REINFORCEMENT LEARNING

Gergely Neu
Univ. Pompeu Fabra

A PRIMAL-DUAL VIEW OF

REINFORCEMENT LEARNING

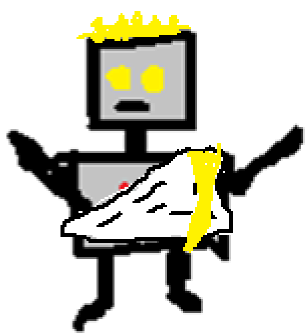
Gergely Neu
Univ. Pompeu Fabra

A PRIMAL-DUAL VIEW OF REINFORCEMENT LEARNING

GERGELY NEU
UNIV. POMPEU FABRA

WHAT IS REINFORCEMENT LEARNING?

Agent



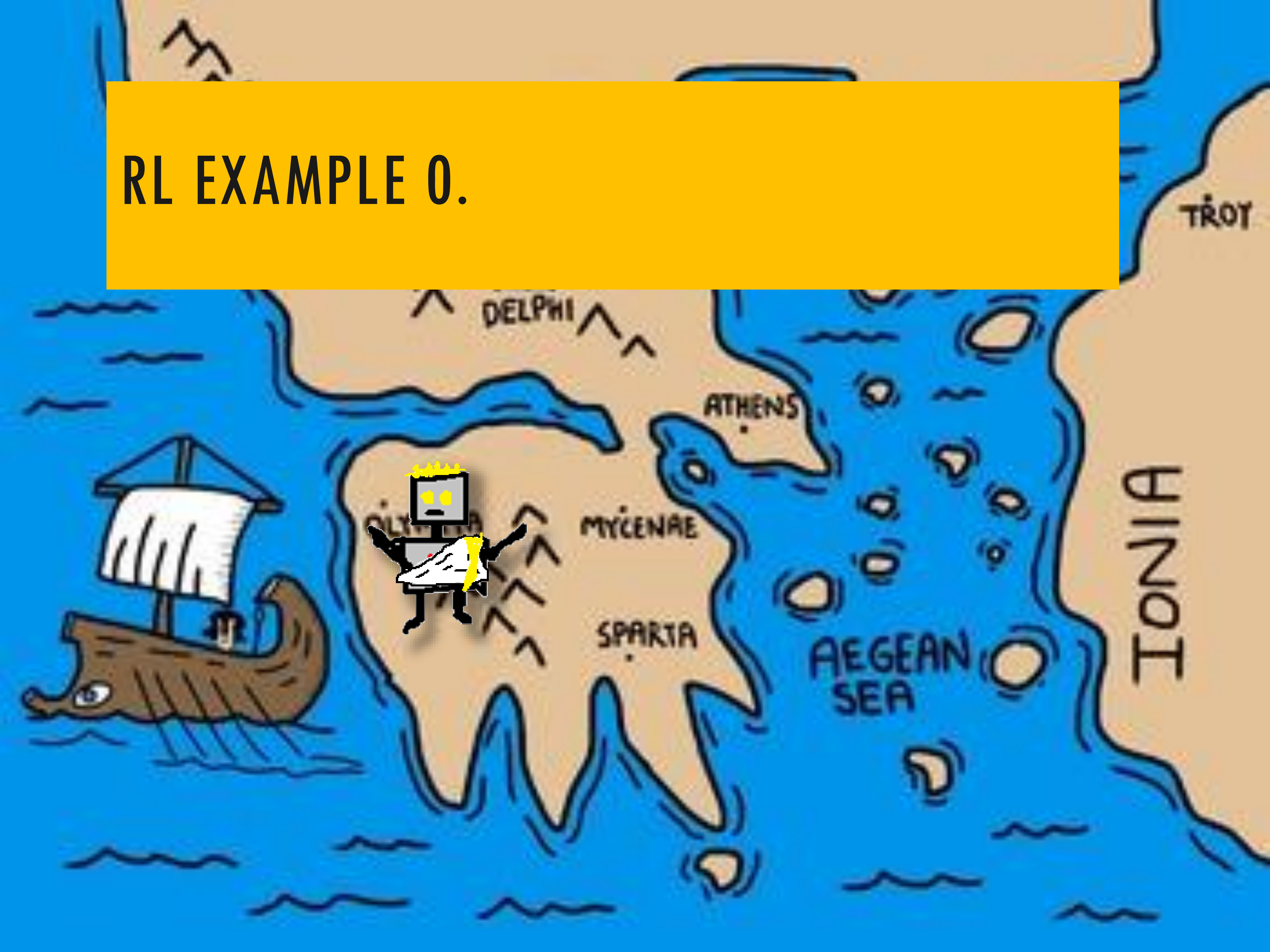
In state s,
take action a

Environment

Reward r,
new state s'

Learning to
- maximize reward
- in a reactive environment
- under partial feedback

RL EXAMPLE 0.



RL EXAMPLE 0.

RL EXAMPLE 0.


TROY
DELPHI
ATHENS
MYCENAE
SPARTA
AEGEAN SEA
IONIA
x
state
actions

RL EXAMPLE 0.


DELPHI
ATHENS
MYCENAE
SPARTA
AEGEAN SEA
IONIA
TROY
state
actions

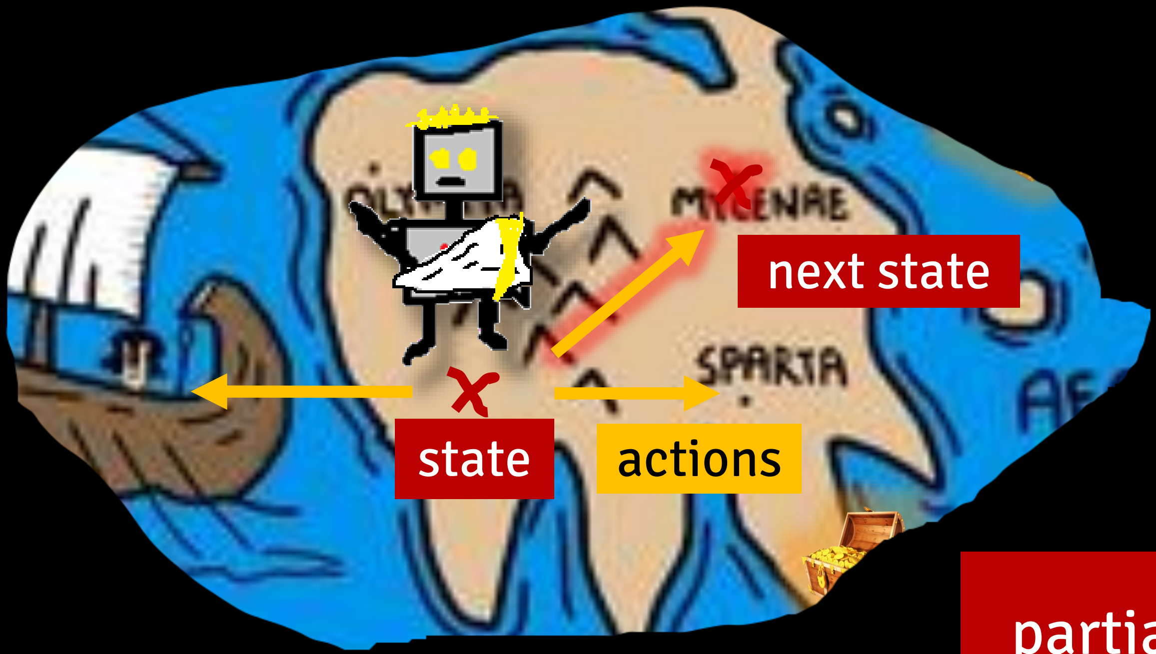
RL EXAMPLE 0.


DELPHI
ATHENS
TROY
next state
SPARTA
AEGEAN SEA
IONIA
state
actions

RL EXAMPLE 0.

DELPHI
ATHENS
TROY
reward
MYCENAE
next state
SPARTA
AEGEAN SEA
IONIA
state
actions

RL EXAMPLE 0.



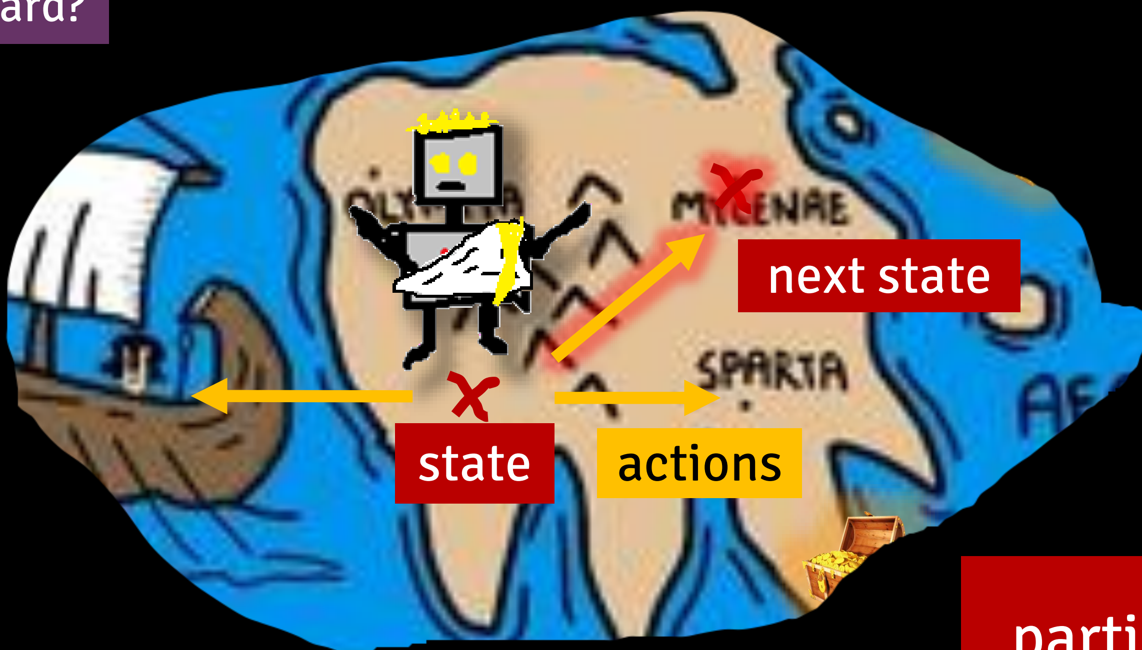
partial observability

RL EXAMPLE 0.

reward?

reward?

reward?



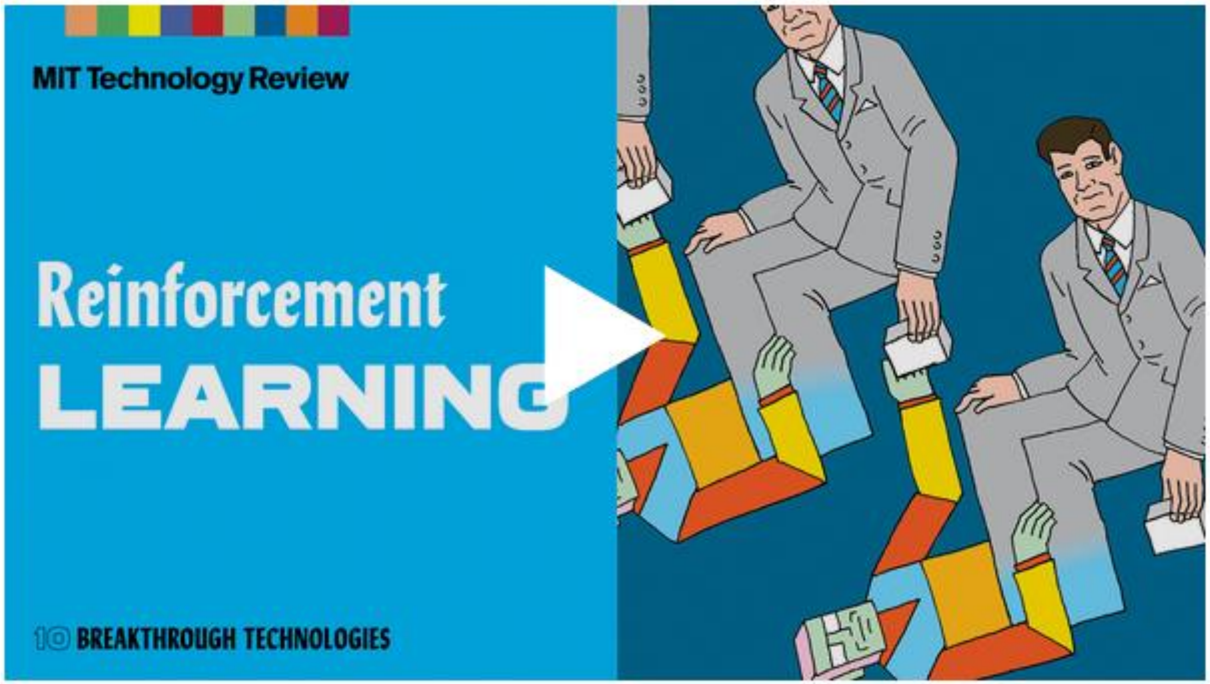
reward?

reward?

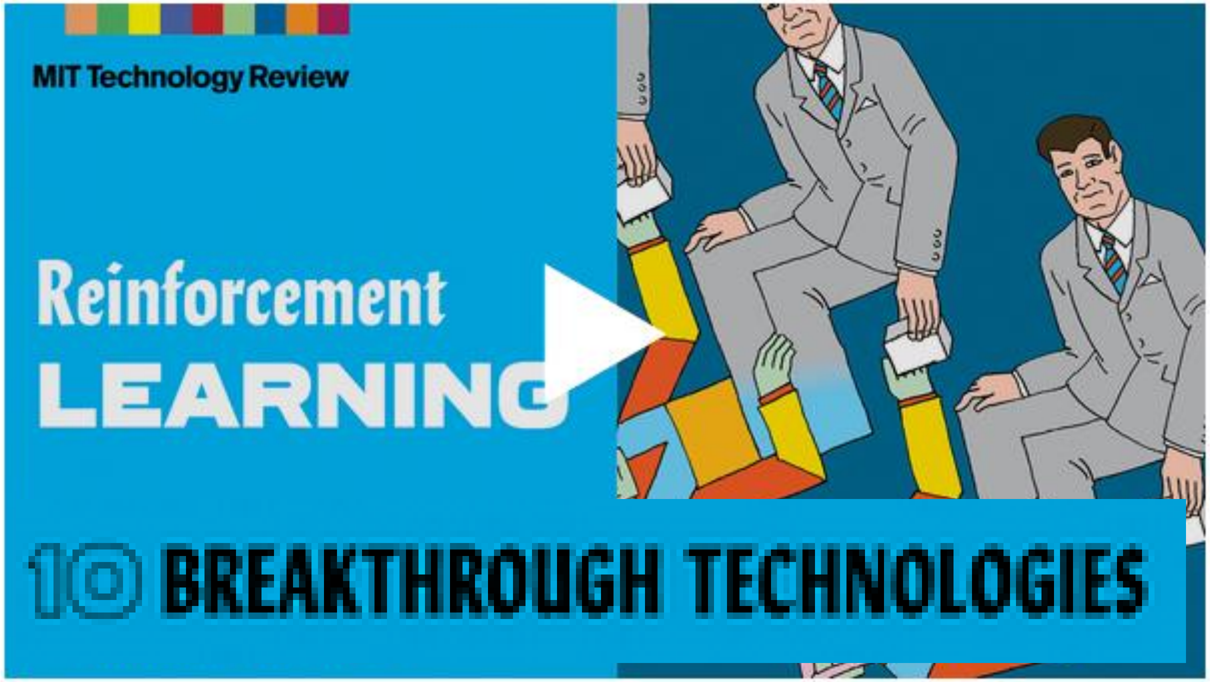
reward?

partial observability

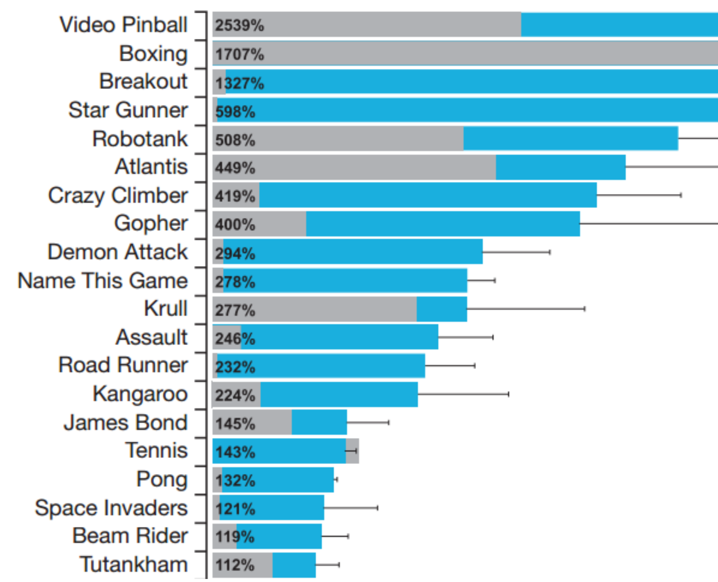
WHY SHOULD I CARE?



WHY SHOULD I CARE?



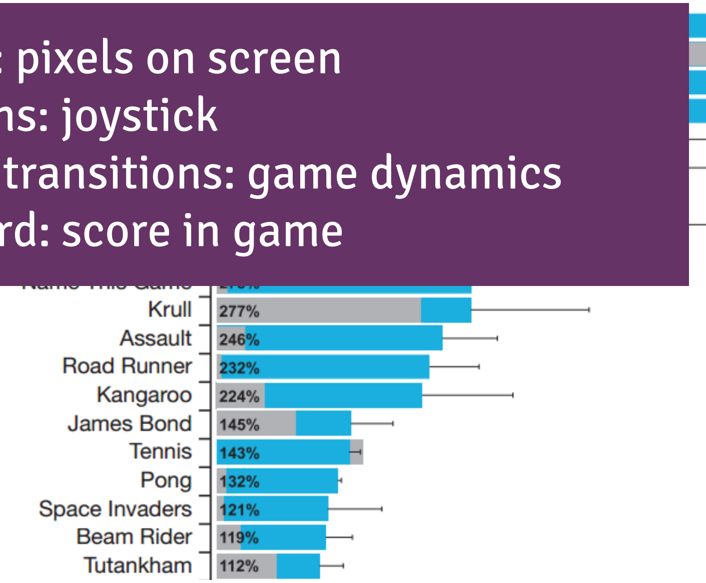
WHY SHOULD I CARE?



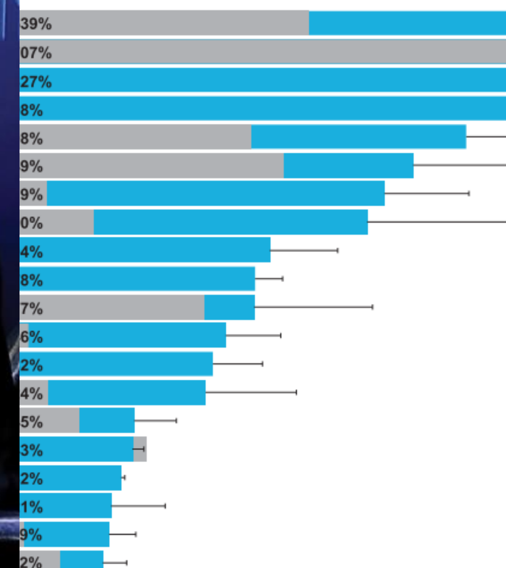
WHY SHOULD I CARE?

Breakthrough in Atari game playing

- State: pixels on screen
- Actions: joystick
- State transitions: game dynamics
- Reward: score in game



WHY SHOULD I CARE?

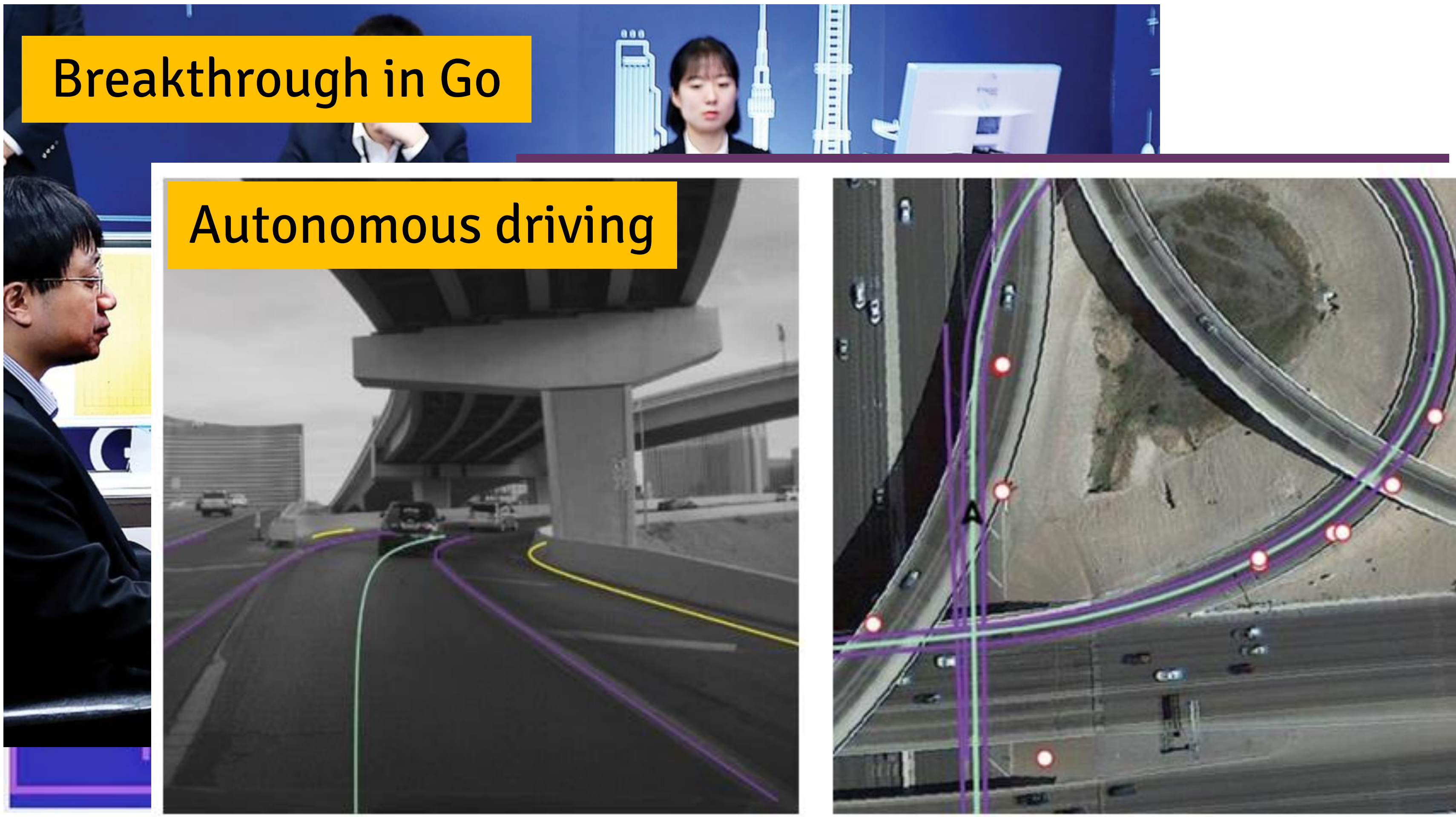


WHY SHOULD I CARE?

Breakthrough in Go

- State: stones currently on board
- Actions: place stone on board
- State transitions: own move + adversary's move
- Reward: +1 for winning the game

WHY SHOULD I CARE?

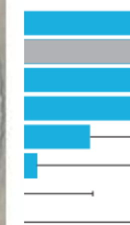


WHY SHOULD I CARE?

Breakthrough in Go

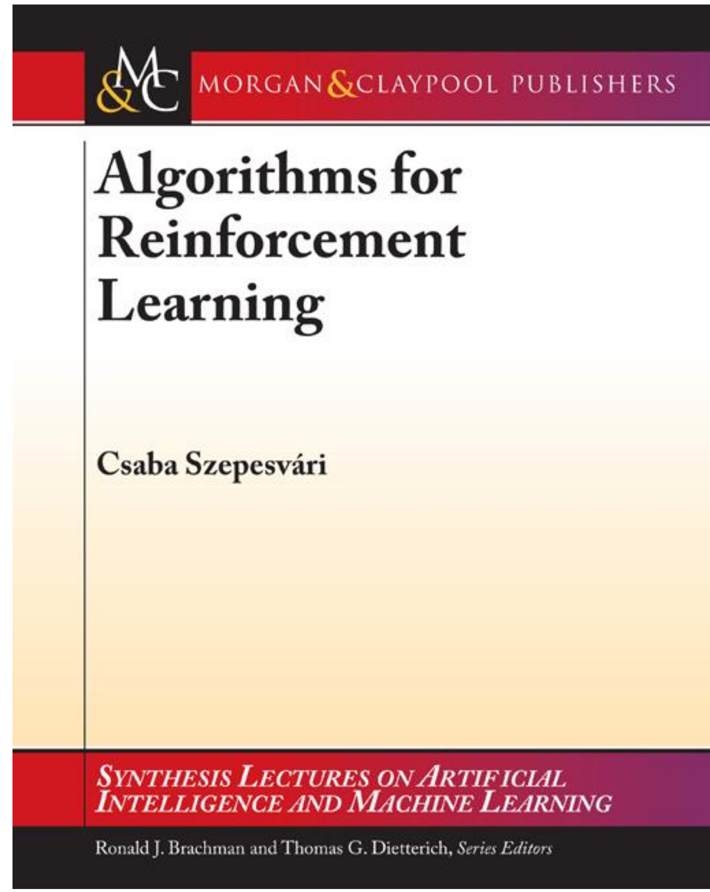
Autonomous driving

- State: road conditions, other vehicles, obstacles,…
- Actions: turn left/right, accelerate/brake,…
- State transitions: depending on state+action+randomness
- Reward: +100 for reaching destination, -100 for accidents,…



RECOMMENDED READING

- Richard Sutton and Andrew Barto (2018): “Reinforcement Learning: An Introduction”
  - For an enjoyable (but not very rigorous) introduction
- Dimitri Bertsekas (2012): “Dynamic Programming and Optimal Control”
  - For a rigorous treatment of the basics
- Csaba Szepesvári (2012): “Algorithms for RL”
  - For a rigorous description of basic RL algorithms



THIS SHORT COURSE: A PRIMAL-DUAL VIEW

- Markov decision processes
  - Value functions and optimal policies
- Primal view: Dynamic programming
  - Policy evaluation, value and policy iteration
  - Value-function-based methods
    - Temporal differences, Q-learning, LSTD, deep Q networks,…

THIS SHORT COURSE: A PRIMAL-DUAL VIEW

- Markov decision processes
  - Value functions and optimal policies
- Primal view: Dynamic programming
  - Policy evaluation, value and policy iteration
  - Value-function-based methods
    - Temporal differences, Q-learning, LSTD, deep Q networks,…
- Dual view: Linear programming
  - LP duality in MDPs
  - Direct policy optimization methods
    - Policy gradients, REPS, TRPO,…

THIS SHORT COURSE: A PRIMAL-DUAL VIEW

part 1

- Markov decision processes
  - Value functions and optimal policies
- Primal view: Dynamic programming
  - Policy evaluation, value and policy iteration
  - Value-function-based methods
    - Temporal differences, Q-learning, LSTD, deep Q networks,…

part 2

- Dual view: Linear programming
  - LP duality in MDPs
  - Direct policy optimization methods
    - Policy gradients, REPS, TRPO,…

THIS SHORT COURSE: A PRIMAL-DUAL VIEW

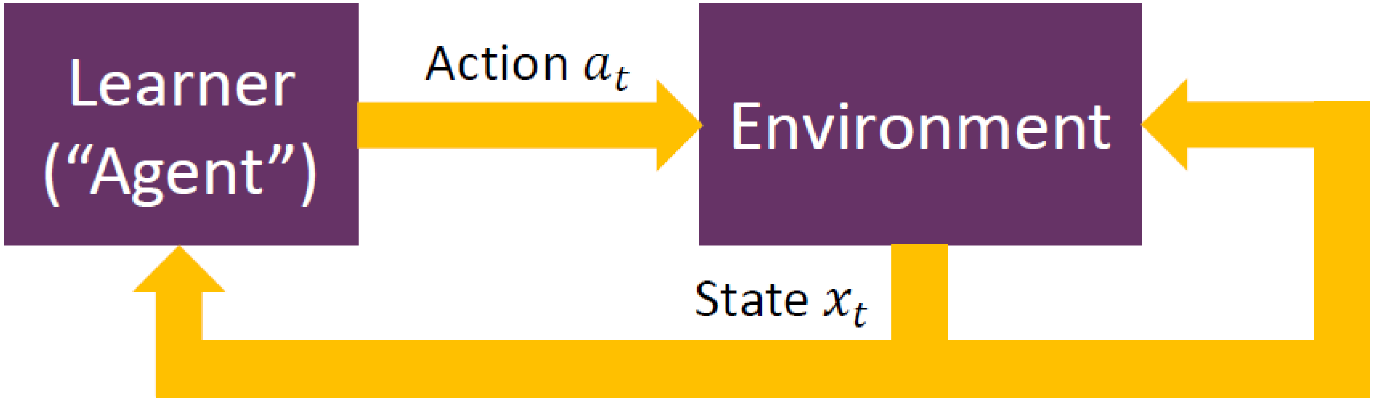
part 1

- Markov decision processes
  - Value functions and optimal policies
- Primal view: Dynamic programming
  - Policy evaluation, value and policy iteration
  - Value-function-based methods
    - Temporal differences, Q-learning, LSTD, deep Q networks,…

part 2

- Dual view: Linear programming
  - LP duality in MDPs
  - Direct policy optimization methods
    - Policy gradients, REPS, TRPO,…

