{b}) + \max_{\mathbf{u}} Q(\mathrm{d}, \mathbf{u})$$

$$Q(\mathrm{d}, \mathrm{R}) = r(\mathrm{d}) + \max_{\mathbf{u}} Q(\mathrm{e}, \mathbf{u})$$

$$Q(\mathrm{a}, \mathrm{R}) = r(\mathrm{a}) + \max_{\mathbf{u}} Q(\mathrm{b}, \mathbf{u})$$

Q	R - right	D - down
a	?	?
b	?	?
c	?	?
d	?	?
e	?	?

Having a trajectory, each transition gives one equation



$$\tau_2:$$

$$(a, R, \text{-}1),\ (b, D, \text{-}1),$$

$$(d, R, \text{-}1),\ (e, R, \text{-}10)$$

$$Q(\mathrm{e}, \mathrm{R}) = r(\mathrm{e})$$

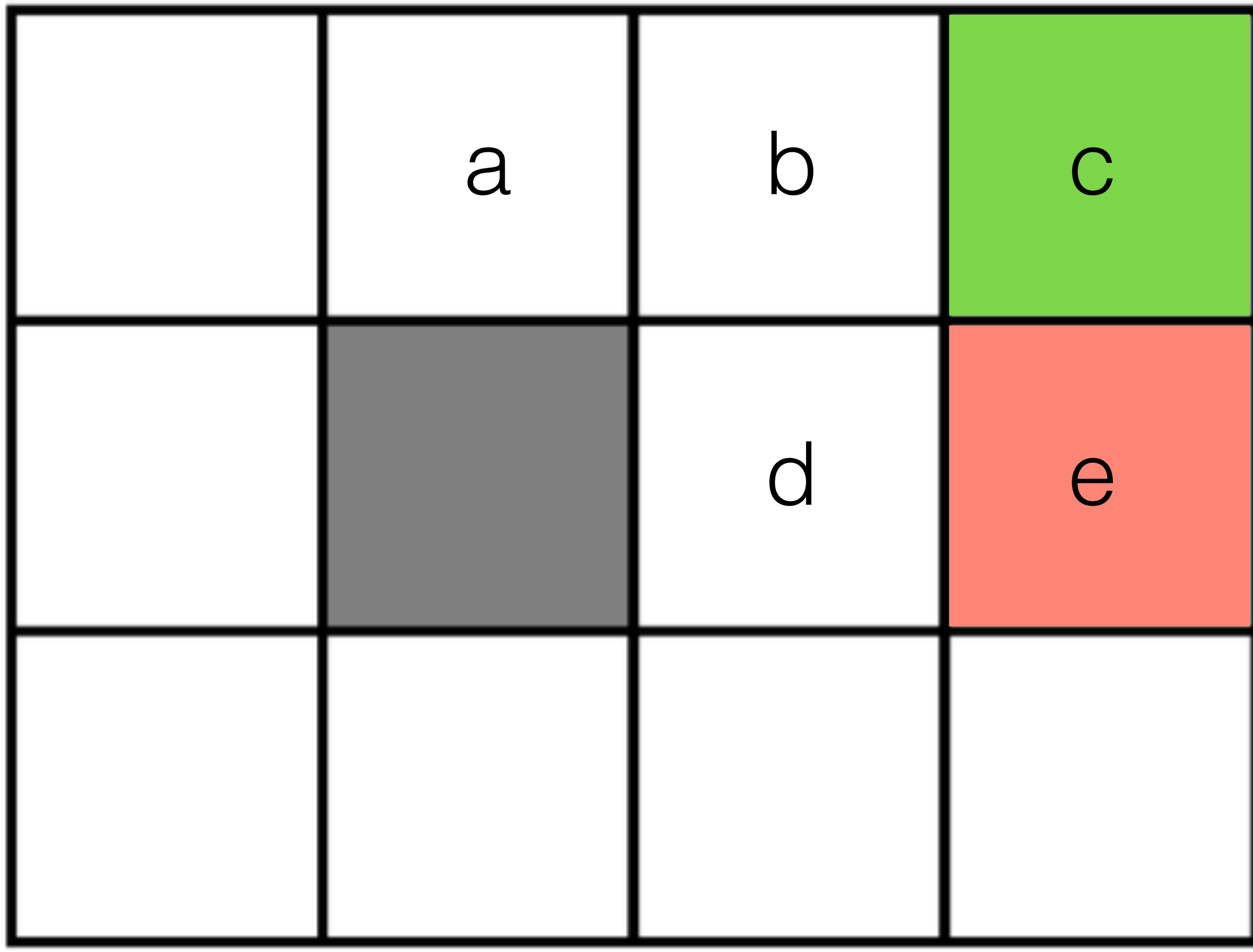
$$Q(\mathrm{b}, \mathrm{D}) = r(\mathrm{b}) + \max_{\mathbf{u}} Q(\mathrm{d}, \mathbf{u})$$

$$Q(\mathrm{d}, \mathrm{R}) = r(\mathrm{d}) + \max_{\mathbf{u}} Q(\mathrm{e}, \mathbf{u})$$

$$Q(\mathrm{a}, \mathrm{R}) = r(\mathrm{a}) + \max_{\mathbf{u}} Q(\mathrm{b}, \mathbf{u})$$

Q	R - right	D - down
a	?	?
b	?	?
c	?	?
d	?	?
e	?	?

Having a trajectory, each transition gives one equation



τ_2 :

$(a, R, \text{-}1),\ (b, D, \text{-}1),$

$(d, R, \text{-}1),\ (e, R, \text{-}10)$

$$Q(\mathrm{e}, \mathrm{R}) = r(\mathrm{e})$$

$$Q(\mathrm{b}, \mathrm{D}) = r(\mathrm{b}) + \max_{\mathbf{u}} Q(\mathrm{d}, \mathbf{u})$$

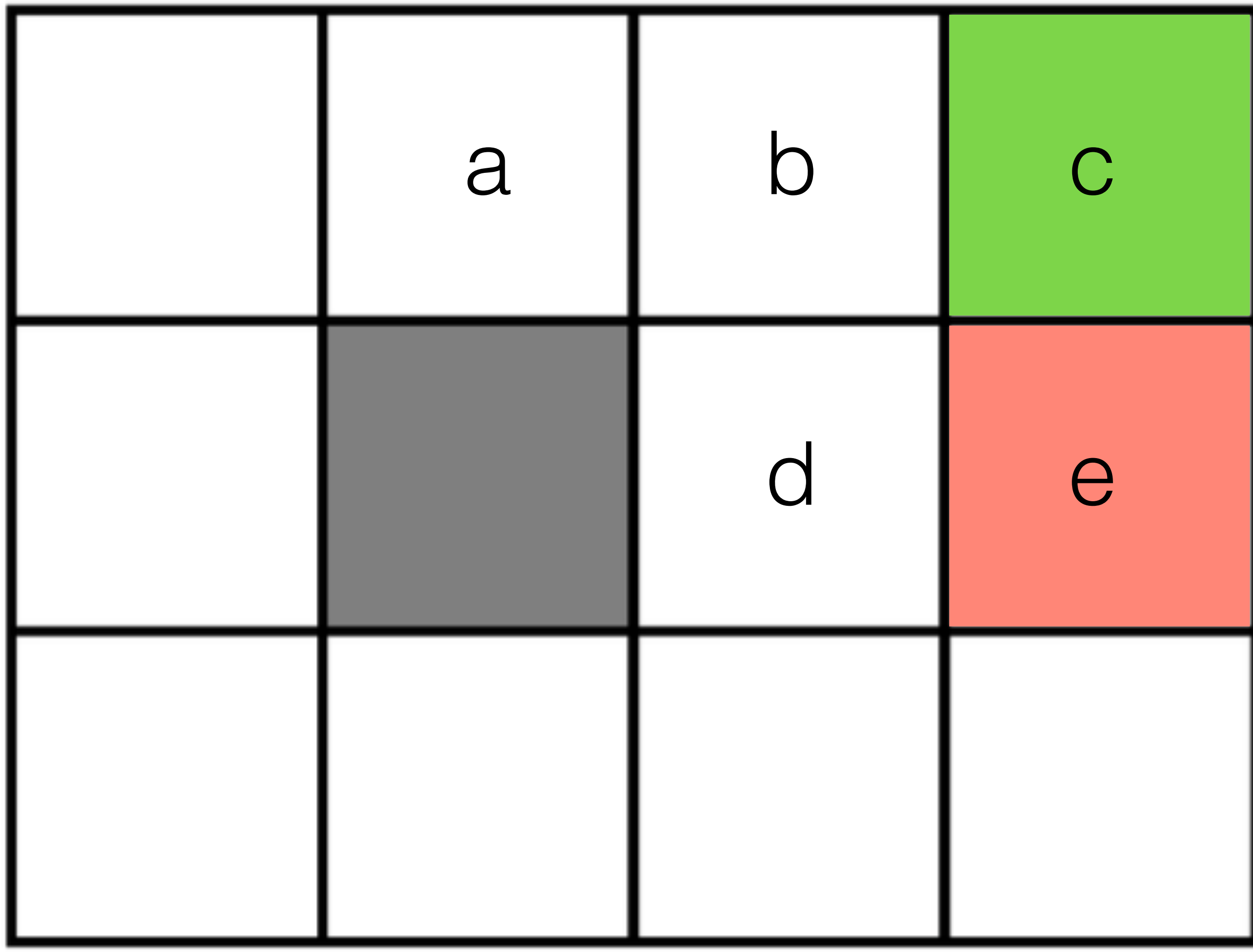
$$Q(\mathrm{d}, \mathrm{R}) = r(\mathrm{d}) + \max_{\mathbf{u}} Q(\mathrm{e}, \mathbf{u})$$

$$Q(\mathrm{a}, \mathrm{R}) = r(\mathrm{a}) + \max_{\mathbf{u}} Q(\mathrm{b}, \mathbf{u})$$

Q	R - right	D - down
a	?	?
b	?	?
c	?	?
d	?	?
e	?	?

unknowns

Having a trajectory, each transition gives one equation



τ_2 :

$(a, R, \text{-}1), \ (b, D, \text{-}1),$

$(d, R, \text{-}1), \ (e, R, \text{-}10)$

$$Q(\mathrm{e}, \mathrm{R}) = r(\mathrm{e})$$

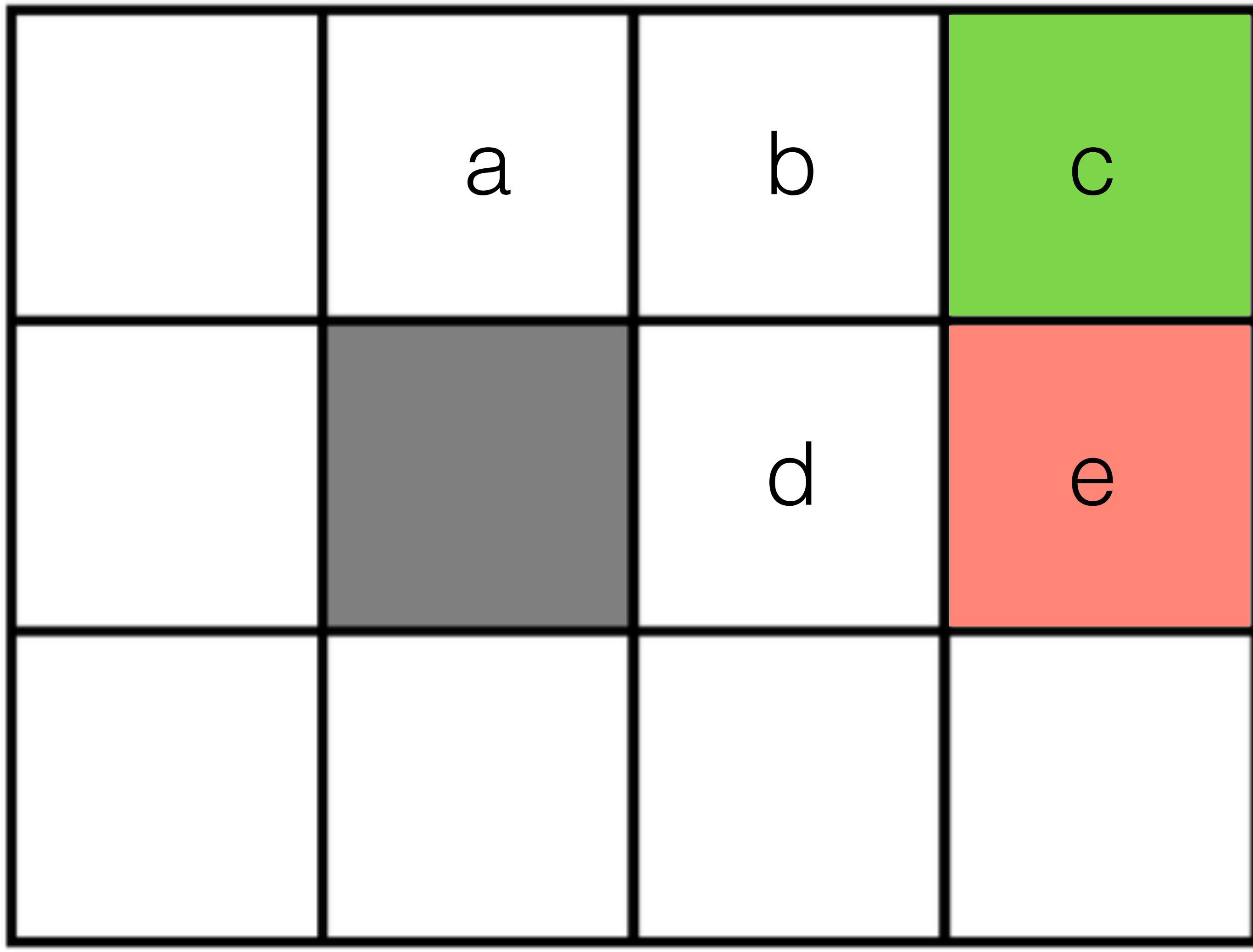
$$Q(\mathrm{b}, \mathrm{D}) = r(\mathrm{b}) + \max_{\mathbf{u}} Q(\mathrm{d}, \mathbf{u})$$

$$Q(\mathrm{d}, \mathrm{R}) = r(\mathrm{d}) + \max_{\mathbf{u}} Q(\mathrm{e}, \mathbf{u})$$

$$Q(\mathrm{a}, \mathrm{R}) = r(\mathrm{a}) + \max_{\mathbf{u}} Q(\mathrm{b}, \mathbf{u})$$

Q	R - right	D - down
a	0	0
b	0	0
c	0	0
d	0	0
e	0	0

(1) Substitute transitions and current Q-values to the right side and solve for left side.



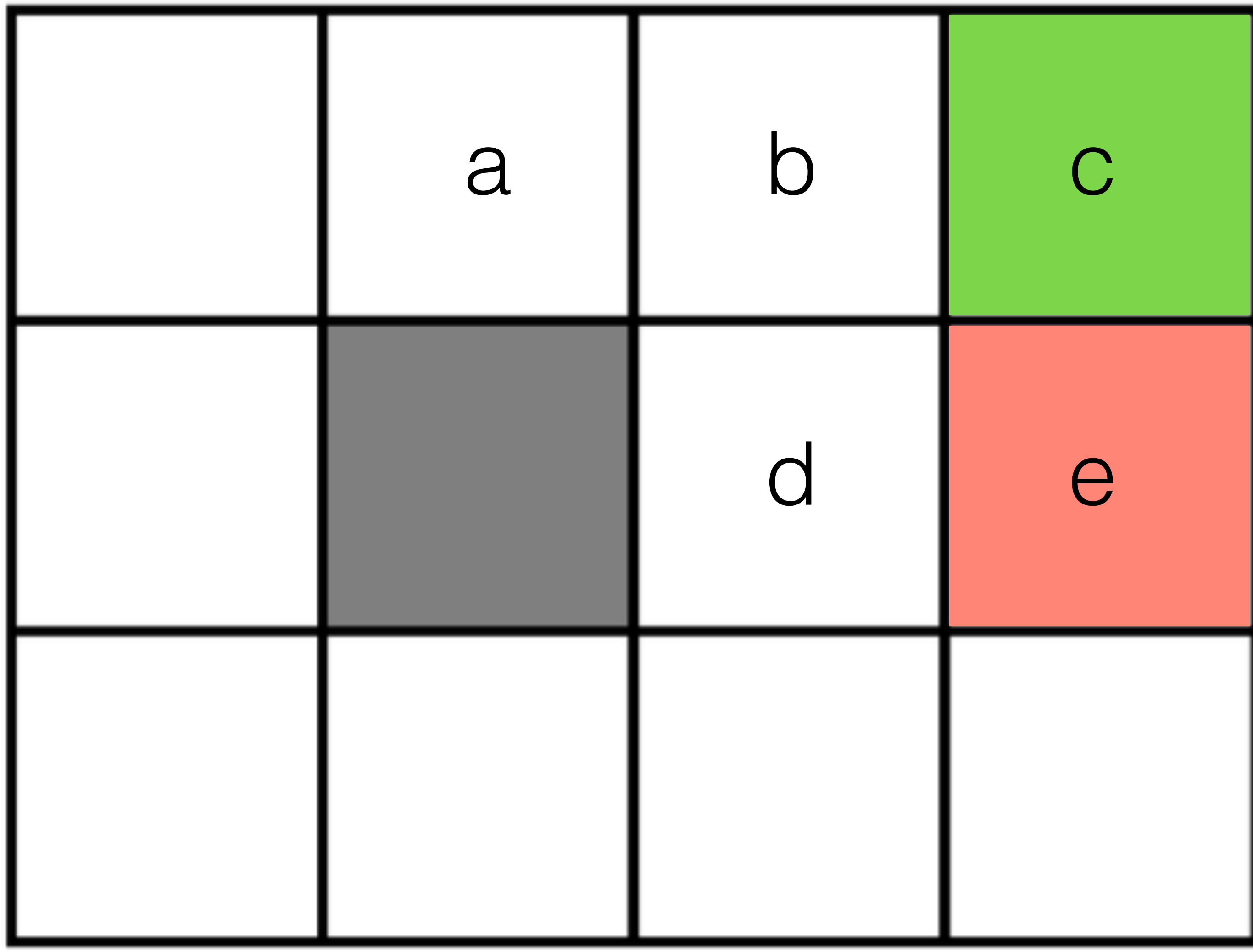
τ_2 :

$(a, R, \text{-}1),\ (b, D, \text{-}1),$
$(d, R, \text{-}1),\ (e, R, \text{-}10)$

$$Q(\mathrm{e}, \mathrm{R}) = -10$$
$$Q(\mathrm{b}, \mathrm{D}) = r(\mathrm{b}) + \max_{\mathbf{u}} Q(\mathrm{d}, \mathbf{u})$$
$$Q(\mathrm{d}, \mathrm{R}) = r(\mathrm{d}) + \max_{\mathbf{u}} Q(\mathrm{e}, \mathbf{u})$$
$$Q(\mathrm{a}, \mathrm{R}) = r(\mathrm{a}) + \max_{\mathbf{u}} Q(\mathrm{b}, \mathbf{u})$$

Q	R - right	D - down
a	0	0
b	0	0
c	0	0
d	0	0
e	-10	0

(1) Substitute transitions and current Q-values to the right side and solve for left side.



τ_2 :

$(a, R, \text{-}1)$, $(b, D, \text{-}1)$,

$(d, R, \text{-}1)$, $(e, R, \text{-}10)$

$$Q(\mathrm{e}, \mathrm{R}) = \text{-}10$$

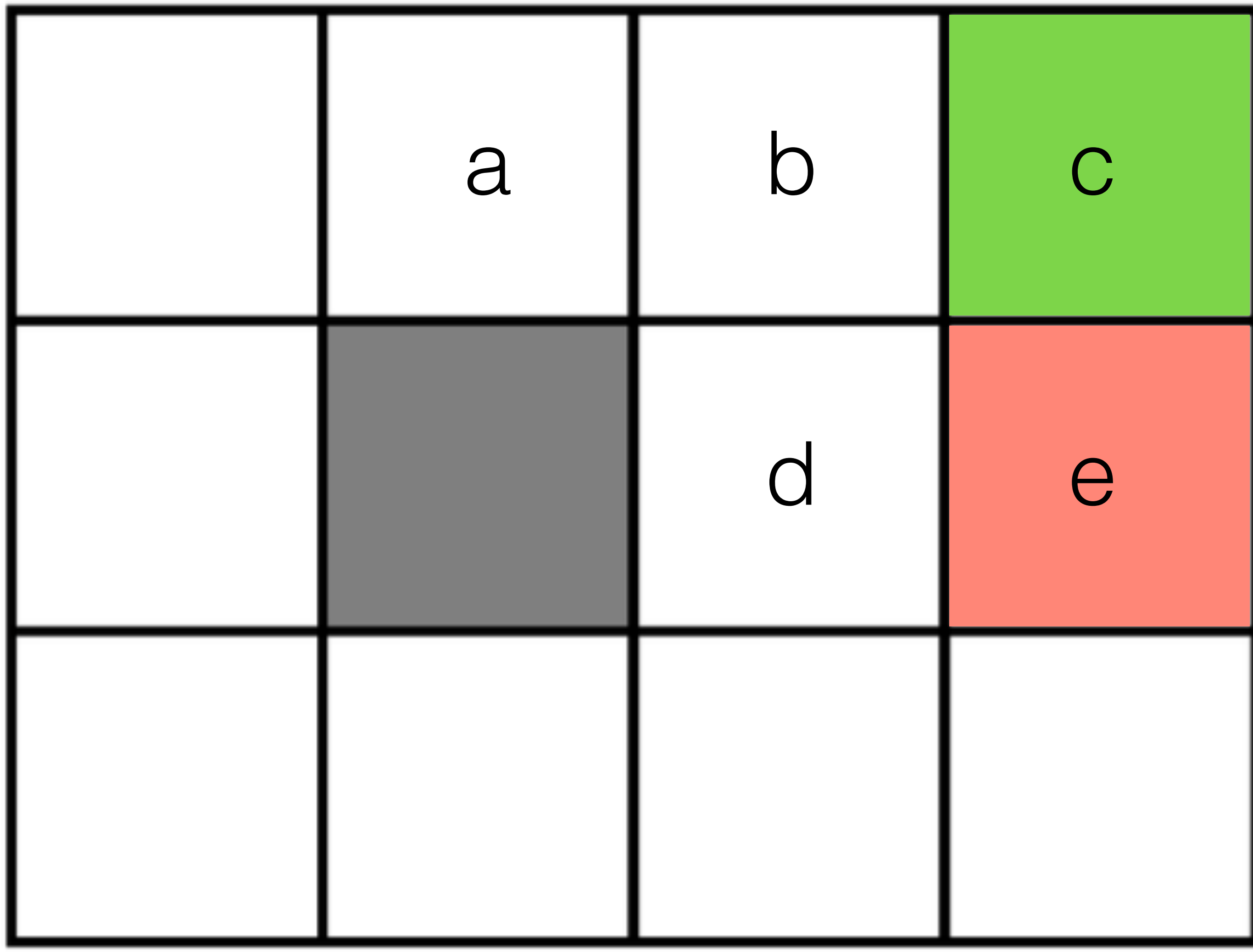
$$Q(\mathrm{b}, \mathrm{D}) = \textbf{???}$$

$$Q(\mathrm{d}, \mathrm{R}) = r(\mathrm{d}) + \max_{\mathbf{u}} Q(\mathrm{e}, \mathbf{u})$$

$$Q(\mathrm{a}, \mathrm{R}) = r(\mathrm{a}) + \max_{\mathbf{u}} Q(\mathrm{b}, \mathbf{u})$$

Q	R - right	D - down
a	0	0
b	0	-1
c	0	0
d	0	0
e	-10	0

(1) Substitute transitions and current Q-values to the right side and solve for left side.



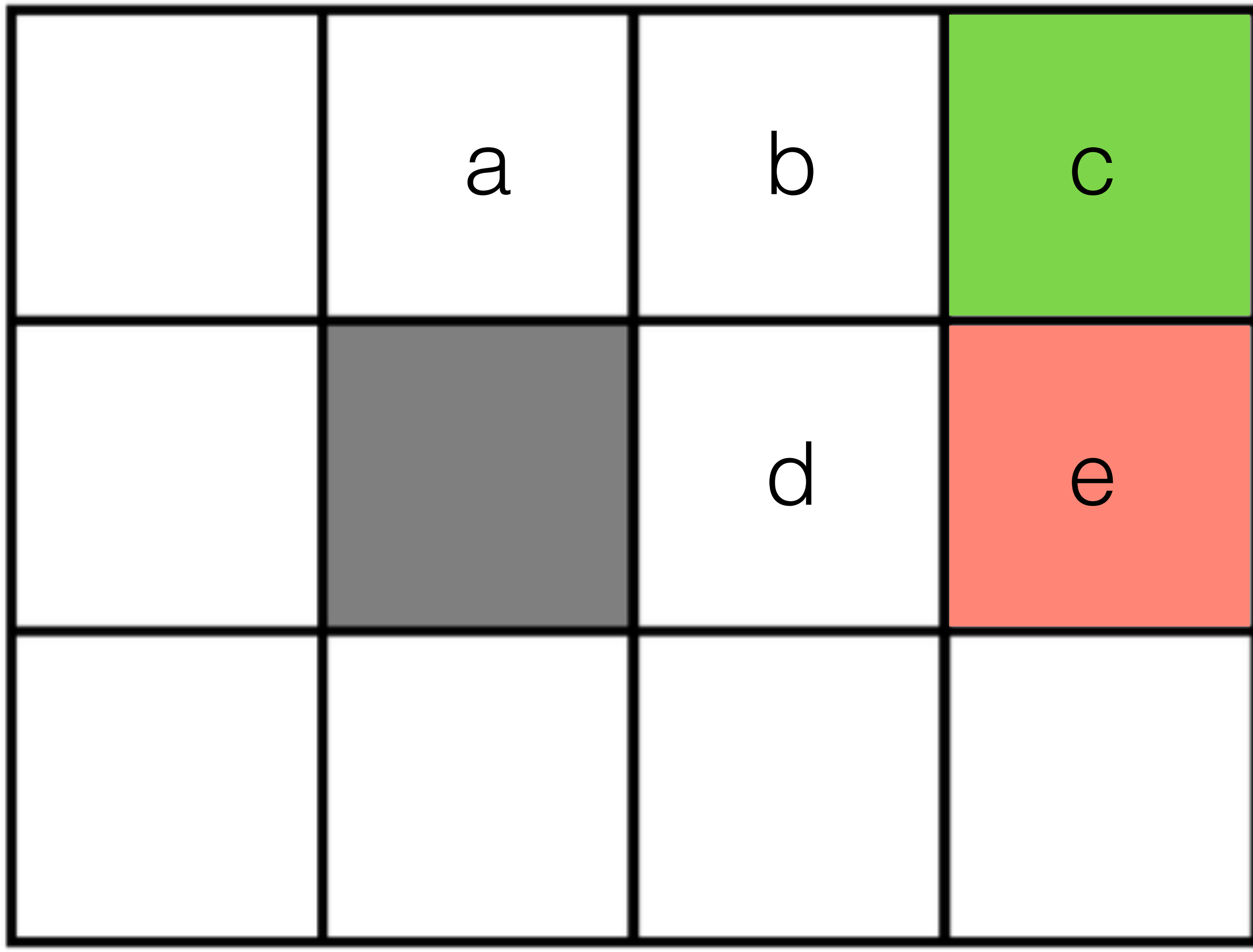
τ_2 :

$(a, R, \text{-}1)$, $(b, D, \text{-}1)$,
$(d, R, \text{-}1)$, $(e, R, \text{-}10)$

$$\mathrm{Q(e, R)} = -10$$
$$\mathrm{Q(b, D)} = -1$$
$$\mathrm{Q(d, R)} = r(\mathrm{d}) + \max_{\mathbf{u}} Q(\mathrm{e}, \mathbf{u})$$
$$\mathrm{Q(a, R)} = r(\mathrm{a}) + \max_{\mathbf{u}} Q(\mathrm{b}, \mathbf{u})$$

Q	R - right	D - down
a	0	0
b	0	-1
c	0	0
d	0	0
e	-10	0

(1) Substitute transitions and current Q-values to the right side and solve for left side.



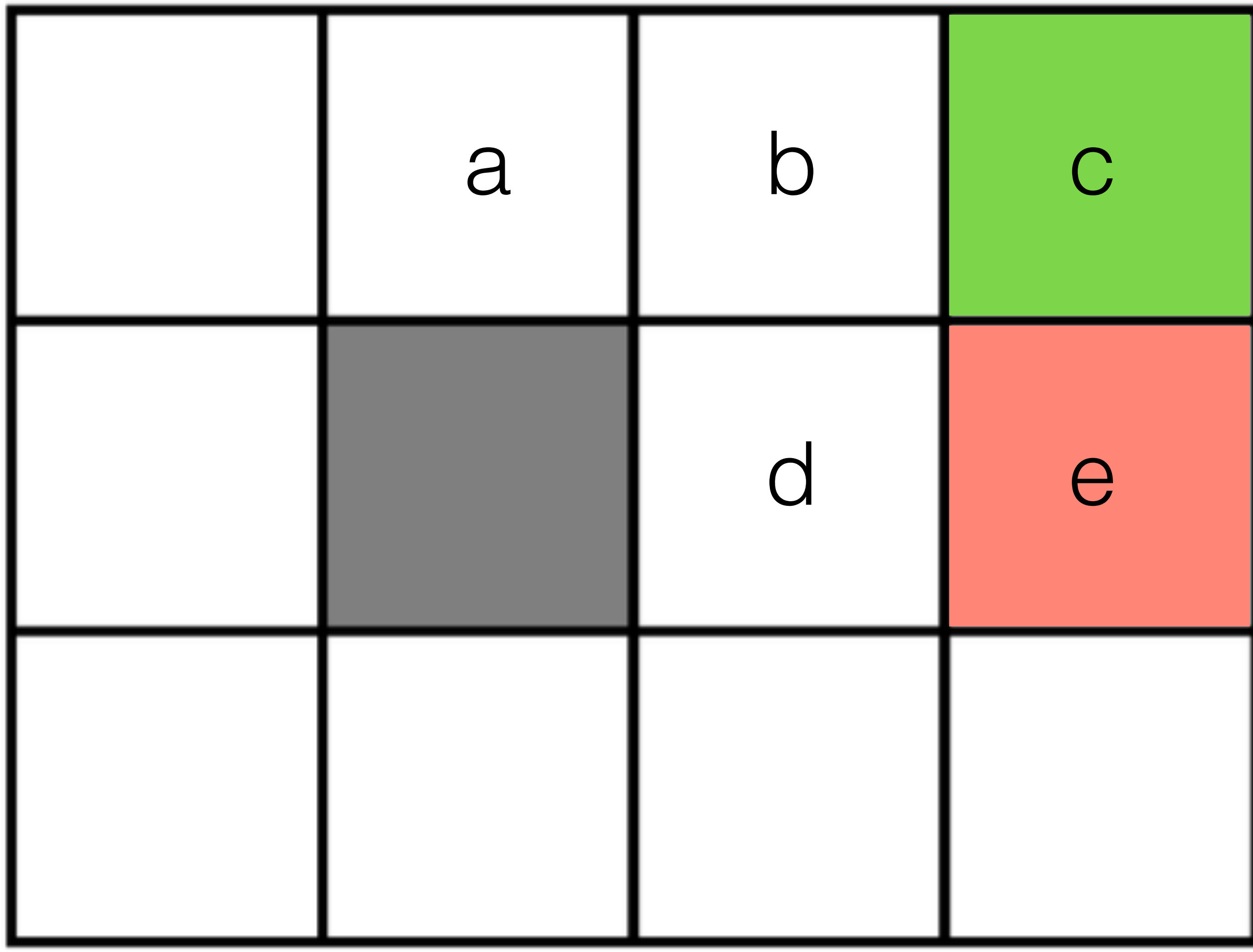
τ_2 :

$(a, R, \text{-}1),\ (b, D, \text{-}1),$
$(d, R, \text{-}1),\ (e, R, \text{-}10)$

$$Q(\mathrm{e}, \mathrm{R}) = -10$$
$$Q(\mathrm{b}, \mathrm{D}) = -1$$
$$Q(\mathrm{d}, \mathrm{R}) = -1$$
$$Q(\mathrm{a}, \mathrm{R}) = r(\mathrm{a}) + \max_{\mathbf{u}} Q(\mathrm{b}, \mathbf{u})$$

Q	R - right	D - down
a	0	0
b	0	-1
c	0	0
d	-1	0
e	-10	0

(1) Substitute transitions and current Q-values to the right side and solve for left side.



τ_2 :

$(a, R, \text{-}1),\ (b, D, \text{-}1),$
$(d, R, \text{-}1),\ (e, R, \text{-}10)$

$Q(\mathrm{e}, \mathrm{R}) =$ -10

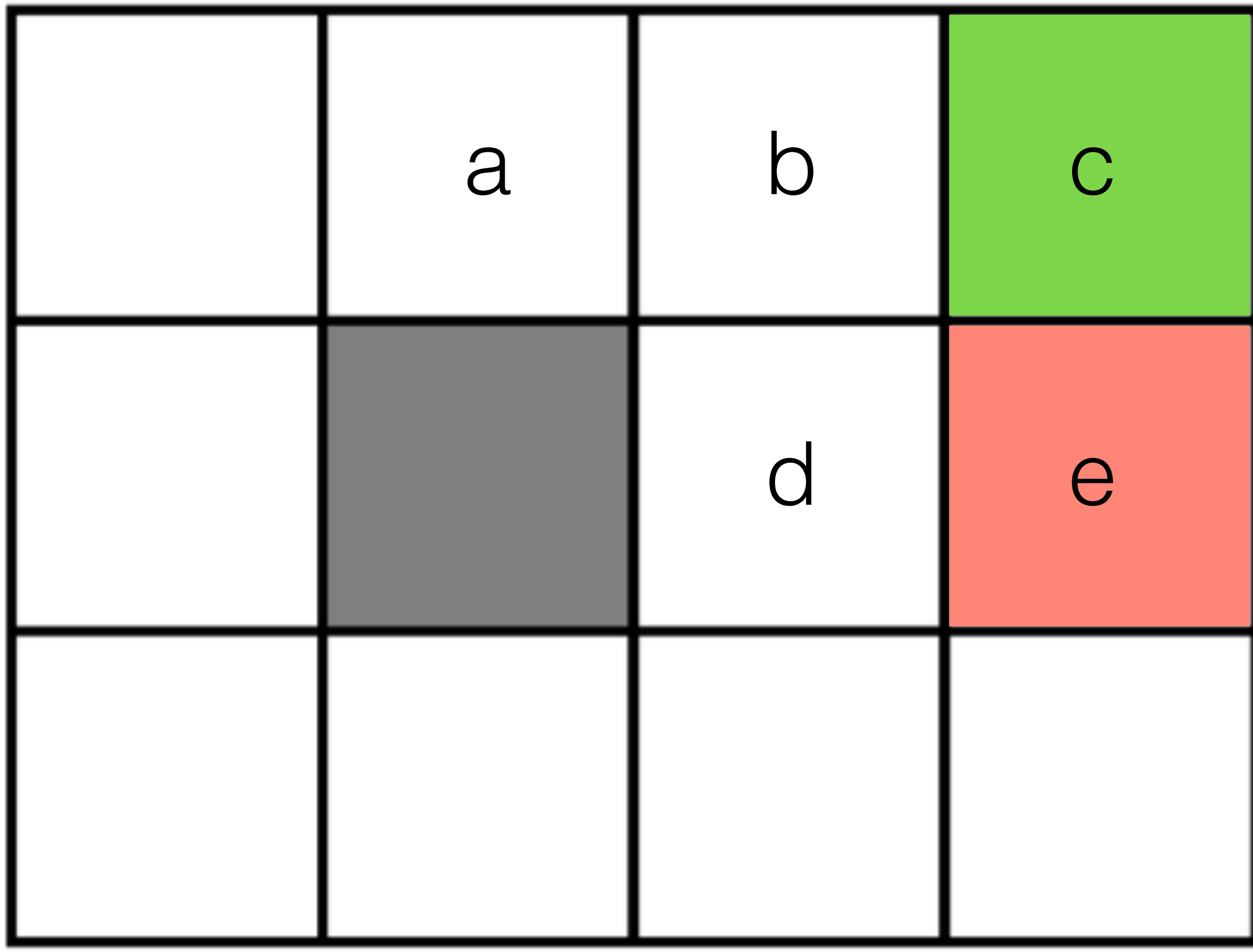
$Q(\mathrm{b}, \mathrm{D}) =$ -1

$Q(\mathrm{d}, \mathrm{R}) =$ -1

$Q(\mathrm{a}, \mathrm{R}) =$ -1

Q	R - right	D - down
a	-1	0
b	0	-1
c	0	0
d	-1	0
e	-10	0

(1) Substitute transitions and current Q-values to the right side and solve for left side.



τ_2 :

$(a, R, \text{-}1),\ (b, D, \text{-}1),$

$(d, R, \text{-}1),\ (e, R, \text{-}10)$

$$Q(\mathrm{e}, \mathrm{R}) = r(\mathrm{e})$$

$$Q(\mathrm{b}, \mathrm{D}) = r(\mathrm{b}) + \max_{\mathbf{u}} Q(\mathrm{d}, \mathbf{u})$$

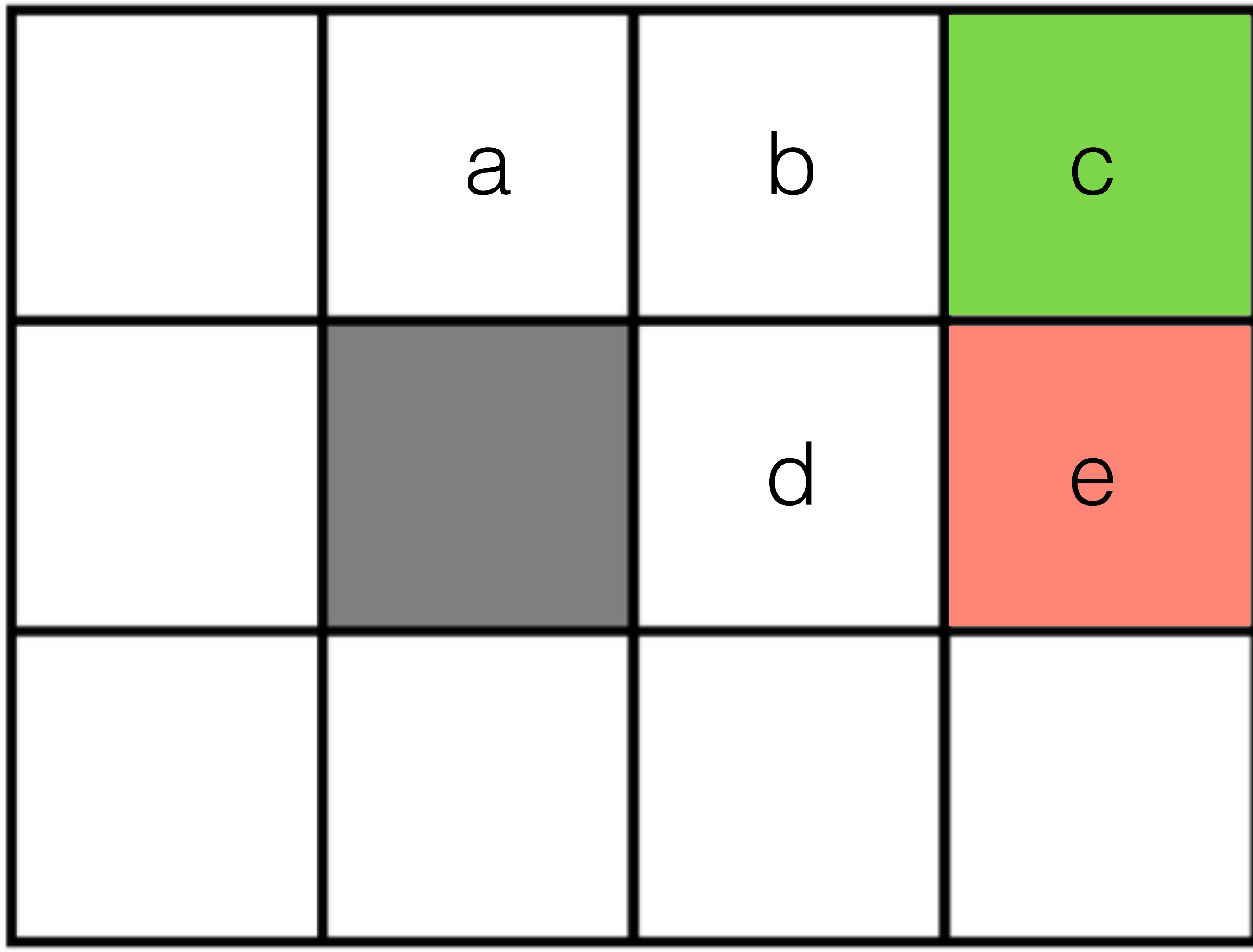
$$Q(\mathrm{d}, \mathrm{R}) = r(\mathrm{d}) + \max_{\mathbf{u}} Q(\mathrm{e}, \mathbf{u})$$

$$Q(\mathrm{a}, \mathrm{R}) = r(\mathrm{a}) + \max_{\mathbf{u}} Q(\mathrm{b}, \mathbf{u})$$

Q	R - right	D - down
a	-1	0
b	0	-1
c	0	0
d	-1	0
e	-10	0

(1) Substitute transitions and current Q-values to the right side and solve for left side.

(2) Repeat several times



$$\tau_2:$$

$$(a, R, \text{-}1),\ (b, D, \text{-}1),$$

$$(d, R, \text{-}1),\ (e, R, \text{-}10)$$

$$Q(\mathrm{e}, \mathrm{R}) = r(\mathrm{e})$$

$$Q(\mathrm{b}, \mathrm{R}) = r(\mathrm{b}) + \max_{\mathbf{u}} Q(\mathrm{d}, \mathbf{u})$$

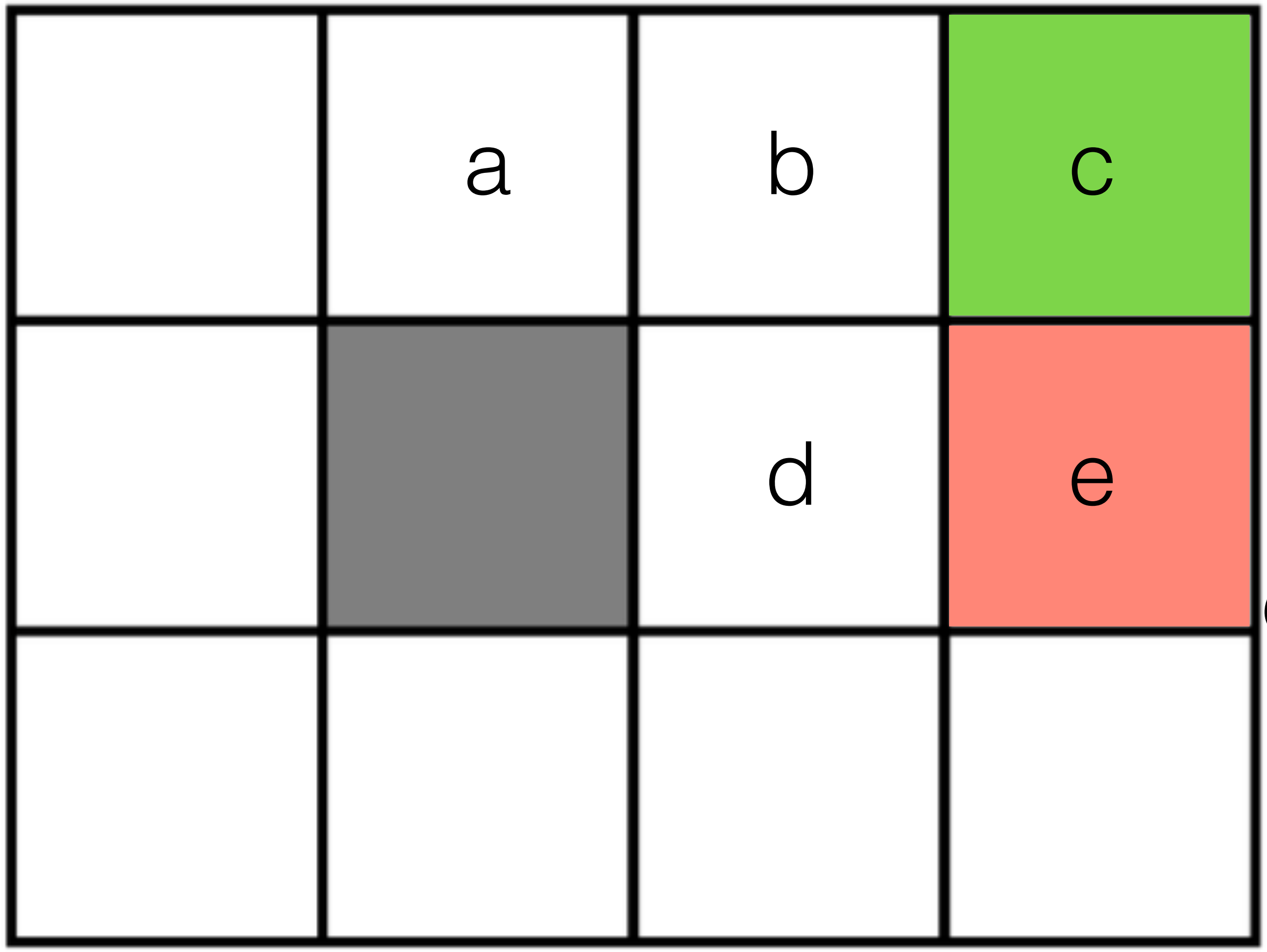
$$Q(\mathrm{d}, \mathrm{R}) = r(\mathrm{d}) + \max_{\mathbf{u}} Q(\mathrm{e}, \mathbf{u})$$

$$Q(\mathrm{a}, \mathrm{R}) = r(\mathrm{a}) + \max_{\mathbf{u}} Q(\mathrm{b}, \mathbf{u})$$

Q	R - right	D - down
a	-1	0
b	0	-1
c	0	0
d	-1	0
e	-10	0

(1) Substitute transitions and current Q-values to the right side and solve for left side.
(2) Repeat several times (search for the fixed point of the Bellman operator)

$$Q = \mathcal{B}(Q)$$



$$\tau_2:$$

$$(a, R, \text{-}1),\ (b, D, \text{-}1),$$
$$(d, R, \text{-}1),\ (e, R, \text{-}10)$$