MARKOV DECISION PROCESSES (MDPs)



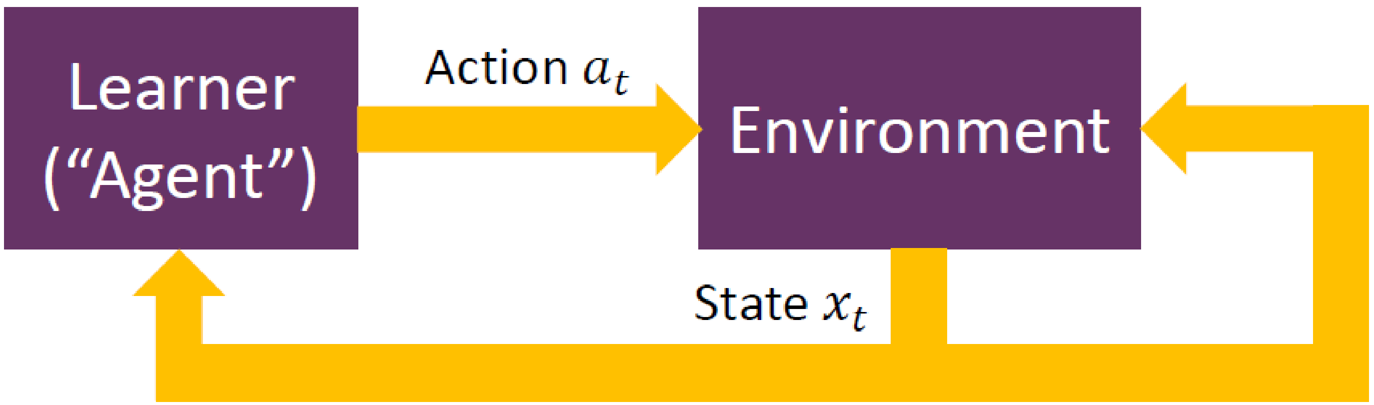
A Markov Decision Process (MDP) is characterized by

- X: a set of states
- A: a set of actions, possibly different in each state
- P: $X \times A \times X \to [0,1]$: a transition function with $P(\cdot\,|x,a)$ being the distribution of the next state given previous state x and action a:

$$\mathbf{P}[x_{t+1} = x' | x_t = x, a_t = a] = P(x'|x,a)$$

- r: $X \times A \to [0,1]$: a reward function

MARKOV DECISION PROCESSES (MDPs)



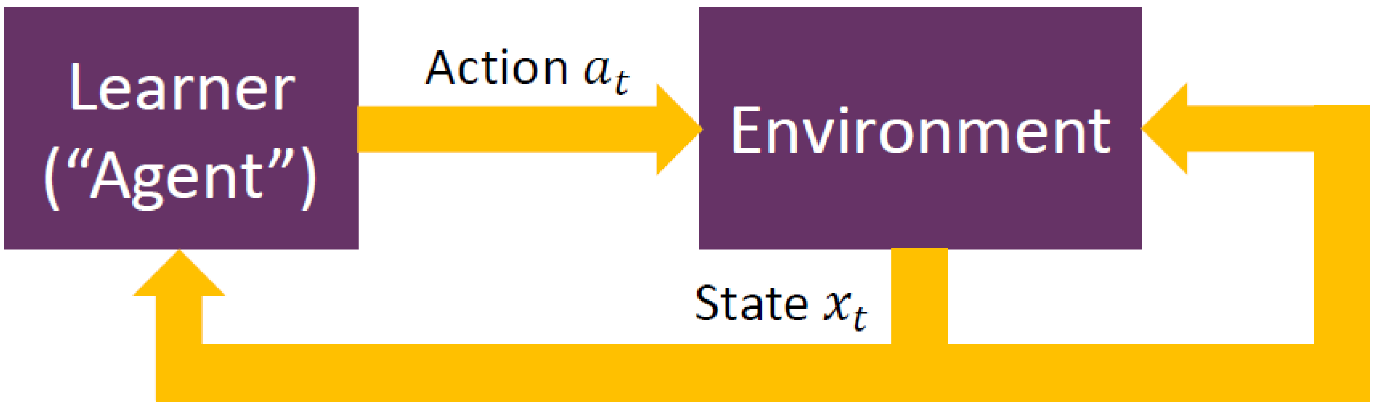
A Markov Decision Process (MDP) is characterized by (X, A, P, r)

- X: a set of states
- A: a set of actions, possibly different in each state
- P: $X \times A \times X \rightarrow [0,1]$: a transition function with $P(\cdot\,|x, a)$ being the distribution of the next state given previous state x and action a:

$$\mathbf{P}[x_{t+1} = x' | x_t = x, a_t = a] = P(x'|x, a)$$

- r: $X \times A \rightarrow [0,1]$: a reward function

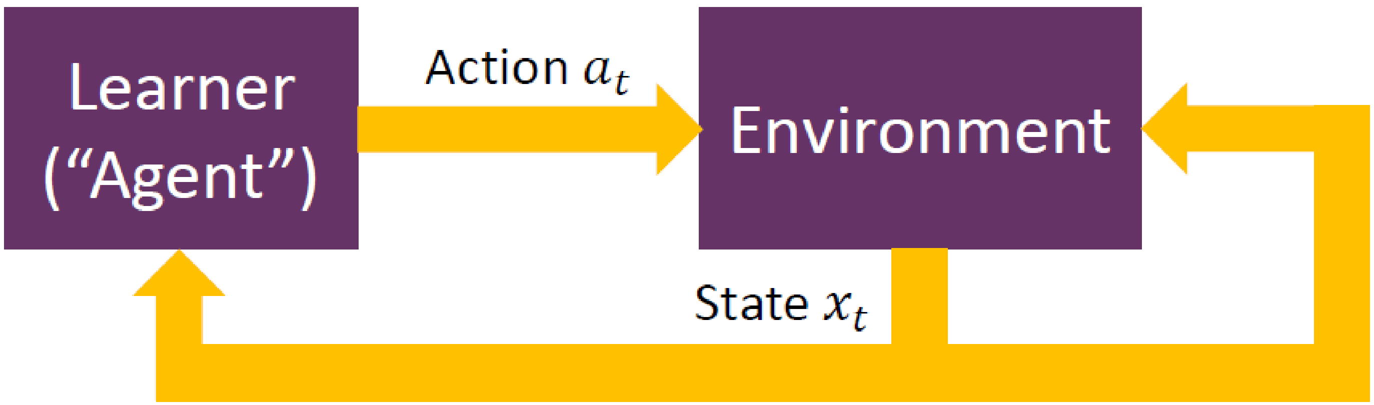
MARKOV DECISION PROCESSES (MDPs)



A Markov Decision Process (MDP) is characterized by (X, A, P, r)
Interaction in an MDP: in each round $t = 1,2, \ldots$

- Agent observes state x_t and selects action a_t
- Environment moves to state $x_{t+1} \sim P(\cdot \mid x_t, a_t)$
- Agent receives reward r_t such that $\mathbf{E}[r_t | x_t, a_t] = r(x_t, a_t)$

MARKOV DECISION PROCESSES (MDPs)



A Markov Decision Process (MDP) is characterized by (X, A, P, r)
Interaction in an MDP: in each round $t = 1,2, \ldots$

- Agent observes state x_t and selects action a_t
- Environment moves to state $x_{t+1} \sim P(\cdot \,|x_t, a_t)$
- Agent receives reward r_t such that $\mathbf{E}[r_t|x_t, a_t] = r(x_t, a_t)$

GOAL:
maximize "total rewards"!

NOTIONS OF "TOTAL REWARD"

Episodic MDPs:

- There is a terminal state x^*
- GOAL: maximize total reward until final round T when x^* is reached:

$$R^* = \mathbf{E}[\sum_{t=0}^{T} r_t]$$

NOTIONS OF “TOTAL REWARD”

Episodic MDPs:

- There is a terminal state x^*
- GOAL: maximize total reward until final round T when x^* is reached:

$$R^* = \mathbf{E}[\sum_{t=0}^{T} r_t]$$

Discounted MDPs:

- No terminal state
- Discount factor $\gamma \in (0,1)$
- GOAL: maximize total discounted reward

$$R_\gamma = \mathbf{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$$

NOTIONS OF "TOTAL REWARD"

+ other notions:
- long-term average reward
- total reward up to fixed horizon
- …

Episodic MDPs:

- There is a terminal state x^*
- GOAL: maximize total reward until final round T when x^* is reached:

$$R^* = \mathbf{E}[\sum_{t=0}^{T} r_t]$$

Discounted MDPs:

- No terminal state
- Discount factor $\gamma \in (0,1)$
- GOAL: maximize total discounted reward

$$R_\gamma = \mathbf{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$$

NOTIONS OF "TOTAL REWARD"

+ other notions:
- long-term average reward (part 2?)
- total reward up to fixed horizon
- …

Episodic MDPs:
- There is a terminal state x^*
- GOAL: maximize total reward until final round T when x^* is reached:

$$R^* = \mathbf{E}[\textstyle\sum_{t=0}^{T} r_t]$$

Discounted MDPs:
- No terminal state
- Discount factor $\gamma \in (0,1)$
- GOAL: maximize total discounted reward

$$R_\gamma = \mathbf{E}[\textstyle\sum_{t=0}^{\infty} \gamma^t r_t]$$

+ we will assume that X and A are finite

POLICIES AND TRAJECTORY DISTRIBUTIONS

Policy: mapping from histories to actions

$$\pi: x_1, a_1, x_2, a_2, \ldots, x_t \mapsto a_t$$

POLICIES AND TRAJECTORY DISTRIBUTIONS

Policy: mapping from histories to actions

$$\pi: x_1, a_1, x_2, a_2, \ldots, x_t \mapsto a_t$$

Stationary policy: mapping from states to actions
(no dependence on history or t)

$$\pi: x \mapsto a$$

POLICIES AND TRAJECTORY DISTRIBUTIONS

Policy: mapping from histories to actions

$$\pi: x_1, a_1, x_2, a_2, \dots, x_t \mapsto a_t$$

Stationary policy: mapping from states to actions
(no dependence on history or t)

$$\pi: x \mapsto a$$

Let $\tau = (x_1, a_1, x_2, a_2, \dots)$ be a trajectory generated by running π in the MDP $\tau \sim (\pi, P)$:

- $a_t = \pi(x_t, a_{t-1}, x_{t-1}, \dots, x_1)$
- $x_{t+1} \sim P(\cdot \mid x_t, a_t)$

POLICIES AND TRAJECTORY DISTRIBUTIONS

Policy: mapping from histories to actions

$$\pi: x_1, a_1, x_2, a_2, \dots, x_t \mapsto a_t$$

Stationary policy: mapping from states to actions
(no dependence on history or t)

$$\pi: x \mapsto a$$

Let $\tau = (x_1, a_1, x_2, a_2, \dots)$ be a trajectory generated by running π in the MDP $\tau \sim (\pi, P)$:

- $a_t = \pi(x_t, a_{t-1}, x_{t-1}, \dots, x_1)$
- $x_{t+1} \sim P(\cdot \mid x_t, a_t)$

Expectation under this distribution: $\mathbf{E}_\pi[\cdot]$

DEFINING OPTIMALITY

Optimal policy π^*: a policy that maximizes

$$\mathbf{E}_\pi\left[R_\gamma\right] = \mathbf{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$$

DEFINING OPTIMALITY

Optimal policy π^*: a policy that maximizes

$$\mathbf{E}_\pi[R_\gamma] = \mathbf{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$$

Theorem

There exists a deterministic optimal policy π^* such that

$$\pi^*(x_1, a_1, \dots, x_t) = \pi^*(x_t)$$

DEFINING OPTIMALITY

Optimal policy π^*: a policy that maximizes

$$\mathbf{E}_\pi[R_\gamma] = \mathbf{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$$

Theorem

There exists a deterministic optimal policy π^* such that

$$\pi^*(x_1, a_1, \dots, x_t) = \pi^*(x_t)$$

Consequence: it's enough to study stationary policies

$$\pi: x \mapsto a$$

DEFINING OPTIMALITY

Theorem

There exists a deterministic optimal policy π^* such that

$$\pi^*(x_1, a_1, \dots, x_t) = \pi^*(x_t)$$

Consequence: it's enough to study stationary policies

$$\pi: x \mapsto a$$

Intuitive "proof": Future transitions $x_{t+1} \sim P(\cdot\,|x_t, a_t)$ do not depend on the previous states $x_1, x_2, \dots$

DEFINING OPTIMALITY

Theorem

There exists a deterministic optimal policy π^* such that

$$\pi^*(x_1, a_1, \ldots, x_t) = \pi^*(x_t)$$

Consequence: it's enough to study stationary policies

$$\pi: x \mapsto a$$

Intuitive "proof": Future transitions $x_{t+1} \sim P(\cdot \, | x_t, a_t)$ do not depend on the previous states $x_1, x_2, \ldots$

=“Markov property”

THIS SHORT COURSE: A PRIMAL-DUAL VIEW

part 1

- Markov decision processes
  - Value functions and optimal policies
- Primal view: Dynamic programming
  - Policy evaluation, value and policy iteration
  - Value-function-based methods
    - Temporal differences, Q-learning, LSTD, deep Q networks,…

part 2

- Dual view: Linear programming
  - LP duality in MDPs
  - Direct policy optimization methods
    - Policy gradients, REPS, TRPO,…

VALUE FUNCTIONS

Value function: evaluates policy π starting from state x:

$$V^{\pi}(x) = \mathbf{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t \,|x_0 = x\right]$$

VALUE FUNCTIONS

Value function: evaluates policy π starting from state x:

$$V^{\pi}(x) = \mathbf{E}_{\pi}[\sum_{t=0}^{\infty} \gamma^t r_t \,| x_0 = x]$$

Action-value function: evaluates policy π starting from state x and action a:

$$Q^{\pi}(x, a) = \mathbf{E}_{\pi}[\sum_{t=0}^{\infty} \gamma^t r_t \,| x_0 = x, a_0 = a]$$

VALUE FUNCTIONS

Value function: evaluates policy π starting from state x:

$$V^{\pi}(x) = \mathbf{E}_{\pi}[\sum_{t=0}^{\infty} \gamma^t r_t \,|x_0 = x]$$

Action-value function: evaluates policy π starting from state x and action a:

$$Q^{\pi}(x, a) = \mathbf{E}_{\pi}[\sum_{t=0}^{\infty} \gamma^t r_t \,|x_0 = x, a_0 = a]$$

"Optimal policy π^* $= \arg\max_{\pi} V^{\pi}(x_0)$"

VALUE FUNCTIONS AND THE OPTIMAL POLICY

Theorem

There exists a policy π^* that satisfies

$$V^{\pi^*}(x) = \max_{\pi} V^{\pi}(x) \quad (\forall x)$$

VALUE FUNCTIONS AND THE OPTIMAL POLICY

Theorem

There exists a policy π^* that satisfies

$$V^{\pi^*}(x) = \max_{\pi} V^{\pi}(x) \quad (\forall x)$$

VALUE FUNCTIONS AND THE OPTIMAL POLICY

Theorem

There exists a policy π^* that satisfies

$$V^{\pi^*}(x) = \max_{\pi} V^{\pi}(x) \quad (\forall x)$$

Optimal policy: a policy π^* that satisfies the above

VALUE FUNCTIONS AND THE OPTIMAL POLICY

Theorem

There exists a policy π^* that satisfies

$$V^{\pi^*}(x) = \max_{\pi} V^{\pi}(x) \quad (\forall x)$$

Optimal policy: a policy π^* that satisfies the above

The optimal value function:

$$V^* = V^{\pi^*}$$

THE BELLMAN EQUATIONS

Theorem

The value function of a stationary policy π satisfies the system of equations $(\forall x \in X)$

$$V^{\pi}(x) = r\big(x, \pi(x)\big) + \gamma \sum_y P(y|x, \pi(x))\, V^{\pi}(y)$$

THE BELLMAN EQUATIONS

Theorem

The value function of a stationary policy π satisfies the system of equations $(\forall x \in X)$

$$V^{\pi}(x) = r\big(x, \pi(x)\big) + \gamma \sum_y P(y|x, \pi(x))\, V^{\pi}(y)$$

Proof:

$$V^{\pi}(x) = \mathbf{E}_{\pi}[\sum_{t=0}^{\infty} \gamma^t r(x_t, a_t)\,|x_0 = x]$$

THE BELLMAN EQUATIONS

Theorem

The value function of a stationary policy π satisfies the system of equations ($\forall x \in X$)

$$V^{\pi}(x) = r\big(x, \pi(x)\big) + \gamma \sum_y P(y|x, \pi(x))\, V^{\pi}(y)$$

Proof:

$$V^{\pi}(x) = \mathbf{E}_{\pi}[\sum_{t=0}^{\infty} \gamma^t r(x_t, a_t)\,|x_0 = x]$$

$$= r\big(x, \pi(x)\big) + \mathbf{E}_{\pi}[\sum_{t=1}^{\infty} \gamma^t r(x_t, a_t)\,|x_0 = x]$$

THE BELLMAN EQUATIONS

Theorem
The value function of a stationary policy π satisfies the system of equations ($\forall x \in X$)

$$V^\pi(x) = r\big(x, \pi(x)\big) + \gamma \sum_y P(y|x, \pi(x))\, V^\pi(y)$$

Proof:

$$V^\pi(x) = \mathbf{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(x_t, a_t)\,|x_0 = x]$$

$$= r\big(x, \pi(x)\big) + \mathbf{E}_\pi[\sum_{t=1}^{\infty} \gamma^t r(x_t, a_t)\,|x_0 = x]$$

$$= r\big(x, \pi(x)\big) + \gamma \sum_y P(y|x, \pi(x))\, \mathbf{E}_\pi[\sum_{t=1}^{\infty} \gamma^{t-1} r(x_t, a_t)\,|x_1 = y]$$

THE BELLMAN EQUATIONS

Theorem

The value function of a stationary policy π satisfies the system of equations ($\forall x \in X$)

$$V^\pi(x) = r\big(x, \pi(x)\big) + \gamma \sum_y P(y|x, \pi(x))\, V^\pi(y)$$

Proof:

$$V^\pi(x) = \mathbf{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(x_t, a_t)\,|x_0 = x]$$

$$= r\big(x, \pi(x)\big) + \mathbf{E}_\pi[\sum_{t=1}^{\infty} \gamma^t r(x_t, a_t)\,|x_0 = x]$$

$$= r\big(x, \pi(x)\big) + \gamma \sum_y P(y|x, \pi(x))\, \mathbf{E}_\pi[\sum_{t=1}^{\infty} \gamma^{t-1} r(x_t, a_t)\,|x_1 = y]$$

$$= r\big(x, \pi(x)\big) + \gamma \sum_y P(y|x, \pi(x))\, V^\pi(y) \qquad \blacksquare$$

THE BELLMAN OPTIMALITY EQUATIONS

Theorem

The optimal value function satisfies the system of equations

$$V^*(x) = \max_a \left\{ r(x, a) + \gamma \sum_y P(y|x, a)\, V^*(y) \right\}$$

THE BELLMAN OPTIMALITY EQUATIONS

Theorem

The optimal value function satisfies the system of equations

$$V^*(x) = \max_a \left\{ r(x,a) + \gamma \sum_y P(y|x,a)\, V^*(y) \right\}$$

Theorem

An optimal policy π^* satisfies

$$\pi^*(x) \in \arg\max_a \left\{ r(x,a) + \gamma \sum_y P(y|x,a)\, V^*(y) \right\}$$

OPTIMAL ACTION-VALUE FUNCTIONS

Theorem

The optimal action-value function satisfies

$$Q^*(x,a) = r(x,a) + \gamma \sum_y P(y|x,a) \max_b Q^*(y,b)$$

OPTIMAL ACTION-VALUE FUNCTIONS

Theorem

The optimal action-value function satisfies

$$Q^*(x,a) = r(x,a) + \gamma \sum_y P(y|x,a) \max_b Q^*(y,b)$$

Theorem

An optimal policy π^* satisfies

$$\pi^*(x) \in \arg\max_a Q^*(x,a)$$

OPTIMAL ACTION-VALUE FUNCTIONS

Theorem

The optimal action-value function satisfies

$$Q^*(x,a) = r(x,a) + \gamma \sum_y P(y|x,a) \max_b Q^*(y,b)$$

Theorem

An optimal policy π^* satisfies

$$\pi^*(x) \in \arg\max_a Q^*(x,a)$$

= greedy with respect to Q^*

SHORT SUMMARY SO FAR

So far, we have characterized

- The value functions of a given policy
- The optimal policy through value functions
- The optimal value functions
- The optimal policy through the optimal value functions

SHORT SUMMARY SO FAR

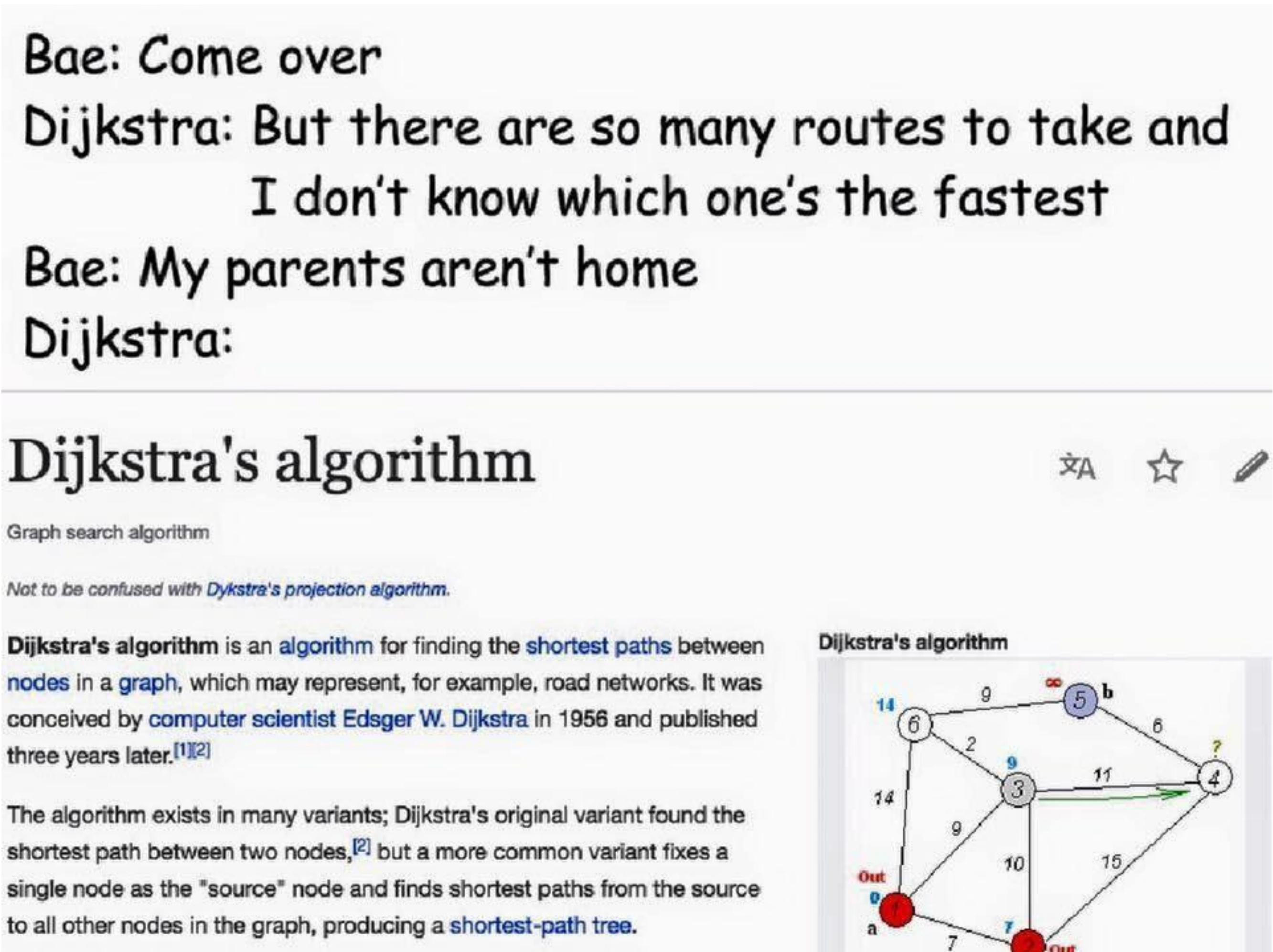
So far, we have characterized

- The value functions of a given policy
- The optimal policy through value functions
- The optimal value functions
- The optimal policy through the optimal value functions

BUT HOW DO WE FIND THE OPTIMAL VALUE FUNCTION??

… also, is there a way to clean up this mess? See part 2!

EASY ANSWER FOR FINITE-HORIZON PROBLEMS



THIS SHORT COURSE:
A PRIMAL-DUAL VIEW

part 1

- Markov decision processes
  - Value functions and optimal policies
- Primal view: Dynamic programming
  - Policy evaluation, value and policy iteration
  - Value-function-based methods
    - Temporal differences, Q-learning, LSTD, deep Q networks,…

part 2

- Dual view: Linear programming
  - LP duality in MDPs
  - Direct policy optimization methods
    - Policy gradients, REPS, TRPO,…

DYNAMIC PROGRAMMING

Dynamic programming

=

computing value functions through repeated use of the "Bellman operators"

THE BELLMAN OPERATOR

Bellman operator T^{π}:
maps a function $V \in \mathbb{R}^X$ to another function $T^{\pi}V \in \mathbb{R}^X$:

$$(T^{\pi}V)(x) = r\big(x, \pi(x)\big) + \gamma \sum_y P(y|x, \pi(x))V(y)$$

THE BELLMAN OPERATOR

Bellman operator T^{π}:
maps a function $V \in \mathbb{R}^X$ to another function $T^{\pi}V \in$

$$(T^{\pi}V)(x) = r\big(x, \pi(x)\big) + \gamma \sum_y P(y|x, \pi(x))V(y)$$

r.h.s. of BE

THE BELLMAN OPERATOR

Bellman operator T^{π}:
maps a function $V \in \mathbb{R}^X$ to another function $T^{\pi}V \in$

r.h.s. of BE

$$(T^{\pi}V)(x) = r\big(x, \pi(x)\big) + \gamma \sum_y P(y|x, \pi(x))V(y)$$

The Bellman Equations:

$$V^{\pi}(x) = r\big(x, \pi(x)\big) + \gamma \sum_y P(y|x, \pi(x))\, V^{\pi}(y)$$

THE BELLMAN OPERATOR

Bellman operator T^{π}:
maps a function $V \in \mathbb{R}^X$ to another function $T^{\pi}V \in$

r.h.s. of BE

$$(T^{\pi}V)(x) = r\big(x, \pi(x)\big) + \gamma \sum_y P(y|x, \pi(x))V(y)$$

The Bellman Equations:

$$V^{\pi} = T^{\pi}V^{\pi}$$

THE BELLMAN OPERATOR

Bellman operator T^π:
maps a function $V \in \mathbb{R}^X$ to another function $T^\pi V \in$

$$(T^\pi V)(x) = r\big(x, \pi(x)\big) + \gamma \sum_y P(y|x, \pi(x))V(y)$$

r.h.s. of BE

V^π is the fixed point of T^π

The Bellman Equations:

$$V^\pi = T^\pi V^\pi$$

POLICY EVALUATION USING THE BELLMAN OPERATOR

Idea: repeated application of T^{π} on any function V_0 should converge to V^{π}…

POLICY EVALUATION USING THE BELLMAN OPERATOR

Idea: repeated application of T^{π} on any function V_0 should converge to V^{π}…

…and it works!!

Power iteration

Input: arbitrary $V_0 : X \to \mathrm{R}$ and π

For $k = 1,2, \ldots,$ compute

$$V_{k+1} = T^{\pi} V_k$$

POLICY EVALUATION USING THE BELLMAN OPERATOR

Idea: repeated application of T^{π} on any function V_0 should converge to V^{π}…

…and it works!!