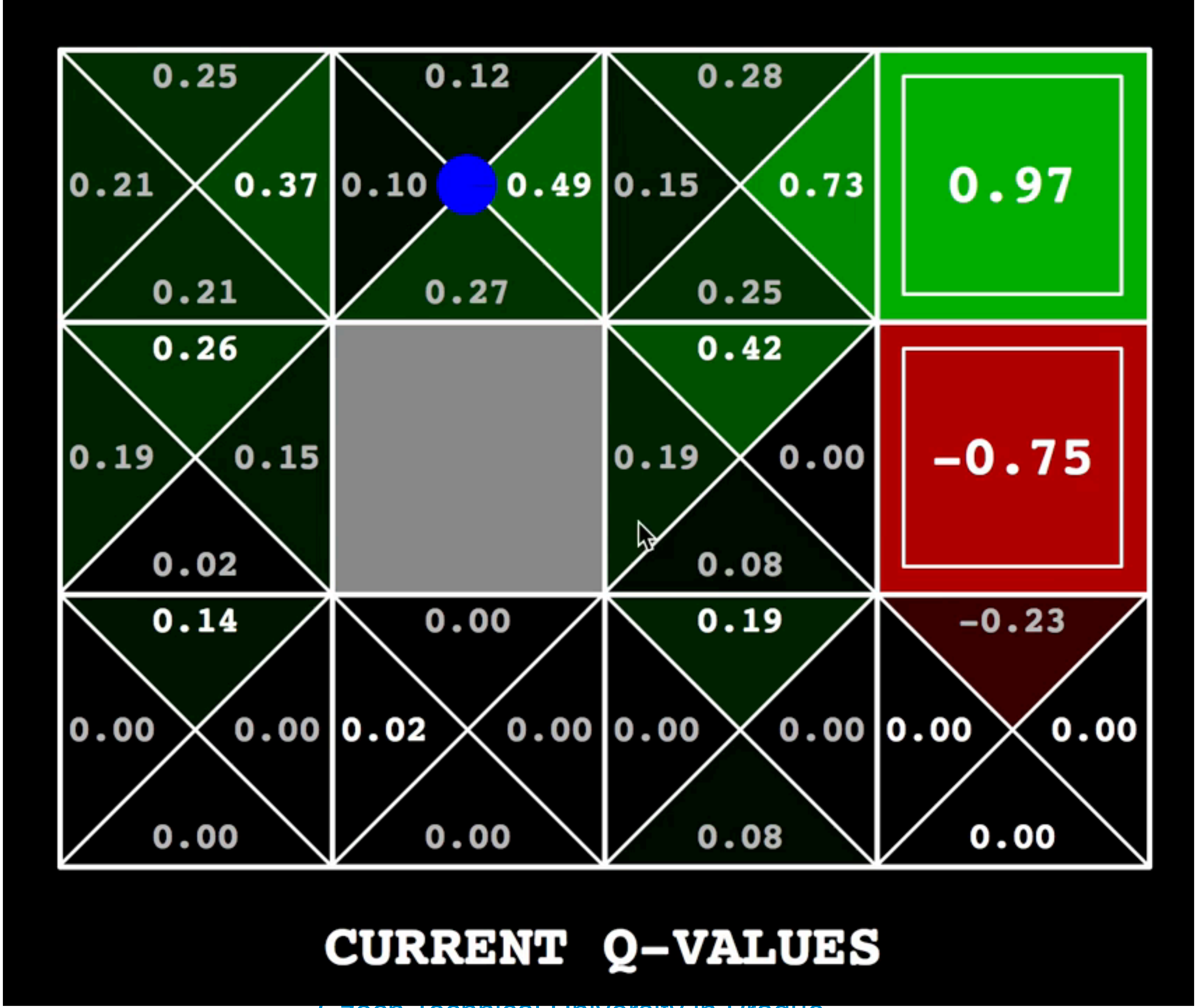
$$Q(\mathrm{e}, \mathrm{R}) = r(\mathrm{e})$$
$$Q(\mathrm{b}, \mathrm{R}) = r(\mathrm{b}) + \max_{\mathbf{u}} Q(\mathrm{d}, \mathbf{u})$$
$$Q(\mathrm{d}, \mathrm{R}) = r(\mathrm{d}) + \max_{\mathbf{u}} Q(\mathrm{e}, \mathbf{u})$$
$$Q(\mathrm{a}, \mathrm{R}) = r(\mathrm{a}) + \max_{\mathbf{u}} Q(\mathrm{b}, \mathbf{u})$$

(1) Substitute transitions and current Q-values to the right side and solve for left side.

(2) Repeat several times (search for the fixed point of the Bellman operator)

$$Q = \mathcal{B}(Q)$$

Iterations of the Bellman operator converge to a fixed point !!!


0.25
0.21
0.37
0.21
0.12
0.10
0.49
0.27
0.28
0.15
0.73
0.25
0.97
0.26
0.19
0.15
0.02
0.42
0.19
0.00
0.08
-0.75
0.14
0.00
0.00
0.00
0.00
0.02
0.00
0.00
0.19
0.00
0.00
0.08
-0.23
0.00
0.00
0.00
CURRENT Q-VALUES

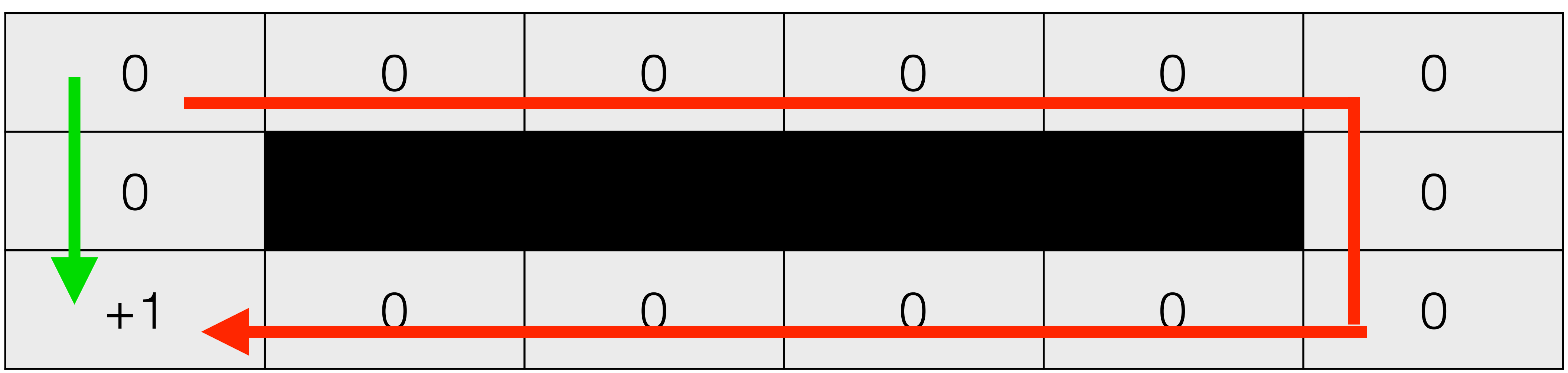
Bellman equation

$$Q(\mathbf{x}, \mathbf{u}) = r(\mathbf{x}, \mathbf{u}, \mathbf{x}') + \max_{\mathbf{u}'} Q(\mathbf{x}', \mathbf{u}')$$

reward for transition

the best you can do from the following state

Which path is better?



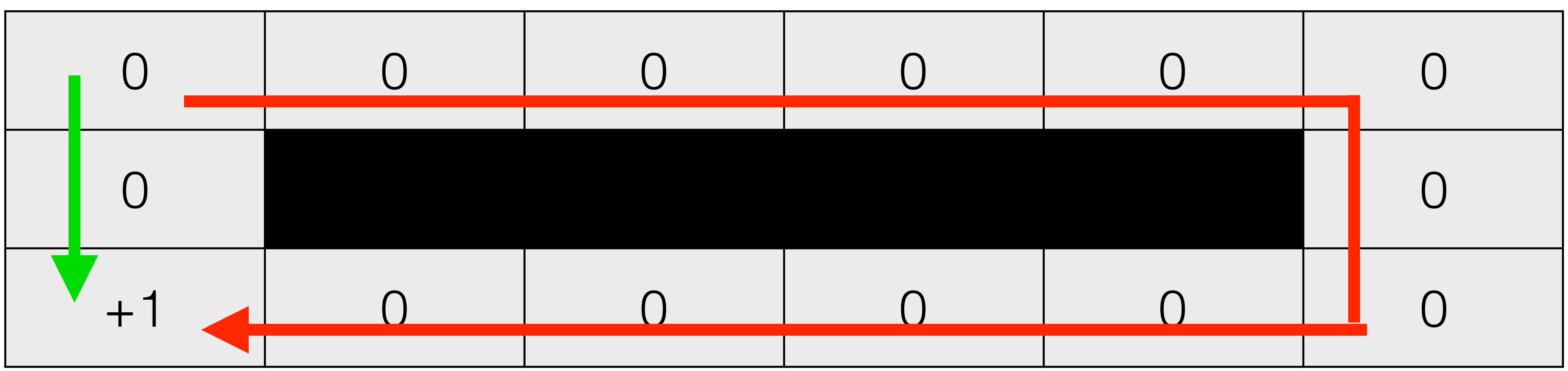
Bellman equation

$$Q(\mathbf{x}, \mathbf{u}) = r(\mathbf{x}, \mathbf{u}, \mathbf{x}') + \gamma \max_{\mathbf{u}'} Q(\mathbf{x}', \mathbf{u}')$$

reward for transition

the best you can do from the following state

discount factor $\gamma \in [0; 1]$



Q-learning

1. Collect trajectories $\tau_1, \tau_2, \tau_3, \dots$
2. Solve $Q(\mathbf{x}, \mathbf{u}) = r(\mathbf{x}, \mathbf{u}, \mathbf{x}') + \gamma \max_{\mathbf{u}'} Q(\mathbf{x}', \mathbf{u}')$
3. Repeat from 1

Q-learning

1. Collect trajectories $\tau_1, \tau_2, \tau_3, \ldots$
2. Solve $Q(\mathbf{x}, \mathbf{u}) = r(\mathbf{x}, \mathbf{u}, \mathbf{x}') + \gamma \max_{\mathbf{u}'} Q(\mathbf{x}', \mathbf{u}')$
3. Repeat from 1

- Curse of dimensionality

Q-learning

1. Collect trajectories $\tau_1, \tau_2, \tau_3, \ldots$
2. Solve $Q(\mathbf{x}, \mathbf{u}) = r(\mathbf{x}, \mathbf{u}, \mathbf{x}') + \gamma \max_{\mathbf{u}'} Q(\mathbf{x}', \mathbf{u}')$
3. Repeat from 1

- Curse of dimensionality
- Replace table $Q(\mathbf{x}, \mathbf{u})$ by function $Q_\theta(\mathbf{x}, \mathbf{u})$

Q-learning

1. Collect trajectories $\tau_1, \tau_2, \tau_3, \ldots$
2. Solve $Q(\mathbf{x}, \mathbf{u}) = r(\mathbf{x}, \mathbf{u}, \mathbf{x}') + \gamma \max_{\mathbf{u}'} Q(\mathbf{x}', \mathbf{u}')$
3. Repeat from 1

- Curse of dimensionality
- Replace table $Q(\mathbf{x}, \mathbf{u})$ by function $Q_\theta(\mathbf{x}, \mathbf{u})$

Approximate Q-learning

1. Collect trajectories $\tau_1, \tau_2, \tau_3, \ldots$, initialize $\theta = \mathrm{rand}$
2. Estimate $\mathbf{y} = r(\mathbf{x}, \mathbf{u}, \mathbf{x}') + \gamma \max_{\mathbf{u}'} Q_\theta(\mathbf{x}', \mathbf{u}')$
3. Update parameters by learning

$$\arg\min_\theta \sum_{\mathbf{x},\mathbf{u},\mathbf{y}} \|Q_\theta(\mathbf{x}, \mathbf{u}) - \mathbf{y}\|$$

4. Repeat from 2
5. Repeat from 1

Q-learning

1. Collect trajectories $\tau_1, \tau_2, \tau_3, \dots$
2. Solve $Q(\mathbf{x}, \mathbf{u}) = r(\mathbf{x}, \mathbf{u}, \mathbf{x}') + \gamma \max_{\mathbf{u}'} Q(\mathbf{x}', \mathbf{u}')$
3. Repeat from 1

- Curse of dimensionality
- Replace table $Q(\mathbf{x}, \mathbf{u})$ by function $Q_\theta(\mathbf{x}, \mathbf{u})$

Approximate Q-learning

1. Collect trajectories $\tau_1, \tau_2, \tau_3, \dots$, initialize $\theta = \mathrm{rand}$
2. Estimate $\mathbf{y} = r(\mathbf{x}, \mathbf{u}, \mathbf{x}') + \gamma \max_{\mathbf{u}'} Q_\theta(\mathbf{x}', \mathbf{u}')$
3. Update parameters by learning

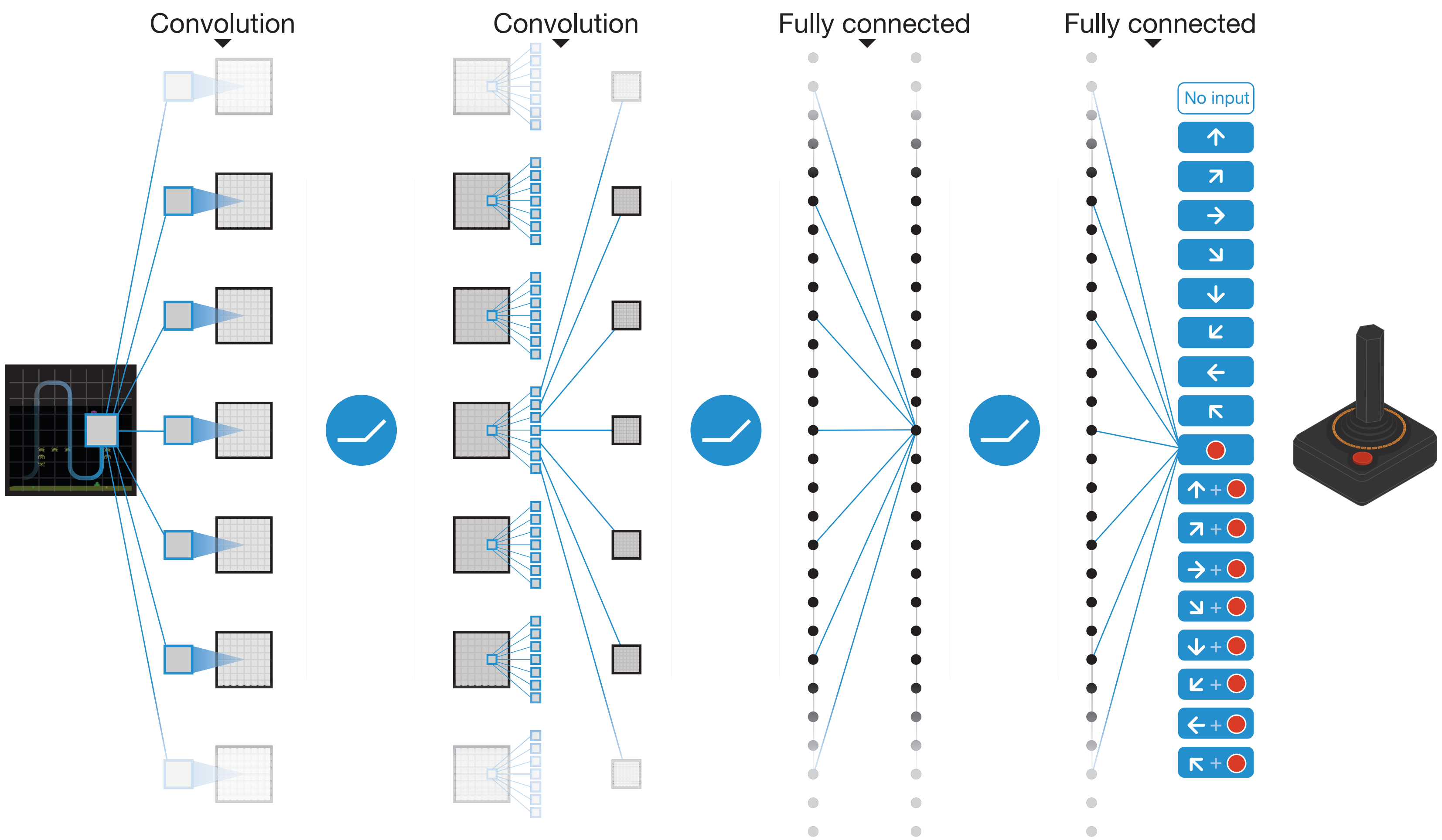
$$\arg\min_\theta \sum_{\mathbf{x},\mathbf{u},\mathbf{y}} \|Q_\theta(\mathbf{x}, \mathbf{u}) - \mathbf{y}\|$$

4. Repeat from 2
5. Repeat from 1

Approximated Q-learning does not have to converge to a fixed-point !!!

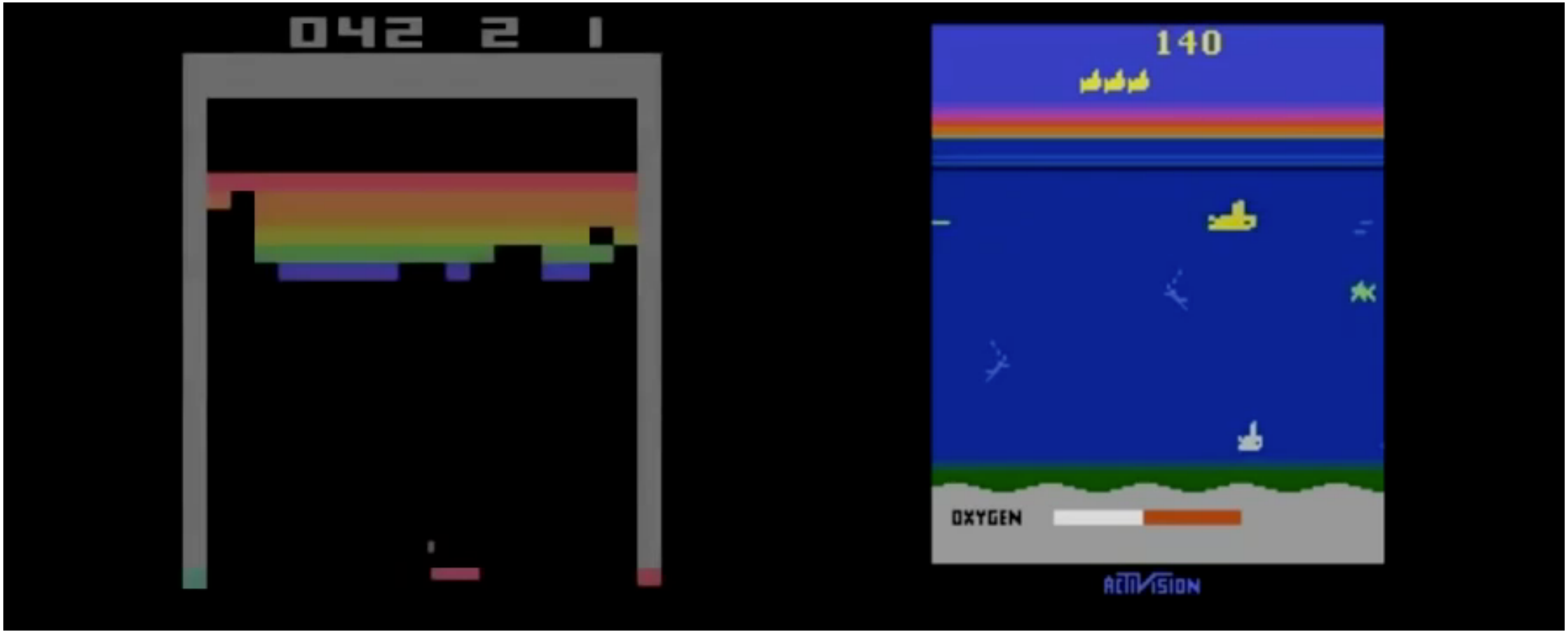
Mnih et al. Nature 2015

- 2600 atari games
- **state space:** pixels (e.g. VGA resolution)
- **action space:** discrete joystic actions (8 direction + 8 direction with button + neutral action)
- replay buffer (decorrelates samples to be “more i.i.d”)
- two Q-networks (suppress oscilations)

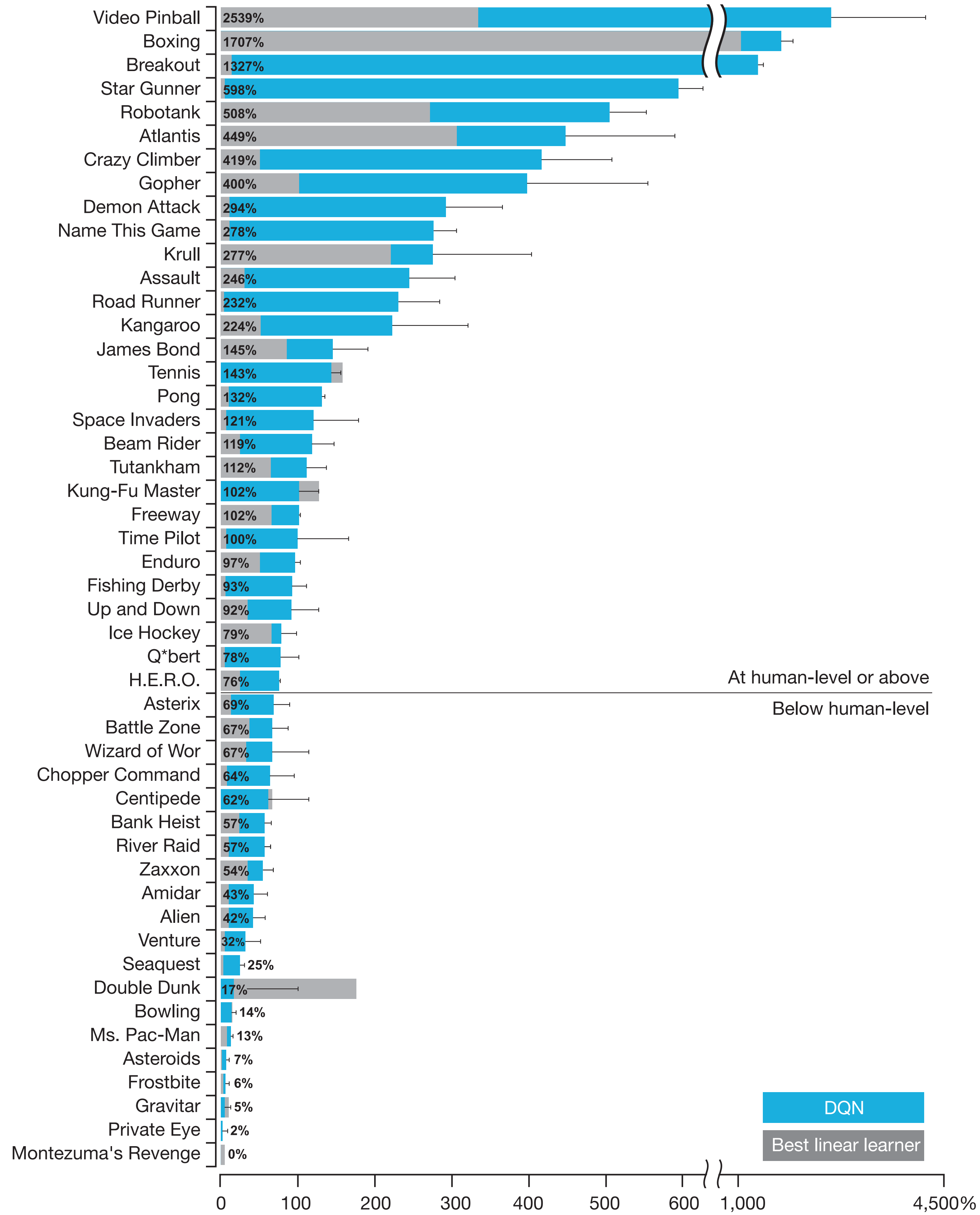


Mnih et al. Nature 2015

- 2600 atari games
- **state space:** pixels (e.g. VGA resolution)
- **action space:** discrete joystic actions (8 directions + 8 directions with button)
- collection of control tasks: https://gym.openai.com

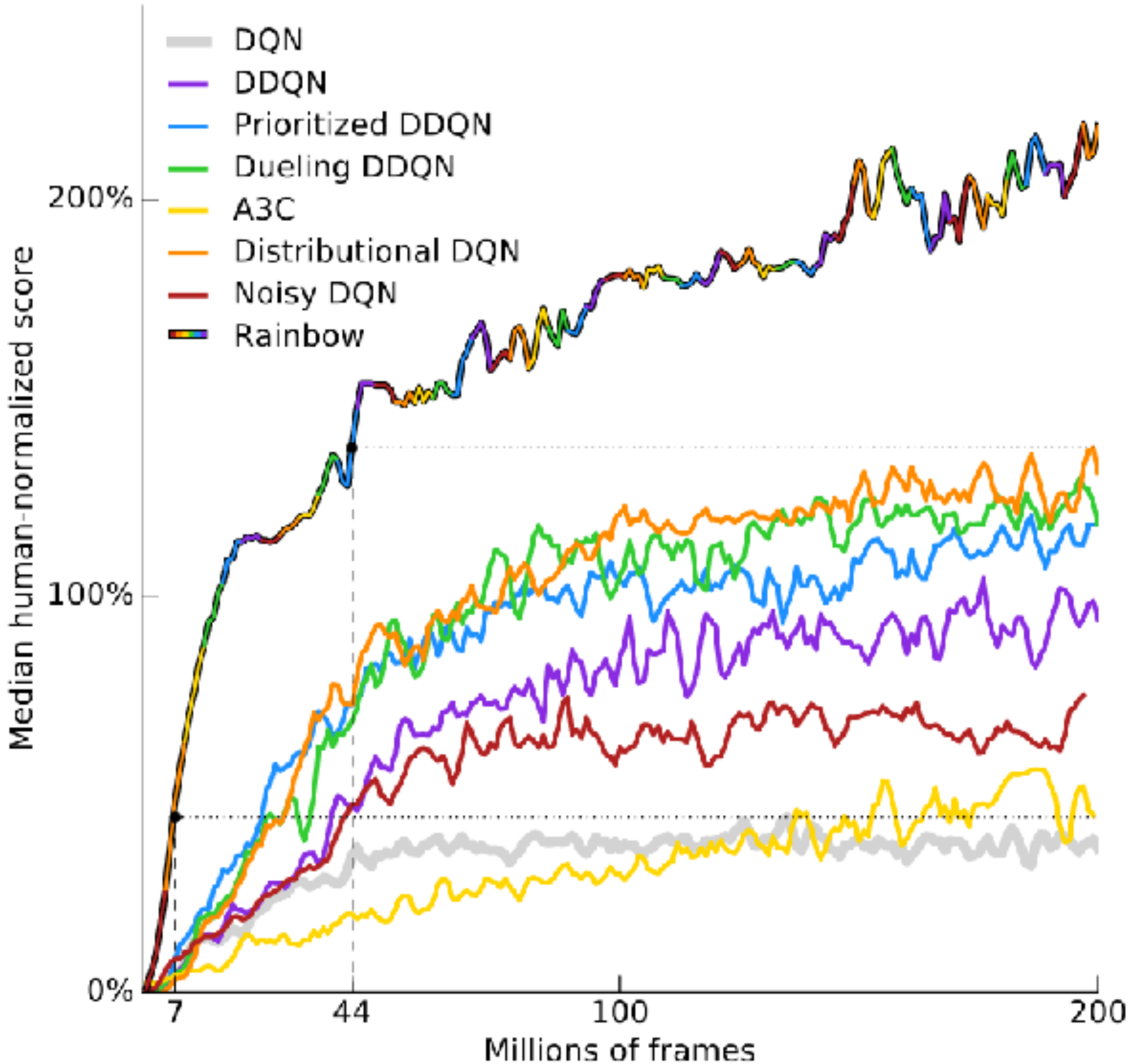


Mnih et al. Nature 2015



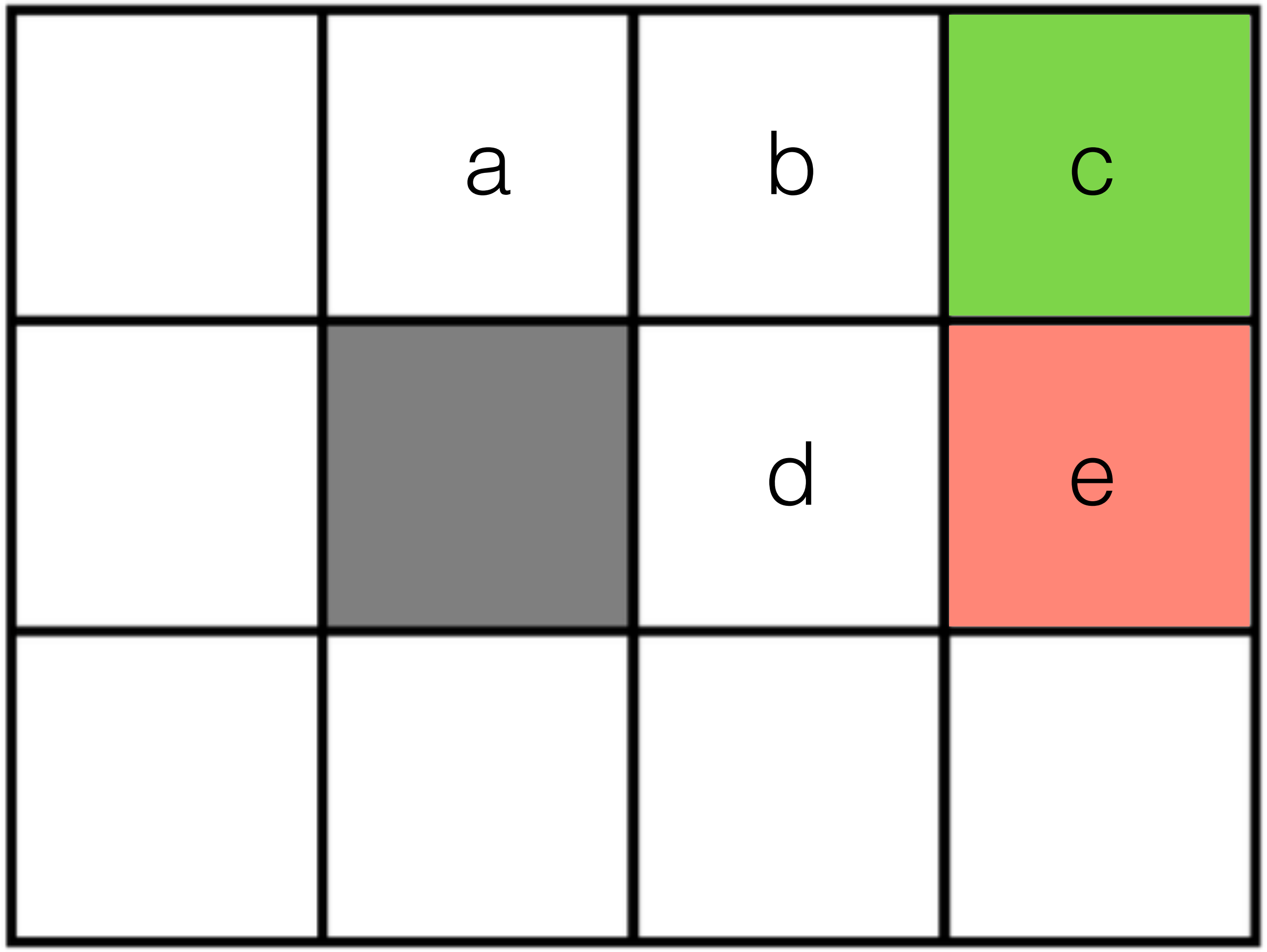
Hessel et. al Rainbow DQN, 2017

Average of different estimates helps a lot



Value function

- If model available it is often better to train the state-value function.



τ_1 :

$(a, R, \text{-}1),\ (b, R, \text{-}1),\ (c, R, 10)$

τ_2 :

$(a, R, \text{-}1),\ (b, D, \text{-}1),$

$(d, R, \text{-}1),\ (e, R, \text{-}10)$

Return of a trajectory starting from the state x:

$$G = r_1 + \gamma r_2 + \gamma^2 r_3 + \ldots$$

$$\tau_1 : \quad G_1 = (-1) + \gamma(-1) + \gamma^2 10$$

$$\tau_2 : \quad G_2 = (-1) + \gamma(-1) + \gamma^2(-1) + \gamma^3(-10)$$

$$V(\mathbf{x}) = \mathbb{E}_{G \sim \pi}[G] \approx \frac{1}{N} \sum_i G_i$$

$$V(\mathbf{x}) = \mathbb{E}_{G\sim\pi}[G] \approx \frac{1}{N}\sum_i G_i$$

Such estimate has high variance =>
re-use older estimates of $V(\mathbf{x})$ and estimate exponentially weighting average

$$V(\mathbf{x}) = \mathbb{E}_{G\sim\pi}[G] \approx \frac{1}{N}\sum_i G_i$$

Such estimate has high variance =>
re-use older estimates of $V(\mathbf{x})$ and estimate exponentially weighting average

$$V(\mathbf{x}) \approx (1-\alpha)V(\mathbf{x}) + \alpha G_i = V(\mathbf{x}) + \alpha(G_i - V(\mathbf{x}))$$

$$V(\mathbf{x}) = \mathbb{E}_{G\sim\pi}[G] \approx \frac{1}{N}\sum_i G_i$$

Such estimate has high variance =>
re-use older estimates of $V(\mathbf{x})$ and estimate exponentially weighting average

$$V(\mathbf{x}) \approx (1-\alpha)V(\mathbf{x}) + \alpha G_i = V(\mathbf{x}) + \alpha(G_i - V(\mathbf{x}))$$

Such estimate has smaller variance but is still bad =>

$$\begin{aligned}
G^{(\infty)} &= r_1 + \gamma r_2 + \gamma^2 r_3 + \dots \\
&\vdots \\
G^{(2)} &= r_1 + \gamma r_2 + \gamma^2 V(\mathbf{x}_3) \\
G^{(1)} &= r_1 + \gamma V(\mathbf{x}_2)
\end{aligned}$$

$$V(\mathbf{x}) = \mathbb{E}_{G\sim\pi}[G] \approx \frac{1}{N}\sum_i G_i$$

Such estimate has high variance =>
re-use older estimates of $V(\mathbf{x})$ and estimate exponentially weighting average

$$V(\mathbf{x}) \approx (1-\alpha)V(\mathbf{x}) + \alpha G_i = V(\mathbf{x}) + \alpha(G_i - V(\mathbf{x}))$$

Such estimate has smaller variance but is still bad =>

$$G^{(\infty)} = r_1 + \gamma r_2 + \gamma^2 r_3 + \dots$$

$$\vdots$$

$$G^{(2)} = r_1 + \gamma r_2 + \gamma^2 V(\mathbf{x}_3)$$

$$G^{(1)} = r_1 + \gamma V(\mathbf{x}_2)$$

MC estimate:
high variance, no bias

TD estimate:
small variance,
strong bias

$$G^{(\infty)} = r_1 + \gamma r_2 + \gamma^2 r_3 + \dots$$

$$\vdots$$

$$G^{(2)} = r_1 + \gamma r_2 + \gamma^2 V(\mathbf{x}_3)$$

$$G^{(1)} = r_1 + \gamma V(\mathbf{x}_2)$$

Convex combination of all possible return estimates

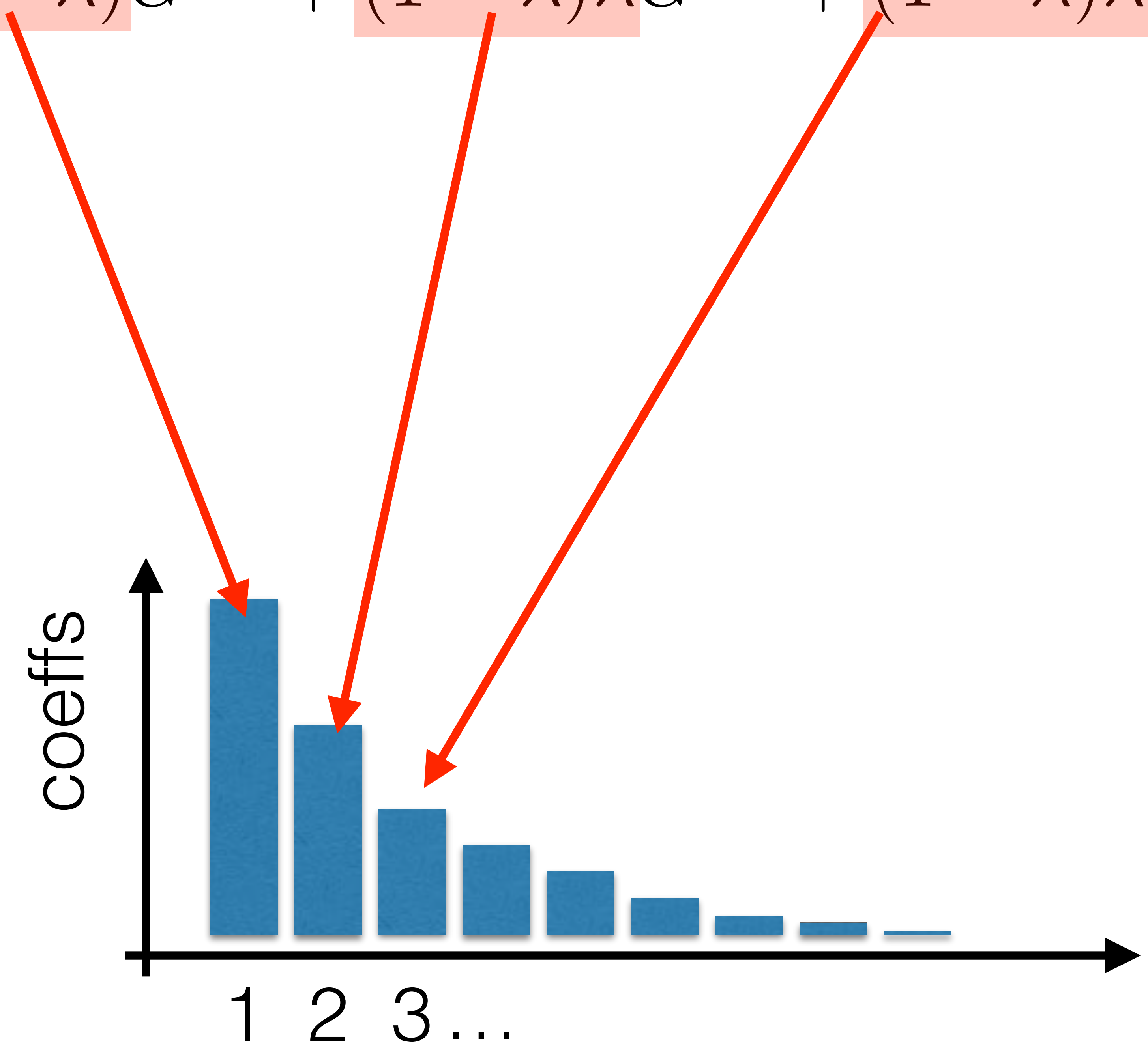
$$G^{\lambda} = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} G^{(n)} =$$

$$= (1-\lambda)G^{(1)} + (1-\lambda)\lambda G^{(2)} + (1-\lambda)\lambda^2 G^{(3)} + \dots$$

coeffs sums to 1

$$G^{\lambda} = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} G^{(n)} =$$

$$= (1-\lambda)G^{(1)} + (1-\lambda)\lambda G^{(2)} + (1-\lambda)\lambda^2 G^{(3)} + \ldots$$

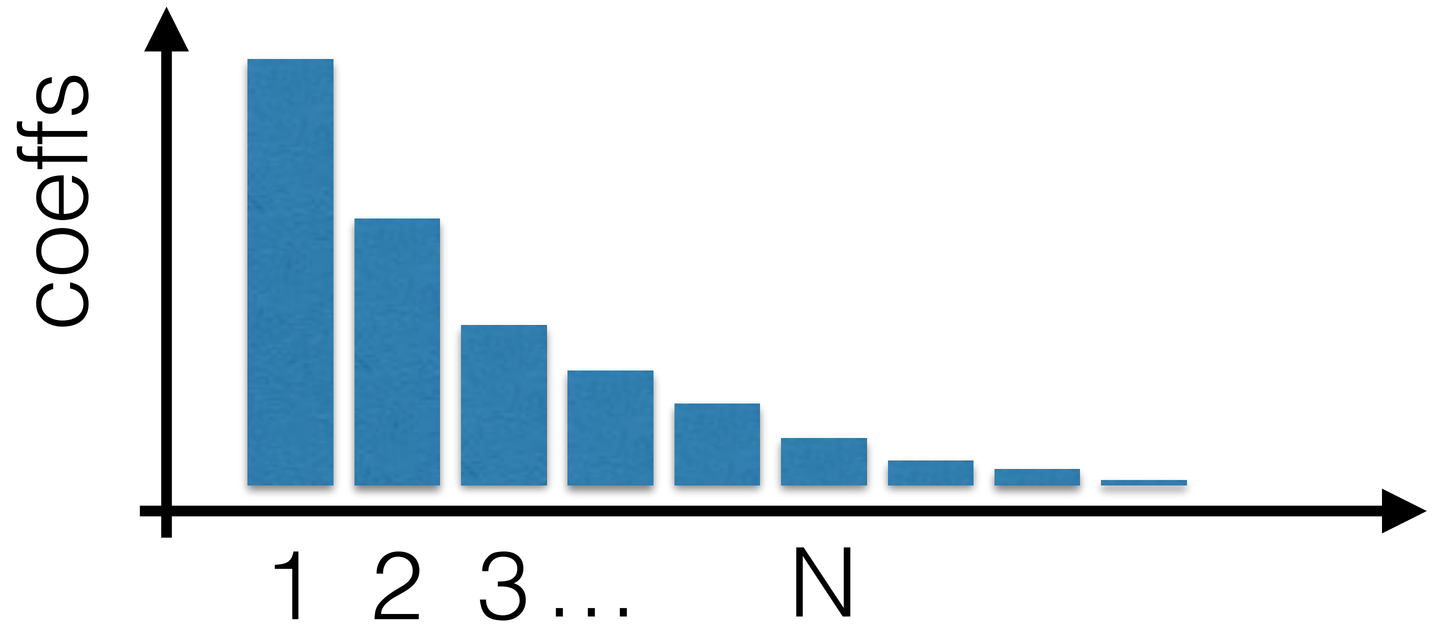


$$G^{\lambda} = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}G^{(n)} =$$

$$= (1-\lambda)G^{(1)} + (1-\lambda)\lambda G^{(2)} + (1-\lambda)\lambda^{2}G^{(3)} + \ldots$$

$$\ldots (1-\lambda)\lambda^{(N-2)}G^{(N-1)}$$

- In reality, sequences have finite length

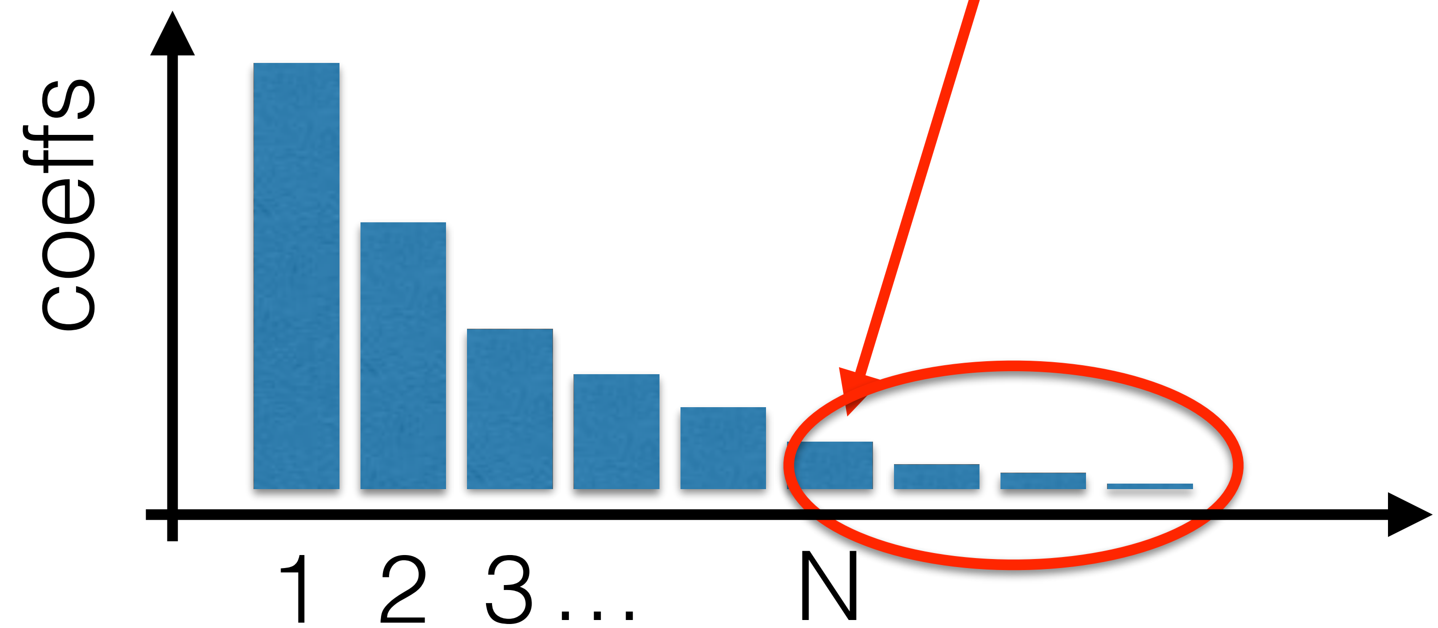


$$G^{\lambda} = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} G^{(n)} =$$

$$= (1-\lambda)G^{(1)} + (1-\lambda)\lambda G^{(2)} + (1-\lambda)\lambda^2 G^{(3)} + \dots$$

$$\dots (1-\lambda)\lambda^{(N-2)} G^{(N-1)} + \lambda^{(N-1)} G^{(N)}$$

- In reality, sequences have finite length
- Last coeff sums up all coeffs from N to infinity.



$$G^{\lambda} = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}G^{(n)} =$$

$$= (1-\lambda)G^{(1)} + (1-\lambda)\lambda G^{(2)} + (1-\lambda)\lambda^{2}G^{(3)} + \dots$$

$$\dots (1-\lambda)\lambda^{(N-2)}G^{(N-1)} + \lambda^{(N-1)}G^{(N)}$$

$TD(\lambda)$ learning algorithm

1. collect trajectories
2. for each state $\mathbf{x}$ estimate G^{λ}
3. Update state-value function:
   $V(\mathbf{x}) = V(\mathbf{x}) + \alpha(G^{\lambda} - V(\mathbf{x}))$
4. repeat from 1

State value function $V(\mathbf{x}_1)$ is approximated from traj. which

- started in $\mathbf{x}_1$

$$G^{(\infty)} = r_1 + \gamma r_2 + \gamma^2 r_3 + \dots$$

$$\vdots$$

$$G^{(2)} = r_1 + \gamma r_2 + \gamma^2 V(\mathbf{x}_3)$$

$$G^{(1)} = r_1 + \gamma V(\mathbf{x}_2)$$

Similarly state-action function $Q(\mathbf{x}_1, \mathbf{u}_1)$ can be approximated but only from trajectories which