Power iteration

Input: arbitrary $V_0: X \to \mathrm{R}$ and π

For $k = 1,2, \dots,$ compute

$$V_{k+1} = T^{\pi} V_k$$

Theorem: $\lim_{k \to \infty} V_k = V^{\pi}$

CONVERGENCE OF POWER ITERATION: PROOF SKETCH

- Power iteration can be written as the linear recursion
  $V_{k+1} = r + \gamma P^{\pi} V_k$

CONVERGENCE OF POWER ITERATION: PROOF SKETCH

- Power iteration can be written as the linear recursion
$$V_{k+1} = r + \gamma P^{\pi} V_k = r + \gamma P^{\pi}(r + \gamma P^{\pi} V_{k-1})$$

CONVERGENCE OF POWER ITERATION: PROOF SKETCH

- Power iteration can be written as the linear recursion

$$\begin{aligned} V_{k+1} &= r + \gamma P^{\pi} V_k = r + \gamma P^{\pi}(r + \gamma P^{\pi} V_{k-1}) \\ &= r + \gamma P^{\pi} r + (\gamma P^{\pi})^2 r + \cdots + (\gamma P^{\pi})^k r \end{aligned}$$

CONVERGENCE OF POWER ITERATION: PROOF SKETCH

- Power iteration can be written as the linear recursion

$$
\begin{aligned}
V_{k+1} &= r + \gamma P^{\pi} V_k = r + \gamma P^{\pi}(r + \gamma P^{\pi} V_{k-1}) \\
&= r + \gamma P^{\pi} r + (\gamma P^{\pi})^2 r + \cdots + (\gamma P^{\pi})^k r \\
&= \sum_{t=0}^{k} (\gamma P^{\pi})^k r
\end{aligned}
$$

CONVERGENCE OF POWER ITERATION: PROOF SKETCH

- Power iteration can be written as the linear recursion

$$\begin{aligned} V_{k+1} &= r + \gamma P^{\pi} V_k = r + \gamma P^{\pi}(r + \gamma P^{\pi} V_{k-1}) \\ &= r + \gamma P^{\pi} r + (\gamma P^{\pi})^2 r + \cdots + (\gamma P^{\pi})^k r \\ &= \sum_{t=0}^{k} (\gamma P^{\pi})^k r \\ &= (I - \gamma P^{\pi})^{-1} \cdot \left(I - (\gamma P^{\pi})^k\right) r \end{aligned}$$

Geometric sum!
(von Neumann series)

CONVERGENCE OF POWER ITERATION: PROOF SKETCH

- Power iteration can be written as the linear recursion

$$
\begin{aligned}
V_{k+1} &= r + \gamma P^{\pi} V_k = r + \gamma P^{\pi}(r + \gamma P^{\pi} V_{k-1}) \\
&= r + \gamma P^{\pi} r + (\gamma P^{\pi})^2 r + \cdots + (\gamma P^{\pi})^k r \\
&= \sum_{t=0}^{k} (\gamma P^{\pi})^k r \\
&= (I - \gamma P^{\pi})^{-1} \cdot \big(I - (\gamma P^{\pi})^k\big) r \\
&\to (I - \gamma P^{\pi})^{-1} r \qquad\qquad (k \to \infty)
\end{aligned}
$$

Geometric sum!
(von Neumann series)

$(\gamma P^{\pi})^k \to 0$

CONVERGENCE OF POWER ITERATION: PROOF SKETCH

- Power iteration can be written as the linear recursion

$$\begin{aligned} V_{k+1} &= r + \gamma P^\pi V_k = r + \gamma P^\pi (r + \gamma P^\pi V_{k-1}) \\ &= r + \gamma P^\pi r + (\gamma P^\pi)^2 r + \cdots + (\gamma P^\pi)^k r \\ &= \sum_{t=0}^{k} (\gamma P^\pi)^k r \\ &= (I - \gamma P^\pi)^{-1} \cdot \big(I - (\gamma P^\pi)^k\big) r \\ &\to (I - \gamma P^\pi)^{-1} r \qquad\qquad (k \to \infty) \end{aligned}$$

Geometric sum! (von Neumann series)

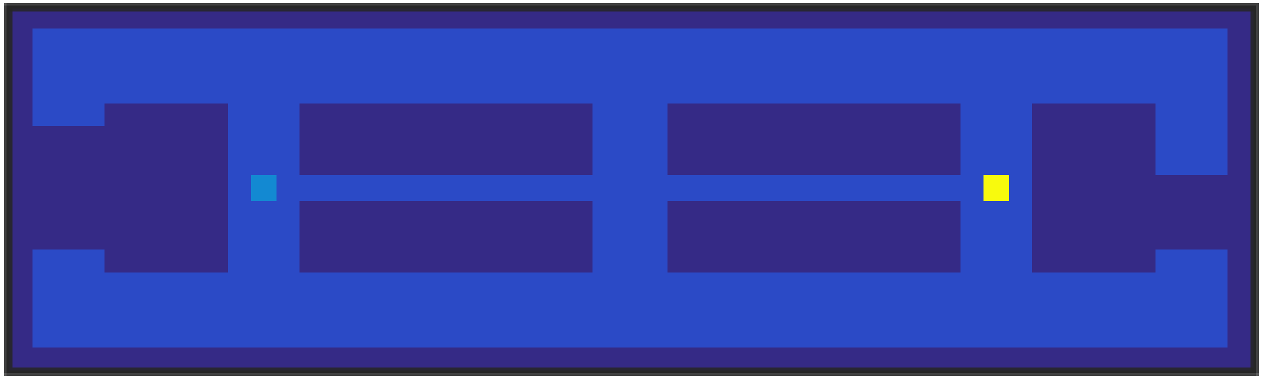
$(\gamma P^\pi)^k \to 0$

- The value function V^π satisfies

$$V^\pi = r + \gamma P^\pi V^\pi \iff V^\pi = (I - \gamma P^\pi)^{-1} r$$

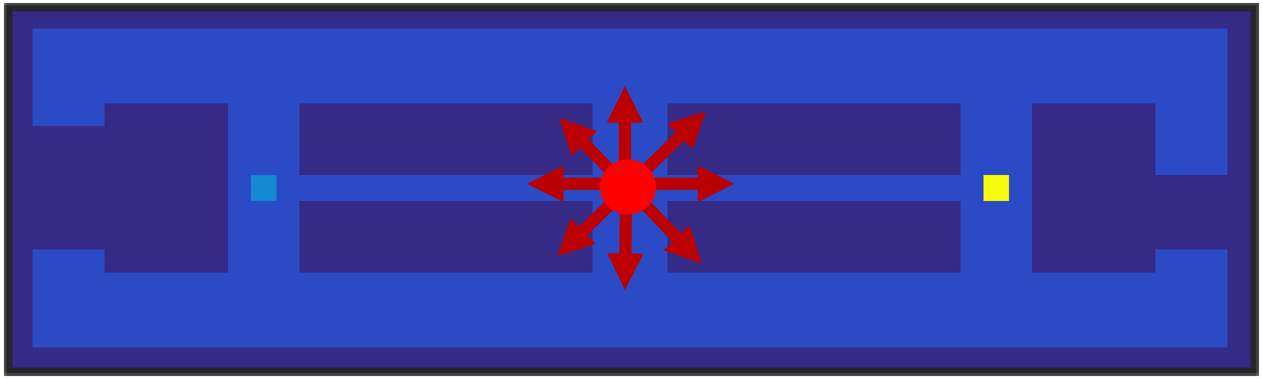
POWER ITERATION IN ACTION

Gridworld MDP



POWER ITERATION IN ACTION

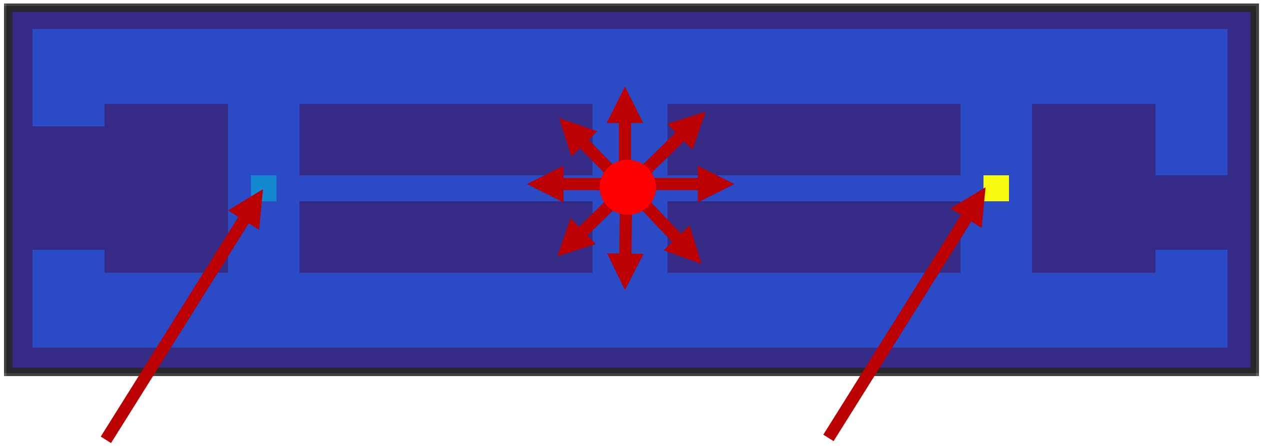
Gridworld MDP



- State: location on the grid
- Actions: try to move in one of 8 directions or stay put
- Transition probabilities:
  - move successfully w.p. $p = 0.5$
  - otherwise move in neighboring direction

POWER ITERATION IN ACTION

Gridworld MDP

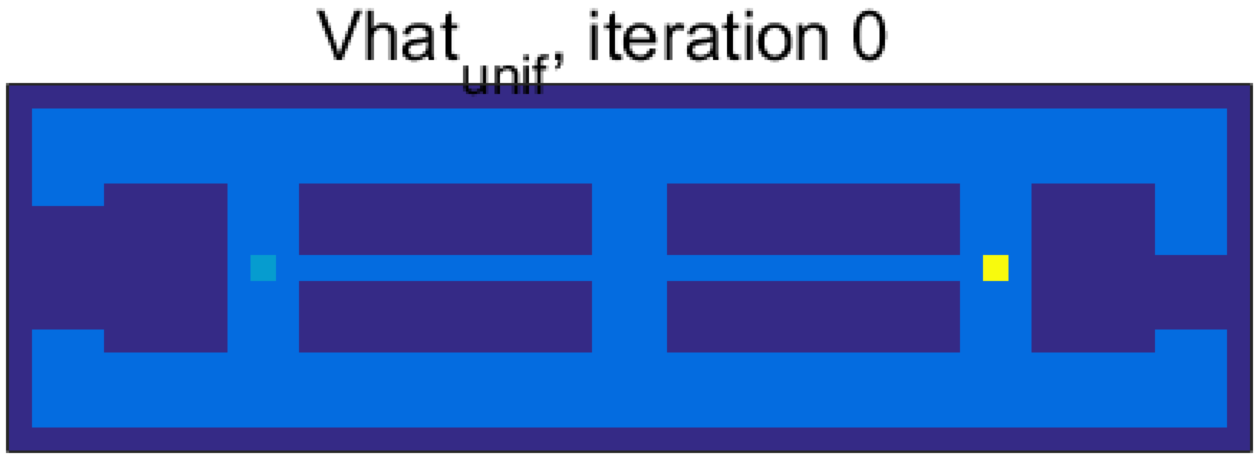


Reward: +100

Reward: +500

- State: location on the grid
- Actions: try to move in one of 8 directions or stay put
- Transition probabilities:
    - move successfully w.p. $p = 0.5$
    - otherwise move in neighboring direction

POWER ITERATION IN ACTION

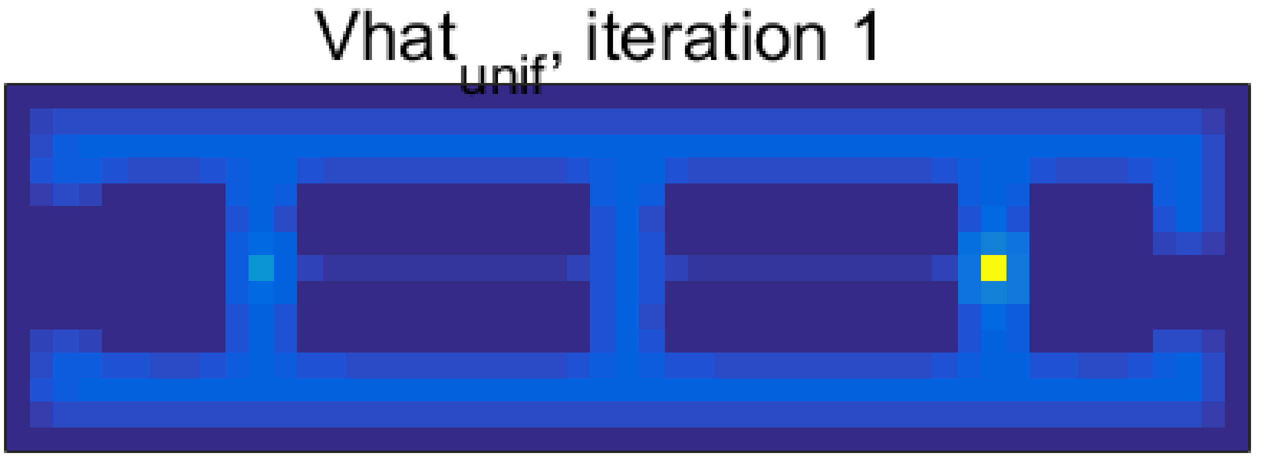


Uniform policy:

$$\pi(a|x) = \frac{1}{9}$$

for all actions $a \in \{1,2,\dots,9\}$

POWER ITERATION IN ACTION

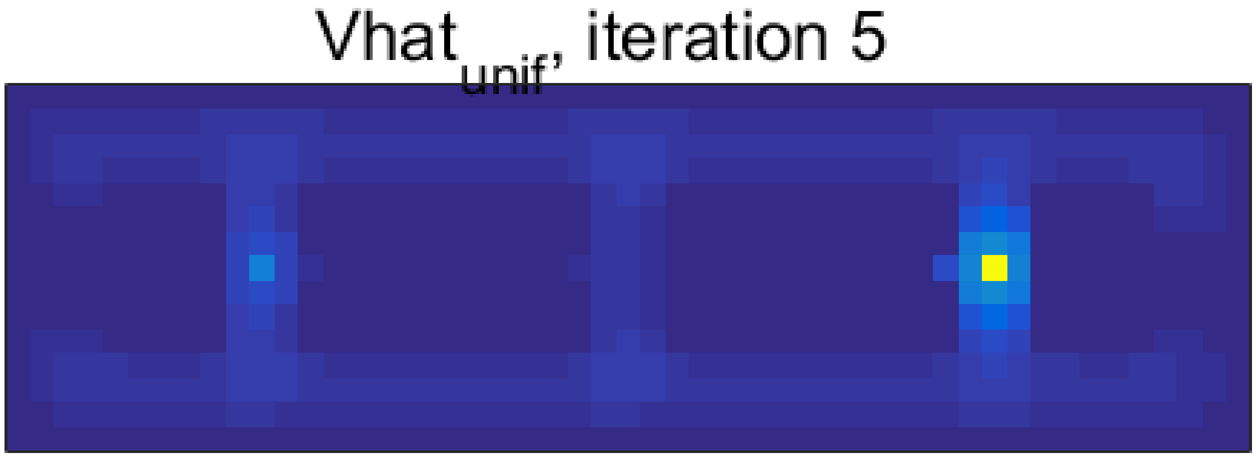


Uniform policy:

$$\pi(a|x) = \frac{1}{9}$$

for all actions $a \in \{1,2,\dots,9\}$

POWER ITERATION IN ACTION

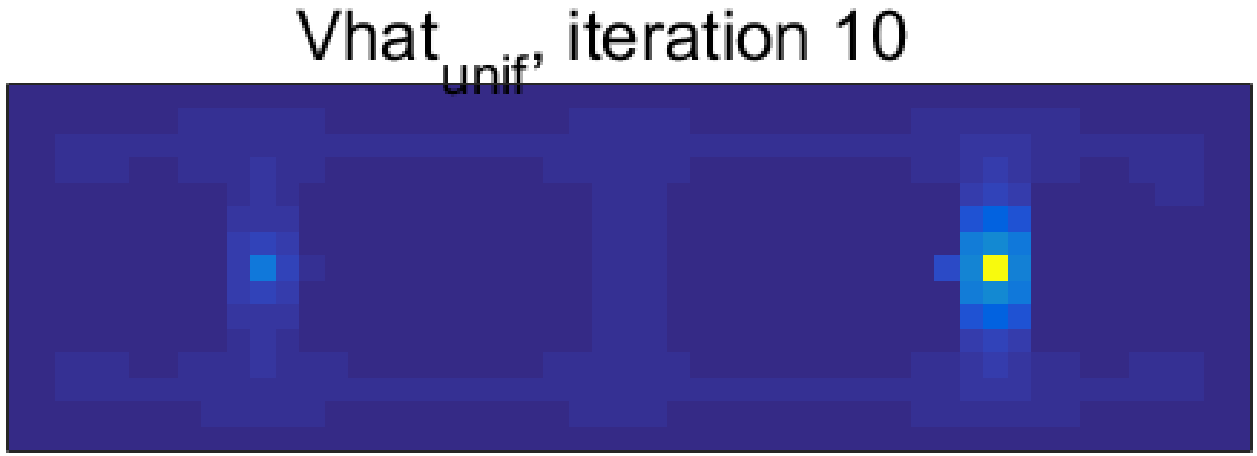


Uniform policy:

$$\pi(a|x) = \frac{1}{9}$$

for all actions $a \in \{1,2,\ldots,9\}$

POWER ITERATION IN ACTION

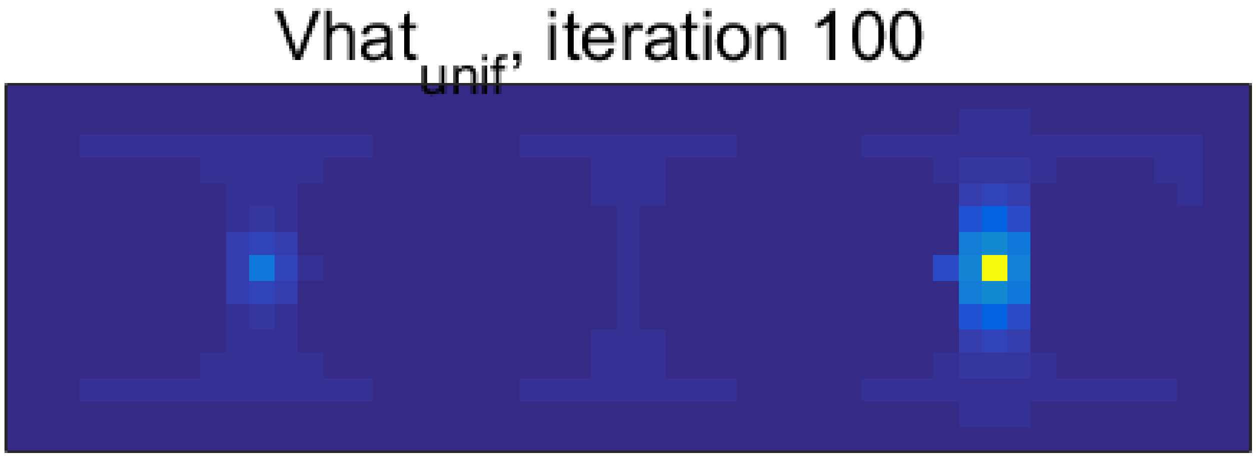


Uniform policy:

$$\pi(a|x) = \frac{1}{9}$$

for all actions $a \in \{1,2,\ldots,9\}$

POWER ITERATION IN ACTION

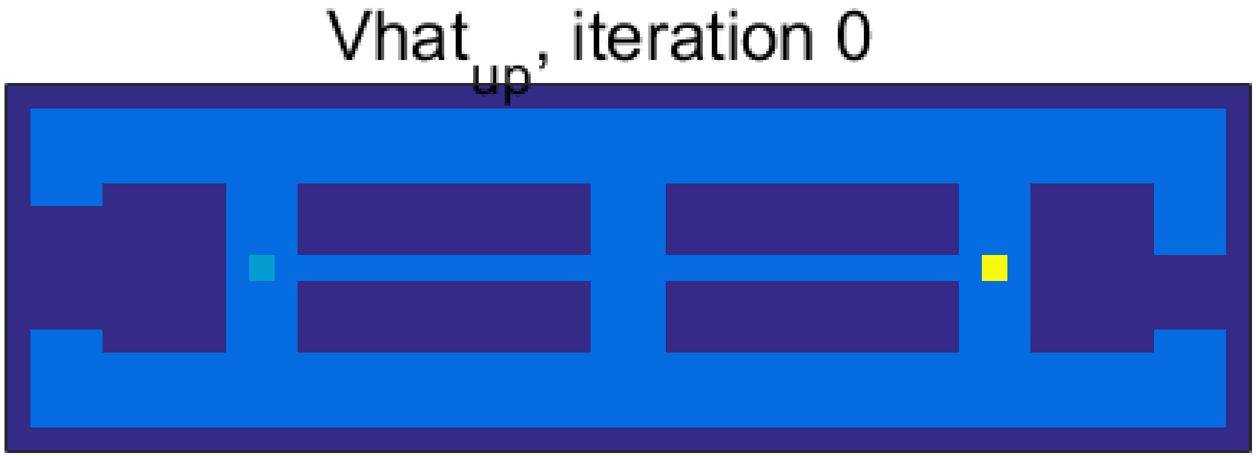


Uniform policy:

$$\pi(a|x) = \frac{1}{9}$$

for all actions $a \in \{1,2,\ldots,9\}$

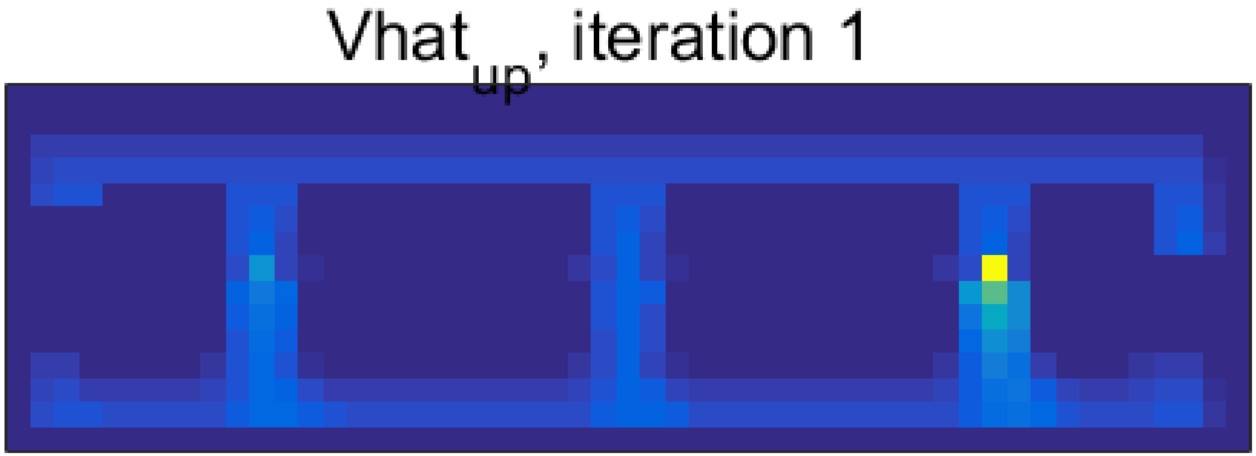
POWER ITERATION IN ACTION



“Upwards” policy:

$\pi(\mathrm{up}|x) = 1$

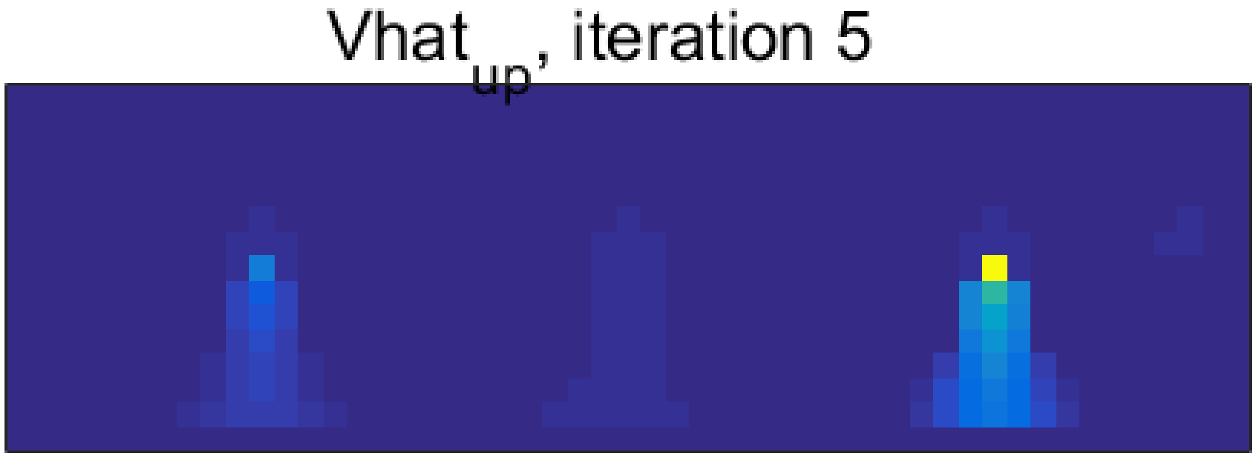
POWER ITERATION IN ACTION



“Upwards” policy:

$\pi(\text{up}|x) = 1$

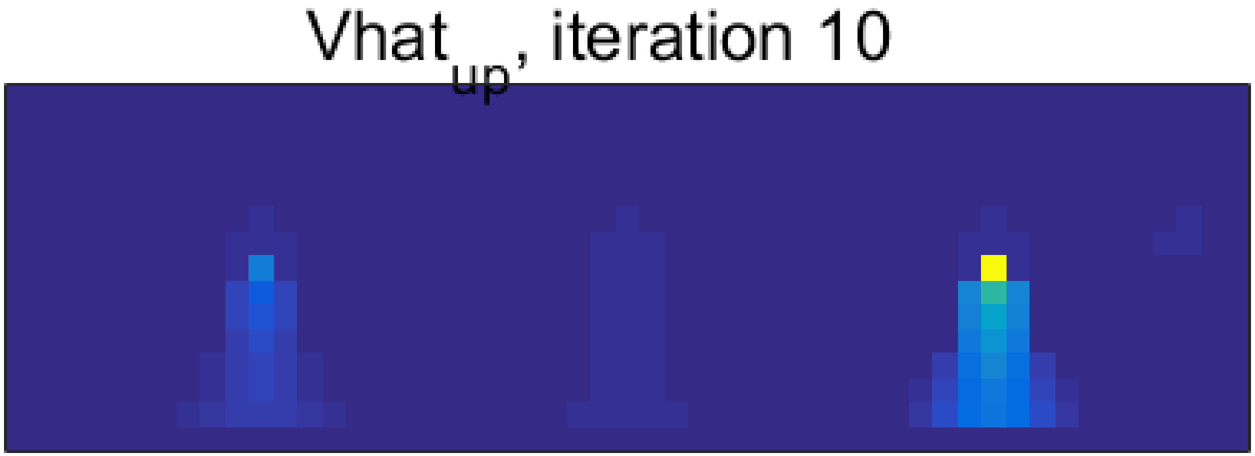
POWER ITERATION IN ACTION



"Upwards" policy:

$\pi(\text{up}|x) = 1$

POWER ITERATION IN ACTION



"Upwards" policy:

$\pi(\text{up}|x) = 1$

THE BELLMAN OPTIMALITY OPERATOR

Bellman optimality operator T^*:
maps a function $V \in \mathbb{R}^X$ to another function $T^*V \in \mathbb{R}^X$:

$$(T^*V)(x) = \max_a \{r(x,a) + \gamma \sum_y P(y|x,a)V(y)\}$$

THE BELLMAN OPTIMALITY OPERATOR

r.h.s. of BOE

Bellman optimality operator T^*:
maps a function $V \in \mathbb{R}^X$ to another function $T^*V \in \mathbb{R}^X$:

$$(T^*V)(x) = \max_a \{r(x,a) + \gamma \sum_y P(y|x,a)V(y)\}$$

THE BELLMAN OPTIMALITY OPERATOR

r.h.s. of BOE

Bellman optimality operator T^*:
maps a function $V \in \mathbb{R}^X$ to another function $T^*V \in \mathbb{R}^X$:

$$(T^*V)(x) = \max_a \{r(x,a) + \gamma \sum_y P(y|x,a) V(y)\}$$

The Bellman Optimality Equations:

$$V^*(x) = \max_a \left\{r(x,a) + \gamma \sum_y P(y|x,a)\, V^*(y)\right\}$$

THE BELLMAN OPTIMALITY OPERATOR

r.h.s. of BOE

Bellman optimality operator T^*:
maps a function $V \in \mathbb{R}^X$ to another function $T^*V \in \mathbb{R}^X$:

$$(T^*V)(x) = \max_a \{r(x,a) + \gamma \sum_y P(y|x,a)V(y)\}$$

V^* is the fixed point of T^*

The Bellman Optimality Equations:

$$V^* = T^*V^*$$

VALUE ITERATION

Idea: repeated application of T^* on any function V_0 should converge to V^*...

...and it works!!

VALUE ITERATION

Idea: repeated application of T^* on any function V_0 should converge to V^*...

...and it works!!

Value iteration

Input: arbitrary function $V_0 : X \to \mathbf{R}$

For $k = 1,2, \dots ,$ compute

$$V_{k+1} = T^* V_k$$

VALUE ITERATION

Idea: repeated application of T^* on any function V_0 should converge to V^*…

…and it works!!