- started in $\mathbf{x}_1$
- followed action $\mathbf{u}_1$

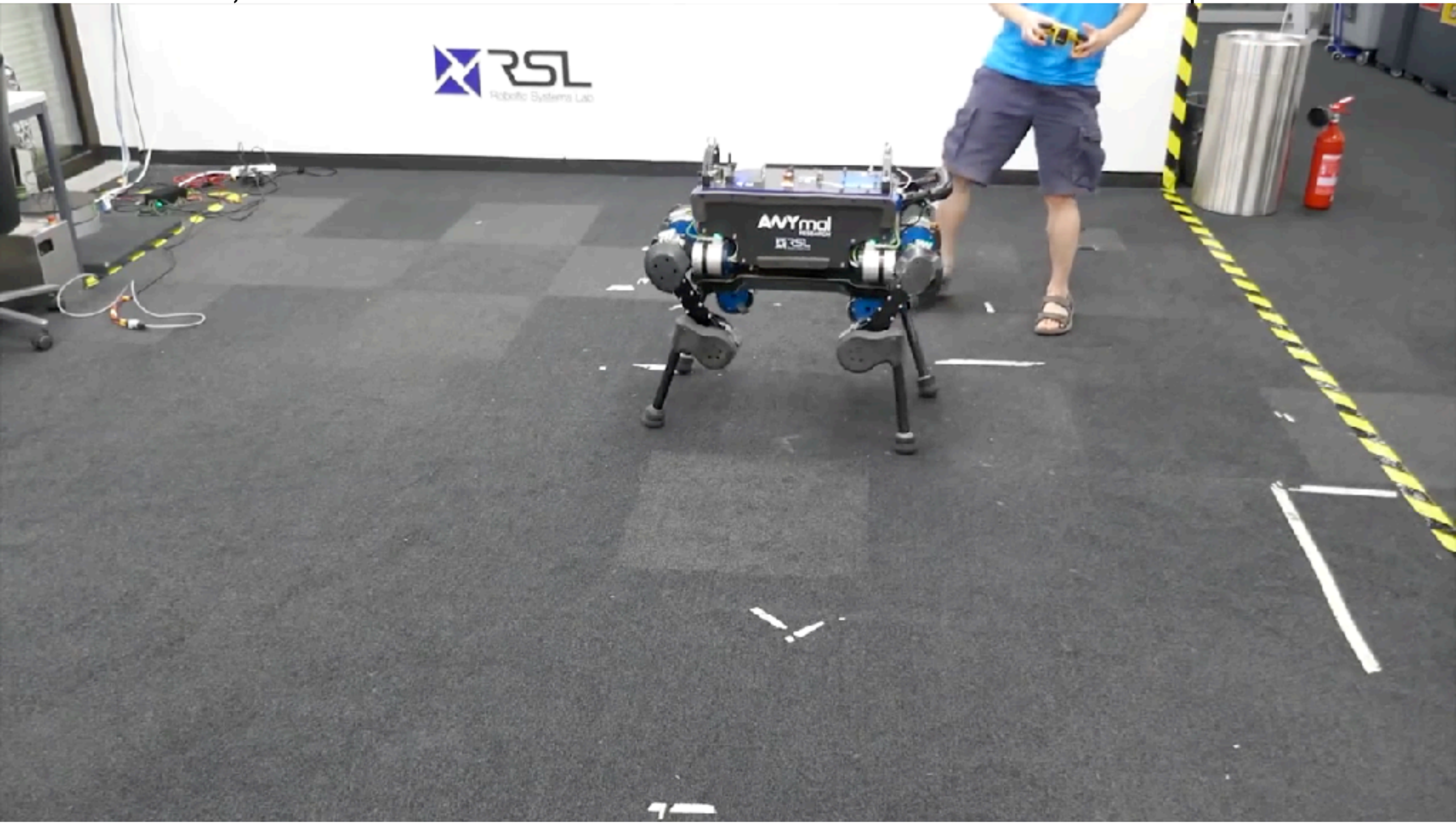
$$\hat{Q}^{(\infty)} = r_1 + \gamma r_2 + \gamma^2 r_3 + \dots$$

$$\vdots$$

$$\hat{Q}^{(1)} = r_1 + \gamma V(\mathbf{x}_2)$$

- Learning has been shown to be possible in simulation
- but can I use it on a real robot??
- !!! millions (or billions) of real-world trials are needed

If exteroceptive sensors are not used and terrain is trivial, then transfer from accurate simulation is possible

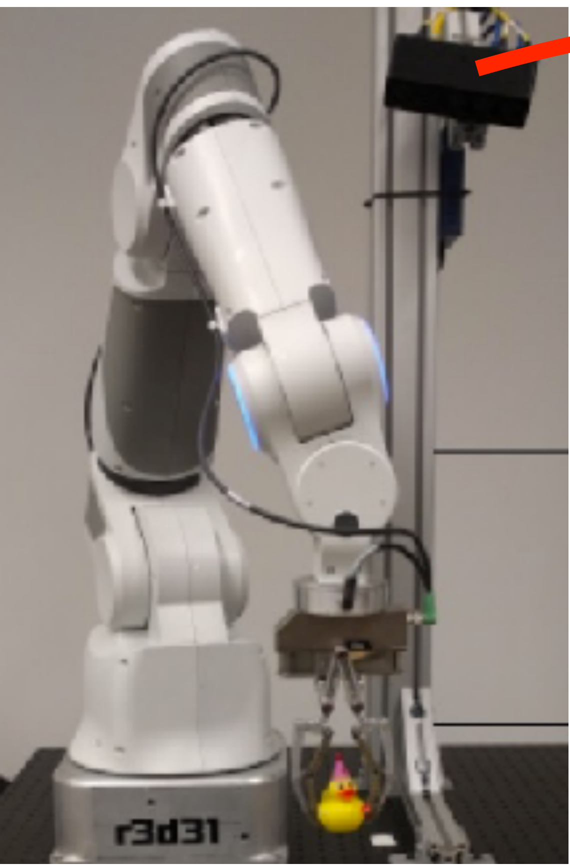


[Hwangbo, ETH Zurich, Science Robotics, 2018]

[Levine IJRR 2017] https://arxiv.org/abs/1603.02199

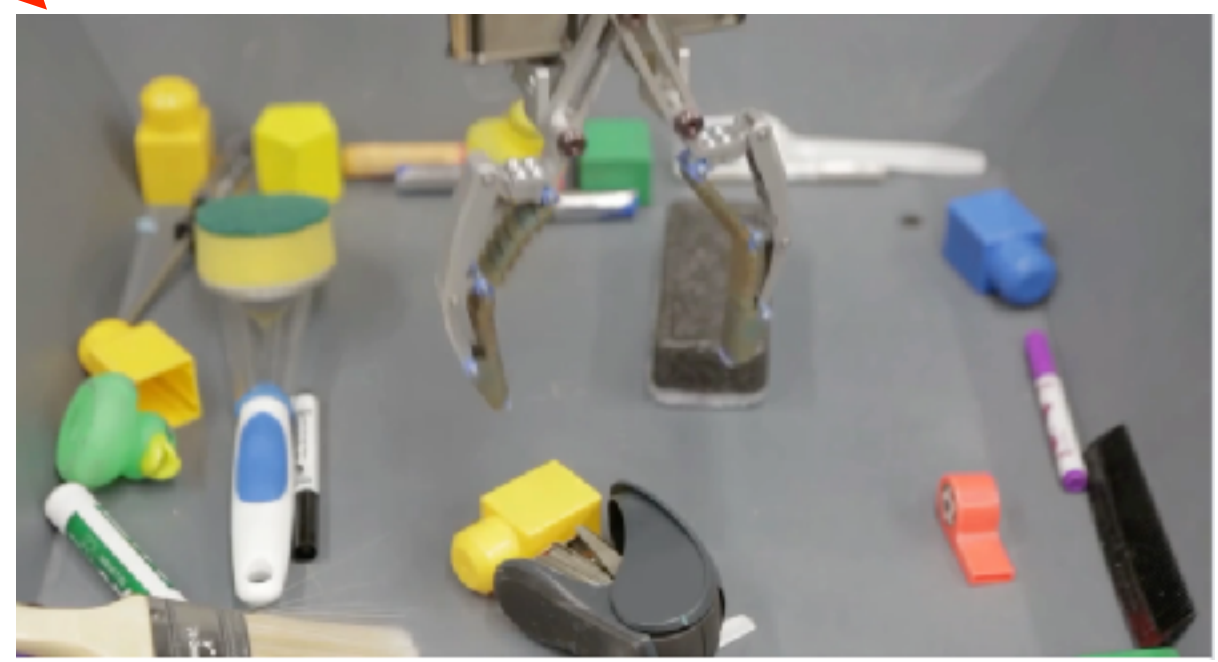
Another option is to avoid simulation completely !!!

manipulator+ RGB camera



image

$$= \pi_\theta \left(\right.$$



$$\left. \right)$$

joint torques

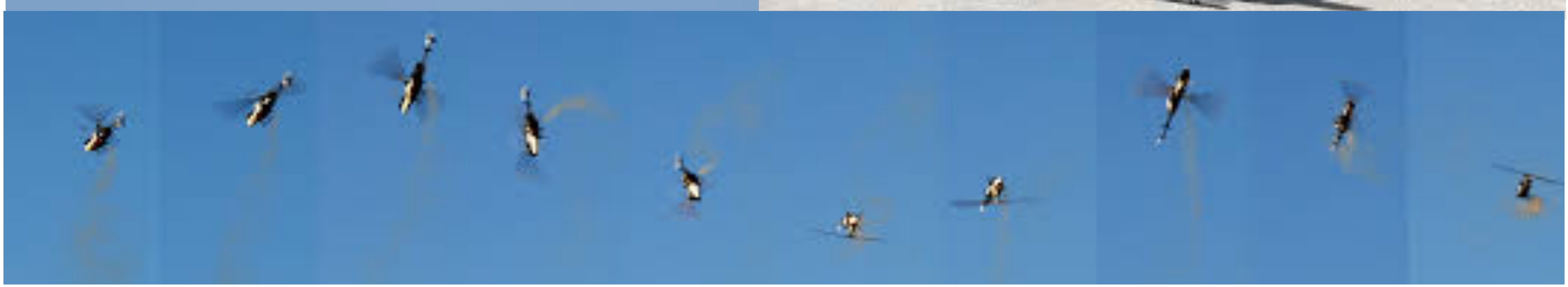
Continues motion control from RGB(D)

[Levine IJRR 2017] https://arxiv.org/abs/1603.02199

Learning from expert demonstrations

- Sometimes easier to provide good trajectories than good rewards.





Learning from expert demonstrations

- Sometimes easier to provide good trajectories than good rewards.
- Imitation learning setup

Learning from expert demonstrations

- Sometimes easier to provide good trajectories than good rewards.

- Imitation learning setup
  1. Collect expert trajectories $\tau_1^*, \tau_2^*, \tau_3^*, \ldots$
  2. Find policy $\arg\min_\theta \sum_{(\mathbf{x}_i, \mathbf{a}_i) \in \tau^*} \|\pi_\theta(\mathbf{x}_i) - \mathbf{a}_i\|_2^2$

Learning from expert demonstrations

- Sometimes easier to provide good trajectories than good rewards.

- Imitation learning setup (statistically inconsistent+ blackbox)
  1. Collect expert trajectories $\tau_1^*, \tau_2^*, \tau_3^*, \ldots$
  2. Find policy $\arg\min_\theta \sum_{(\mathbf{x}_i, \mathbf{a}_i) \in \tau^*} \|\pi_\theta(\mathbf{x}_i) - \mathbf{a}_i\|_2^2$

- Inverse reinforcement learning setup

Learning from expert demonstrations

- Sometimes easier to provide good trajectories than good rewards.

- Imitation learning setup (statistically inconsistent+ blackbox)
  1. Collect expert trajectories $\tau_1^*, \tau_2^*, \tau_3^*, \ldots$
  2. Find policy $\arg\min_\theta \sum_{(\mathbf{x}_i, \mathbf{a}_i) \in \tau^*} \|\pi_\theta(\mathbf{x}_i) - \mathbf{a}_i\|_2^2$

- Inverse reinforcement learning setup
  1. Collect expert trajectories $\tau_1^*, \tau_2^*, \tau_3^*, \ldots$
  2. Find reward function $r_{\mathbf{w}}$

$$\arg\min_{\mathbf{w}} \|\mathbf{w}\|_2^2$$

$$\text{subject to:} \quad \sum_{(\mathbf{x},\mathbf{u},\mathbf{x}') \in \tau^*} r_{\mathbf{w}}(\mathbf{x}, \mathbf{u}, \mathbf{x}') \leq \sum_{(\mathbf{x},\mathbf{u},\mathbf{x}') \in \{\mathcal{T} \setminus \tau^*\}} r_{\mathbf{w}}(\mathbf{x}, \mathbf{u}, \mathbf{x}')$$

Learning from expert demonstrations

- Sometimes easier to provide good trajectories than good rewards.
- Imitation learning setup (statistically inconsistent+ blackbox)
  1. Collect expert trajectories $\tau_1^*, \tau_2^*, \tau_3^*, \ldots$
  2. Find policy $\arg\min_\theta \sum_{(\mathbf{x}_i, \mathbf{a}_i) \in \tau^*} \|\pi_\theta(\mathbf{x}_i) - \mathbf{a}_i\|_2^2$
- Inverse reinforcement learning setup
  1. Collect expert trajectories $\tau_1^*, \tau_2^*, \tau_3^*, \ldots$
  2. Find reward function $r_\mathbf{w}$

$$\arg\min_\mathbf{w} \|\mathbf{w}\|_2^2$$

$$\text{subject to:} \quad \sum_{(\mathbf{x},\mathbf{u},\mathbf{x}') \in \tau^*} r_\mathbf{w}(\mathbf{x}, \mathbf{u}, \mathbf{x}') \leq \sum_{(\mathbf{x},\mathbf{u},\mathbf{x}') \in \{\mathcal{T} \setminus \tau^*\}} r_\mathbf{w}(\mathbf{x}, \mathbf{u}, \mathbf{x}')$$

  3. Solve underlying RL task

Abbeel et al. IJRR 2010

- inverse reinforcement learning
- **state space:** angular and euclidean position, velocity, acceleration
- **action space:** motor torques
- learning reward function from expert pilot

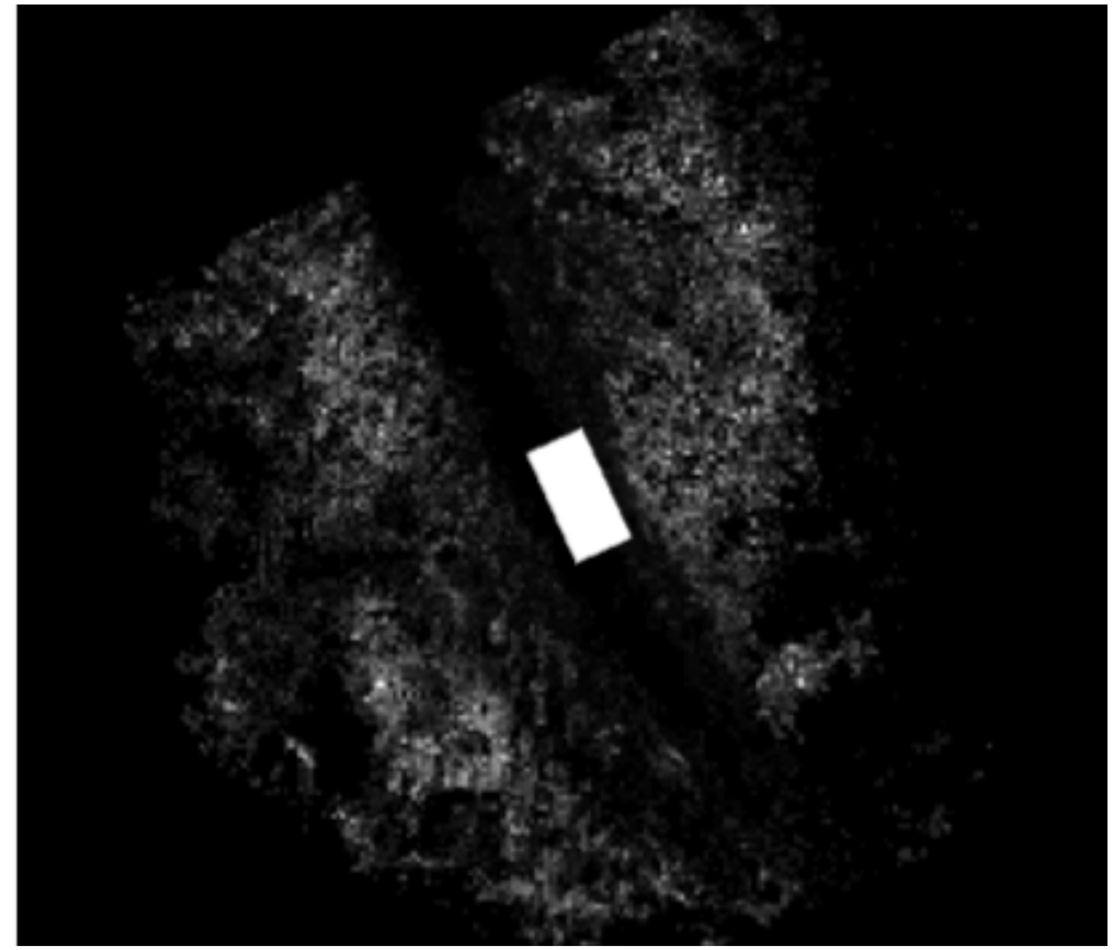
Abbeel et al. IJRR 2010

Silver et al. IJRR 2010

http://www.dtic.mil/dtic/tr/fulltext/u2/a525288.pdf

Silver et al. IJRR 2010

input image (state)

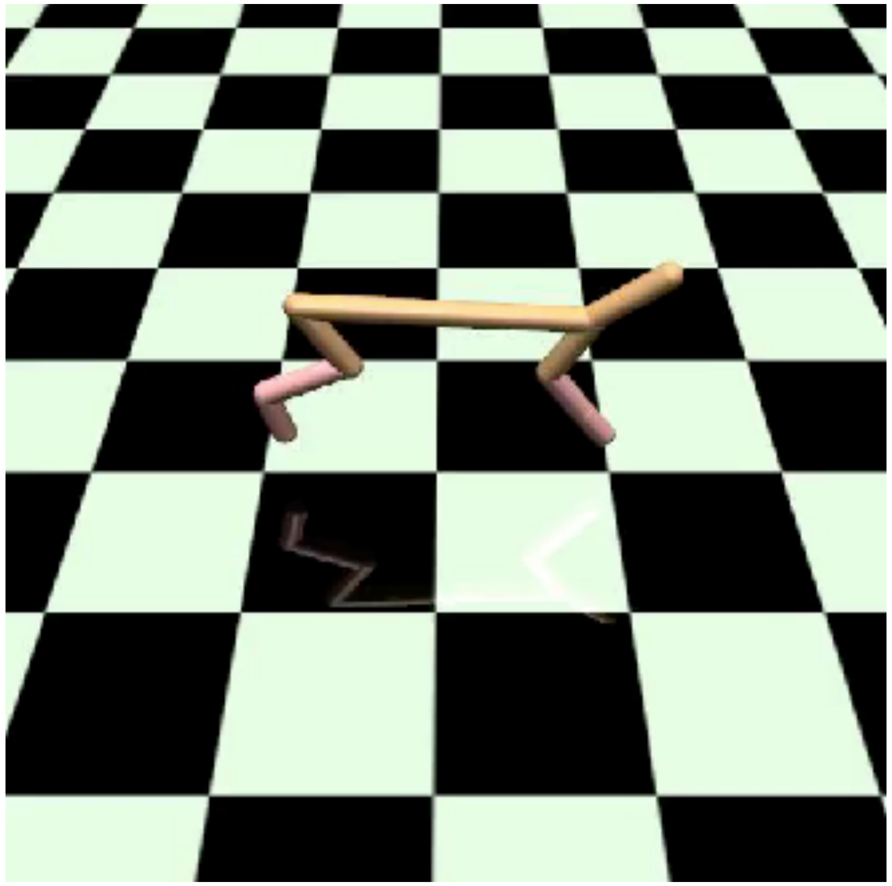


learned reward function
(traversability map)

http://www.dtic.mil/dtic/tr/fulltext/u2/a525288.pdf

Reward shaping

- Sparse rewards are easier to design correctly
- Dense rewards are easier to learn
- Half cheetah:
  - sparse rewards (for reaching the goal position fast)
  - dense rewards (for velocity)



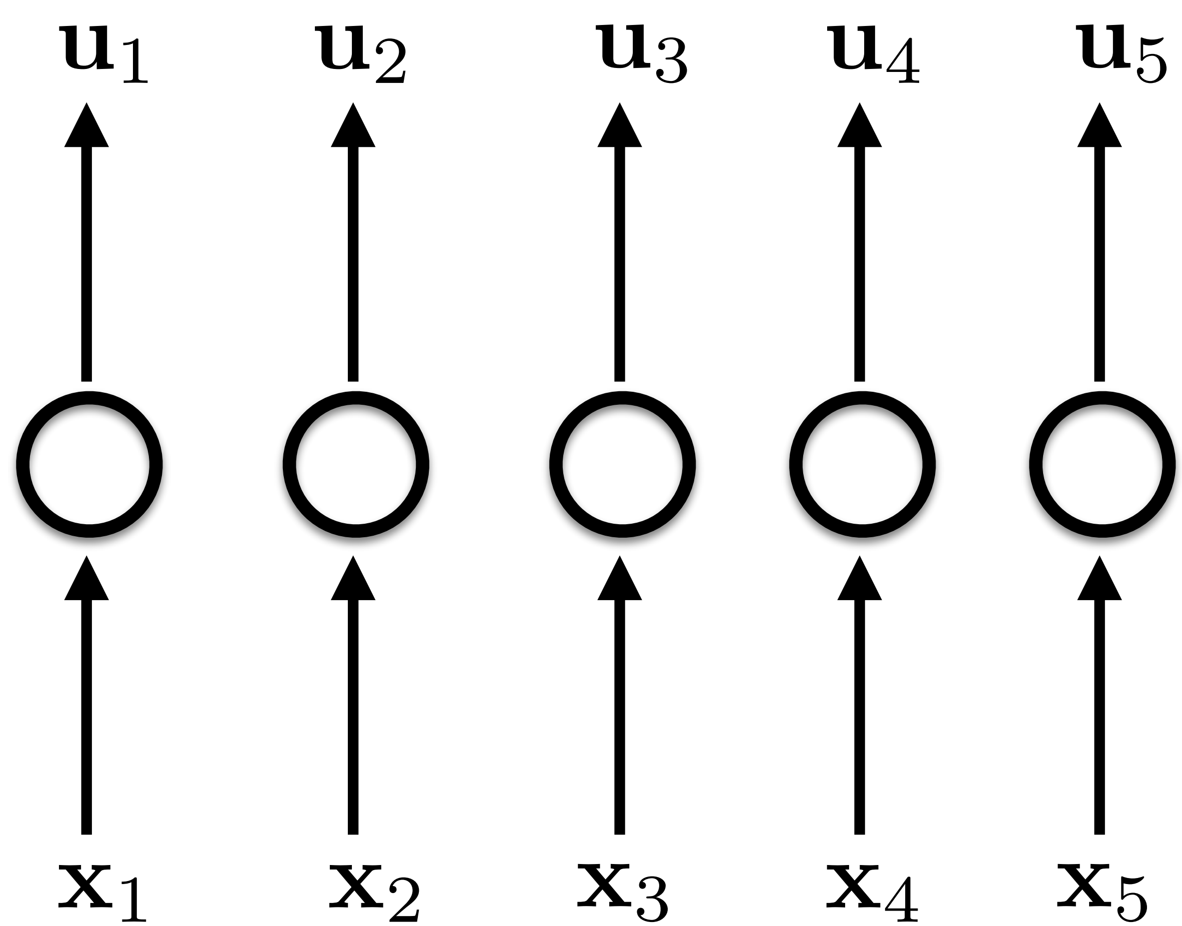
Czech Technical University in Prague
Faculty of Electrical Engineering, Department of Cybernetics

Reward shaping

- Sparse rewards are easier to design correctly
- Dense rewards are easier to learn

Reward shaping

- Sparse rewards are easier to design correctly
- Dense rewards are easier to learn



Sparse rewards	0	0	0	0	10
Dense rewards	2	2	2	2	2

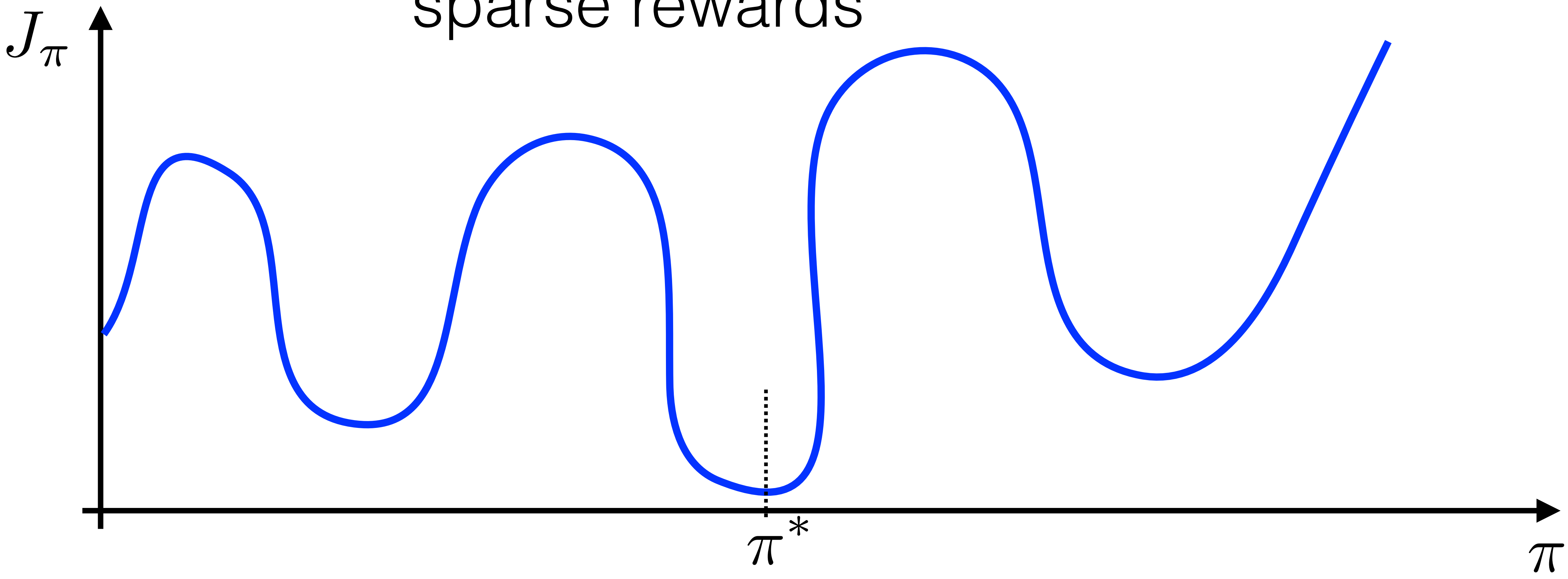
Reward shaping

- Sparse rewards are easier to design correctly
- Dense rewards are easier to learn

Reward shaping

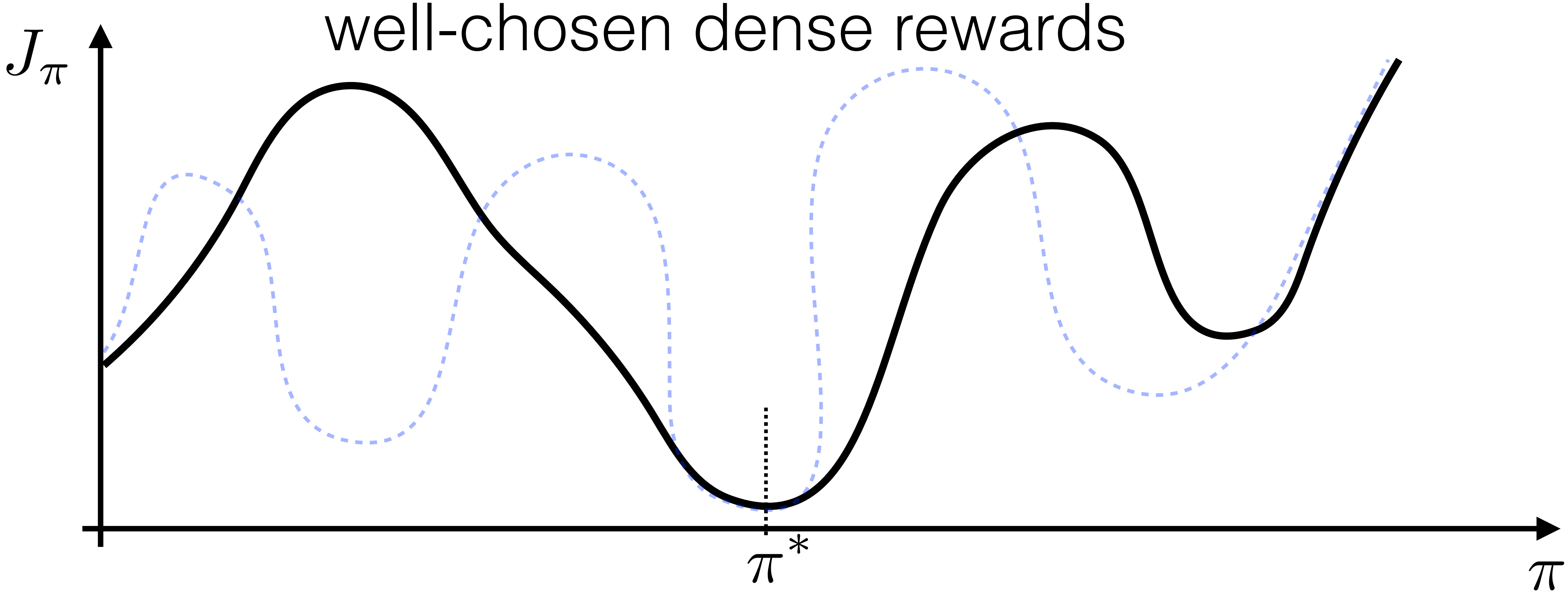
- Sparse rewards are easier to design correctly
- Dense rewards are easier to learn

sparse rewards



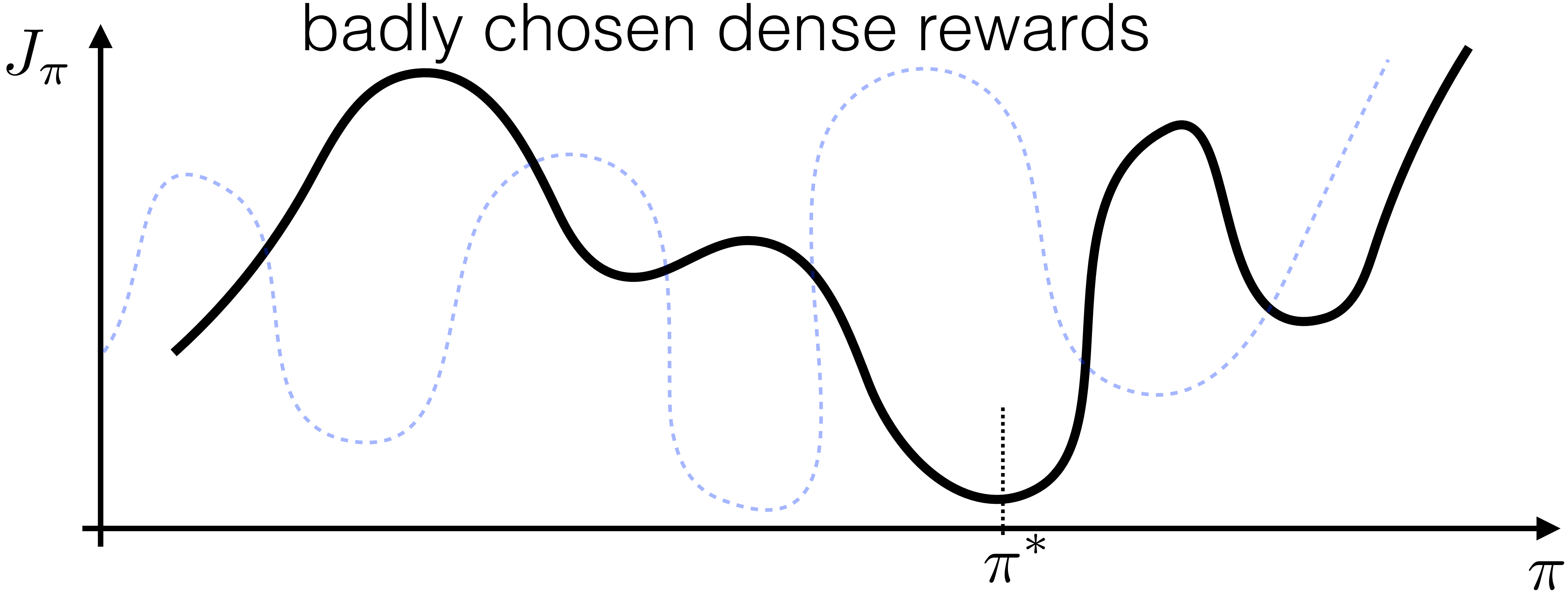
Reward shaping

- Sparse rewards are easier to design correctly
- Dense rewards are easier to learn



Reward shaping

- Sparse rewards are easier to design correctly
- Dense rewards are easier to learn



Reward shaping

- Dense reward allows to easier find the corresponding action but they are more likely to introduce bias.
- Boat racing (bad dense rewards):
  - sparse rewards (winning the race)
  - dense rewards (collecting powerups, checkpoints …)

Disadvantages of value-based methods

- Resulting policy is deterministic => exploration unclear=> eps-greedy exploration is often inefficient
- Handling continuous action-space is complicated (requires online optimization during inference)
- Learning of value based methods minimize estimation error of Q-function (does not directly maximize policy rewards).

Taxonomy of policy search methods

- Direct policy search (primal task)

  e.g. gradient ascent for $\pi^* = \arg\max_{\pi} J_{\pi}$

grey zone

Episodic REPS [Peters, 2010]

PILCO [Deisenroth, ICML 2011]

Actor-critic (e.g. DPG [Silver,JMLR 2014])

Deep Q-learning (e.g. [Mnih,Nature 2015])

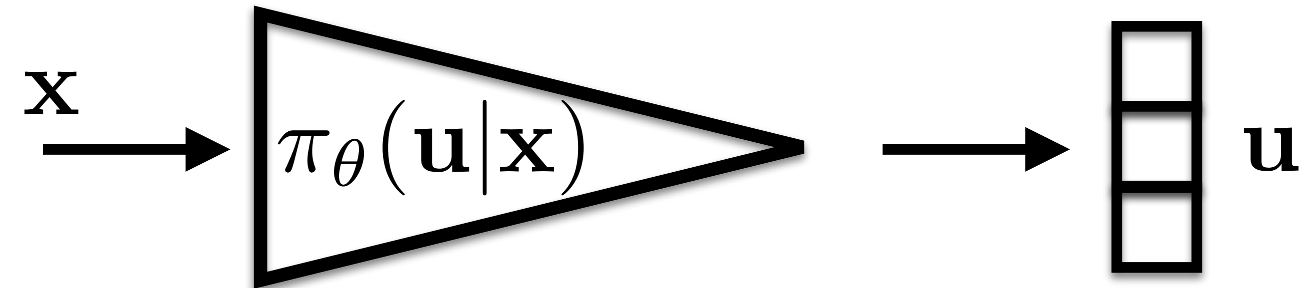
- Value-based methods (dual function [Kober, 2013])

  e.g. search for $Q(\mathbf{x}, \mathbf{a}) = r(\mathbf{x}, \mathbf{a}, \mathbf{x}') + \gamma \max_{\mathbf{a}'} Q(\mathbf{x}', \mathbf{a}')$

$$\pi^* = \arg\max_{a} Q(\mathbf{x}, \mathbf{a})$$

Primal task

Stochastic policy for discrete control:



$$\pi_\theta(\mathbf{u}|\mathbf{x}) = f(\mathbf{x}, \theta)$$

- Let us consider episodic setting:
  - Initialize in some start state
  - Run the policy in the envornment
  - Generate trajectory τ
  - Obtain reward for the generated trajectory $r(\tau)$
  - Update policy parameters

Example:

- Throwing a ball into a basket (what is suitable reward?)

Primal task - episodic setting

1. Randomly initialize policy π_θ

Primal task - episodic setting

1. Randomly initialize policy π_θ
2. Collect trajectories τ with policy π_θ

Primal task - episodic setting

1. Randomly initialize policy π_θ
2. Collect trajectories τ with policy π_θ
3. Denote $p(\tau|\pi_\theta)$ probability of τ occurs when following π_θ

Primal task - episodic setting

1. Randomly initialize policy π_θ
2. Collect trajectories τ with policy π_θ
3. Denote $p(\tau|\pi_\theta)$ probability of τ occurs when following π_θ
4. Define criterion

$$J(\theta) = \mathbb{E}_{\tau \sim p(\tau|\pi_\theta)}\{r(\tau)\} = \int\limits_{\tau \in \mathcal{T}} p(\tau|\pi_\theta) r(\tau)\, \mathrm{d}\tau \approx \frac{1}{N}\sum_{i=1}^{N} r(\tau_i)$$

Primal task - episodic setting

1. Randomly initialize policy π_θ
2. Collect trajectories τ with policy π_θ
3. Denote $p(\tau|\pi_\theta)$ probability of τ occurs when following π_θ
4. Define criterion

$$J(\theta) = \mathbb{E}_{\tau \sim p(\tau|\pi_\theta)}\{r(\tau)\} = \int\limits_{\tau \in \mathcal{T}} p(\tau|\pi_\theta) r(\tau)\, \mathrm{d}\tau \approx \frac{1}{N}\sum_{i=1}^{N} r(\tau_i)$$

5. Optimize criterion (e.g. gradient descent)

$$\theta^* = \arg\min_\theta J(\theta)$$

6. Repeat from 2

Primal task - episodic setting

$$J(\theta) = \mathbb{E}_{\tau\sim p(\tau|\pi_\theta)}\{r(\tau)\} = \int\limits_{\tau\in\mathcal{T}} p(\tau|\pi_\theta) r(\tau)\,\mathrm{d}\tau \approx \frac{1}{N}\sum_{i=1}^{N} r(\tau_i)$$

$$\theta^* = \arg\min_\theta J(\theta)$$

- What do I need for gradient descent optimization? $\frac{\partial J(\theta)}{\partial \theta}^\top$
- Perturb parameters by $\Delta\theta_i$ and estimate $J(\theta+\Delta\theta_i)$

$$J(\theta+\Delta\theta_i) = J(\theta) + \frac{\partial J(\theta)}{\partial \theta}^\top \Delta\theta_i$$

$$\Delta\theta_i^\top \frac{\partial J(\theta)}{\partial \theta} = J(\theta) - J(\theta+\Delta\theta_i)$$

Primal task - episodic setting

$$J(\theta) = \mathbb{E}_{\tau \sim p(\tau|\pi_\theta)}\{r(\tau)\} = \int_{\tau \in \mathcal{T}} p(\tau|\pi_\theta) r(\tau)\, \mathrm{d}\tau \approx \frac{1}{N}\sum_{i=1}^{N} r(\tau_i)$$

$$\theta^* = \arg\min_\theta J(\theta)$$

- What do I need for gradient descent optimization? $\frac{\partial J(\theta)}{\partial \theta}^\top$
- Perturb parameters by $\Delta\theta_i$ and estimate $J(\theta + \Delta\theta_i)$

$$J(\theta + \Delta\theta_i) = J(\theta) + \frac{\partial J(\theta)}{\partial \theta}^\top \Delta\theta_i$$

$$\Delta\theta_i^\top \frac{\partial J(\theta)}{\partial \theta} = J(\theta) - J(\theta + \Delta\theta_i)$$

$$\underbrace{\begin{bmatrix} \Delta\theta_1^\top \\ \vdots \\ \Delta\theta_n^\top \end{bmatrix}}_{\text{matrix } \mathtt{A}} \frac{\partial J(\theta)}{\partial \theta} = \underbrace{\begin{bmatrix} J(\theta) - J(\theta + \Delta\theta_1)) \\ \vdots \\ J(\theta) - J(\theta + \Delta\theta_n)) \end{bmatrix}}_{\text{vector } \mathbf{b}}$$

Primal task - episodic setting

$$\underbrace{\begin{bmatrix} \Delta\theta_1^\top \\ \vdots \\ \Delta\theta_n^\top \end{bmatrix}}_{\text{matrix } \mathtt{A}} \frac{\partial J(\theta)}{\partial \theta} = \underbrace{\begin{bmatrix} J(\theta) - J(\theta + \Delta\theta_1)) \\ \vdots \\ J(\theta) - J(\theta + \Delta\theta_n)) \end{bmatrix}}_{\text{vector } \mathbf{b}}$$

$$\frac{\partial J(\theta)}{\partial \theta} = \begin{bmatrix} \Delta\theta_1^\top \\ \vdots \\ \Delta\theta_n^\top \end{bmatrix}^+ \cdot \begin{bmatrix} J(\theta) - J(\theta + \Delta\theta_1)) \\ \vdots \\ J(\theta) - J(\theta + \Delta\theta_n)) \end{bmatrix}$$

Primal task - episodic setting

1. Randomly initialize θ

2. Collect trajectories randomly perturbed policy $\pi_{\theta+\Delta\theta_i}$

3. Compute gradient $\frac{\partial J(\theta)}{\partial \theta}^\top$ using pseudo-inverse

$$\frac{\partial J(\theta)}{\partial \theta} = \begin{bmatrix} \Delta\theta_1^\top \\ \vdots \\ \Delta\theta_n^\top \end{bmatrix}^+ \cdot \begin{bmatrix} J(\theta) - J(\theta + \Delta\theta_1)) \\ \vdots \\ J(\theta) - J(\theta + \Delta\theta_n)) \end{bmatrix}$$

4. Update parameters

$$\theta \leftarrow \theta + \alpha \frac{\partial J(\theta)}{\partial \theta}$$

Primal task - episodic setting

REINFORCE: better gradient approximation

- stochastic policy $\pi_\theta(\mathbf{u}|\mathbf{x}) : X \times U \to [0; 1]$

- criterion

$$J(\theta) = \int_T p(\tau|\pi_\theta) r(\tau) d\tau$$

- gradient of the criterion

$$\frac{\partial J(\theta)}{\partial \theta} = \int_T \frac{\partial p(\tau|\pi_\theta)}{\partial \theta} r(\tau) d\tau$$

- likelihood ratio trick expresses gradient of the prob distr.

Primal task - episodic setting

REINFORCE: better gradient approximation

- stochastic policy $\pi_\theta(\mathbf{u}|\mathbf{x}) : X \times U \rightarrow [0; 1]$

- criterion

$$J(\theta) = \int_T p(\tau|\pi_\theta) r(\tau) d\tau$$

- gradient of the criterion

$$\frac{\partial J(\theta)}{\partial \theta} = \int_T \frac{\partial p(\tau|\pi_\theta)}{\partial \theta} r(\tau) d\tau$$

- likelihood ratio trick expresses gradient of the prob distr.

$$\frac{\partial p(\tau|\pi_\theta)}{\partial \theta} = p(\tau|\pi_\theta) \frac{\partial \log p(\tau|\pi_\theta)}{\partial \theta}$$

Primal task - episodic setting

- after substitution

$$\frac{\partial J(\theta)}{\partial \theta} = \int_T p(\tau|\pi_\theta) \frac{\partial \log p(\tau|\pi_\theta)}{\partial \theta} r(\tau) d\tau =$$

$$= \mathbb{E}_{\tau \sim p(\tau|\pi_\theta)} [\frac{\partial \log p(\tau|\pi_\theta)}{\partial \theta} r(\tau)] \approx \frac{1}{N} \sum_{i=1}^{N} \frac{\partial \log p(\tau_i|\pi_\theta)}{\partial \theta} r(\tau_i)$$

Primal task - episodic setting

- after substitution

$$\frac{\partial J(\theta)}{\partial \theta} = \int_T p(\tau|\pi_\theta) \frac{\partial \log p(\tau|\pi_\theta)}{\partial \theta} r(\tau) d\tau =$$

$$= \mathbb{E}_{\tau \sim p(\tau|\pi_\theta)} [\frac{\partial \log p(\tau|\pi_\theta)}{\partial \theta} r(\tau)] \approx \frac{1}{N} \sum_{i=1}^{N} \frac{\partial \log p(\tau_i|\pi_\theta)}{\partial \theta} r(\tau_i)$$

- where prob distribution simplified using MDP assumption

$$p(\tau|\pi_\theta) = p(\mathbf{x}_0) \prod_k p(\mathbf{x}_{k+1}|\mathbf{x}_k, \mathbf{u}_k) \pi_\theta(\mathbf{u}_k|\mathbf{x}_\mathbf{k})$$

$$\frac{\partial \log p(\tau|\pi_\theta)}{\partial \theta} = \frac{\partial}{\partial \theta} [\log p(\mathbf{x}_0) + \sum_k \log(p(\mathbf{x}_{k+1}|\mathbf{x}_k, \mathbf{u}_k)) +$$

$$+ \sum_k \log(\pi_\theta(\mathbf{u}_k|\mathbf{x}_\mathbf{k})] = \sum_k \frac{\partial \log \pi_\theta(\mathbf{u}_k|\mathbf{x}_k)}{\partial \theta}$$

Primal task - episodic setting

- after substitution

$$\frac{\partial J(\theta)}{\partial \theta} = \int_T p(\tau|\pi_\theta) \frac{\partial \log p(\tau|\pi_\theta)}{\partial \theta} r(\tau) d\tau =$$

$$= \mathbb{E}_{\tau \sim p(\tau|\pi_\theta)} [\frac{\partial \log p(\tau|\pi_\theta)}{\partial \theta} r(\tau)] \approx \frac{1}{N} \sum_{i=1}^{N} \frac{\partial \log p(\tau_i|\pi_\theta)}{\partial \theta} r(\tau_i)$$

- where prob distribution simplified using MDP assumption

$$p(\tau|\pi_\theta) = p(\mathbf{x}_0) \prod_k p(\mathbf{x}_{k+1}|\mathbf{x}_k, \mathbf{u}_k) \pi_\theta(\mathbf{u}_k|\mathbf{x_k})$$

$$\frac{\partial \log p(\tau|\pi_\theta)}{\partial \theta} = \frac{\partial}{\partial \theta} [\log p(\mathbf{x}_0) + \sum_k \log(p(\mathbf{x}_{k+1}|\mathbf{x}_k, \mathbf{u}_k)) +$$

$$+ \sum_k \log(\pi_\theta(\mathbf{u}_k|\mathbf{x_k})] = \sum_k \frac{\partial \log \pi_\theta(\mathbf{u}_k|\mathbf{x}_k)}{\partial \theta}$$