Value iteration

Input: arbitrary function $V_0 : X \to \mathbf{R}$

For $k = 1,2, \ldots,$ compute

$$V_{k+1} = T^* V_k$$

Theorem: $\lim_{k \to \infty} V_k = V^*$

THE CONVERGENCE OF VALUE ITERATION: PROOF SKETCH

Key idea: T^* is a contraction

- for any two functions V and V', we have
$$\|T^*V - T^*V'\|_\infty \leq \gamma\|V - V'\|_\infty$$

THE CONVERGENCE OF VALUE ITERATION: PROOF SKETCH

Key idea: T^* is a contraction

- for any two functions V and V', we have
$$\|T^*V - T^*V'\|_\infty \leq \gamma\|V - V'\|_\infty$$
- repeated application gives
$$\|V_{k+1} - V^*\|_\infty = \|T^*V_k - T^*V^*\|_\infty$$

THE CONVERGENCE OF VALUE ITERATION: PROOF SKETCH

Key idea: T^* is a contraction

- for any two functions V and V', we have
$$\|T^*V - T^*V'\|_\infty \leq \gamma \|V - V'\|_\infty$$
- repeated application gives
$$\|V_{k+1} - V^*\|_\infty = \|T^*V_k - T^*V^*\|_\infty$$
$$\leq \gamma \|V_k - V^*\|_\infty$$

THE CONVERGENCE OF VALUE ITERATION: PROOF SKETCH

Key idea: T^* is a contraction

- for any two functions V and V', we have
$$\|T^*V - T^*V'\|_\infty \leq \gamma\|V - V'\|_\infty$$
- repeated application gives
$$\begin{aligned}\|V_{k+1} - V^*\|_\infty &= \|T^*V_k - T^*V^*\|_\infty \\ &\leq \gamma\|V_k - V^*\|_\infty \\ &\leq \gamma^2\|V_{k-1} - V^*\|_\infty\end{aligned}$$

THE CONVERGENCE OF VALUE ITERATION: PROOF SKETCH

Key idea: T^* is a contraction

- for any two functions V and V', we have
$$\|T^*V - T^*V'\|_\infty \leq \gamma \|V - V'\|_\infty$$
- repeated application gives
$$\begin{aligned} \|V_{k+1} - V^*\|_\infty &= \|T^*V_k - T^*V^*\|_\infty \\ &\leq \gamma \|V_k - V^*\|_\infty \\ &\leq \gamma^2 \|V_{k-1} - V^*\|_\infty \\ &\leq \cdots \leq \gamma^k \|V_0 - V^*\|_\infty \end{aligned}$$

THE CONVERGENCE OF VALUE ITERATION: PROOF SKETCH

Key idea: T^* is a contraction

- for any two functions V and V', we have

$$\|T^*V - T^*V'\|_\infty \leq \gamma\|V - V'\|_\infty$$

- repeated application gives

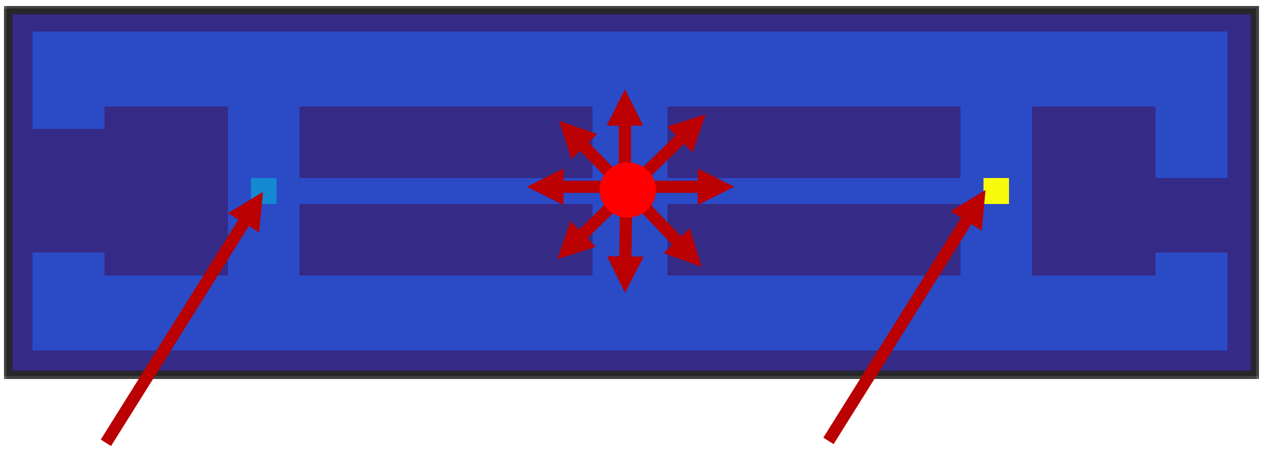
$$\begin{aligned}\|V_{k+1} - V^*\|_\infty &= \|T^*V_k - T^*V^*\|_\infty \\ &\leq \gamma\|V_k - V^*\|_\infty \\ &\leq \gamma^2\|V_{k-1} - V^*\|_\infty \\ &\leq \cdots \leq \gamma^k\|V_0 - V^*\|_\infty\end{aligned}$$

- thus

$$\lim_{k\to\infty}\|V_{k+1} - V^*\|_\infty = 0$$

VALUE ITERATION IN ACTION

Gridworld MDP

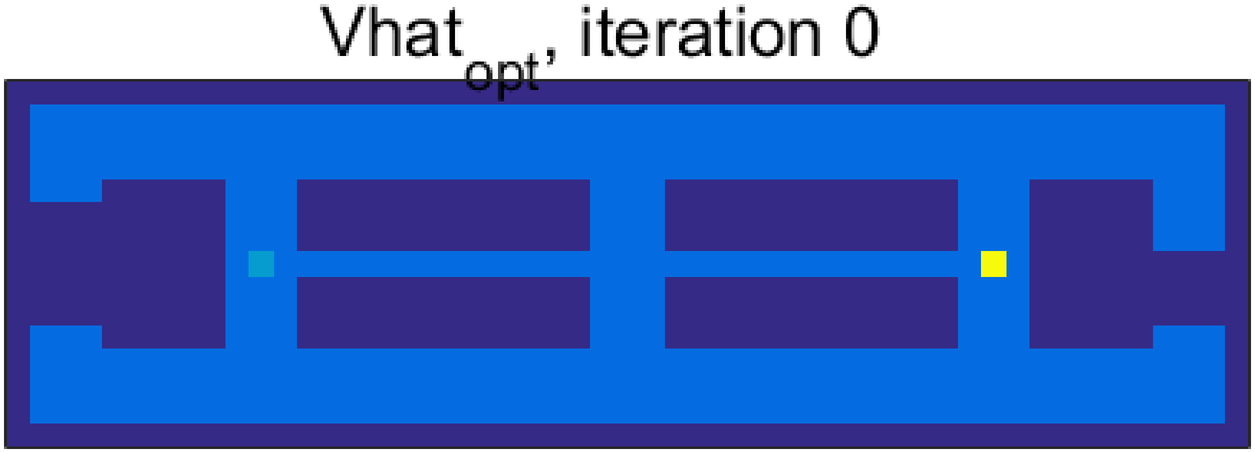


Reward: +100

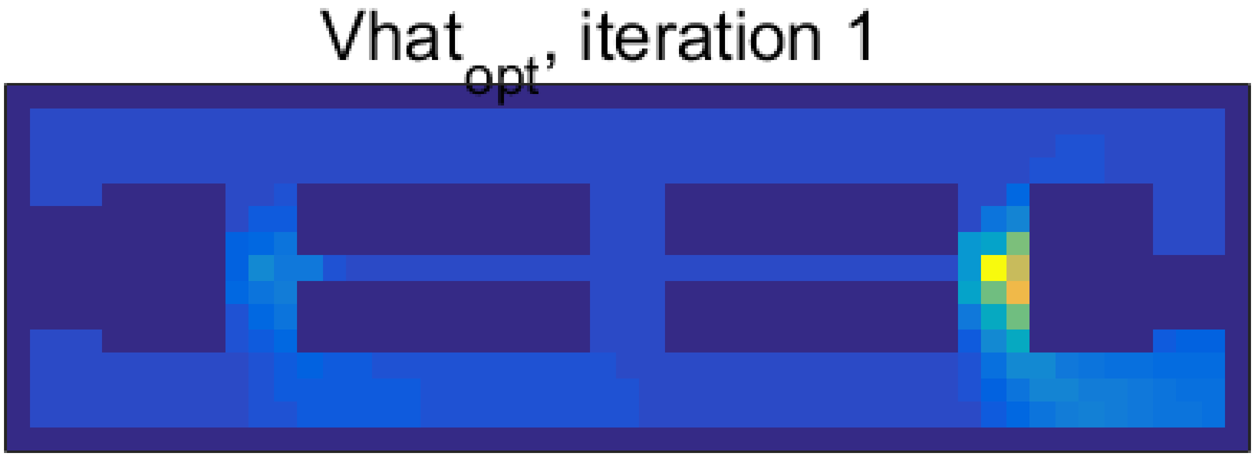
Reward: +500

- **State:** location on the grid
- **Actions:** try to move in one of 8 directions or stay put
- **Transition probabilities:**
    - move successfully w.p. $p = 0.5$
    - otherwise move in neighboring direction

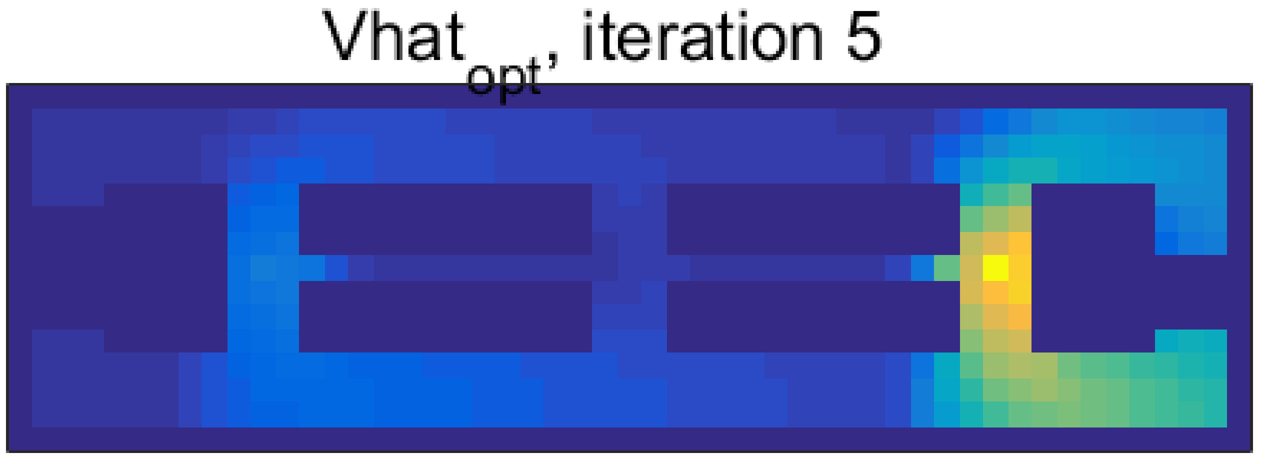
VALUE ITERATION IN ACTION



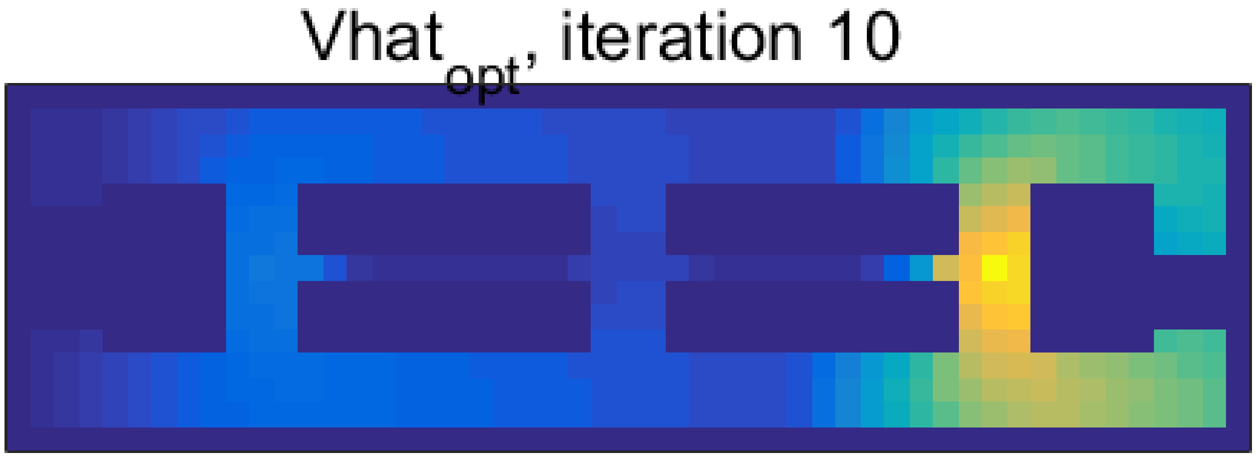
VALUE ITERATION IN ACTION



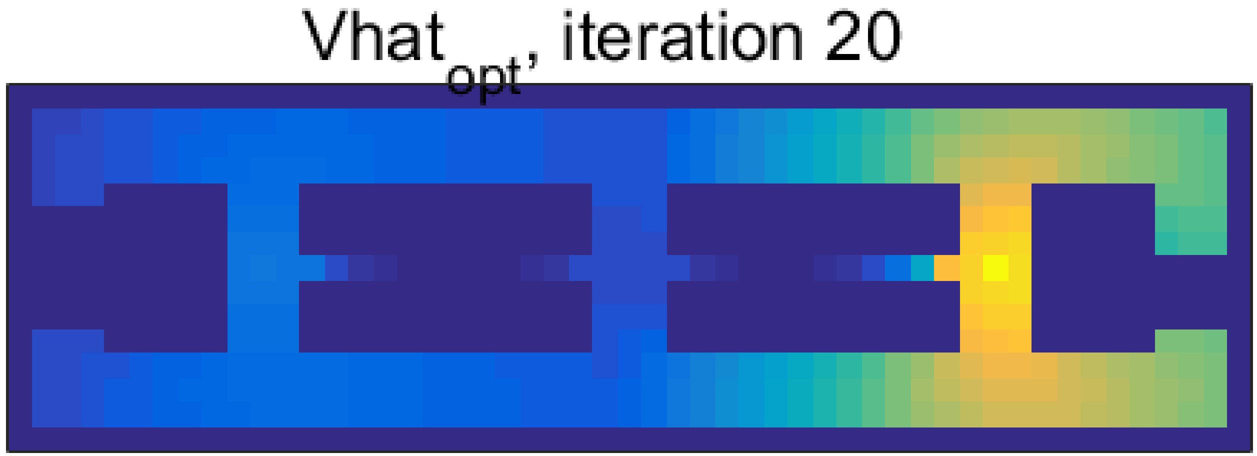
VALUE ITERATION IN ACTION



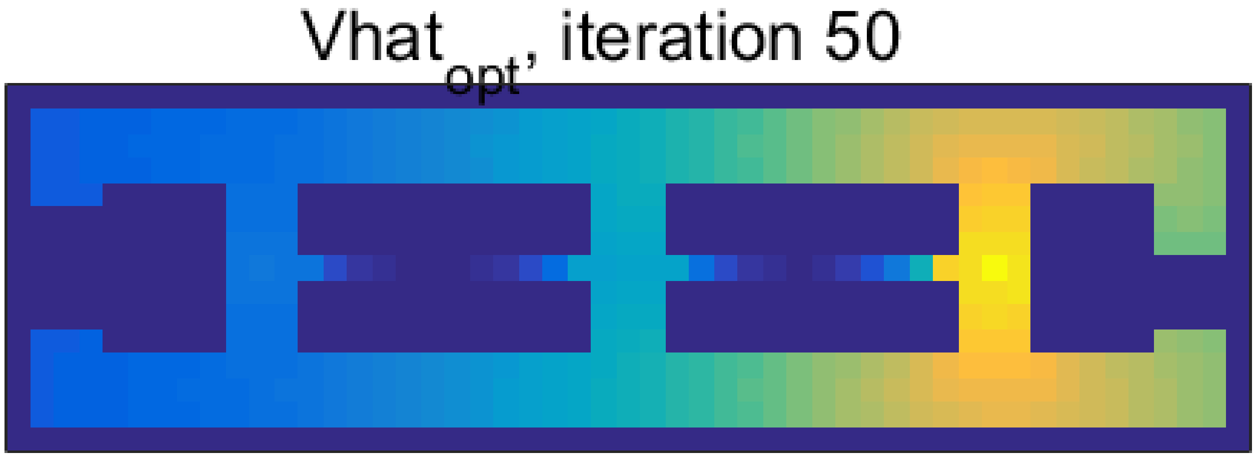
VALUE ITERATION IN ACTION



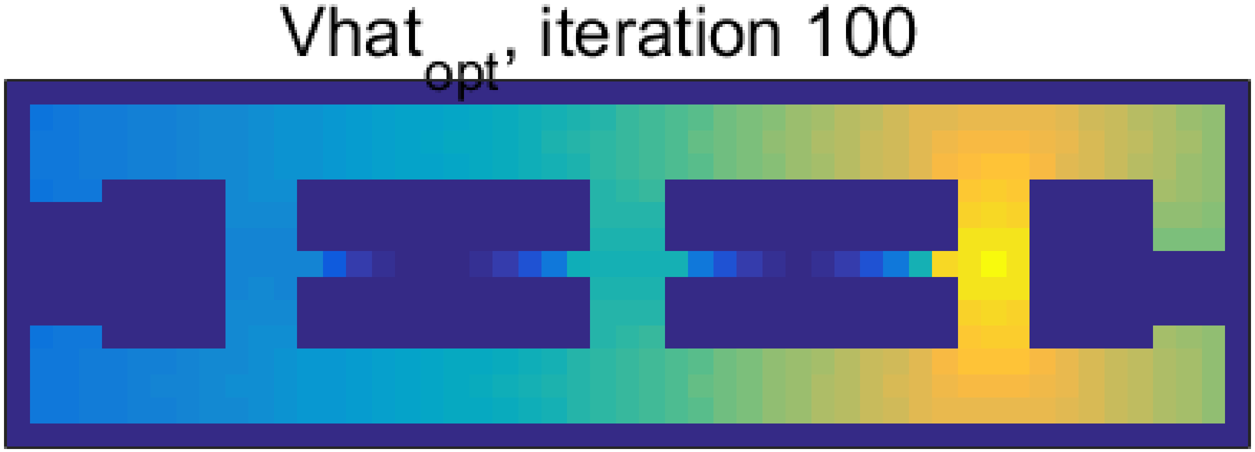
VALUE ITERATION IN ACTION



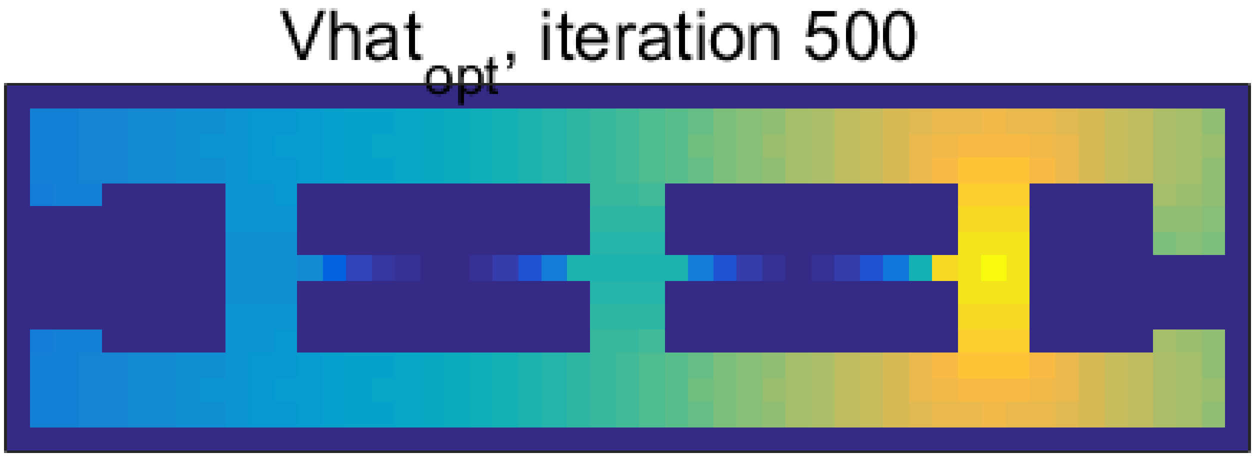
VALUE ITERATION IN ACTION



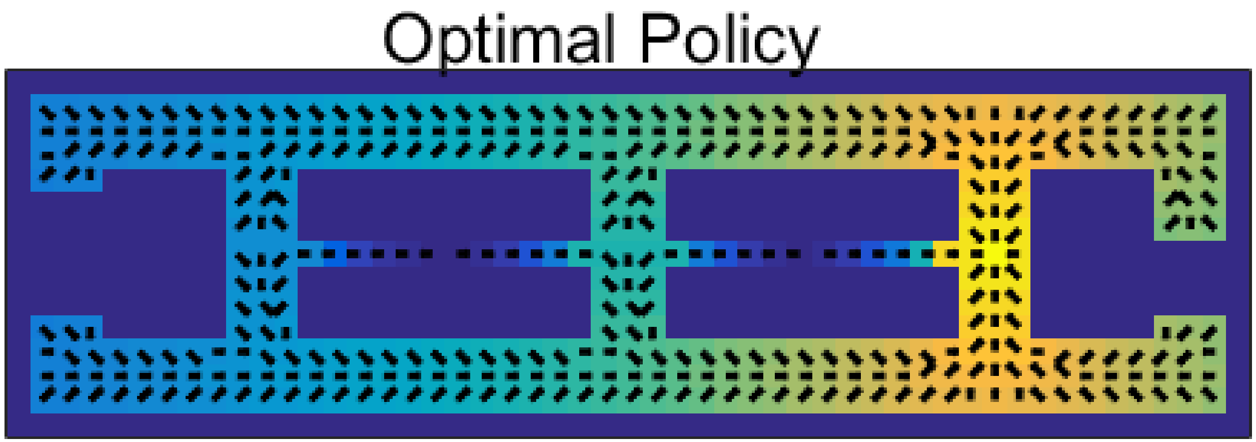
VALUE ITERATION IN ACTION



VALUE ITERATION IN ACTION



VALUE ITERATION IN ACTION



POLICY ITERATION

Greedy policy with respect to V:

$$(GV)(x) = \arg\max_a \left\{ r(x,a) + \sum_y P(y|x,a)V(x) \right\}$$

POLICY ITERATION

Recall: $\pi^* = GV^*$

Greedy policy with respect to V:

$$(GV)(x) = \arg\max_a \left\{ r(x,a) + \sum_y P(y|x,a)V(x) \right\}$$

POLICY ITERATION

Recall: $\pi^* = GV^*$

Greedy policy with respect to V:

$$(GV)(x) = \arg\max_a \{ r(x,a) + \sum_y P(y|x,a)V(x) \}$$

Policy Iteration

Input: arbitrary function $V_0 : X \to \mathbf{R}$

For $k = 0,1, \ldots,$ compute

$$\pi_k = G(V_k)\,, \qquad V_{k+1} = V^{\pi_k}$$

POLICY ITERATION

Recall: $\pi^* = GV^*$

Greedy policy with respect to V:

$$(GV)(x) = \arg\max_a \{ r(x,a) + \sum_y P(y|x,a)V(x) \}$$

Policy Iteration

Input: arbitrary function $V_0 : X \to \mathbf{R}$

For $k = 0,1,\ldots,$ compute

$$\pi_k = G(V_k)\,, \qquad V_{k+1} = V^{\pi_k}$$

Theorem: $\lim_{k\to\infty} V_k = V^*$

THE CONVERGENCE OF VALUE ITERATION: PROOF SKETCH

Key idea: T^* is a contraction

- for any two functions V and V', we have
$$\|T^*V - T^*V'\|_\infty \leq \gamma \|V - V'\|_\infty$$
- repeated application gives
$$\begin{aligned}\|V_{k+1} - V^*\|_\infty &= \|T^*V_k - T^*V^*\|_\infty \\ &\leq \gamma \|V_k - V^*\|_\infty \\ &\leq \gamma^2 \|V_{k-1} - V^*\|_\infty \\ &\leq \cdots \leq \gamma^k \|V_0 - V^*\|_\infty\end{aligned}$$
- thus
$$\lim_{k\to\infty} \|V_{k+1} - V^*\|_\infty = 0$$

THE CONVERGENCE OF ~~VALUE~~ policy ITERATION: PROOF SKETCH

Just replace T^* with the operator
$B^*: V \mapsto \left(T^{G(V)}\right)^{\infty}$

Key idea: T^* is a contraction

- for any two functions V and V', we have
$$\|T^*V - T^*V'\|_\infty \leq \gamma \|V - V'\|_\infty$$
- repeated application gives
$$\begin{aligned}\|V_{k+1} - V^*\|_\infty &= \|T^*V_k - T^*V^*\|_\infty \\ &\leq \gamma \|V_k - V^*\|_\infty \\ &\leq \gamma^2 \|V_{k-1} - V^*\|_\infty \\ &\leq \cdots \leq \gamma^k \|V_0 - V^*\|_\infty\end{aligned}$$
- thus
$$\lim_{k \to \infty} \|V_{k+1} - V^*\|_\infty = 0$$

THIS SHORT COURSE: A PRIMAL-DUAL VIEW

part 1

- Markov decision processes
  - Value functions and optimal policies
- Primal view: Dynamic programming
  - Policy evaluation, value and policy iteration
  - Value-function-based methods
    - Temporal differences, Q-learning, LSTD, deep Q networks,…

part 2

- Dual view: Linear programming
  - LP duality in MDPs
  - Direct policy optimization methods
    - Policy gradients, REPS, TRPO,…

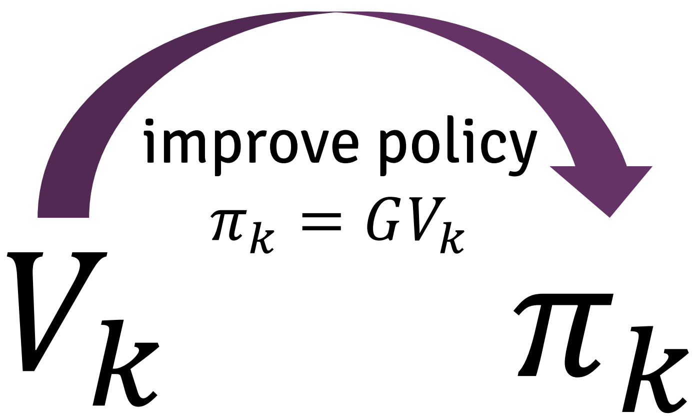
FROM DYNAMIC PROGRAMMING TO VALUE-BASED REINFORCEMENT LEARNING

Policy iteration:

$$V_k$$

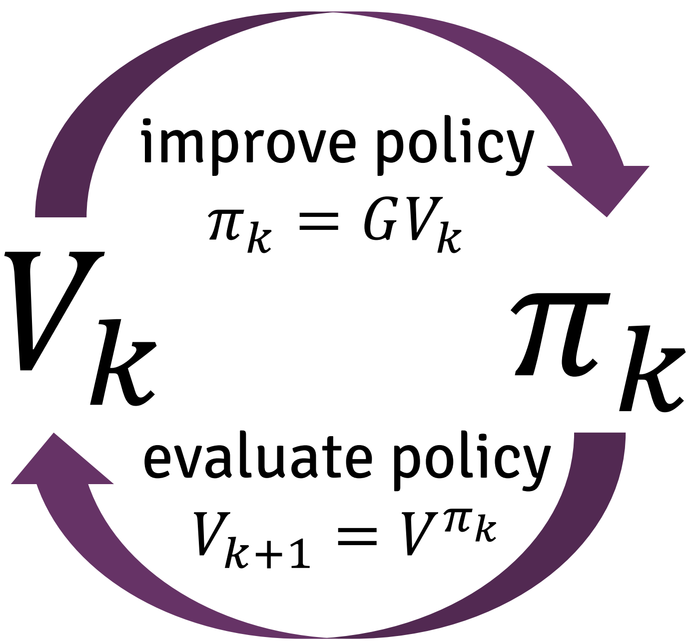
FROM DYNAMIC PROGRAMMING TO VALUE-BASED REINFORCEMENT LEARNING

Policy iteration:



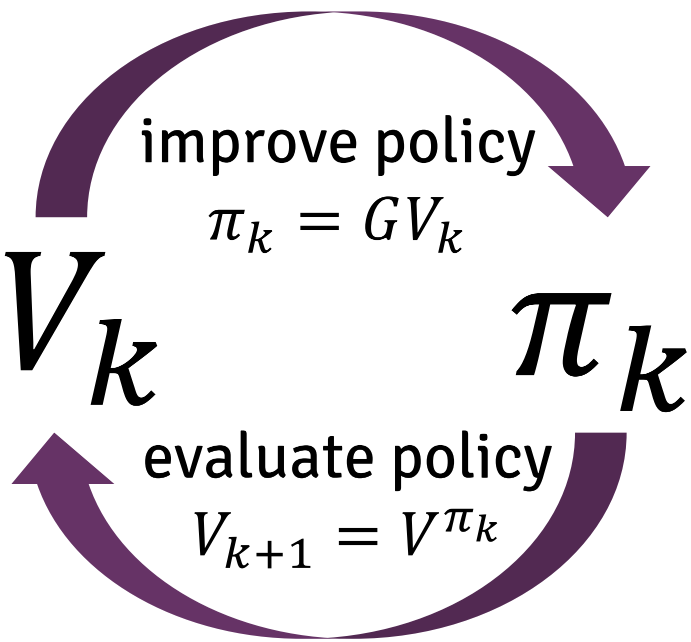
FROM DYNAMIC PROGRAMMING TO VALUE-BASED REINFORCEMENT LEARNING

Policy iteration:

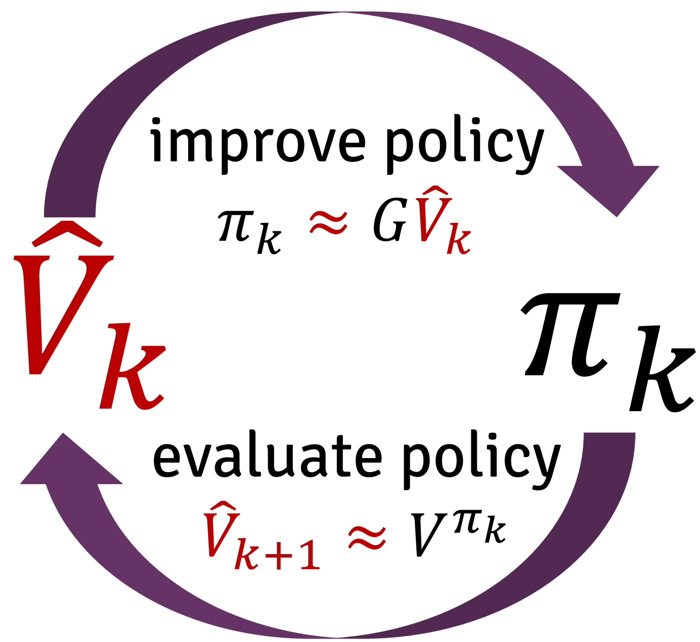


FROM DYNAMIC PROGRAMMING TO VALUE-BASED REINFORCEMENT LEARNING

Policy iteration:



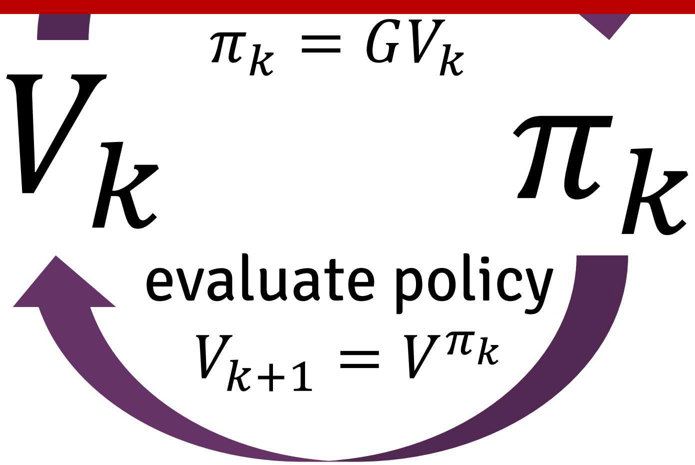
Approximate policy iteration:



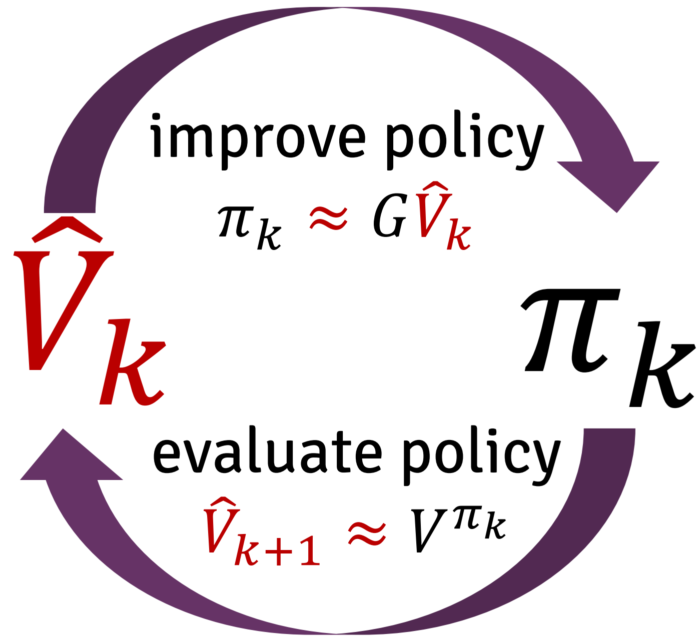
FROM DYNAMIC PROGRAMMING TO VALUE-BASED REINFORCEMENT LEARNING

Fundamental RL tasks:

- Policy evaluation
- Policy improvement



Approximate policy iteration:



FROM DYNAMIC PROGRAMMING TO VALUE-BASED REINFORCEMENT LEARNING

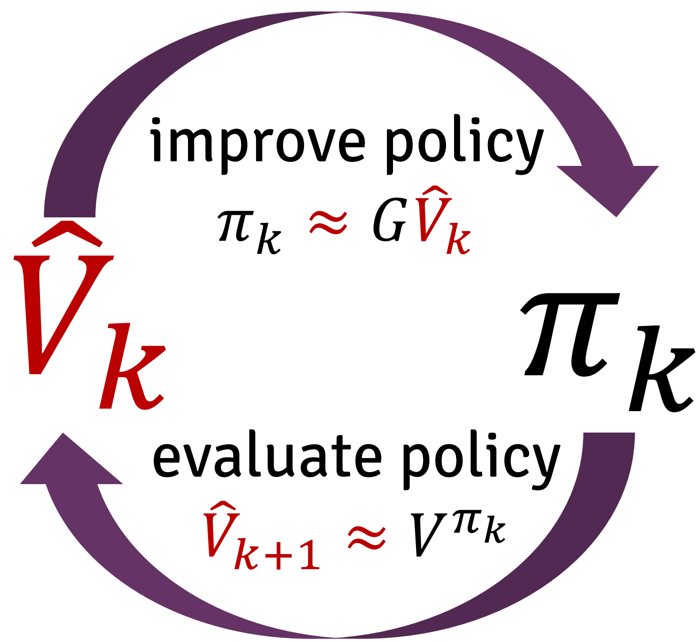
Fundamental RL tasks:

- Policy evaluation
- Policy improvement

Challenges in RL:

- Unknown transition and reward functions ⇒ have to learn from sample access only
- State/action space can be large ⇒ V^* and π^* cannot be stored in memory

Approximate policy iteration:

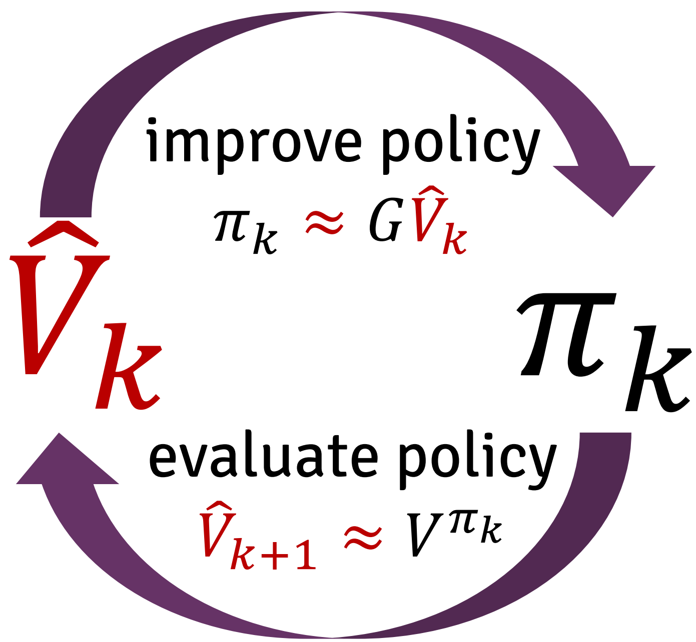


FROM DYNAMIC PROGRAMMING TO VALUE-BASED REINFORCEMENT LEARNING

Fundamental RL tasks:
- Policy evaluation
- Policy improvement

Challenges in RL:
- Unknown transition and reward functions ⇒ have to learn from sample access only
- State/action space can be large ⇒ V^* and π^* cannot be stored in memory

Approximate policy iteration:



Unknown transition and reward functions
⇒ have to learn from sample access only

LEVELS OF SAMPLE ACCESS

Unknown transition and reward functions
⇒ have to learn from sample access only

LEVELS OF SAMPLE ACCESS

Full knowledge of P
⇒ Planning (not RL)

Unknown transition and reward functions
⇒ have to learn from sample access only

LEVELS OF SAMPLE ACCESS

Generative model:
Full sample access to $P(\cdot\,|x,a)$ for any (x,a)

Full knowledge of P
⇒ Planning (not RL)

Unknown transition and reward functions
⇒ have to learn from sample access only

LEVELS OF SAMPLE ACCESS

Samples from full trajectories
+ reset action or save states

Generative model:
Full sample access to $P(\cdot \mid x, a)$ for any (x, a)

Full knowledge of P
⇒ Planning (not RL)

Unknown transition and reward functions
⇒ have to learn from sample access only

LEVELS OF SAMPLE ACCESS

Samples from a single trajectory
⇒ online RL

Samples from full trajectories
+ reset action or save states

Generative model:
Full sample access to $P(\cdot \mid x, a)$ for any (x, a)

Full knowledge of P
⇒ Planning (not RL)

Unknown transition and reward functions
$\Rightarrow$ have to learn from sample access only

LEVELS OF SAMPLE ACCESS

Samples from a single trajectory
$\Rightarrow$ online RL

Samples from full trajectories
+ reset action or save states

Generative model:
Full sample access to $P(\cdot\,|x,a)$ for any (x,a)

Full knowledge of P
$\Rightarrow$ Planning (not RL)

State/action space can be large
⇒ V^* and π^* cannot be stored in memory

DEALING WITH LARGE STATE SPACES

Idea: approximate V^* and/or π^* in a computationally tractable way!

State/action space can be large
$\Rightarrow V^*$ and π^* cannot be stored in memory

DEALING WITH LARGE STATE SPACES

Idea: approximate V^* and/or π^* in a computationally tractable way!

Approximating V^*: linear function approximation

- Define a set of d features:
$$\phi_i : X \to \mathbf{R}$$
- Parametrize value functions as
$$V_\theta(x) = \theta^\top \phi(x)$$
- Learning $V^* \Leftrightarrow$ Learning a good θ_*
$$V_{\theta^*} \approx V^*$$

State/action space can be large
$\Rightarrow V^*$ and π^* cannot be stored in memory

DEALING WITH LARGE STATE SPACES

Idea: approximate V^* and/or π^* in a computationally tractable way!

Approximating V^*: linear function approximation

- Define a set of d features:
  $$\phi_i: X \to \mathbf{R}$$
- Parametrize value functions as
  $$V_\theta(x) = \theta^\top \phi(x)$$
- Learning $V^* \Leftrightarrow$ Learning a good θ_*
  $$V_{\theta^*} \approx V^*$$

Approximating π^*: parametrized policies

- Define a set of d features:
  $$\phi_i: X \times A \to \mathbf{R}$$
- Parametrize (stochastic) policies as
  $$\pi_\theta(a|x) \propto \exp\left(\theta^\top \phi(x)\right)$$
- Learning $\pi^* \Leftrightarrow$ Learning a good θ_*
  $$\pi_{\theta^*} \approx \pi^*$$

State/action space can be large
$\Rightarrow V^*$ and π^* cannot be stored in memory

DEALING WITH LARGE STATE SPACES

Idea: approximate V^* and/or π^* in a computationally tractable way!

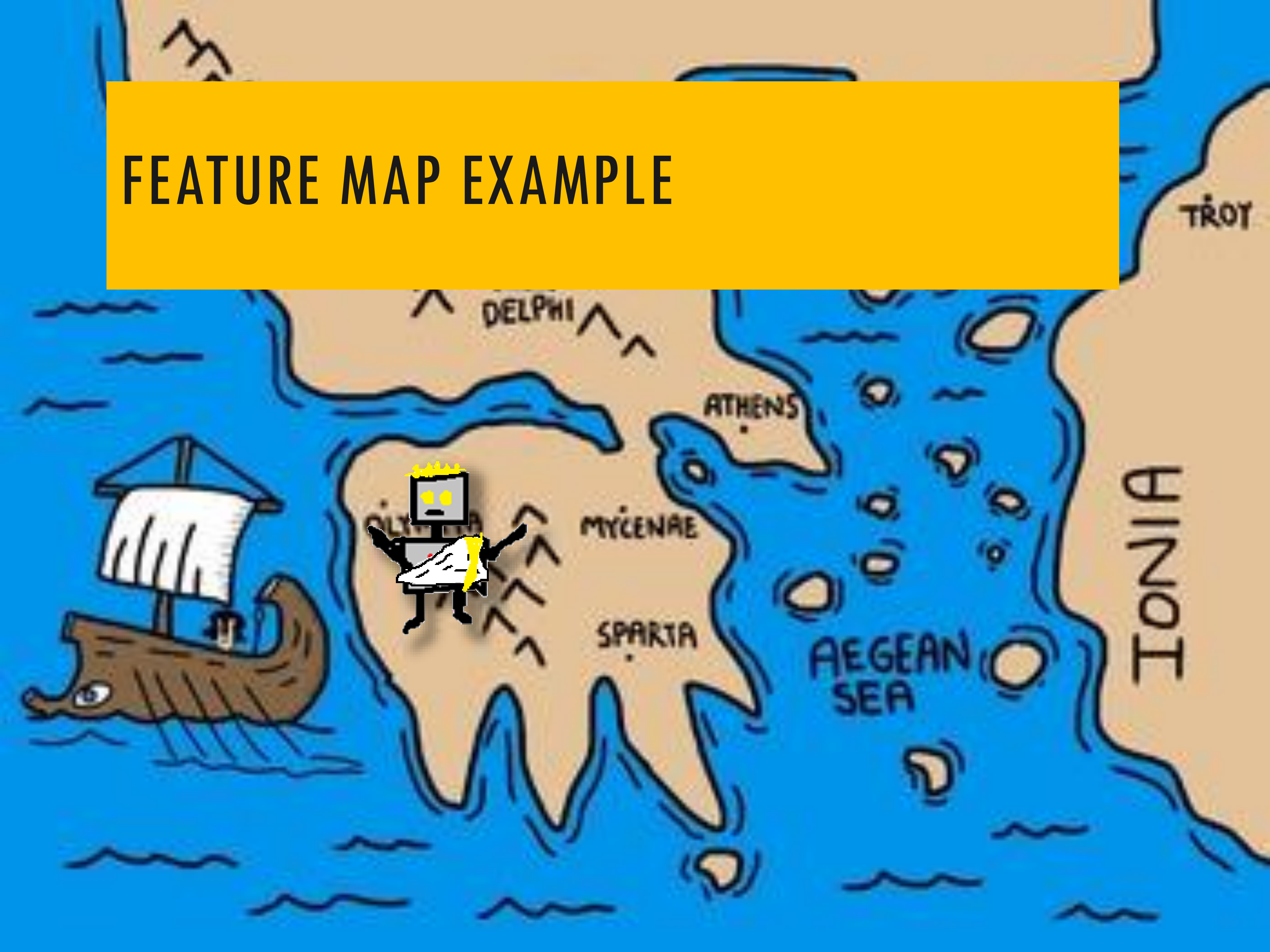
Approximating V^*: linear function approximation

- Define a set of d features:
$$\phi_i: X \to \mathbf{R}$$
- Parametrize value functions as
$$V_\theta(x) = \theta^\top \phi(x)$$
- Learning $V^* \Leftrightarrow$ Learning a good θ_*
$$V_{\theta^*} \approx V^*$$

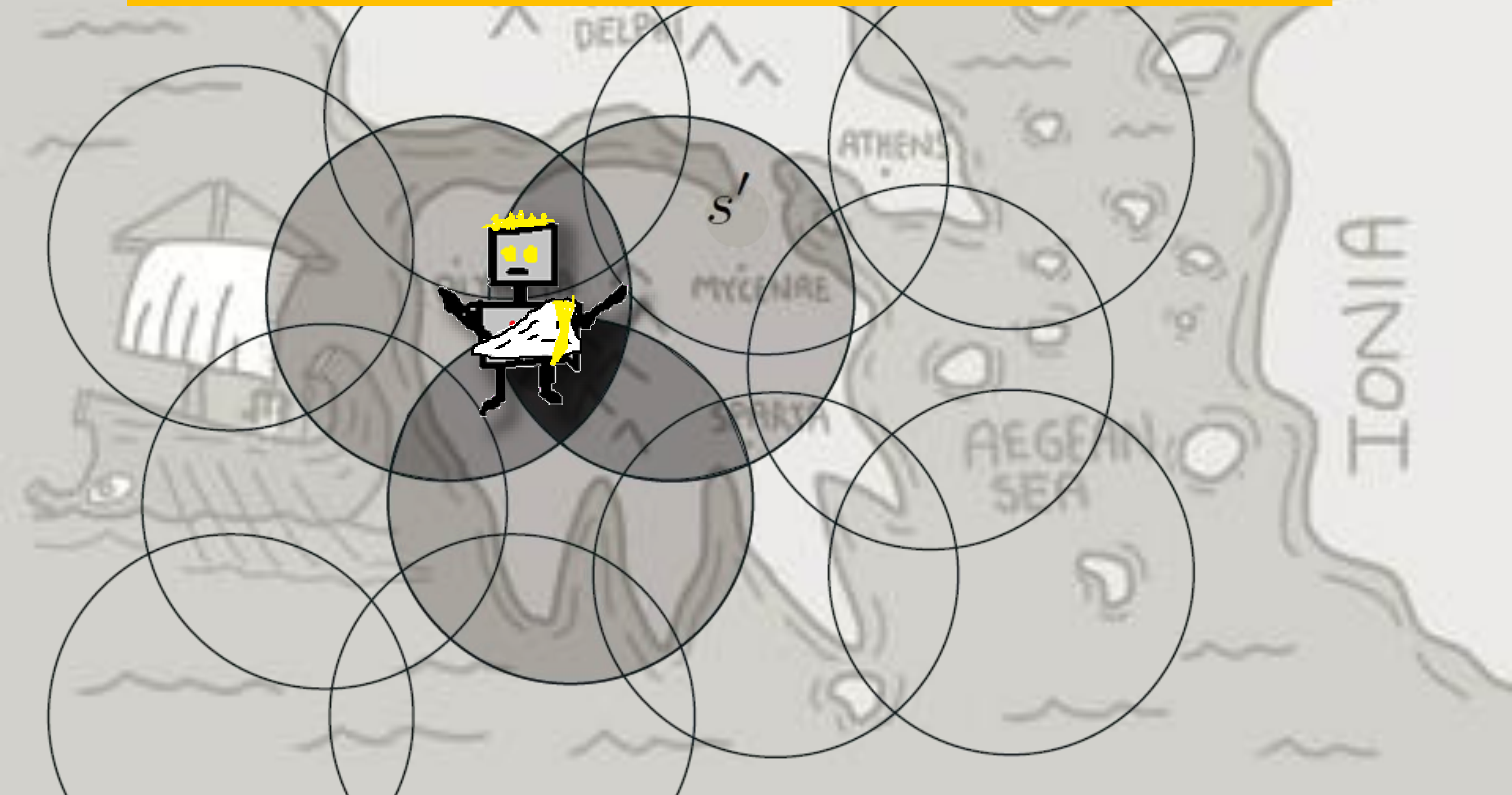
Approximating π^*: parametrized policies

- Define a set of d features:
$$\phi_i: X \times A \to \mathbf{R}$$
- Parametrize (stochastic) policies as
$$\pi_\theta(a|x) \propto \exp\left(\theta^\top \phi(x)\right)$$
- Learning $\pi^* \Leftrightarrow$ Learning a good θ_*
$$\pi_{\theta^*} \approx \pi^*$$

FEATURE MAP EXAMPLE



FEATURE MAP EXAMPLE



FEATURE MAP EXAMPLE

"coarse coding"

≈

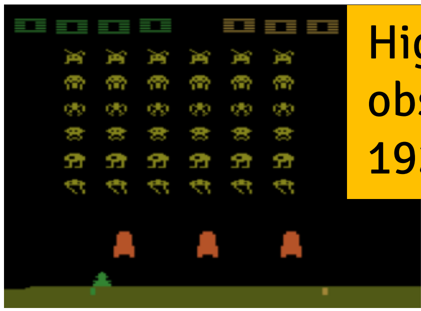
indicator features

$\phi_i(x) = \mathbf{1}\{x \in X_i\}$

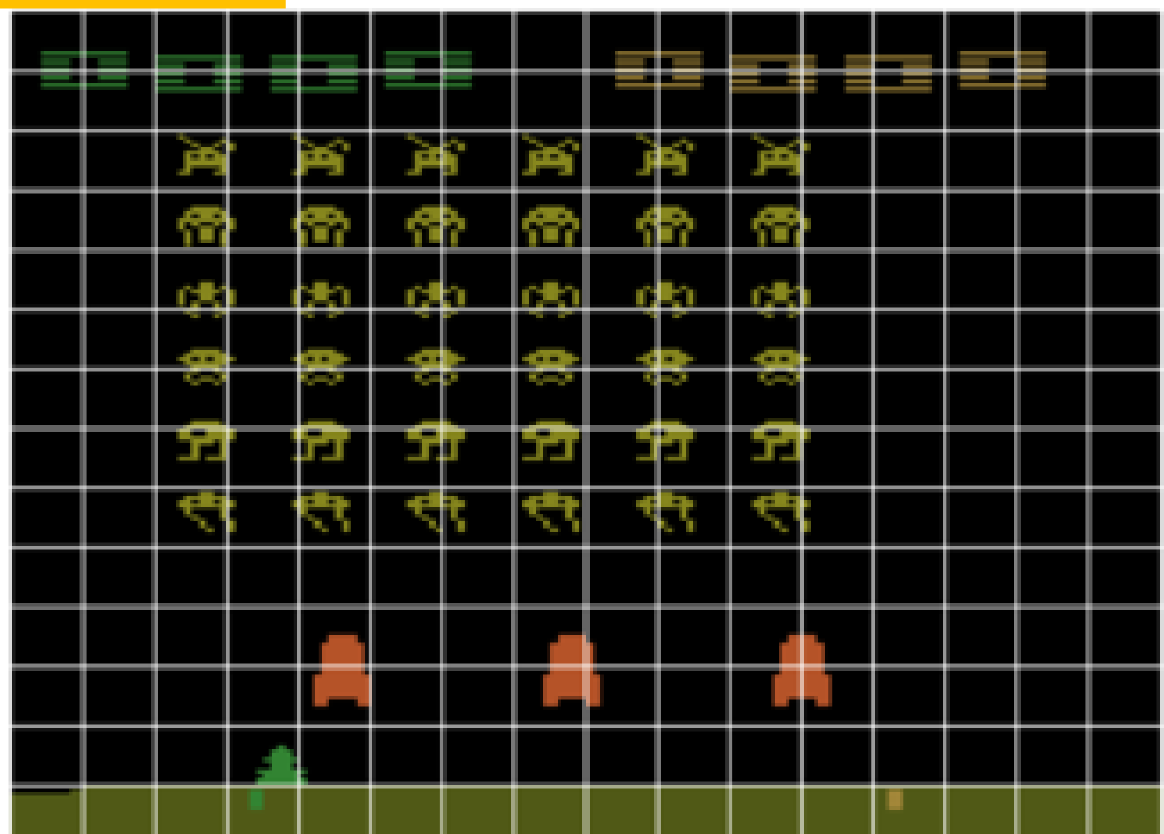
"PROST" FEATURES FOR ATARI GAMES

High-dimensional observations: 192×160 pixels

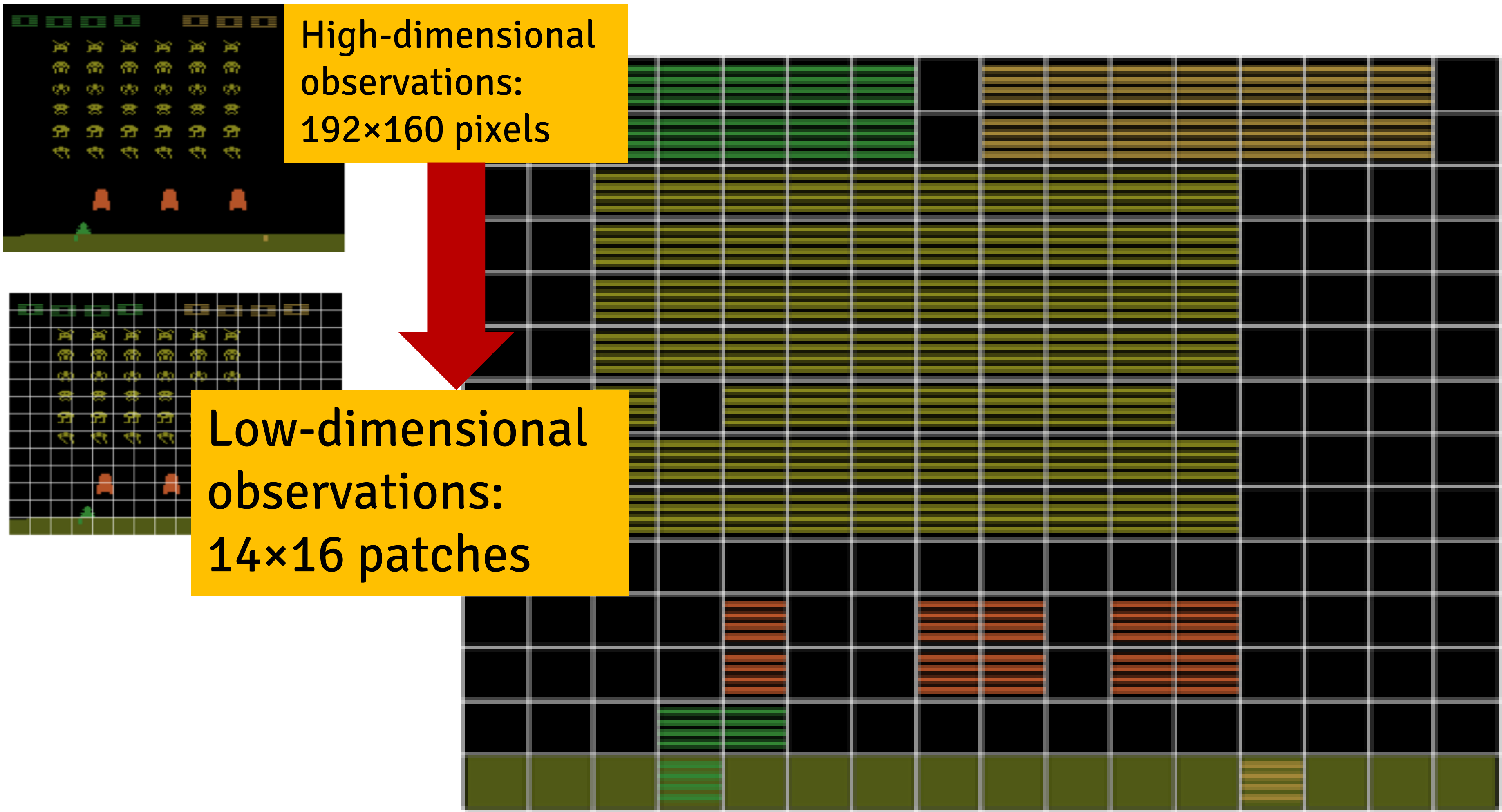
“PROST” FEATURES FOR ATARI GAMES

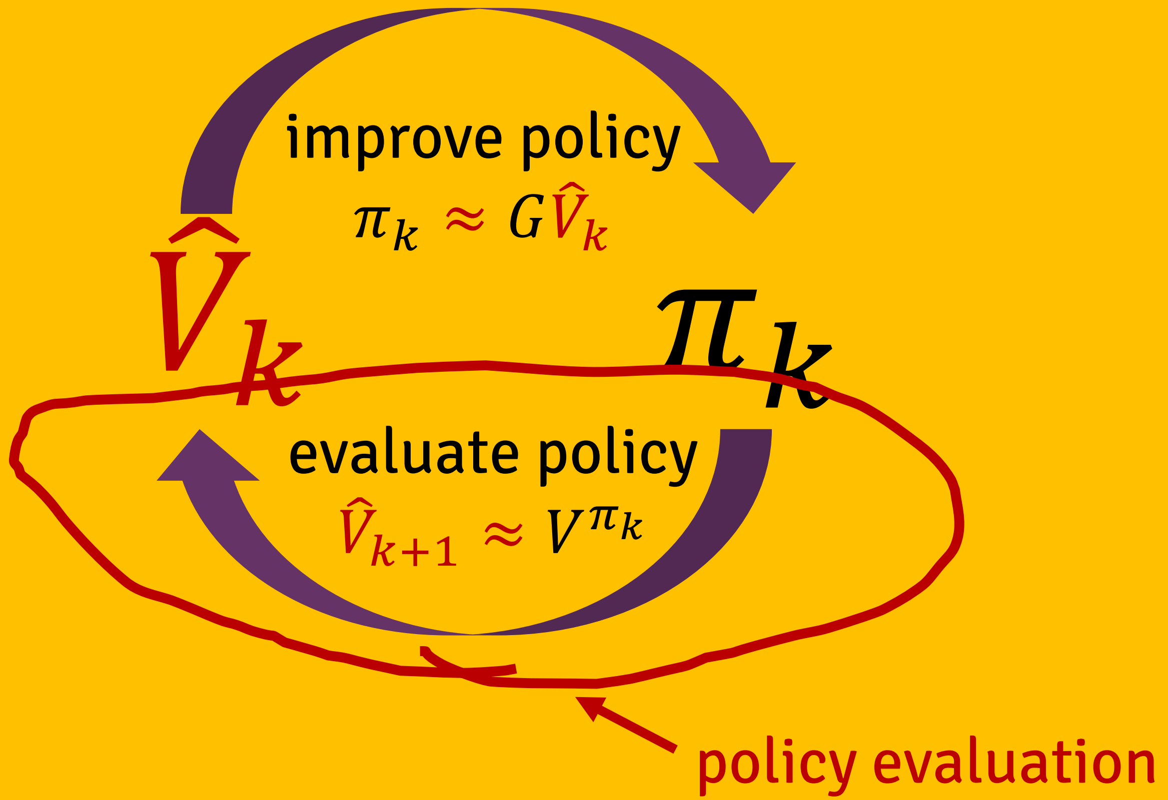


High-dimensional observations: 192×160 pixels



“PROST” FEATURES FOR ATARI GAMES





METHODS FOR POLICY EVALUATION

A GENTLE START: MONTE CARLO

Observe:

Policy evaluation = estimating V^{π}:

$V^{\pi}(x) = \mathbf{E}_{\pi}[\sum_{t=0}^{\infty} \gamma^{t} r(x_t, a_t) \,|x_0 = x]$

A GENTLE START: MONTE CARLO

Observe:
Policy evaluation = estimating V^π:
$V^\pi(x) = \mathbf{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(x_t, a_t)\,|x_0 = x]$

Idea:
approximate $\mathbf{E}_\pi[\cdot]$ by sample averages!