Primal task - episodic setting

- after substitution

$$\frac{\partial J(\theta)}{\partial\theta} = \int_T p(\tau|\pi_\theta)\frac{\partial \log p(\tau|\pi_\theta)}{\partial\theta} r(\tau)d\tau =$$

$$= \mathbb{E}_{\tau\sim p(\tau|\pi_\theta)}[\frac{\partial \log p(\tau|\pi_\theta)}{\partial\theta} r(\tau)] \approx \frac{1}{N}\sum_{i=1}^{N}\frac{\partial \log p(\tau_i|\pi_\theta)}{\partial\theta} r(\tau_i)$$

$$\frac{\partial J(\theta)}{\partial\theta} = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{\infty}\frac{\partial \log \pi_\theta(\mathbf{u}_{ki}|\mathbf{x}_{ki})}{\partial\theta}\cdot\sum_{j=1}^{\infty} r(\mathbf{u}_{ji}, \mathbf{x}_{ji})$$

Primal task - discrete control in episodic settings

- policy (random ini)
- collect N trajectories

$$\tau_1 = \{(\mathbf{x}_{11}, \mathbf{u}_{11}), (\mathbf{x}_{21}, \mathbf{u}_{21}), \dots, (\mathbf{x}_{M1}, \mathbf{u}_{M1})\}$$
$$\vdots$$
$$\tau_N = \{(\mathbf{x}_{1N}, \mathbf{u}_{1N}), (\mathbf{x}_{2N}, \mathbf{u}_{2N}), \dots, (\mathbf{x}_{MN}, \mathbf{u}_{MN})\}$$

- compute gradient

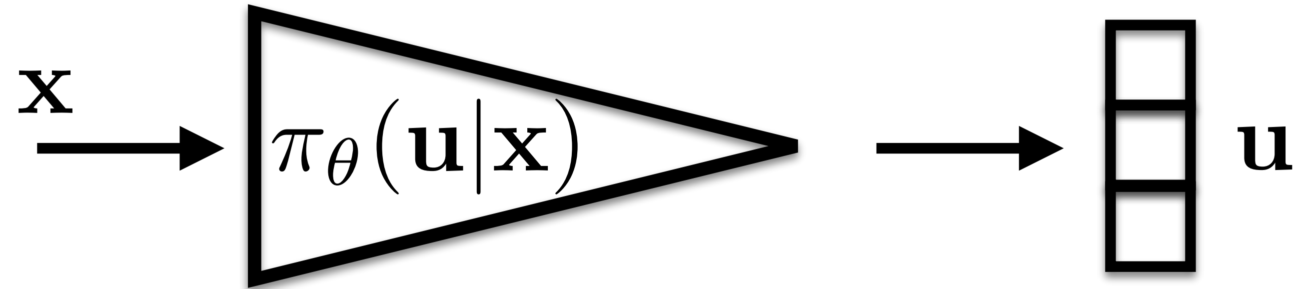
$$\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{M} \frac{\partial \log \pi_\theta(\mathbf{u}_{ki} | \mathbf{x}_{ki})}{\partial \theta} \cdot \sum_{j=1}^{M} r(\mathbf{u}_{ji}, \mathbf{x}_{ji})$$

- update parameters

$$\theta \leftarrow \theta + \alpha \frac{\partial J(\theta)}{\partial \theta}$$

Primal task - discrete control in episodic settings

- policy (random ini)



$$\pi_\theta(\mathbf{u}|\mathbf{x}) = f(\mathbf{x}, \theta)$$

- collect N trajectories

$$\tau_1 = \{(\mathbf{x}_{11}, \mathbf{u}_{11}), (\mathbf{x}_{21}, \mathbf{u}_{21}), \ldots, (\mathbf{x}_{M1}, \mathbf{u}_{M1})\}$$
$$\vdots$$
$$\tau_N = \{(\mathbf{x}_{1N}, \mathbf{u}_{1N}), (\mathbf{x}_{2N}, \mathbf{u}_{2N}), \ldots, (\mathbf{x}_{MN}, \mathbf{u}_{MN})\}$$

- compute gradient

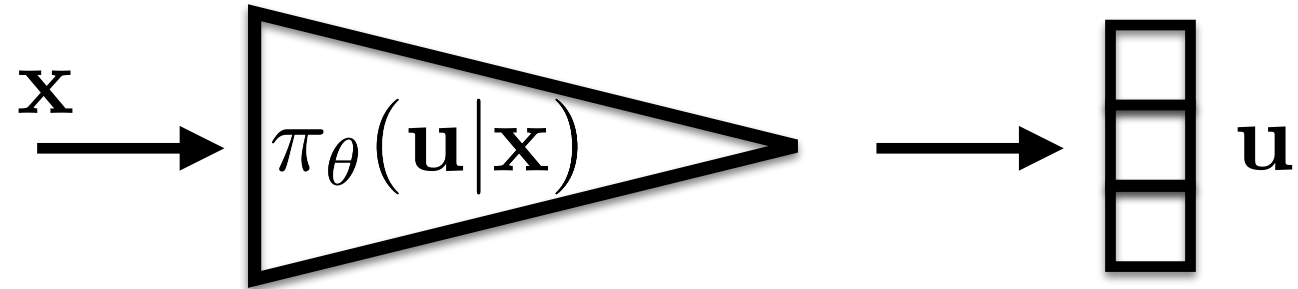
$$\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{M}\frac{\partial \log \pi_\theta(\mathbf{u}_{ki}|\mathbf{x}_{ki})}{\partial \theta} \cdot \sum_{j=1}^{M} r(\mathbf{u}_{ji}, \mathbf{x}_{ji})$$

- update parameters

$$\theta \leftarrow \theta + \alpha \frac{\partial J(\theta)}{\partial \theta}$$

Primal task - discrete control in episodic settings

- policy (random ini)



$$\pi_\theta(\mathbf{u}|\mathbf{x}) = f(\mathbf{x}, \theta)$$

- collect N trajectories

$$\tau_1 = \{(\mathbf{x}_{11}, \mathbf{u}_{11}), (\mathbf{x}_{21}, \mathbf{u}_{21}), \dots, (\mathbf{x}_{M1}, \mathbf{u}_{M1})\}$$
$$\vdots$$
$$\tau_N = \{(\mathbf{x}_{1N}, \mathbf{u}_{1N}), (\mathbf{x}_{2N}, \mathbf{u}_{2N}), \dots, (\mathbf{x}_{MN}, \mathbf{u}_{MN})\}$$

- compute gradient

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{M} \frac{\partial(-\mathcal{L}\{f(\mathbf{x}, \theta), \mathbf{u}_{ki}\})}{\partial \theta} \cdot \sum_{j=1}^{M} r(\mathbf{u}_{ji}, \mathbf{x}_{ji})$$

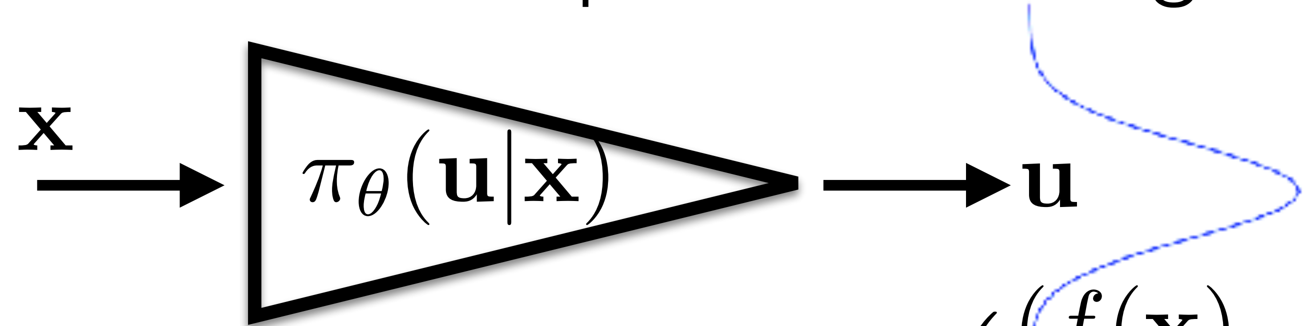
- update parameters

$$\theta \leftarrow \theta + \alpha \frac{\partial J(\theta)}{\partial \theta}$$

Minus cross-entropy loss

Primal task - continuous control in episodic settings

- policy (random ini)



$$\pi_\theta(\mathbf{u}|\mathbf{x}) = C \cdot \exp\left(\frac{(f(\mathbf{x}) - \mathbf{u})^2}{\sigma^2}\right)$$

- collect N trajectories

$$\tau_1 = \{(\mathbf{x}_{11}, \mathbf{u}_{11}), (\mathbf{x}_{21}, \mathbf{u}_{21}), \dots, (\mathbf{x}_{M1}, \mathbf{u}_{M1})\}$$

$$\vdots$$

$$\tau_N = \{(\mathbf{x}_{1N}, \mathbf{u}_{1N}), (\mathbf{x}_{2N}, \mathbf{u}_{2N}), \dots, (\mathbf{x}_{MN}, \mathbf{u}_{MN})\}$$

- compute gradient

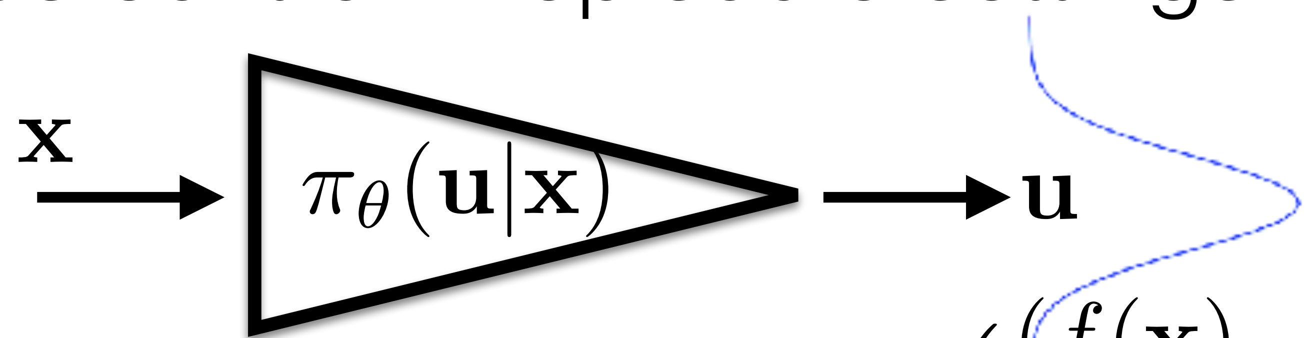
$$\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{M} \frac{\partial \log \pi_\theta(\mathbf{u}_{ki}|\mathbf{x}_{ki})}{\partial \theta} \cdot \sum_{j=1}^{M} r(\mathbf{u}_{ji}, \mathbf{x}_{ji})$$

- update parameters

$$\theta \leftarrow \theta + \alpha \frac{\partial J(\theta)}{\partial \theta}$$

Primal task - continuous control in episodic settings

- policy (random ini)



- collect N trajectories

$$\pi_\theta(\mathbf{u}|\mathbf{x}) = C \cdot \exp\left(\frac{(f(\mathbf{x}) - \mathbf{u})^2}{\sigma^2}\right)$$

$$\tau_1 = \{(\mathbf{x}_{11}, \mathbf{u}_{11}), (\mathbf{x}_{21}, \mathbf{u}_{21}), \dots, (\mathbf{x}_{M1}, \mathbf{u}_{M1})\}$$

$$\vdots$$

$$\tau_N = \{(\mathbf{x}_{1N}, \mathbf{u}_{1N}), (\mathbf{x}_{2N}, \mathbf{u}_{2N}), \dots, (\mathbf{x}_{MN}, \mathbf{u}_{MN})\}$$

- compute gradient

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{M} \frac{\partial \|f(\mathbf{x}, \theta) - \mathbf{u}_{ki}\|_2^2}{\partial \theta} \cdot \sum_{j=1}^{M} r(\mathbf{u}_{ji}, \mathbf{x}_{ji})$$

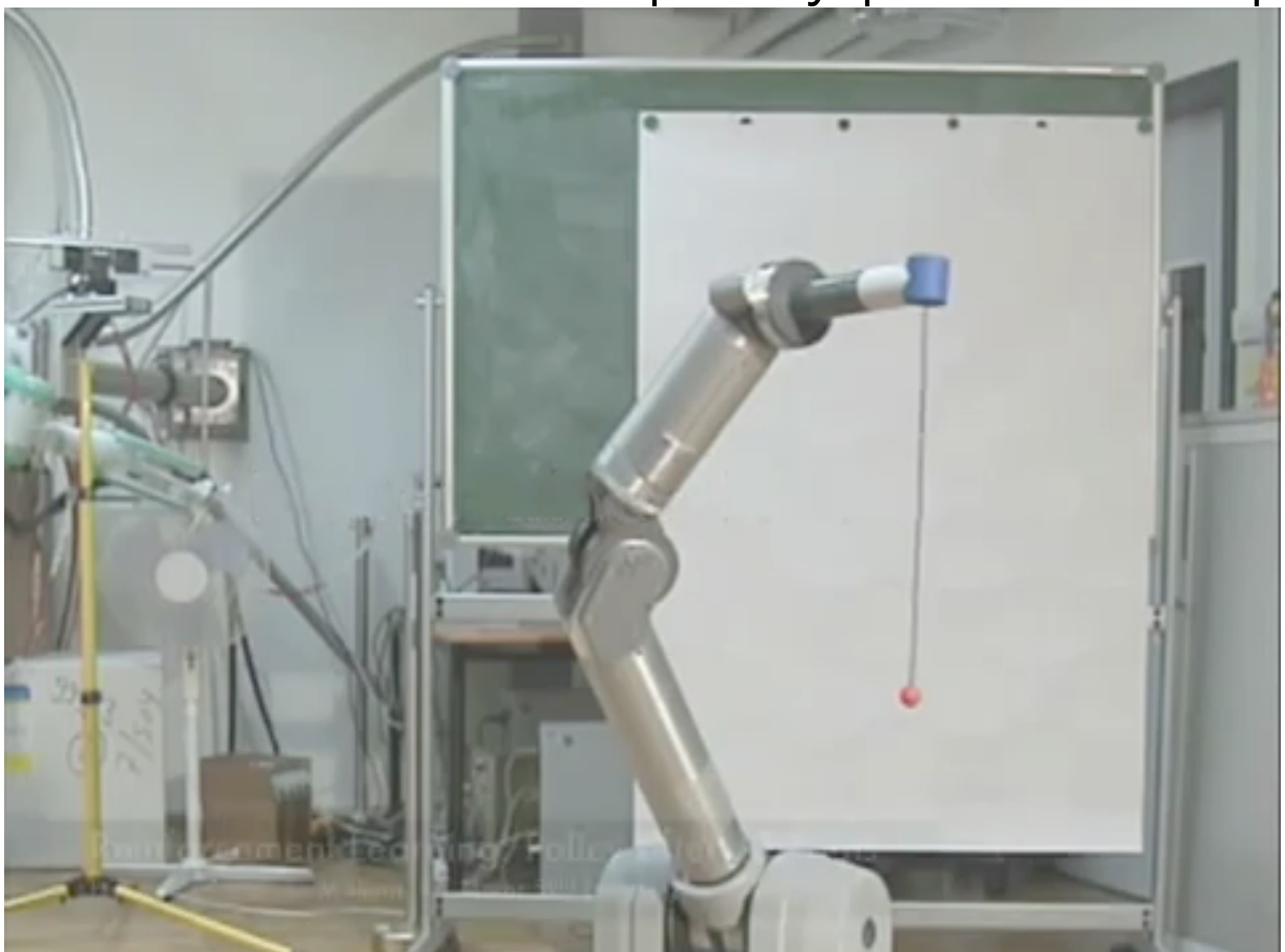
- update parameters

$$\theta \leftarrow \theta + \alpha \frac{\partial J(\theta)}{\partial \theta}$$

L2 loss

Peters et al. NOW 2013

- imitation learning from human demonstration
- **state space:** joint positions, velocities, acceler.
- **action space:** motor torques
- gradient minimization in policy parameter space



Primal task - REINFORCE alternatives [Schulman et al 2016]

- temporal coherence

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{\infty}\frac{\partial \log \pi_\theta(\mathbf{u}_{ki}|\mathbf{x}_{ki})}{\partial \theta}\cdot\Big(\sum_{j=1}^{\infty} r(\mathbf{u}_{ji},\mathbf{x}_{ji})\Big)$$

Primal task - REINFORCE alternatives [Schulman et al 2016]

- temporal coherence

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{\infty}\frac{\partial \log \pi_\theta(\mathbf{u}_{ki}|\mathbf{x}_{ki})}{\partial \theta}\cdot\Big(\sum_{j=k}^{\infty} r(\mathbf{u}_{ji},\mathbf{x}_{ji})\Big)$$

Primal task - REINFORCE alternatives [Schulman et al 2016]

- temporal coherence

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{\infty}\frac{\partial \log \pi_\theta(\mathbf{u}_{ki}|\mathbf{x}_{ki})}{\partial \theta}\cdot\Big(\sum_{j=k}^{\infty} r(\mathbf{u}_{ji},\mathbf{x}_{ji})\Big)$$

- state-action function:

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{\infty}\frac{\partial \log \pi_\theta(\mathbf{u}_{ki}|\mathbf{x}_{ki})}{\partial \theta}\cdot Q(\mathbf{u}_{ki},\mathbf{x}_{ki})$$

Primal task - REINFORCE alternatives [Schulman et al 2016]

- temporal coherence

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{\infty}\frac{\partial \log \pi_\theta(\mathbf{u}_{ki}|\mathbf{x}_{ki})}{\partial \theta}\cdot\Big(\sum_{j=k}^{\infty} r(\mathbf{u}_{ji},\mathbf{x}_{ji})\Big)$$

- state-action function:

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{\infty}\frac{\partial \log \pi_\theta(\mathbf{u}_{ki}|\mathbf{x}_{ki})}{\partial \theta}\cdot Q(\mathbf{u}_{ki},\mathbf{x}_{ki})$$

- baseline

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{\infty}\frac{\partial \log \pi_\theta(\mathbf{u}_{ki}|\mathbf{x}_{ki})}{\partial \theta}\cdot\underbrace{\Big(Q(\mathbf{u}_{ki},\mathbf{x}_{ki}) - V(\mathbf{x}_{ki})\Big)}_{A(\mathbf{u}_{ki},\mathbf{x}_{ki})}$$

Advantage function

- When deciding optimally it is enough to decide which of the actions yields higher Q-values.
- Estimation of exact Q-values is not necessary
- When Q(x,a1) = 99 and Q(x,a2)=101, it is enough to estimate that Q-value of a2 is bigger then Q-value of a1.
- Predicting such values by a deep neural network causes that most of the weights will be sacrificed to unimportant information that Q-values are around 100.
- Consequently advantage function is introduced.

$$A(\mathbf{x},\mathbf{u}) = Q(\mathbf{x},\mathbf{u}) - V(\mathbf{x})$$

- Generalized Advantage Estimation yields lower variance and faster learning

https://arxiv.org/pdf/1506.02438.pdf

Generalized Advantage function Estimation [ICLR 2016]

https://arxiv.org/pdf/1506.02438.pdf

$$A(\mathbf{x},\mathbf{u}) = Q(\mathbf{x},\mathbf{u}) - V(\mathbf{x})$$

$$A^{(1)}(\mathbf{x}_1,\mathbf{u}) = -V(\mathbf{x}_1) + r_1 + \gamma V(\mathbf{x}_2)$$

$$A^{(2)}(\mathbf{x}_1,\mathbf{u}) = -V(\mathbf{x}_1) + r_1 + \gamma r_2 + \gamma^2 V(\mathbf{x}_3)$$

$$\vdots$$

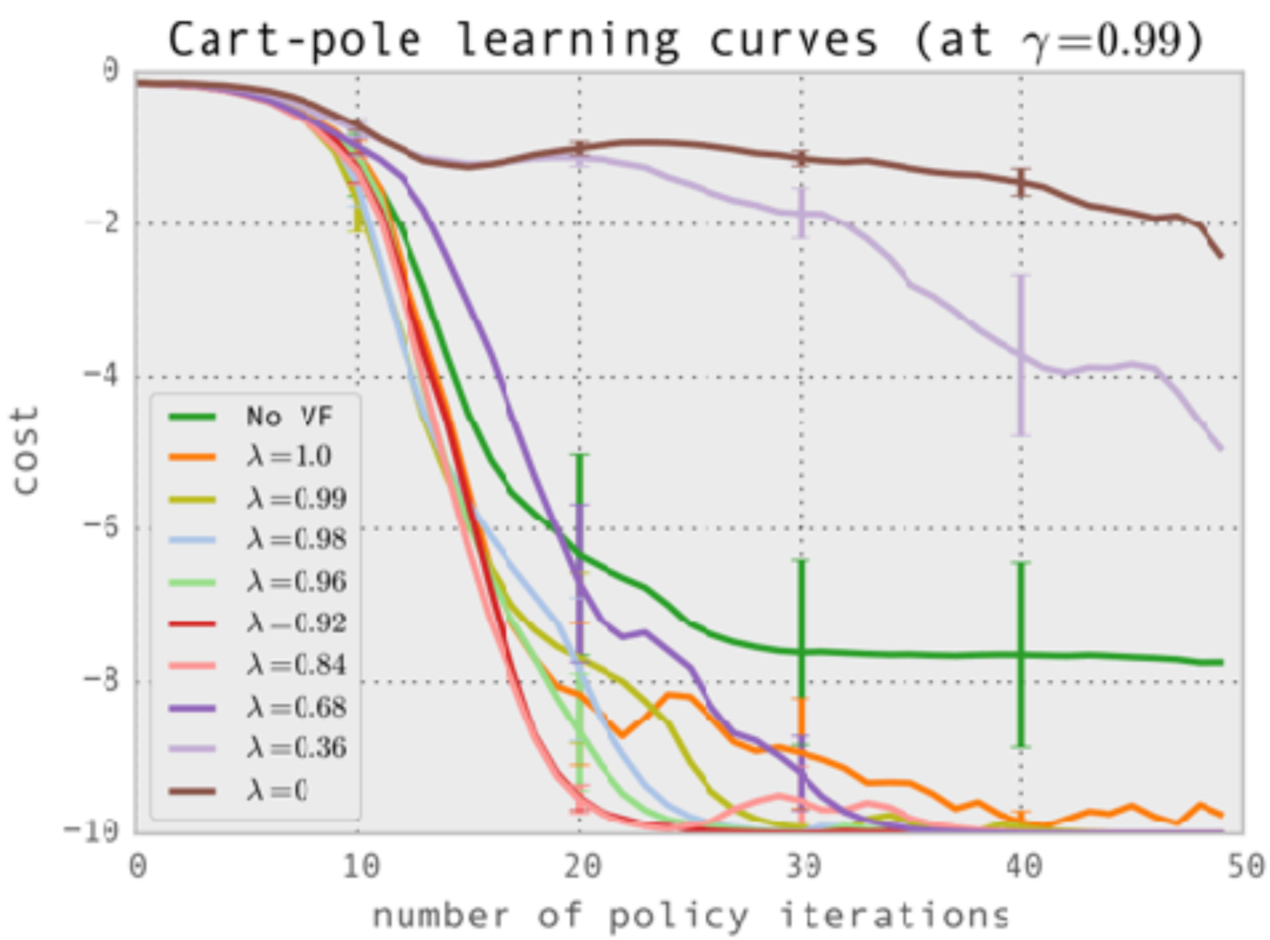
$$A^{(N)}(\mathbf{x}_1,\mathbf{u}) = -V(\mathbf{x}_1) + r_1 + \gamma r_2 + \ldots + \gamma^N V(\mathbf{x}_N)$$