- Simulate N trajectories using policy π
- For every state x that appears in the trajectories, let

$$\hat{V}_N(x) = \text{avg}\big(R_{1:N}(x)\big)$$

A GENTLE START: MONTE CARLO

Idea:
approximate $\mathbf{E}_\pi[\cdot]$ by sample averages!

- Simulate N trajectories using policy π
- For every state x that appears in the trajectories, let
$$\hat{V}_N(x) = \mathrm{avg}\big(R_{1:N}(x)\big)$$

A GENTLE START: MONTE CARLO

Idea:
approximate $\mathbf{E}_\pi[\cdot]$ by sample averages!

- Simulate N trajectories using policy π
- For every state x that appears in the trajectories, let

$$\hat{V}_N(x) = \text{avg}\big(R_{1:N}(x)\big)$$

Collection of discounted returns $\sum_{t=0}^{T'} \gamma^t r_t$ after first visit to x

A GENTLE START: MONTE CARLO

Idea:
approximate $\mathbf{E}_\pi[\cdot]$ by sample averages!

- Simulate N trajectories using policy π
- For every state x that appears in the trajectories, let

$$\hat{V}_N(x) = \text{avg}\big(R_{1:N}(x)\big)$$

Average of i.i.d. random variables:

$$\lim_{N\to\infty} \hat{V}_N = V^\pi$$

Collection of discounted returns $\sum_{t=0}^{T'} \gamma^t r_t$ after first visit to x

MONTE CARLO WITH FEATURES

Monte Carlo policy evaluation

Input:

N trajectories $\sim \pi$, feature map $\phi : X \to \mathbb{R}^d$

Output:

$$\hat{V}_N = \arg\min_{\theta \in \mathbb{R}^d} \mathbf{E}_x \left[\left(\theta^\top \phi(x) - R_{1:N}(x) \right)^2 \right]$$

MONTE CARLO WITH FEATURES

Monte Carlo policy evaluation

Input:

N trajectories $\sim \pi$, feature map $\phi: X \to \mathbb{R}^d$

Output:

$$\hat{V}_N = \arg\min_{\theta \in \mathbb{R}^d} \mathbf{E}_x \left[\left(\theta^\top \phi(x) - R_{1:N}(x) \right)^2 \right]$$

Least-squares fit of discounted returns

PROPERTIES OF MONTE CARLO

☺ Value estimates converge to true values ☺

☺ Doesn't need prior knowledge of P or r ☺

PROPERTIES OF MONTE CARLO

☺ Value estimates converge to true values ☺

☺ Doesn't need prior knowledge of P or r ☺

☹ Doesn't make use of the Bellman equations ☹

A BETTER OBJECTIVE?

Idea: construct an objective that uses the Bellman equations

$$V^{\pi} \approx T^{\pi} V^{\pi}$$

A BETTER OBJECTIVE?

Idea: construct an objective that uses the Bellman equations

$$V^\pi \approx T^\pi V^\pi$$

The Bellman error

$$L(V) = \mathbf{E}_{x\sim\mu}\left[\left(T^\pi V(x) - V(x)\right)^2\right]$$

TEMPORAL DIFFERENCE LEARNING

Idea: use stochastic approximation to reduce instantaneous Bellman error

$$\Delta_t = \left(T^{\pi}\hat{V}_t(x_t) - \hat{V}_t(x_t)\right)^2$$

TEMPORAL DIFFERENCE LEARNING

Idea: use stochastic approximation to reduce instantaneous Bellman error

$$\Delta_t = \left(T^\pi \hat{V}_t(x_t) - \hat{V}_t(x_t)\right)^2$$

TD(0)

Input: arbitrary function $\hat{V}_0 : X \to \mathbf{R}$

For $t = 0,1, \ldots,$

$$\delta_t = r_t + \gamma \hat{V}_t(x_{t+1}) - \hat{V}_t(x_t)$$

$$\hat{V}_{t+1}(x_t) = \hat{V}_t(x_t) + \alpha_t \delta_t$$

TEMPORAL DIFFERENCE LEARNING

TD(0)

Input: arbitrary function $\hat{V}_0 : X \to \mathbf{R}$

For $t = 0,1,\ldots,$

$$\delta_t = r_t + \gamma \hat{V}_t(x_{t+1}) - \hat{V}_t(x_t)$$
$$\hat{V}_{t+1}(x_t) = \hat{V}_t(x_t) + \alpha_t \delta_t$$

Converges if step-sizes satisfy

$\sum_{t=0}^{\infty} \alpha_t = \infty$ and $\sum_{t=0}^{\infty} \alpha_t^2 < \infty$

(e.g., $\alpha_t = c/t$ does the job)

TEMPORAL DIFFERENCE LEARNING

TD(0)

Input: arbitrary function $\hat{V}_0 : X \to \mathbf{R}$

For $t = 0, 1, \ldots,$

$$\delta_t = r_t + \gamma \hat{V}_t(x_{t+1}) - \hat{V}_t(x_t)$$
$$\hat{V}_{t+1}(x_t) = \hat{V}_t(x_t) + \alpha_t \delta_t$$

Converges if step-sizes satisfy

$\sum_{t=0}^{\infty} \alpha_t = \infty$ and $\sum_{t=0}^{\infty} \alpha_t^2 < \infty$

(e.g., $\alpha_t = c/t$ does the job)

In equilibrium,

$$\mathbf{E}\left[r_t + \gamma \hat{V}_t(x_{t+1}) - \hat{V}_t(x_t)\right] = 0$$

TD(0) WITH LINEAR FUNCTION APPROXIMATION

Let $\phi: X \to \mathbf{R}^d$ be a feature vector

TD(0) WITH LINEAR FUNCTION APPROXIMATION

Let $\phi: X \to \mathbf{R}^d$ be a feature vector

Approximating $V^\pi(x) \approx \theta^\top \phi(x)$ by TD(0):

TD(0) with LFA

Input: arbitrary param. vector $\theta_0 \in \mathbf{R}^d$

For $t = 0,1, \ldots,$

$$\delta_t = r_t + \gamma \theta_t^\top \phi(x_{t+1}) - \theta_t^\top \phi(x_t)$$
$$\theta_{t+1} = \theta_t + \alpha_t \delta_t \phi(x_t)$$

TD(0) WITH LINEAR FUNCTION APPROXIMATION

Let $\phi: X \to \mathbf{R}^d$ be a feature vector

Approximating $V^\pi(x) \approx \theta^\top \phi(x)$ by TD(0):

TD(0) with LFA

Input: arbitrary param. vector $\theta_0 \in \mathbf{R}^d$

For $t = 0,1, \dots,$

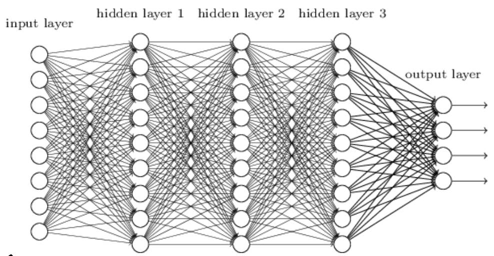
$$\delta_t = r_t + \gamma \theta_t^\top \phi(x_{t+1}) - \theta_t^\top \phi(x_t)$$
$$\theta_{t+1} = \theta_t + \alpha_t \delta_t \phi(x_t)$$

This still converges to V^π!!!

OK, well, somewhere nearby…

TD(0) WITH NONLINEAR FUNCTION APPROXIMATION

Let $V_\theta: X \to R$ be a parametric class of functions (e.g., deep neural network)



Approximating $V^\pi(x) \approx V_\theta(x)$ by TD(0):

TD(0) with general FA

Input: arbitrary param. vector $\theta_0 \in \mathbf{R}^d$

For $t = 0,1, \dots,$

$$\delta_t = r_t + \gamma V_{\theta_t}(x_{t+1}) - V_{\theta_t}(x_t)$$

$$\theta_{t+1} = \theta_t + \alpha_t \delta_t \nabla_\theta V_{\theta_t}(x_t)$$

TD(0) WITH NONLINEAR FUNCTION APPROXIMATION

Let $V_\theta : X \to R$ be a p… functions (e.g., deep…

Approximating $V^\pi(x) \approx V_\theta(x)$ by TD(0):

Not much is known about convergence ☹

TD(0) with general FA

Input: arbitrary param. vector $\theta_0 \in \mathbf{R}^d$

For $t = 0,1, \dots,$

$$\delta_t = r_t + \gamma V_{\theta_t}(x_{t+1}) - V_{\theta_t}(x_t)$$

$$\theta_{t+1} = \theta_t + \alpha_t \delta_t \nabla_\theta V_{\theta_t}(x_t)$$

PROPERTIES OF TD(0)

☺ Value estimates converge to true values ☺

☺ Doesn't need prior knowledge of P or r ☺

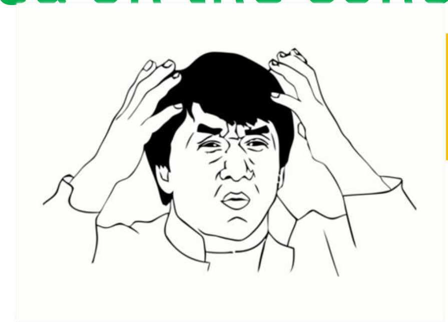
☺ Based on the concept of Bellman error ☺

PROPERTIES OF TD(0)

☺ Value estimates converge to true values ☺

☺ Doesn't need prior knowledge of P or r ☺

☺ Based on the concept of Bellman error ☺



= "bootstrapping"

WHERE DOES TD(0) CONVERGE TO?

TD(0) with LFA

Input: arbitrary param. vector $\theta_0 \in \mathbf{R}^d$

For $t = 0,1, \dots,$

$$\delta_t(\theta) = r_t + \gamma\theta^\top\phi(x_{t+1}) - \theta^\top\phi(x_t)$$

$$\theta_{t+1} = \theta_t + \alpha_t \delta_t(\theta_t)\phi(x_t)$$

WHERE DOES TD(0) CONVERGE TO?

TD(0) with LFA

Input: arbitrary param. vector $\theta_0 \in \mathbf{R}^d$

For $t = 0,1, \dots,$

$$\delta_t(\theta) = r_t + \gamma\theta^\top\phi(x_{t+1}) - \theta^\top\phi(x_t)$$

$$\theta_{t+1} = \theta_t + \alpha_t\delta_t(\theta_t)\phi(x_t)$$

In the limit, TD(0) finds a θ^* such that

$$\mathbf{E}[\delta_t(\theta^*)\phi(x_t)] = 0$$

WHERE DOES TD(0) CONVERGE TO?

Idea: given a finite trajectory, approximate the TD fixed point by solving

$$\mathbf{E}[\delta_t(\theta)\phi(x_t)] \approx \frac{1}{T}\sum_{t=1}^{T}\delta_t(\theta)\phi(x_t) = 0$$

WHERE DOES TD(0) CONVERGE TO?

Idea: given a finite trajectory, approximate the TD fixed point by solving

$$\mathbf{E}[\delta_t(\theta)\phi(x_t)] \approx \frac{1}{T}\sum_{t=1}^{T}\delta_t(\theta)\phi(x_t) = 0$$

Equivalently:

$$\frac{1}{T}\sum_{t=1}^{T}\phi(x_t)\big(\phi(x_t) - \gamma\phi(x_{t+1})\big)^\top\theta = \frac{1}{T}\sum_{t=1}^{T} r_t\phi(x_t)$$

WHERE DOES TD(0) CONVERGE TO?

This is a linear system

$$A_T \theta = b_T$$

Solution:

Equivalently:

$$\underbrace{\frac{1}{T}\sum_{t=1}^{T}\phi(x_t)\big(\phi(x_t) - \gamma\phi(x_{t+1})\big)^{\top}}_{A_T}\theta = \underbrace{\frac{1}{T}\sum_{t=1}^{T} r_t \phi(x_t)}_{b_T}$$

WHERE DOES TD(0) CONVERGE TO?

This is a linear system

$$A_T \theta = b_T$$

Solution: $\theta_T = A_T^{-1} b_T$

e

Equivalently:

$$\frac{1}{T}\sum_{t=1}^{T} \phi(x_t)\big(\phi(x_t) - \gamma\phi(x_{t+1})\big)^\top \theta = \frac{1}{T}\sum_{t=1}^{T} r_t \phi(x_t)$$

A_T $\qquad$ b_T

LEAST-SQUARES TEMPORAL DIFFERENCE LEARNING (LSTD)

LSTD(0)

Input: trajectory $(x_t, a_t, r_t)_{t=1}^{T}$

$$\theta_T = A_T^{-1} b_T$$

$$\hat{V}_T = \theta_T^{\top} \phi$$

LEAST-SQUARES TEMPORAL DIFFERENCE LEARNING (LSTD)

LSTD(0)

Input: trajectory $(x_t, a_t, r_t)_{t=1}^{T}$

$$\theta_T = A_T^{-1} b_T$$
$$\hat{V}_T = \theta_T^{\top} \phi$$

☺ converges to same θ^* as TD(0) ☺

☺ no need to set step sizes α_t ☺

LEAST-SQUARES TEMPORAL DIFFERENCE LEARNING (LSTD)

LSTD(0)

Input: trajectory $(x_t, a_t, r_t)_{t=1}^{T}$

$$\theta_T = A_T^{-1} b_T$$
$$\hat{V}_T = \theta_T^\top \phi$$

☺ converges to same θ^* as TD(0) ☺

☺ no need to set step sizes α_t ☺

☹ computational complexity: $O(Td^2 + d^3)$ ☹

☹ A_T^{-1} may not exist for small T ☹

TD(0):
$O(Td)$

THE CONVERGENCE OF TD(0) AND LSTD(0)

Theorem

In the limit $T \to \infty$, LSTD(0) and TD(0) both minimize the projected Bellman error

$$L(V) = \mathbf{E}_{x\sim\mu}\left[\left(\Pi_{\phi}[T^{\pi}V(x)] - V(x)\right)^{2}\right]$$

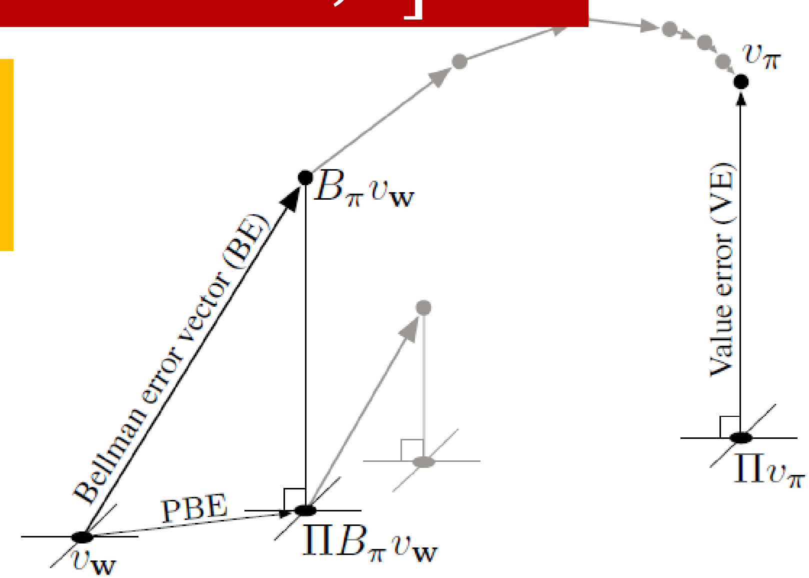
THE CONVERGENCE OF TD(0) AND LSTD(0)

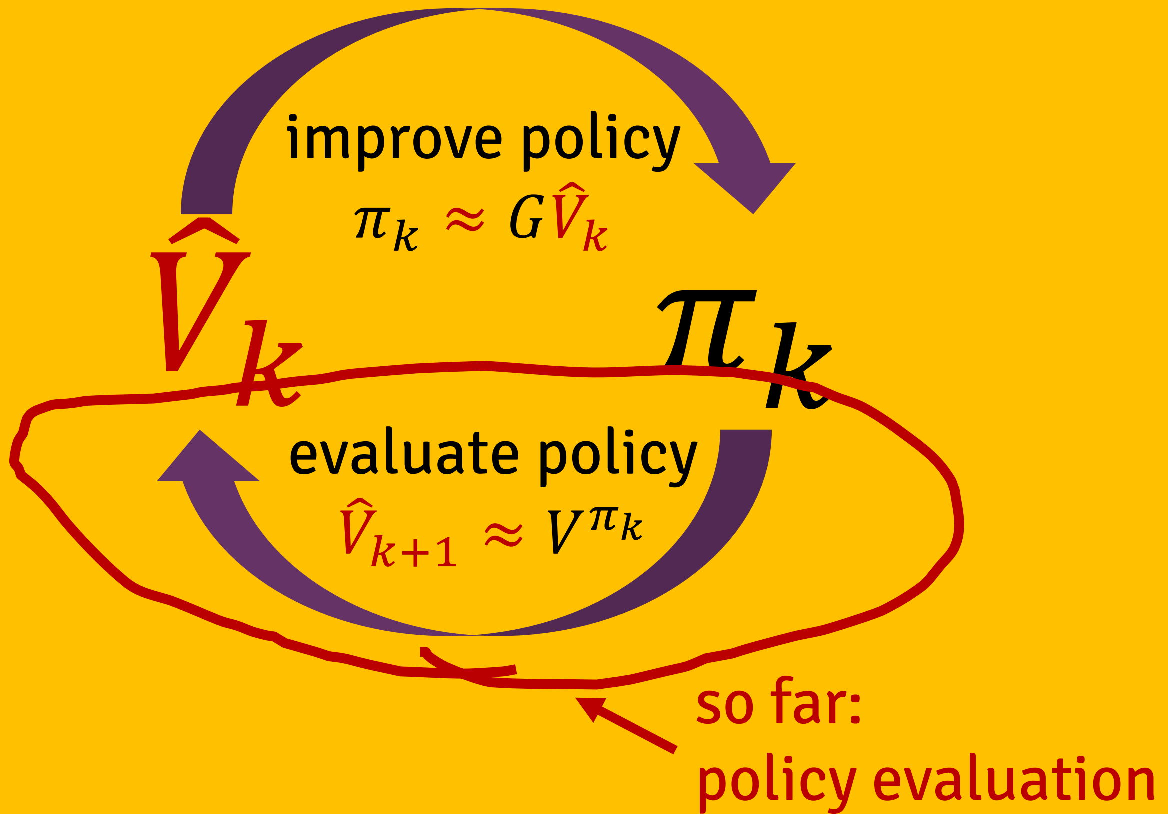
Theorem

In the limit $T \to \infty$, LSTD(0) and TD(0) both minimize the projected Bellman error

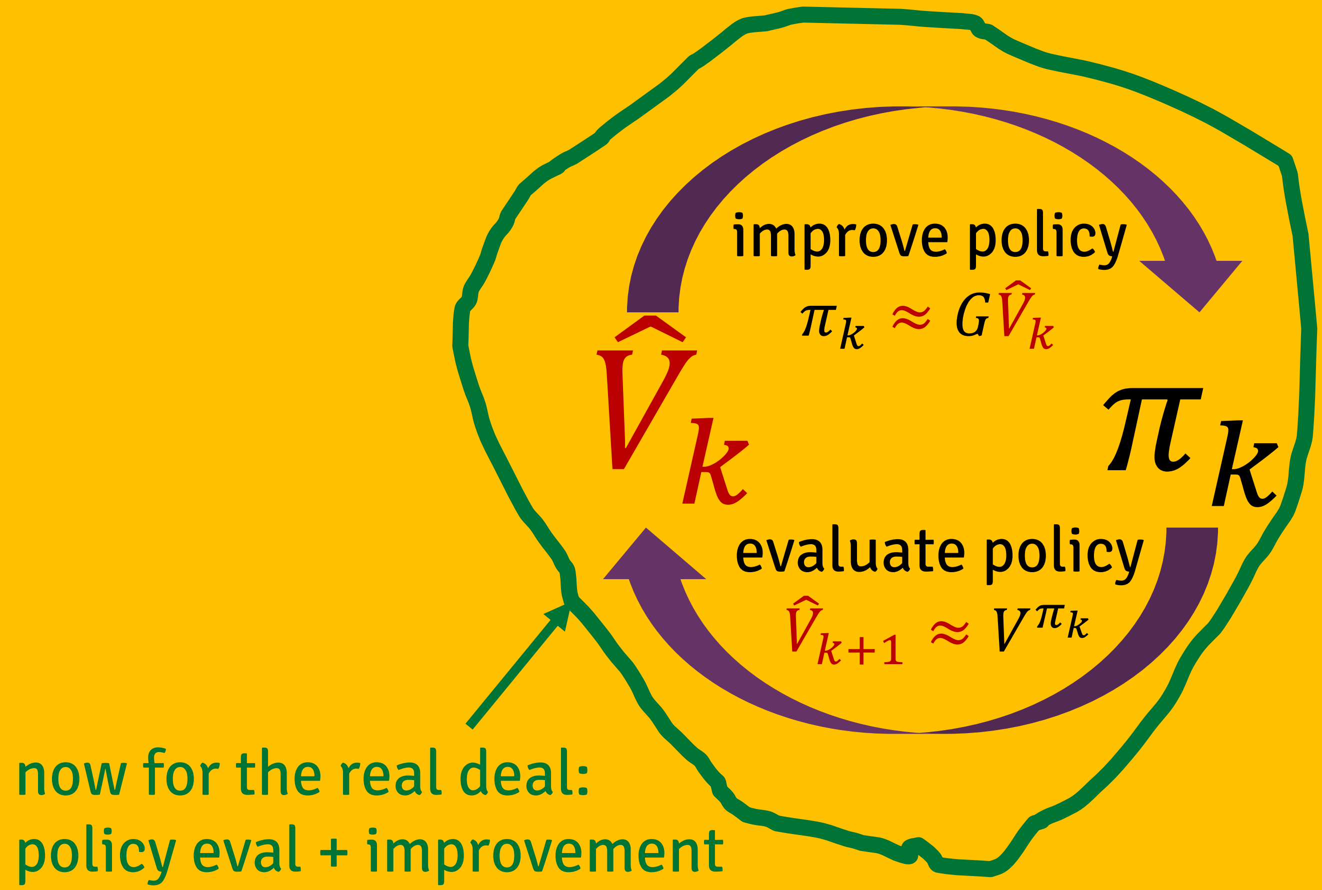
$$L(V) = \mathbf{E}_{x \sim \mu}\left[\left(\Pi_\phi[T^\pi V(x)] - V(x)\right)^2\right]$$

Projection onto span of features





FROM POLICY EVALUATION POLICY IMPROVEMENT



FROM POLICY EVALUATION POLICY IMPROVEMENT

OFF-POLICY CONTROL: Q-LEARNING

Idea: Let's try to

- directly learn about Q^*, and
- improve the policy on the fly!

OFF-POLICY CONTROL: Q-LEARNING

Idea: Let's try to

- directly learn about Q^*, and
- improve the policy on the fly!

- Compute ε-greedy policy w.r.t. $\hat{Q}_t$:

$$\pi_t(x) = \begin{cases} \arg\max \hat{Q}_t(x,a)\,, & \text{w.p. } 1-\varepsilon \\ \text{uniform random action}, & \text{w.p. } \varepsilon \end{cases}$$

- Improve estimated $\hat{Q}_{t+1}$ by reducing Bellman error

$$\Delta_t = \left(\mathbf{E}\left[r_t + \gamma \max_a \hat{Q}_t(x_{t+1},a)\right] - \hat{Q}_t(x_t,a_t)\right)^2$$

OFF-POLICY CONTROL: Q-LEARNING

Off-policy learning: evaluating π^* while following suboptimal policy!

Idea: Let's try to
- directly learn about Q^*, and
- improve the policy on the fly!

- Compute ε-greedy policy w.r.t. $\hat{Q}_t$:

$$\pi_t(x) = \begin{cases} \arg\max \hat{Q}_t(x,a), & \text{w.p. } 1-\varepsilon \\ \text{uniform random action}, & \text{w.p. } \varepsilon \end{cases}$$

- Improve estimated $\hat{Q}_{t+1}$ by reducing Bellman error

$$\Delta_t = \left(\mathbf{E}\left[r_t + \gamma \max_a \hat{Q}_t(x_{t+1},a)\right] - \hat{Q}_t(x_t,a_t)\right)^2$$

OFF-POLICY CONTROL: Q-LEARNING

Q-learning

Input: arbitrary $\hat{Q}_0 : X \times A \to \mathbf{R}$

For $t = 0,1, \dots,$

- Choose action $a_t \sim \varepsilon$-greedy w.r.t. $\hat{Q}_t$
- Observe r_t, x_{t+1}
- Compute

$$\delta_t = r_t + \gamma \max_a \hat{Q}_t(x_{t+1}, a) - \hat{Q}_t(x_t, a_t)$$

$$\hat{Q}_{t+1}(x_t, a_t) = \hat{Q}_t(x_t, a_t) + \alpha_t \delta_t$$

ON-POLICY CONTROL: SARSA

SARSA

Input: arbitrary $\hat{Q}_0 : X \times A \to \mathbf{R}$

For $t = 0,1, \ldots,$

- Choose action $a_t \sim \varepsilon$-greedy w.r.t. $\hat{Q}_t$
- Observe r_t, x_{t+1}, a'_{t+1}
- Compute

  $\delta_t = r_t + \gamma \hat{Q}_t(x_{t+1}, a'_{t+1}) - \hat{Q}_t(x_t, a_t)$

  $\hat{Q}_{t+1}(x_t, a_t) = \hat{Q}_t(x_t, a_t) + \alpha_t \delta_t$

ON-POLICY CONTROL: SARSA

SARSA

Input: arbitrary $\hat{Q}_0: X \times A \to \mathbf{R}$

For $t = 0,1, \dots,$

- Choose action $a_t \sim \varepsilon$-greedy w.r.t. $\hat{Q}_t$
- Observe r_t, x_{t+1}, a'_{t+1}
- Compute

$$\delta_t = r_t + \gamma \hat{Q}_t(x_{t+1}, a'_{t+1}) - \hat{Q}_t(x_t, a_t)$$

$$\hat{Q}_{t+1}(x_t, a_t) = \hat{Q}_t(x_t, a_t) + \alpha_t \delta_t$$

$a'_{t+1} \sim \varepsilon$ −greedy: on-policy

ON-POLICY CONTROL: SARSA

SARSA

Input: arbitrary $\hat{Q}_0: X \times A \to \mathbf{R}$

For $t = 0,1, \dots,$

- Choose action $a_t \sim \varepsilon$-greedy w.r.t. $\hat{Q}_t$
- Observe r_t, x_{t+1}, a'_{t+1}
- Compute

$$\delta_t = r_t + \gamma \hat{Q}_t(x_{t+1}, a'_{t+1}) - \hat{Q}_t(x_t, a_t)$$
$$\hat{Q}_{t+1}(x_t, a_t) = \hat{Q}_t(x_t, a_t) + \alpha_t \delta_t$$

$a'_{t+1} \sim \varepsilon$ −greedy: on-policy

SARSA = $(s_t, a_t, r_t, s_{t+1}, a'_{t+1})$

ON-POLICY CONTROL: SARSA

SARSA ~ XARXA

Input: arbitrary $\hat{Q}_0 : X \times A \to \mathbf{R}$

For $t = 0,1, \ldots,$

- Choose action $a_t \sim \varepsilon$-greedy w.r.t. $\hat{Q}_t$
- Observe r_t, x_{t+1}, a'_{t+1}
- Compute

  $\delta_t = r_t + \gamma \hat{Q}_t(x_{t+1}, a'_{t+1}) - \hat{Q}_t(x_t, a_t)$

  $\hat{Q}_{t+1}(x_t, a_t) = \hat{Q}_t(x_t, a_t) + \alpha_t \delta_t$

$a'_{t+1} \sim \varepsilon$ −greedy: on-policy

SARSA = $(s_t, a_t, r_t, s_{t+1}, a'_{t+1})$

Q-LEARNING VS. SARSA WITH FUNCTION APPROXIMATION

Both algorithms can be adapted to linear and non-linear FA by using the update rule

$$\theta_{t+1} = \theta_t + \alpha_t \delta_t \nabla_\theta Q_\theta(x_t, a_t)$$

Q-LEARNING VS. SARSA WITH FUNCTION APPROXIMATION

Both algorithms can be adapted to linear and non-linear FA by using the update rule

$$\theta_{t+1} = \theta_t + \alpha_t \delta_t \nabla_\theta Q_\theta(x_t, a_t)$$

- SARSA guarantees bounded error and tends to behave well in practice (may not find optimal policy though)

Q-LEARNING VS. SARSA WITH FUNCTION APPROXIMATION

Both algorithms can be adapted to linear and non-linear FA by using the update rule

$$\theta_{t+1} = \theta_t + \alpha_t \delta_t \nabla_\theta Q_\theta(x_t, a_t)$$

- SARSA guarantees bounded error and tends to behave well in practice (may not find optimal policy though)
- Q-learning may diverge catastrophically