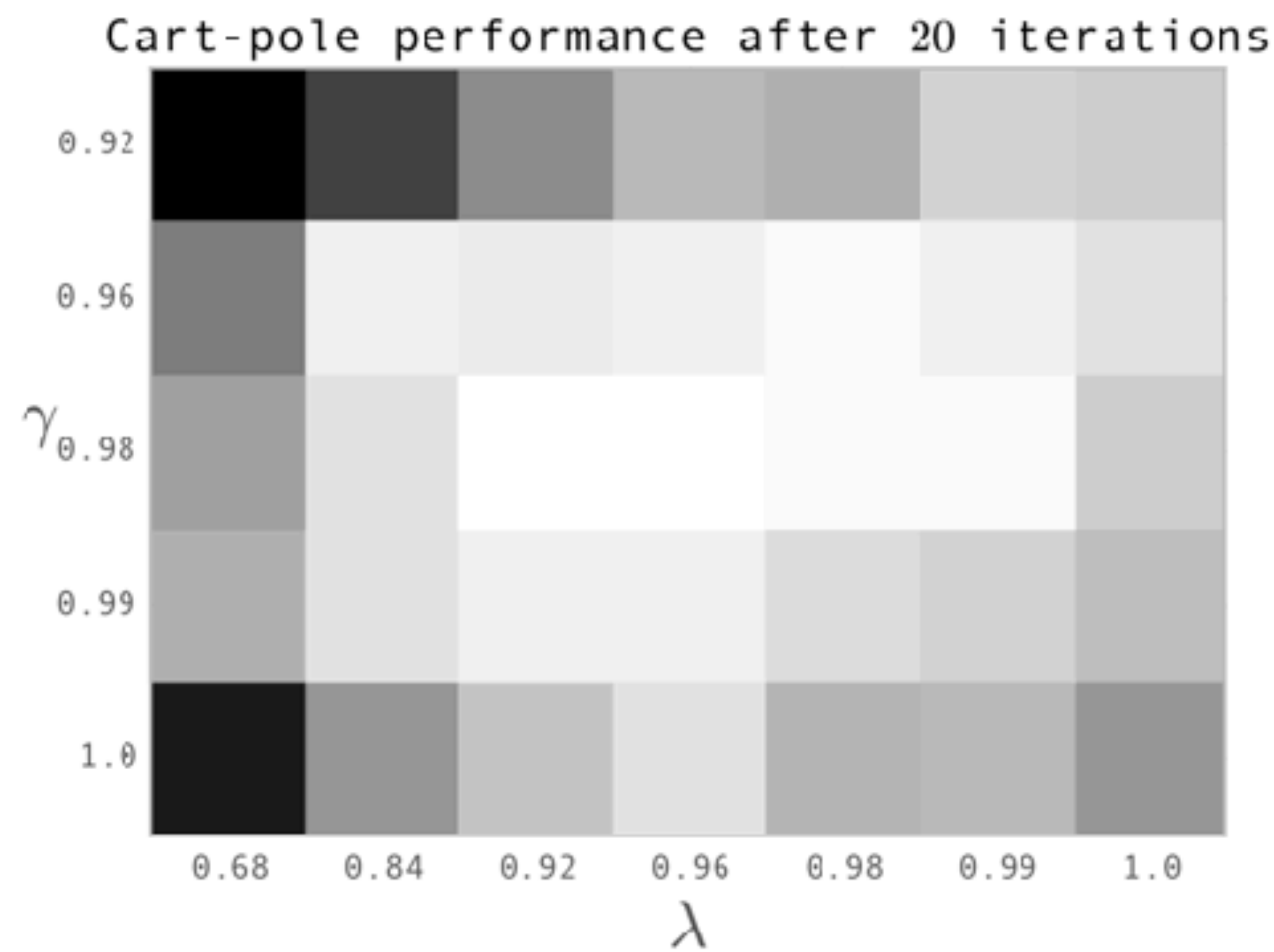
$$\hat{A}^{\lambda,\gamma} = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1}\hat{A}^{(n)}$$

advantage estimate from state-value function

lambda sets trade-off between variance and bias

Generalized Advantage function Estimation [ICLR 2016]
https://arxiv.org/pdf/1506.02438.pdf





Primal task

- No motion model required
- Converges to local optima (good initialization needed)
- High-dimensional parameters requires many samples
- Imitation learning from expert trajectories

Summary RL

- No motion model required
- Converges to local optima (good initialization needed)
- High-dimensional parameters => requires many samples
- Imitation or Inverse RL learning from expert trajectories

- If motion model is available then trajectory optimization [Tassa 2013] Tassa, Synthesis and Stabilization of Complex Behaviors through Online Trajectory Optimization, IROS2013

Taxonomy of policy search methods

- Direct policy search (primal task)

  e.g. gradient ascent for $\pi^* = \arg\max_{\pi} J_{\pi}$

grey zone

Episodic REPS [Peters, 2010]

PILCO [Deisenroth, ICML 2011]

Actor-critic (e.g. DPG [Silver,JMLR 2014])

Deep Q-learning (e.g. [Mnih,Nature 2015])

- Value-based methods (dual function [Kober, 2013])

  e.g. search for $Q(\mathbf{x},\mathbf{a}) = r(\mathbf{x},\mathbf{a},\mathbf{x}') + \gamma \max_{\mathbf{a}'} Q(\mathbf{x}',\mathbf{a}')$

$$\pi^* = \arg\max_{a} Q(\mathbf{x},\mathbf{a})$$

DDPG actor-critic method [Lilicrap et al. 2015]

1. Collect trajectories $\tau_1, \tau_2, \tau_3, \ldots$, initialize $\theta = \text{rand}$
2. Estimate $\mathbf{y} = r(\mathbf{x}, \mathbf{u}, \mathbf{x}') + \gamma \max_{\mathbf{u}'} Q_\theta(\mathbf{x}', \mathbf{u}')$
3. Update parameters by learning

$$\arg\min_{\theta} \sum_{\mathbf{x},\mathbf{u},\mathbf{y}} \|Q_\theta(\mathbf{x}, \mathbf{u}) - \mathbf{y}\|$$

Approximated Q-learning

DDPG actor-critic method [Lilicrap et al. 2015]

1. Collect trajectories $\tau_1, \tau_2, \tau_3, \ldots$, initialize $\theta = \text{rand}$
2. Estimate $\mathbf{y} = r(\mathbf{x}, \mathbf{u}, \mathbf{x}') + \gamma \max_{\mathbf{u}'} Q_\theta(\mathbf{x}', \mathbf{u}')$
3. Update parameters by learning

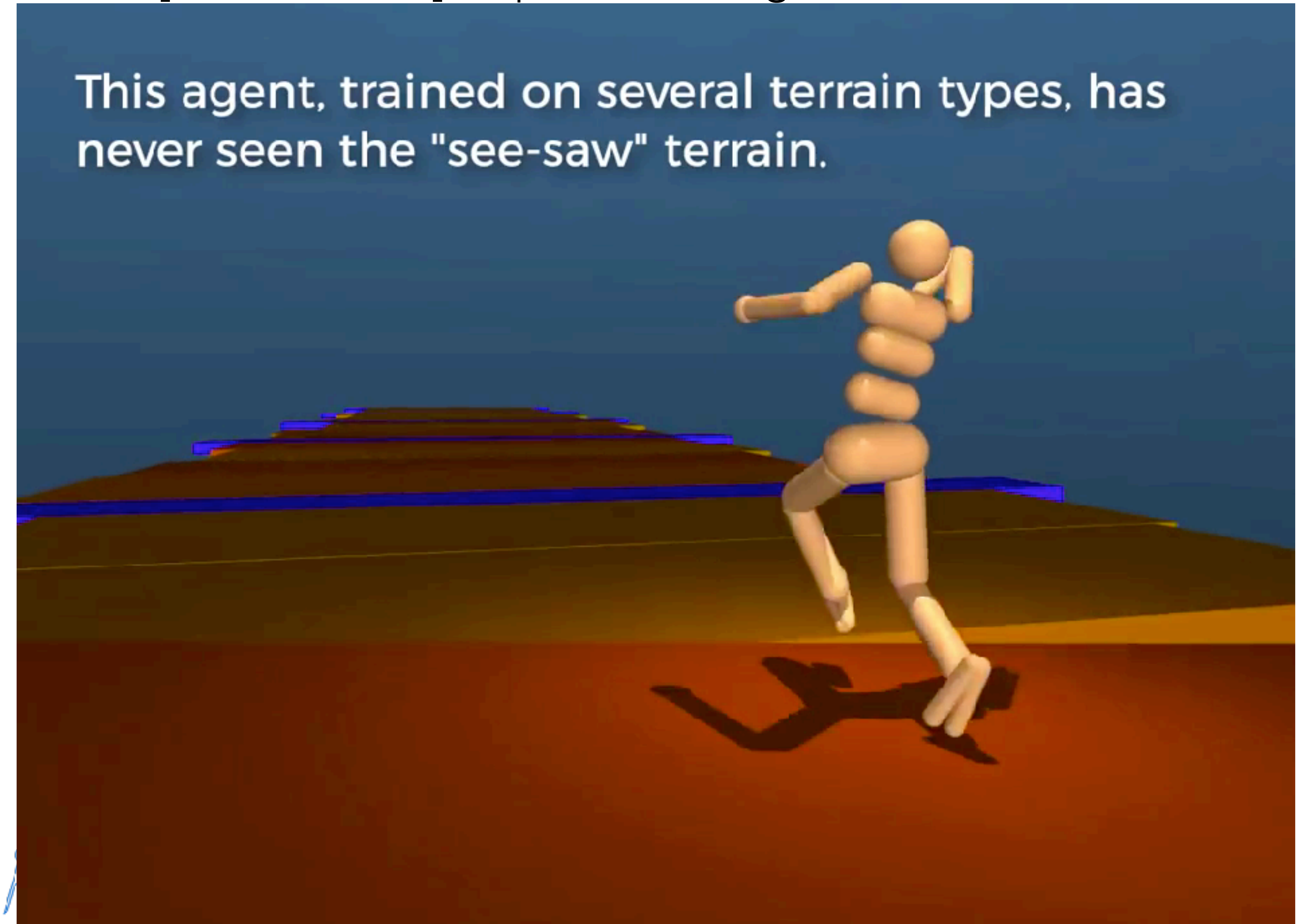
$$\arg\min_\theta \sum_{\mathbf{x},\mathbf{u},\mathbf{y}} \|Q_\theta(\mathbf{x}, \mathbf{u}) - \mathbf{y}\|$$

4. Learn policy π_ω which do actions maximizing the state-action value function on the collected trajectories

$$\arg\max_\omega \sum_{\mathbf{x}\in\tau} Q_\theta(\mathbf{x}, \pi_\omega(\mathbf{x}))$$

Direct policy optimization on Q

Known successes of RL - locomotion in simulation [Heess 2017] https://arxiv.org/abs/1707.02286



Known successes of RL - Starcraft II

- Starcraft II (Deepmind AlphaStart beaten top-end professional human gamers 5:0)

https://medium.com/mlmemoirs/deepminds-ai-alphastar-showcases-significant-progress-towards-agi-93810c94fbe9

Known successes of RL

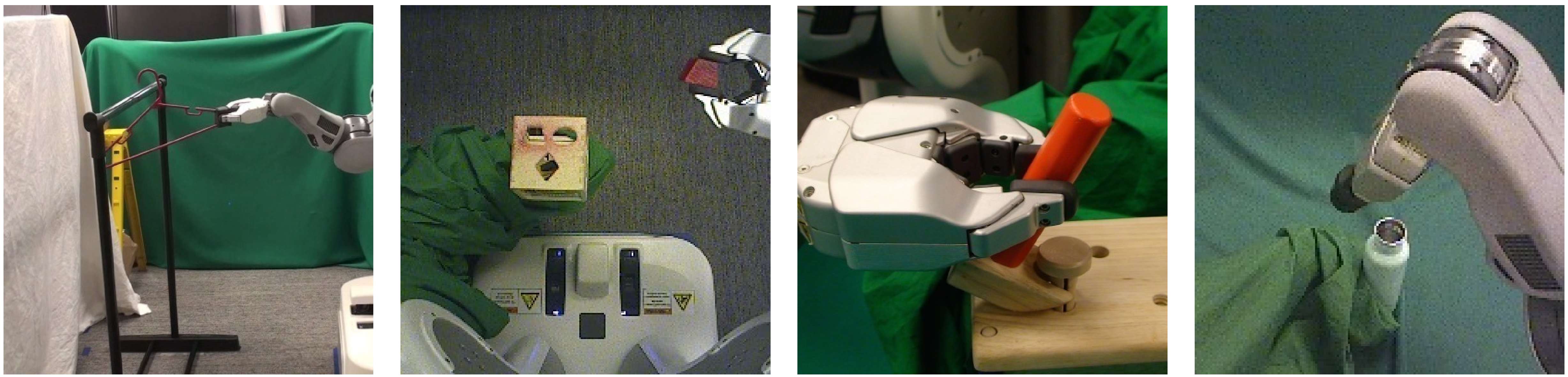
- AlphaGo/Alpha Zero https://en.wikipedia.org/wiki/AlphaZero
- SearchTrees has no chance in huge state-action spaces
  - AlphaGo:
    - beat professional Go player
    - 9 dan professional ranking
  - Alpha Zero: Top Chess Engine Championship 2017
    - 9h of self-play, no openingbooks nor endgames tables
    - 1 minute per move, 1GB RAM
    - 28 wins, 72 withdraws
- DOTA 2 openAI+ bot https://blog.openai.com/dota-2/
- AutoML https://cloud.google.com/automl/
  - [Zoph 2016] REINFORCE learns RCNN policy which generates deep CNN architectures.

Known successes of RL

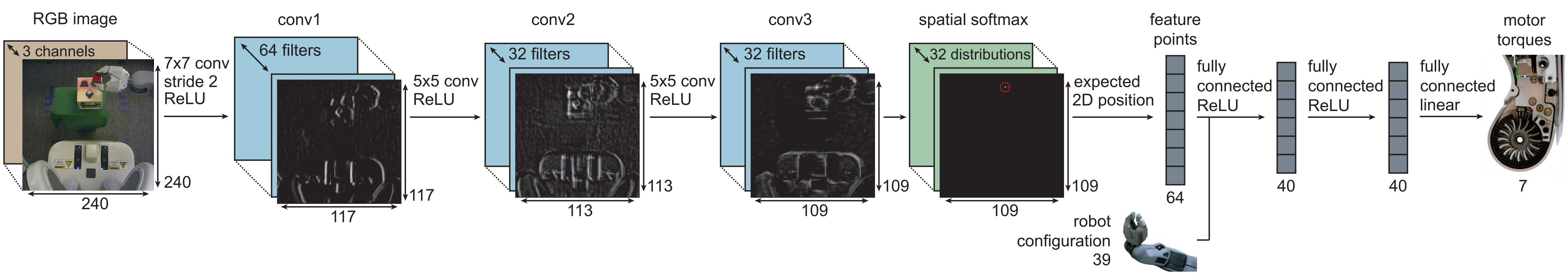
- Application on real robots is still questionable since
  - transfer from simulator suffers from domain bias
  - direct training on robots is impossible due to sample inefficiency of state-of-the-art methods.

Levine et al JMLR 2016

- guides policy gradient method by optimal trajectories
- **state space:** RGB camera images
- **action space:** motor torques



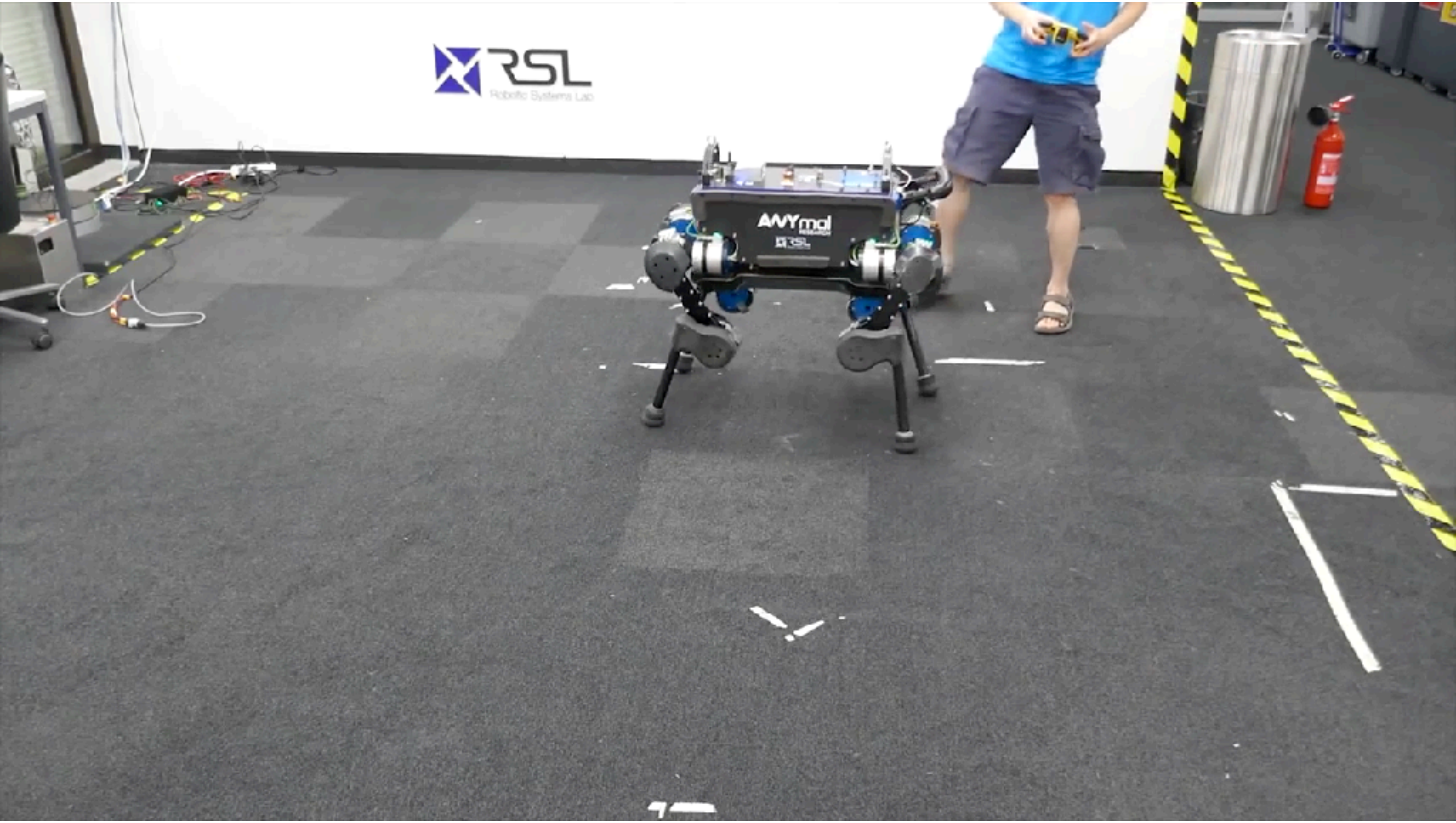
(a) hanger (b) cube (c) hammer (d) bottle



Czech Technical University in Prague
Faculty of Electrical Engineering, Department of Cybernetics

Levine et al JMLR 2016

Can we use it in real world problems?

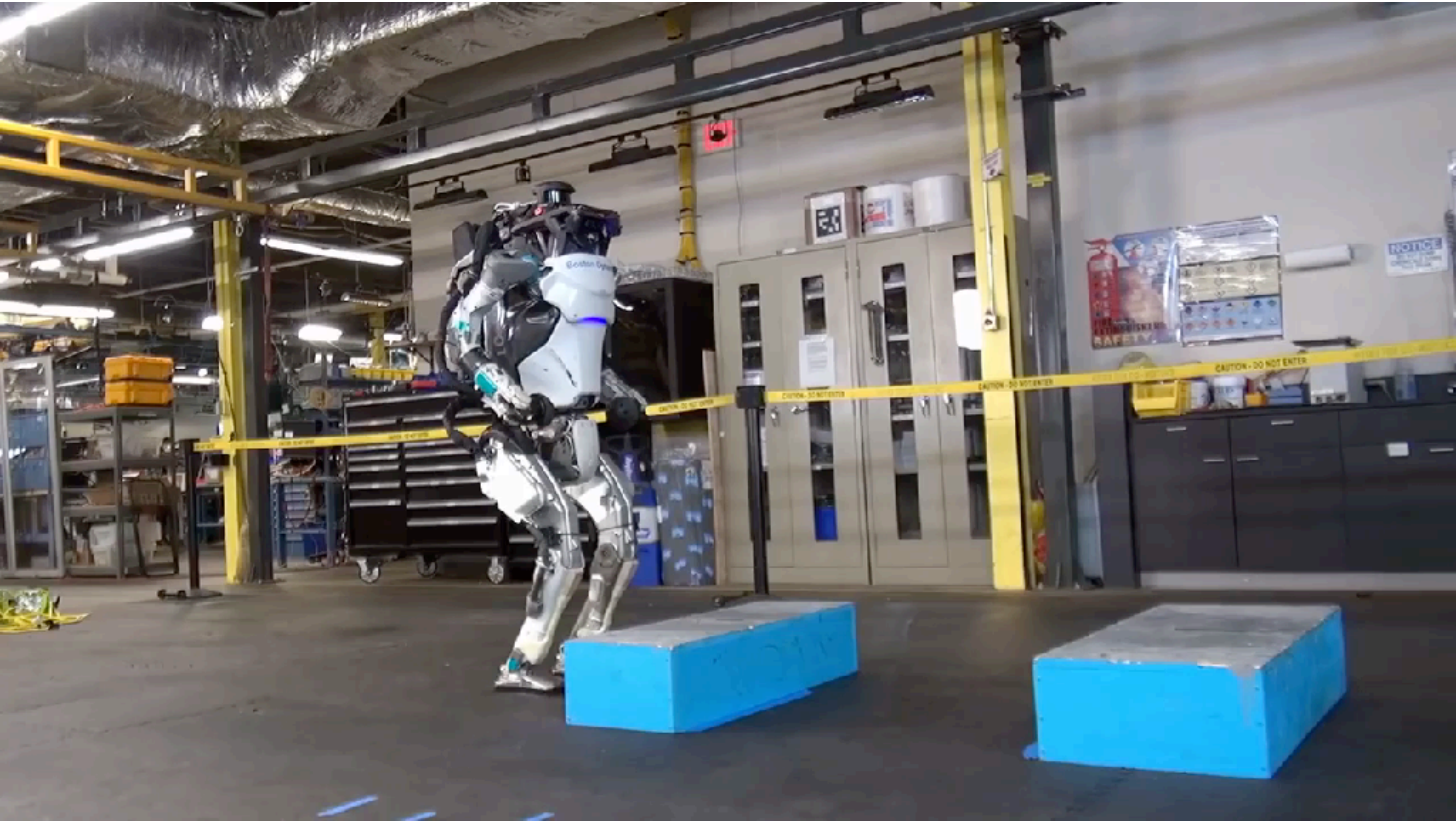


[Hwangbo, ETH Zurich, Science Robotics, 2018]

Motion and compliance control of flippers

[3] Pecka, Zimmermann, Svoboda, et al.
IROS/RAL/TIE(IF=6), 2015-2018

Boston dynamics - Atlas - NO RL AT ALL



Boston dynamics - Big dog - NO RL AT ALL

Summary

- If accurate differentiable motion model and reward functions are known, than optimal control in MDP is straightforward optimization problem (efficiently tackled by DP or DDP)
- State-action value function is dual variable wrt policy. It serves as auxiliary function in the policy optimization:
  - actor-critic methods
  - heuristic in planning methods (LQR trees)
- **Holy grail** is to efficiently combine motion model, state-action value function with efficient planning, learning and exploration.
- RL will be much more useful for motion control, when accurate domain transfer methods (from simulators to reality) become available.