Q-LEARNING VS. SARSA WITH FUNCTION APPROXIMATION

Both algorithms can be adapted to linear and non-linear FA by using the update rule

$$\theta_{t+1} = \theta_t + \alpha_t \delta_t \nabla_\theta Q_\theta(x_t, a_t)$$

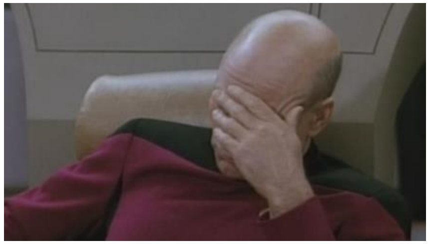
- SARSA guarantees bounded error and tends to behave well in practice (may not find optimal policy though)
- Q-learning may diverge catastrophically
  - Proposed fixes: gradient TD algorithms, emphatic TD algorithms, double Q-learning, soft Q-learning, G-learning,…

Q-LEARNING VS. SARSA WITH FUNCTION APPROXIMATION

Both algorithms can be adapted to linear and non-linear FA by using the update rule

$$\theta_{t+1} = \theta_t + \alpha_t \delta_t \nabla_\theta Q_\theta(x_t, a_t)$$

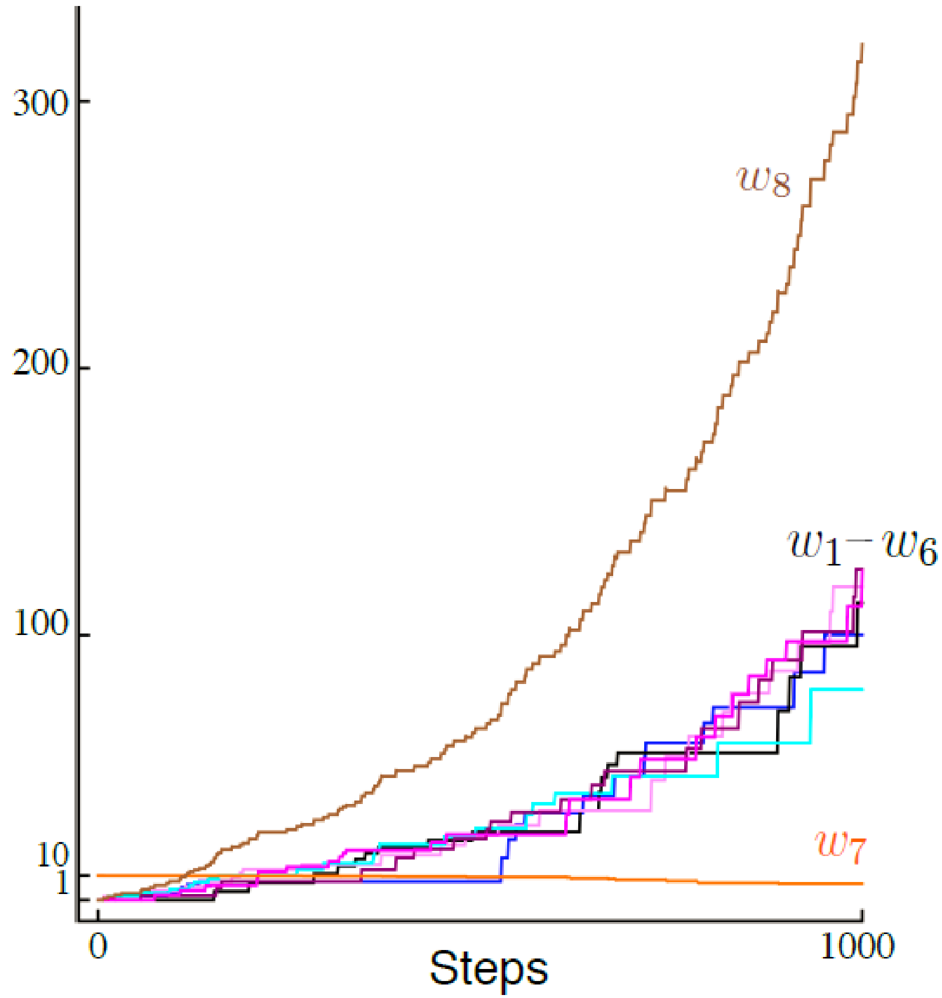
- SARSA guarantees bounded error and tends to behave well in practice (may not find optimal policy though)
- Q-learning may diverge catastrophically
  - Proposed fixes: gradient TD algorithms, emphatic TD algorithms, double Q-learning, soft Q-learning, G-learning,…
  - Practical solution: tune it until it works



DIVERGENCE OF OFF-POLICY TD LEARNING

The "deadly triad":

- Function approximation
- Bootstrapping
- Off-policy learning



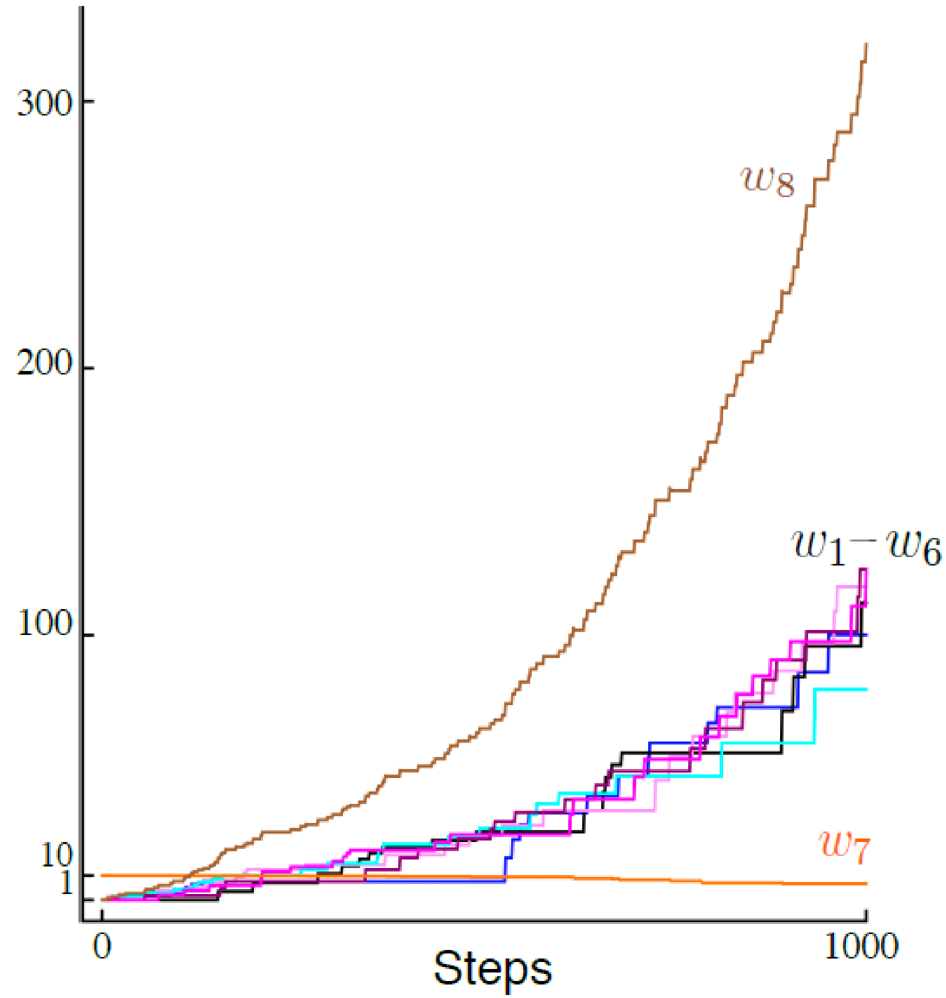
DIVERGENCE OF OFF-POLICY TD LEARNING

The "deadly triad":

- Function approximation
- Bootstrapping
- Off-policy learning

BUT

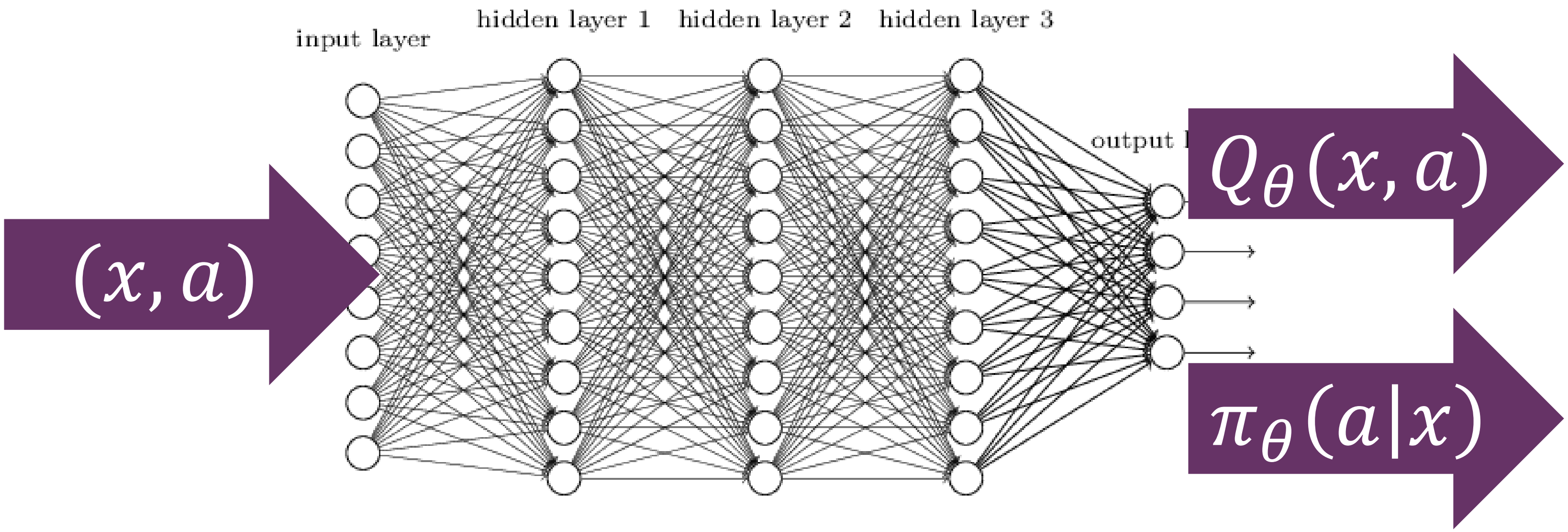
Divergence is typically not too extreme when behavior policy is close to evaluation policy and FA is linear



DEEP REINFORCEMENT LEARNING

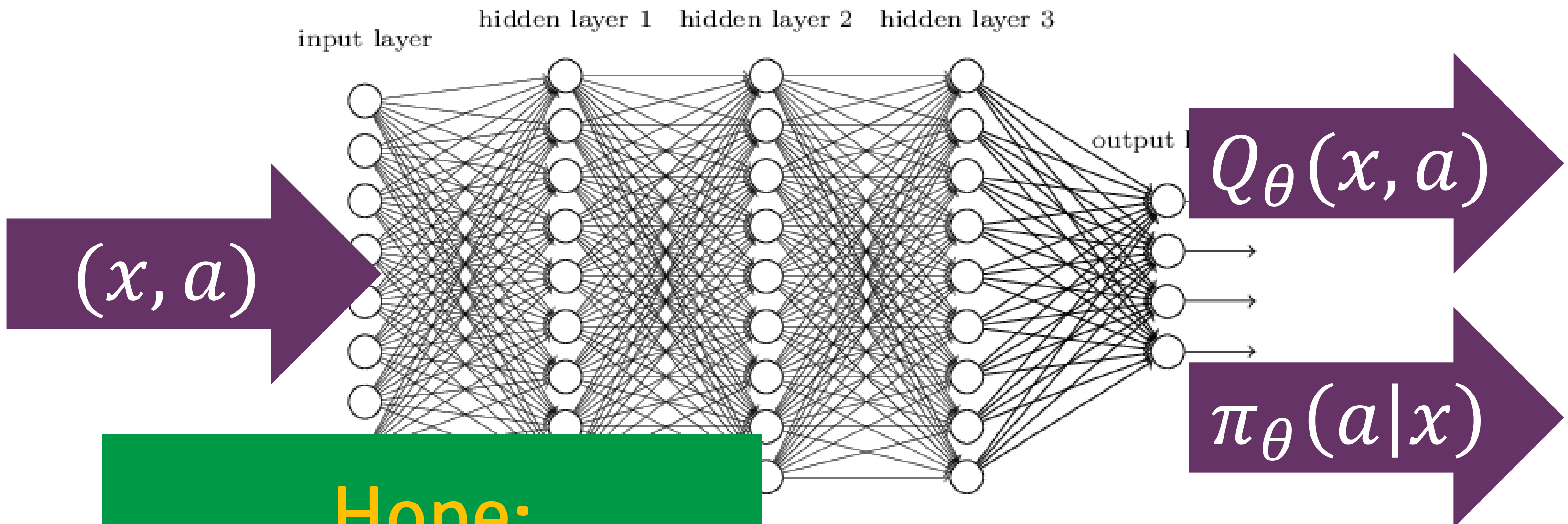
THE PROMISE OF DEEP REINFORCEMENT LEARNING

Parametrize Q-function/policy by a deep net



THE PROMISE OF DEEP REINFORCEMENT LEARNING

Parametrize Q-function/policy by a deep net

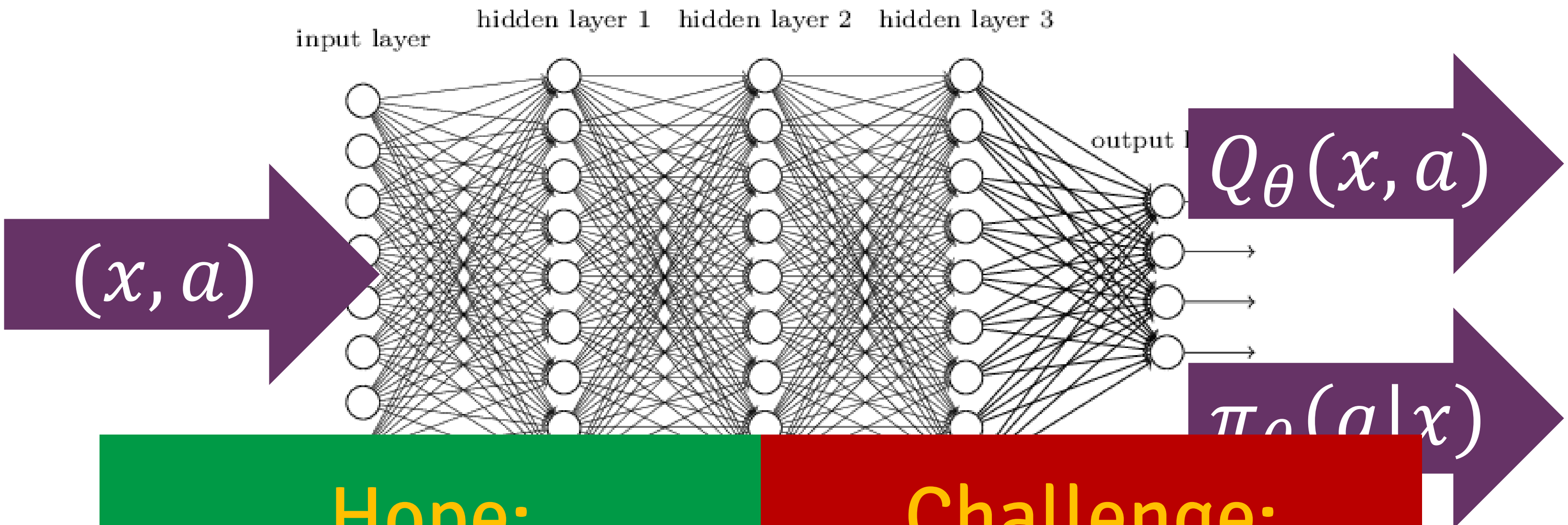


Hope:

Take advantage of representation power!

THE PROMISE OF DEEP REINFORCEMENT LEARNING

Parametrize Q-function/policy by a deep net



Hope:
Take advantage of representation power!

Challenge:
Existing RL methods difficult to generalize

LEAST-SQUARES TEMPORAL DIFFERENCE LEARNING (LSTD)

LSTD(0)

Input: trajectory $(x_t, a_t, r_t)_{t=1}^{T}$

$$\theta_T = A_T^{-1} b_T$$
$$\hat{V}_T = \theta_T^{\top} \phi$$

☹☹☹

Idea not directly applicable to non-linear function approximation!

☹☹☹

LSTD FOR NON-LINEAR FUNCTION APPROXIMATION?

Can we optimize Bellman error

$$L(\theta) = \mathbf{E}_{x\sim\mu}\left[\left(T^{\pi}V_{\theta}(x) - V_{\theta}(x)\right)^{2}\right]$$

by stochastic gradient descent????

LSTD FOR NON-LINEAR FUNCTION APPROXIMATION?

Can we optimize Bellman error

$$L(\theta) = \mathbf{E}_{x\sim\mu}\left[\left(T^{\pi}V_{\theta}(x) - V_{\theta}(x)\right)^2\right]$$

by stochastic gradient descent????

NO!!

Bellman error involves a double expectation:

$$L(\theta) = \mathbf{E}_X[\ell(\theta; X, \mathbf{E}_Y[Y|X])]$$

can't get unbiased gradients!

LSTD FOR NON-LINEAR FUNCTION APPROXIMATION?

Can we optimize Bellman error

$L(\theta) = \mathbf{E}_{x\sim\mu}\Big[\Big(\quad \Big)^2\Big]$

by stochastic g ?

The infamous
"double sampling"
issue of RL

NO!!

Bellman error involves a double expectation:

$$L(\theta) = \mathbf{E}_X[\ell(\theta; X, \mathbf{E}_Y[Y|X])]$$

can't get unbiased gradients!

TACKLING DOUBLE SAMPLING

- Saddle-point optimization:

$$\min_{\theta} \mathbf{E}[f(\theta; X, \mathbf{E}[Y|X])^2]$$

TACKLING DOUBLE SAMPLING

- Saddle-point optimization:

$$\min_{\theta} \mathbf{E}[f(\theta; X, \mathbf{E}[Y|X])^2] =$$

$$\min_{\theta} \max_{z} \mathbf{E}[z(X,Y) \cdot f(\theta; X, \mathbf{E}[Y|X])] - \mathbf{E}[z^2(X,Y)]$$

TACKLING DOUBLE SAMPLING

No nested expectation here!

- Saddle-point optimization:

$$\min_\theta \mathbf{E}[f(\theta; X, \mathbf{E}[Y|X])^2] =$$

$$\min_\theta \max_z \mathbf{E}[z(X,Y) \cdot f(\theta; X, \mathbf{E}[Y|X])] - \mathbf{E}[z^2(X,Y)]$$

$\Rightarrow$ "modified Bellman residual" (Antos et al. 2008), "Gradient TD" methods (Sutton et al. 2009), SBEED (Dai et al., 2018)

TACKLING DOUBLE SAMPLING

No nested expectation here!

- Saddle-point optimization:

$$\min_{\theta} \mathbf{E}[f(\theta; X, \mathbf{E}[Y|X])^2] =$$

$$\min_{\theta} \max_{z} \mathbf{E}[z(X,Y) \cdot f(\theta; X, \mathbf{E}[Y|X])] - \mathbf{E}[z^2(X,Y)]$$

⇒ "modified Bellman residual" (Antos et al. 2008), "Gradient TD" methods (Sutton et al. 2009), SBEED (Dai et al., 2018)

- Iterative optimization schemes

TACKLING DOUBLE SAMPLING

No nested expectation here!

- Saddle-point optimization:

$$\min_{\theta} \mathbf{E}[f(\theta; X, \mathbf{E}[Y|X])^2] =$$

$$\min_{\theta} \max_{z} \mathbf{E}[z(X,Y) \cdot f(\theta; X, \mathbf{E}[Y|X])] - \mathbf{E}[z^2(X,Y)]$$

$\Rightarrow$ "modified Bellman residual" (Antos et al. 2008), "Gradient TD" methods (Sutton et al. 2009), SBEED (Dai et al., 2018)

- Iterative optimization schemes

FITTED POLICY EVALUATION

Idea: compute sequence of value functions by minimizing

$$L_n(\hat{V}; \hat{V}_k) = \frac{1}{n}\sum_{t=1}^{n}\left(r_t + \hat{V}_k(x_{t+1}) - \hat{V}(x_t)\right)^2$$

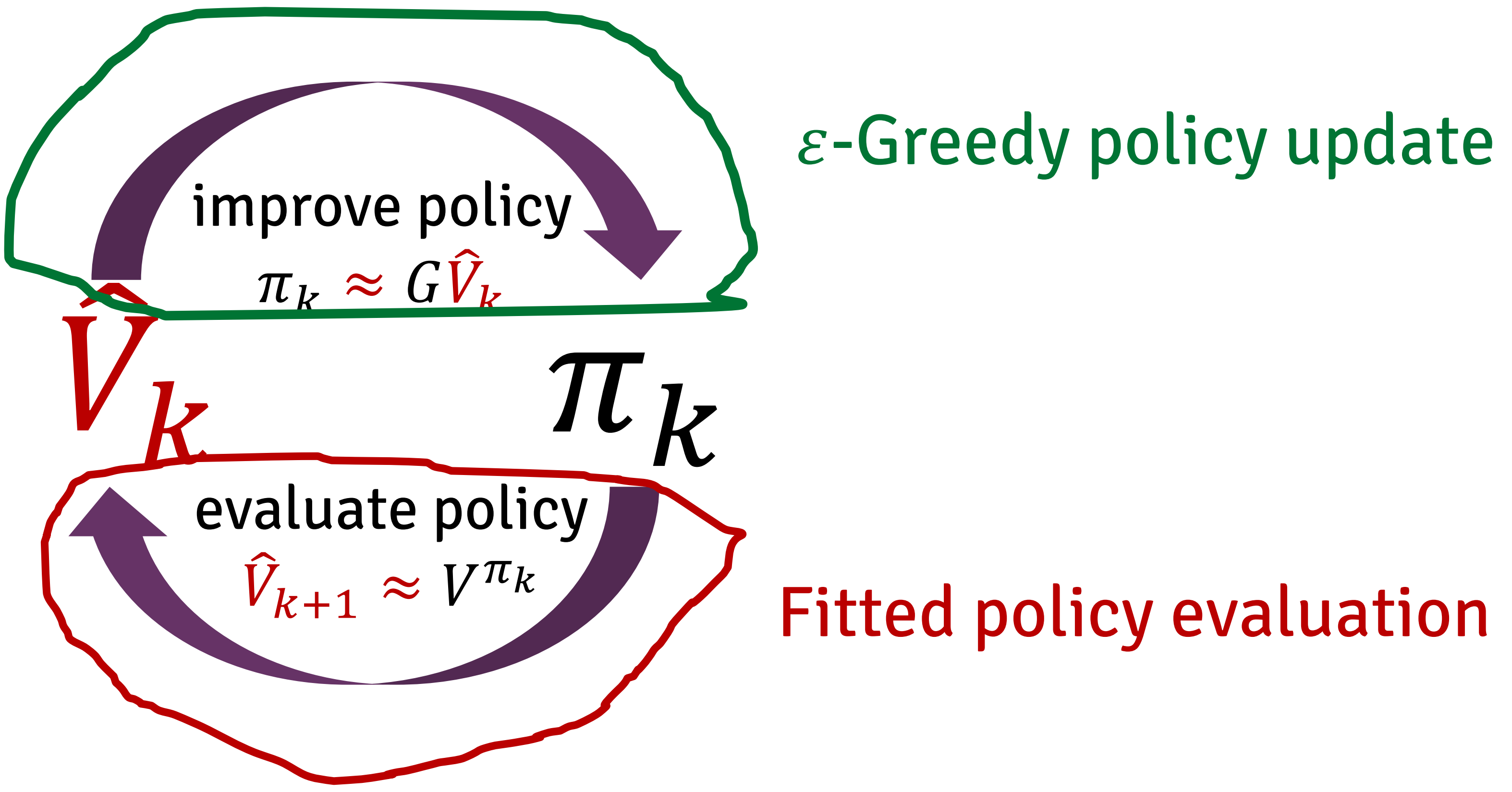
FITTED POLICY EVALUATION

Idea: compute sequence of value functions by minimizing

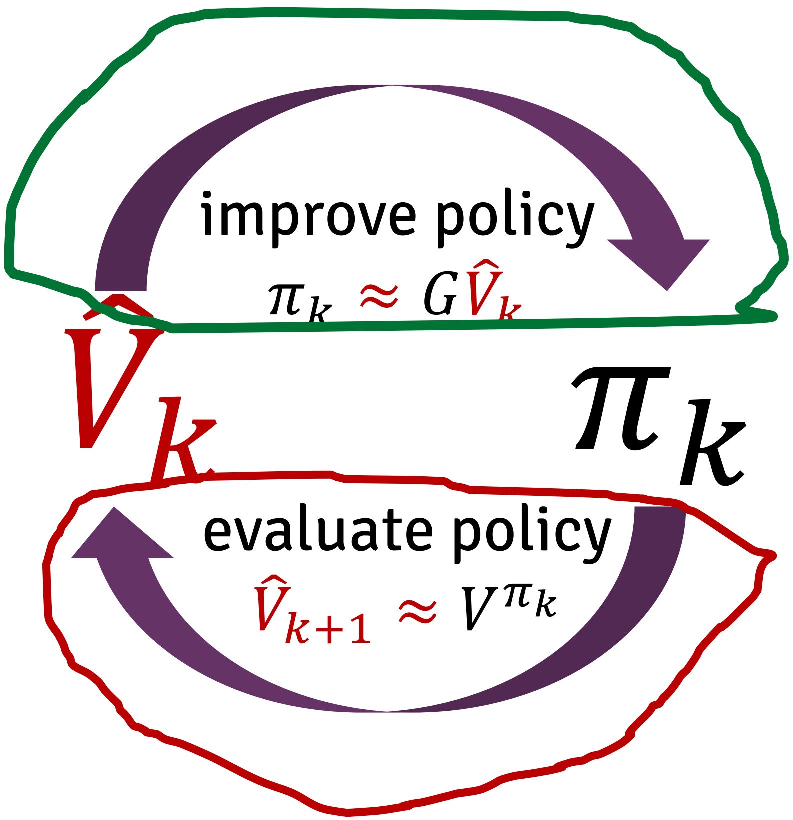
$$L_n(\hat{V}; \hat{V}_k) = \frac{1}{n}\sum_{t=1}^{n}\Big(\underbrace{r_t + \hat{V}_k(x_{t+1})}_{\text{Target}} - \underbrace{\hat{V}(x_t)}_{\text{Free variable}}\Big)^2$$

This can be finally treated as a regression problem & solved by SGD!

FITTED POLICY ITERATION



FITTED POLICY ITERATION



ε-Greedy policy update

Computing policy needs model of P… better use Q-functions!

Fitted policy evaluation

FITTED VALUE ITERATION

Idea: compute sequence of Q-value functions by minimizing

$$L_n(\hat{Q}; \hat{Q}_k) = \frac{1}{n}\sum_{t=1}^{n}\Big(\underbrace{r_t + \max_a \hat{Q}_k(x_{t+1}, a)}_{\text{Target}} - \underbrace{\hat{Q}(x_t, a_t)}_{\text{Free variable}}\Big)^2$$

FITTED VALUE ITERATION

Fitted value iteration

Input: function space F, $\hat{Q}_0 \in F$

For $k = 0,1, \ldots,$

- $\pi_k = G_\varepsilon \hat{Q}_k$
- generate trajectory
$$(x_t, a_t, r_t)_{t=1}^n \sim \pi_k$$
- compute
$$\hat{Q}_{k+1} = \underset{\hat{Q} \in F}{\operatorname{argmin}} L_n(\hat{Q}; \hat{Q}_k)$$

FITTED VALUE ITERATION

Fitted value iteration

Input: function space F, $\hat{Q}_0 \in F$

For $k = 0,1, \ldots,$

- $\pi_k = G_\varepsilon \hat{Q}_k$
- generate trajectory
  $$(x_t, a_t, r_t)_{t=1}^{n} \sim \pi_k$$
- compute
  $$\hat{Q}_{k+1} = \operatorname*{argmin}_{\hat{Q} \in F} L_n\big(\hat{Q}; \hat{Q}_k\big)$$

Computing policy is trivial!

FITTED VALUE ITERATION

Fitted value iteration

Input: function space F, $\hat{Q}_0 \in F$

For $k = 0,1,\ldots,$

- $\pi_k = G_\varepsilon \hat{Q}_k$
- generate trajectory
$$(x_t, a_t, r_t)_{t=1}^{n} \sim \pi_k$$
- compute
$$\hat{Q}_{k+1} = \underset{\hat{Q} \in F}{\operatorname{argmin}}\, L_n\big(\hat{Q}; \hat{Q}_k\big)$$

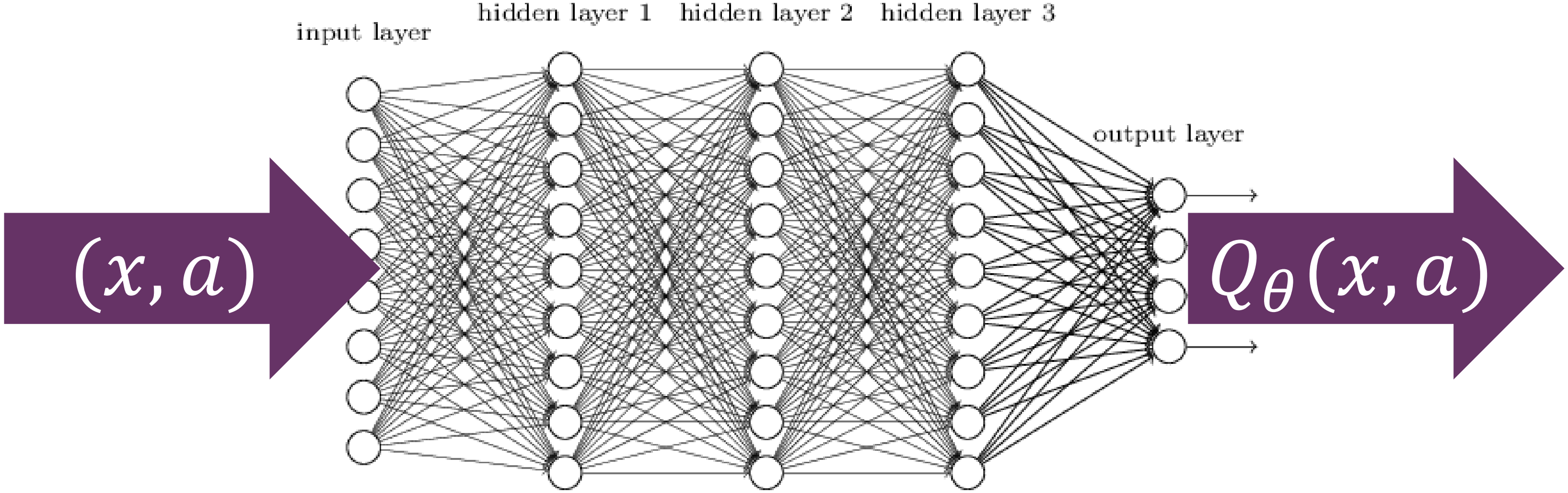
Computing policy is trivial!

Convergence can be guaranteed!

under very technical assumptions…

DEEP Q NETWORKS

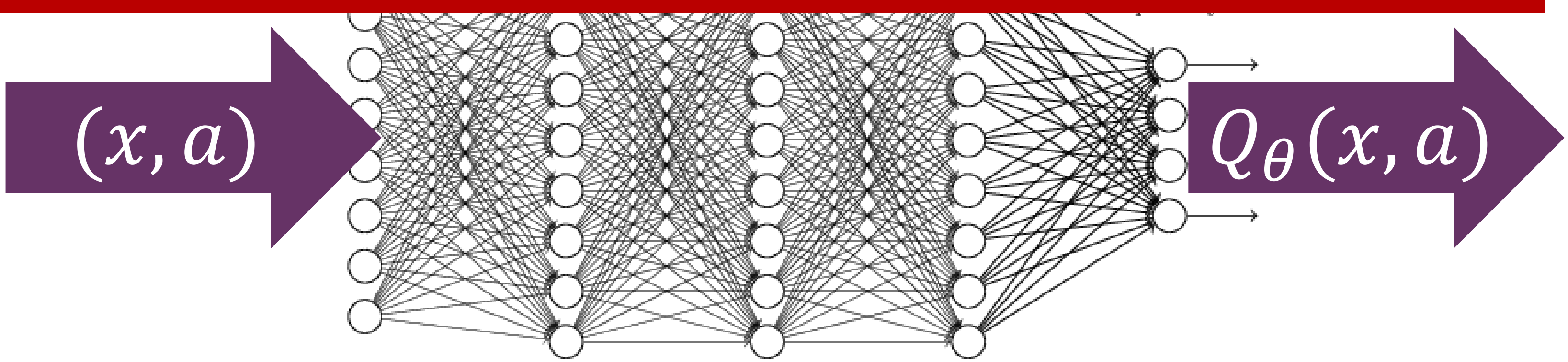
Parametrize Q-function by a deep neural net



DEEP Q NETWORKS

Minimize the loss

$$\mathbf{E}_{(X,A,R,X')\sim D}\left[\left(R+\gamma\max_{b} Q_{\theta_k}(X',b)-Q_\theta(X,A)\right)^2\right]$$



DEEP Q NETWORKS

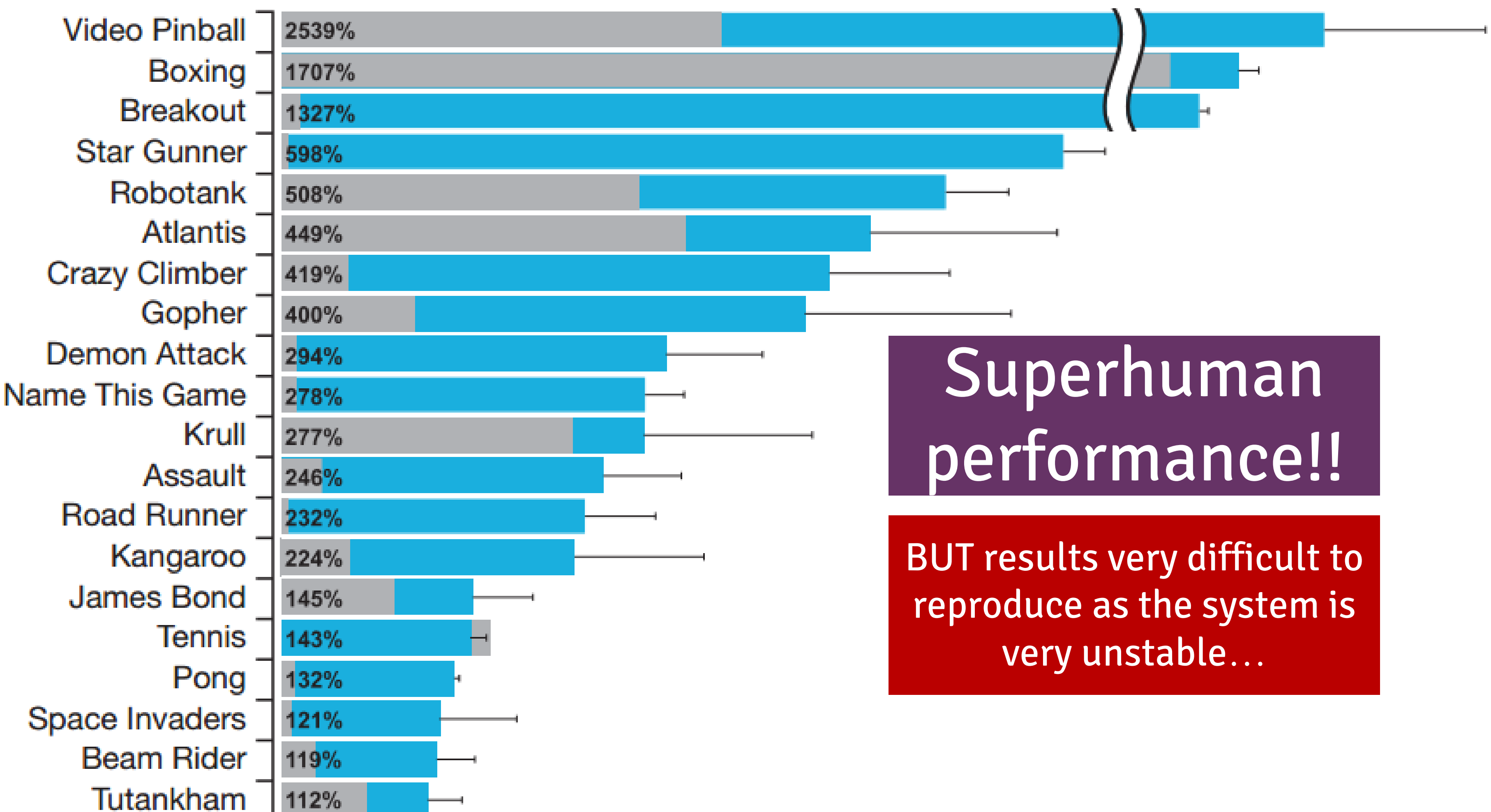
Minimize the loss

$$\mathbf{E}_{(X,A,R,X')\sim D}\left[\left(R+\gamma\max_b Q_{\theta_k}(X',b)-Q_\theta(X,A)\right)^2\right]$$

+ training tricks:

- Store transitions (x,a,r,x') in replay buffer D to break dependence on recent samples
- Compute small updates by mini-batch stochastic gradient descent
- Use an older parameter vector θ_{k-m} in target to avoid oscillations
- …

DEEP Q NETWORKS FOR PLAYING ATARI



Superhuman performance!!

BUT results very difficult to reproduce as the system is very unstable…

THIS SHORT COURSE: A PRIMAL-DUAL VIEW

part 1

- Markov decision processes
  - Value functions and optimal policies
- Primal view: Dynamic programming
  - Policy evaluation, value and policy iteration
  - Value-function-based methods
    - Temporal differences, Q-learning, LSTD, deep Q networks,…

part 2

- Dual view: Linear programming
  - LP duality in MDPs
  - Direct policy optimization methods
    - Policy gradients, REPS, TRPO,…

THIS SHORT COURSE:
A PRIMAL-DUAL VIEW

part 1

- Markov decision processes
  - Value functions and optimal policies
- Primal vi… g
  - Policy ev… tion
  - Value-fu…
    - Temporal … works,…

But first:
some more notation ☺

part 2

- Dual view: Linear programming
  - LP duality in MDPs
  - Direct policy optimization methods
    - Policy gradients, REPS, TRPO,…

POLICIES AND TRAJECTORY DISTRIBUTIONS

Policy: mapping from histories to actions

$$\pi: x_1, a_1, x_2, a_2, \dots, x_t \mapsto a_t$$

Stationary policy: mapping from states to actions
(no dependence on history or t)

$$\pi: x \mapsto a$$

Let $\tau = (x_1, a_1, x_2, a_2, \dots)$ be a trajectory generated by running π in the MDP $\tau \sim (\pi, P)$:

- $a_t = \pi(x_t, a_{t-1}, x_{t-1}, \dots, x_1)$
- $x_{t+1} \sim P(\cdot\,|x_t, a_t)$

Expectation under this distribution: $\mathbf{E}_\pi[\cdot]$

POLICIES AND TRAJECTORY DISTRIBUTIONS

Stationary stochastic policy: mapping from states to action distributions

$$\pi: A \times X \to [0,1]$$

where

$$\pi(a|x) = P[a_t = a | x_t = x]$$

Let $\tau = (x_1, a_1, x_2, a_2, \dots)$ be a trajectory generated by running π in the MDP $\tau \sim (\pi, P)$:

- $a_t = \pi(x_t, a_{t-1}, x_{t-1}, \dots, x_1)$
- $x_{t+1} \sim P(\cdot | x_t, a_t)$

Expectation under this distribution: $\mathbf{E}_\pi[\cdot]$

POLICIES AND TRAJECTORY DISTRIBUTIONS

Stationary stochastic policy: mapping from states to action distributions

$$\pi: A \times X \to [0,1]$$

where

$$\pi(a|x) = P[a_t = a | x_t = x]$$

Let $\tau = (x_1, a_1, x_2, a_2, \dots)$ be a trajectory generated by running π in the MDP $\tau \sim (\pi, P)$:

- $a_t \sim \pi(\cdot\,|x_t)$
- $x_{t+1} \sim P(\cdot\,|x_t, a_t)$

Expectation under this distribution: $\mathbf{E}_\pi[\cdot]$

ANOTHER PERSPECTIVE ON DISCOUNTED REWARDS

Discounted MDPs:

- No terminal state
- Discount factor $\gamma \in (0,1)$
- GOAL: maximize total discounted reward

$$R_\gamma = \mathbf{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$$

ANOTHER PERSPECTIVE ON DISCOUNTED REWARDS

Discounted MDPs:

- No terminal state, initial state $x_0 \sim \mu_0$
- Discount factor $\gamma \in (0,1)$
- GOAL: maximize total discounted reward

$$R_\gamma = \mathbf{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$$

ANOTHER PERSPECTIVE ON DISCOUNTED REWARDS

Discounted MDPs:

- No terminal state, initial state $x_0 \sim \mu_0$
- Discount factor $\gamma \in (0,1)$
- GOAL: maximize total discounted reward

$$R_\gamma = \mathbf{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$$

Observe: the discounted reward of a policy is

$$R_\gamma^\pi = \mathbf{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(x_t, a_t)]$$

ANOTHER PERSPECTIVE ON DISCOUNTED REWARDS

Discounted MDPs:

- No terminal state, initial state $x_0 \sim \mu_0$
- Discount factor $\gamma \in (0,1)$
- GOAL: maximize total discounted reward

$$R_\gamma = \mathbf{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$$

Observe: the discounted reward of a policy is

$$R_\gamma^\pi = \mathbf{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(x_t, a_t)]$$
$$= \mathbf{E}_\pi\left[\sum_{x,a} \sum_{t=0}^{\infty} \gamma^t \mathbf{1}_{\{x_t=x, a_t=a\}} r(x,a)\right]$$

ANOTHER PERSPECTIVE ON DISCOUNTED REWARDS

Discounted MDPs:

- No terminal state, initial state $x_0 \sim \mu_0$
- Discount factor $\gamma \in (0,1)$
- GOAL: maximize total discounted reward

$$R_\gamma = \mathbf{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$$

Observe: the discounted reward of a policy is

$$\begin{aligned} R_\gamma^\pi &= \mathbf{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r(x_t, a_t)\right] \\ &= \mathbf{E}_\pi\left[\sum_{x,a} \sum_{t=0}^{\infty} \gamma^t \mathbf{1}_{\{x_t = x, a_t = a\}} r(x,a)\right] \\ &= \sum_{x,a} \sum_{t=0}^{\infty} \gamma^t \mathbf{P}_\pi[x_t = x, a_t = a]\, r(x,a) \end{aligned}$$

ANOTHER PERSPECTIVE ON DISCOUNTED REWARDS

Discounted MDPs:

- No terminal state, initial state $x_0 \sim \mu_0$
- Discount factor $\gamma \in (0,1)$
- GOAL: maximize total discounted reward

$$R_\gamma = \mathbf{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$$

Observe: the discounted reward of a policy is

$$\begin{aligned}
R_\gamma^\pi &= \mathbf{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r(x_t, a_t)\right] \\
&= \mathbf{E}_\pi\left[\sum_{x,a} \sum_{t=0}^{\infty} \gamma^t \mathbf{1}_{\{x_t = x, a_t = a\}}\, r(x,a)\right] \\
&= \sum_{x,a} \sum_{t=0}^{\infty} \gamma^t \mathbf{P}_\pi[x_t = x, a_t = a]\, r(x,a) \\
&= \sum_{x,a} \mu_\pi(x,a) r(x,a)
\end{aligned}$$

ANOTHER PERSPECTIVE ON DISCOUNTED REWARDS

Discounted MDPs:

- No terminal state, initial state $x_0 \sim \mu_0$
- Discount factor $\gamma \in (0,1)$
- GOAL: maximize total discounted reward

$$R_\gamma = \mathbf{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$$

Observe: the discounted reward of a policy is

$$\begin{aligned} R_\gamma^\pi &= \mathbf{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r(x_t, a_t)\right] \\ &= \mathbf{E}_\pi\left[\sum_{x,a} \sum_{t=0}^{\infty} \gamma^t \mathbf{1}_{\{x_t = x, a_t = a\}} r(x,a)\right] \\ &= \sum_{x,a} \sum_{t=0}^{\infty} \gamma^t \mathbf{P}_\pi[x_t = x, a_t = a]\, r(x,a) \\ &= \sum_{x,a} \mu_\pi(x,a) r(x,a) = \langle \mu_\pi, r \rangle \end{aligned}$$

ANOTHER PERSPECTIVE ON DISCOUNTED REWARDS

Discounted MDPs:

- No terminal state, initial state $x_0 \sim \mu_0$
- Discount factor $\gamma \in (0,1)$
- GOAL: maximize total discounted reward

$$R_\gamma = \mathbf{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$$

Observe: the discounted reward of a policy is

$$R_\gamma^\pi = \langle \mu_\pi, r \rangle$$

μ_π = the discounted occupancy measure induced by policy π:

$$\mu_\pi(x,a) = \sum_{t=0}^{\infty} \gamma^t \mathbf{P}_\pi[x_t = x, a_t = a]$$

ANOTHER PERSPECTIVE ON DISCOUNTED REWARDS

Discounted MDPs:

- No terminal state, i
- Discount factor $\gamma \in$
- GOAL: maximize tot

$$R_\gamma = \mathbf{E}[\ \ _{=0} \gamma^t r_t]$$

A linear optimization problem?!

Observe: the discounted reward of a policy is

$$R_\gamma^\pi = \langle \mu_\pi, r \rangle$$

μ_π = the discounted occupancy measure induced by policy π:

$\mu_\pi(x, a) = \sum_{t=0}^{\infty} \gamma^t \mathbf{P}_\pi[x_t = x, a_t = a]$

TOWARDS A LINEAR-PROGRAM FORMULATION

Theorem

A function μ is a discounted occupancy measure of some (stationary stochastic) policy π if and only if it satisfies

$$\sum_{a'} \mu(x', a') = (1 - \gamma) \sum_{a'} \mu_0(x', a') + \gamma \sum_{x,a} P(x'|x, a)\mu(x, a)$$

and $\sum_{x,a} \mu(x, a) = 1/(1 - \gamma)$.

TOWARDS A LINEAR-PROGRAM FORMULATION

Theorem

A function μ is a discounted occupancy measure of some (stationary stochastic) policy π if and only if it satisfies

$$\sum_{a'} \mu(x', a') = (1-\gamma) \sum_{a'} \mu_0(x', a') + \gamma \sum_{x,a} P(x' | x, a) \mu(x, a)$$

and $\sum_{x,a} \mu(x, a) = 1/(1-\gamma)$.

Linear constraints!

Define $\Delta =$ the set of occupancy measures μ.

OPTIMIZATION IN MDPS AS A LINEAR PROGRAM

LP

$$R_\gamma^* = \max_{\mu \in \Delta} \langle \mu, r \rangle$$

OPTIMIZATION IN MDPS AS A LINEAR PROGRAM

LP

$$R_\gamma^* = \max_{\mu \in \Delta} \langle \mu, r \rangle$$

LP'

$$R_\gamma^* = \min_{V \in \mathbb{R}^X} \langle \mu_0, V \rangle$$

s.t. $V(x) \geq r(x,a) + \gamma \sum_y P(y|x,a) V(y) \quad (\forall x, a)$

OPTIMIZATION IN MDPS AS A LINEAR PROGRAM

Dual LP

$$R_\gamma^* = \max_{\mu \in \Delta} \langle \mu, r \rangle$$

Primal LP

$$R_\gamma^* = \min_{V \in \mathbb{R}^X} \langle \mu_0, V \rangle$$

$$\text{s.t. } V(x) \geq r(x,a) + \gamma \sum_y P(y|x,a) V(y) \quad (\forall x, a)$$

*names are due to tradition

OPTIMIZATION IN MDPS AS A LINEAR PROGRAM

Dual LP

$$R_\gamma^* = \max_{\mu \in \Delta} \langle \mu, r \rangle$$

Primal LP ≡ The Bellman opt. equations

$$V^*(x) = \max_a \left\{ r(x,a) + \gamma \sum_y P(y|x,a) V^*(y) \right\}$$

Assuming $\mu_0 > 0$

*names are due to tradition

OPTIMIZATION IN MDPS AS A LINEAR PROGRAM

A single numerical objective to optimize!

Dual LP

$$R_\gamma^* = \max_{\mu \in \Delta} \langle \mu, r \rangle$$

Primal LP $\equiv$ The Bellman opt. equations

$$V^*(x) = \max_a \left\{ r(x,a) + \gamma \sum_y P(y|x,a) V^*(y) \right\}$$

Assuming $\mu_0 > 0$

*names are due to tradition

OPTIMAL SOLUTIONS OF THE LP

Theorem

There exists a basic solution $\mu^* \in \Delta$ to the dual LP.

OPTIMAL SOLUTIONS OF THE LP

Theorem
There exists a basic solution $\mu^* \in \Delta$ to the dual LP.

"Proof":

objective $\langle \mu, r \rangle$ is bounded on nonempty Δ

$$\Rightarrow$$

there exists optimal solution $\mu^* \in \Delta$

$$\Rightarrow$$

there exists basic solution $\mu^* \in \Delta$

OPTIMAL SOLUTIONS OF THE LP

Theorem

There exists a basic solution $\mu^* \in \Delta$ to the dual LP.

"Proof":

objective $\langle \mu, r \rangle$ is bounded on nonempty Δ

$$\Rightarrow$$

there exists optimal solution $\mu^* \in \Delta$

$$\Rightarrow$$

A "corner" of Δ

there exists basic solution $\mu^* \in \Delta$

EXTRACTING A POLICY

? **Question:** how do we extract a policy from a feasible $\mu \in \Delta$?

EXTRACTING A POLICY

? **Question:** how do we extract a policy from a feasible $\mu \in \Delta$?

Corollary

Assume that $\mu_0(x) > 0$ for all $x \in X$. Then, for any occupancy measure $\mu \in \Delta$, there exists a unique policy π such that $\mu = \mu_\pi$, given by

$$\pi(a|x) = \frac{\mu(x,a)}{\sum_b \mu(x,b)}.$$

EXTRACTING A POLICY

? **Question:** how do we extract a policy from a feasible $\mu \in \Delta$?

Corollary

Assume that $\mu_0(x) > 0$ for all $x \in X$. Then, for any occupancy measure $\mu \in \Delta$, there exists a unique policy π such that $\mu = \mu_\pi$, given by

$$\pi(a|x) = \frac{\mu(x,a)}{\sum_b \mu(x,b)}.$$

Well-defined since $\sum_b \mu(x,b) > 0$ by assumption

EXTRACTING A POLICY

? **Question:** how do we extract a policy from a feasible $\mu \in \Delta$?

Corollary

Assume that $\mu_0(x) > 0$ for all $x \in X$. Then, for any occupancy measure $\mu \in \Delta$, there exists a unique policy π such that $\mu = \mu_\pi$, given by

$$\pi(a|x) = \frac{\mu(x,a)}{\sum_b \mu(x,b)}.$$

Basic solutions
$\Leftrightarrow$
Deterministic policies

Well-defined since
$\sum_b \mu(x,b) > 0$ by assumption

LINEAR PROGRAMMING FOR MDPS

"Why don't they teach this in school?!?"

- Needs some strange conditions that DP theory does not ($\mu_0 > 0$ for existence results and for optimal policy)
- Temporal aspect is rather abstract
- Less intuitive for control theorists and computational neuroscience folks (classic RL crowd)

LINEAR PROGRAMMING FOR MDPS

"Why don't they teach this in school?!?"

- Needs some strange conditions that DP theory does not ($\mu_0 > 0$ for existence results and for optimal policy)
- Temporal aspect is rather abstract
- Less intuitive for control theorists and computational neuroscience folks (classic RL crowd)

Advantages

- Defining optimality is very simple (no value functions, no fixed points, etc.)
- Easily comprehensible with an optimization background (single numerical objective)
- Powerful tool for developing algorithms

LINEAR PROGRAMMING FOR MDPS

"Why don't they teach this in school?!?"

- Needs some strange conditions that DP theory does not ($\mu_0 > 0$ for existence results and for optimal policy)
- Temporal aspect is rather abstract
- Less intuitive for control theorists and computational neuroscience folks (classic RL crowd)

Advantages

- Defining optimality is very simple (no value functions, no fixed points, etc.)
- Easily comprehensible with an optimization background (single numerical objective)
- **Powerful tool for developing algorithms**

THIS SHORT COURSE:
A PRIMAL-DUAL VIEW

part 1

- Markov decision processes
  - Value functions and optimal policies
- Primal view: Dynamic programming
  - Policy evaluation, value and policy iteration
  - Value-function-based methods
    - Temporal differences, Q-learning, LSTD, deep Q networks,…

part 2

- Dual view: Linear programming
  - LP duality in MDPs
  - Direct policy optimization methods
    - Policy gradients, REPS, TRPO,…

DIRECT POLICY OPTIMIZATION

Idea: derive algorithms by thinking of $\mu \in \Delta$ as the decision variable!

DIRECT POLICY OPTIMIZATION

Idea: derive algorithms by thinking of $\mu \in \Delta$ as the decision variable!

Examples

- Policy gradient methods
  $=$ gradient descent on $-R_\gamma^\pi$
- Relative Entropy Policy Search (REPS)
  $=$ mirror descent on $-R_\gamma^\pi$
- Trust-region policy optimization (TRPO)
  $=$ mirror descent on (a surrogate of) $-R_\gamma^\pi$

DIRECT POLICY OPTIMIZATION

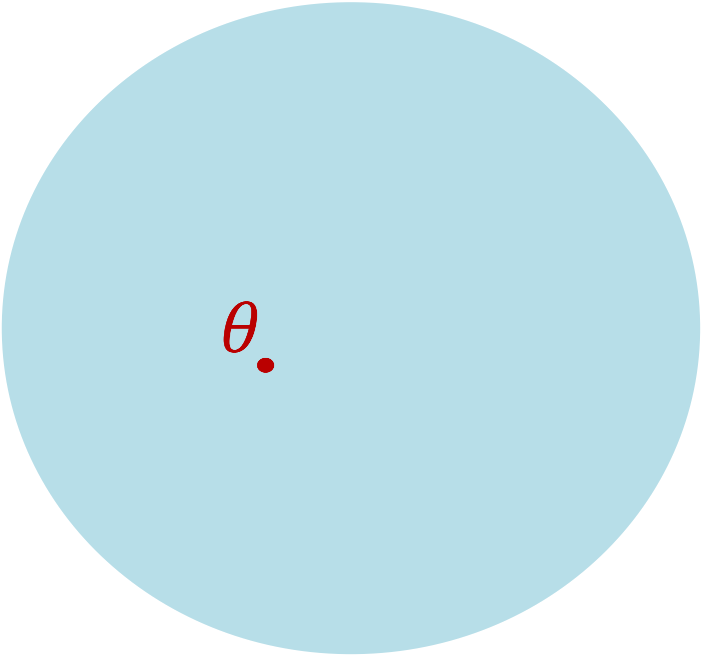
Idea: derive algorithms by thinking of $\mu \in \Delta$ as the decision variable!

Examples

- Policy gradient methods
  $=$ gradient descent on $-R_\gamma^\pi$
- Relative Entropy Policy Search (REPS)
  $=$ mirror descent on $-R_\gamma^\pi$
- Trust-region policy optimization (TRPO)
  $=$ mirror descent on (a surrogate of) $-R_\gamma^\pi$

POLICY GRADIENT METHODS

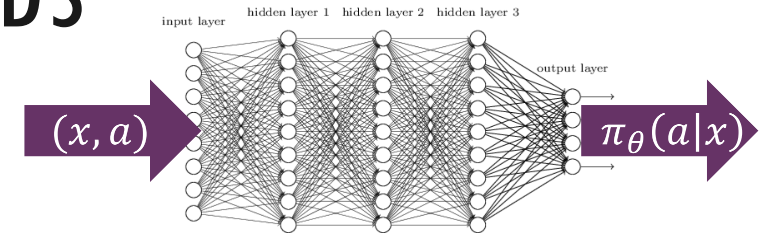
Parameter space Θ



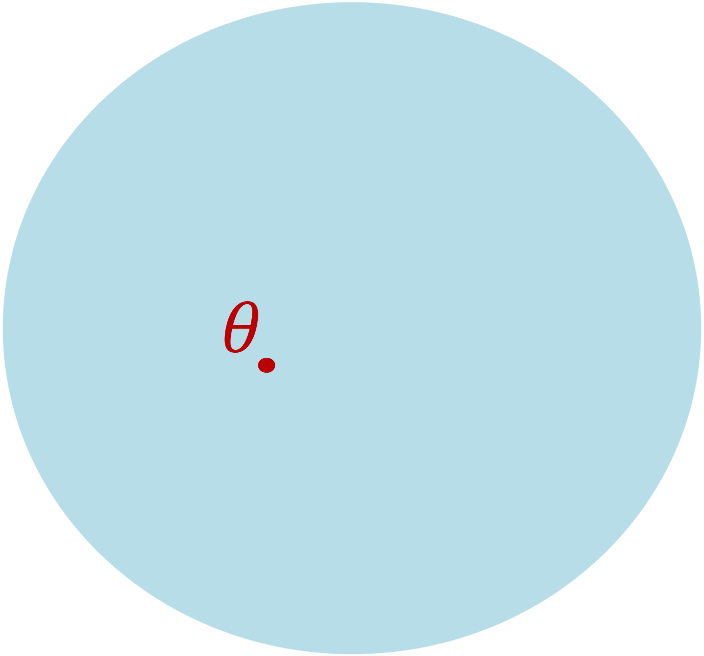
- Construct mapping

$$\theta \mapsto \pi_\theta$$

POLICY GRADIENT METHODS



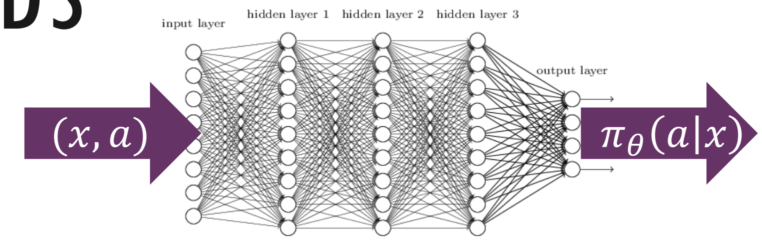
Parameter space Θ

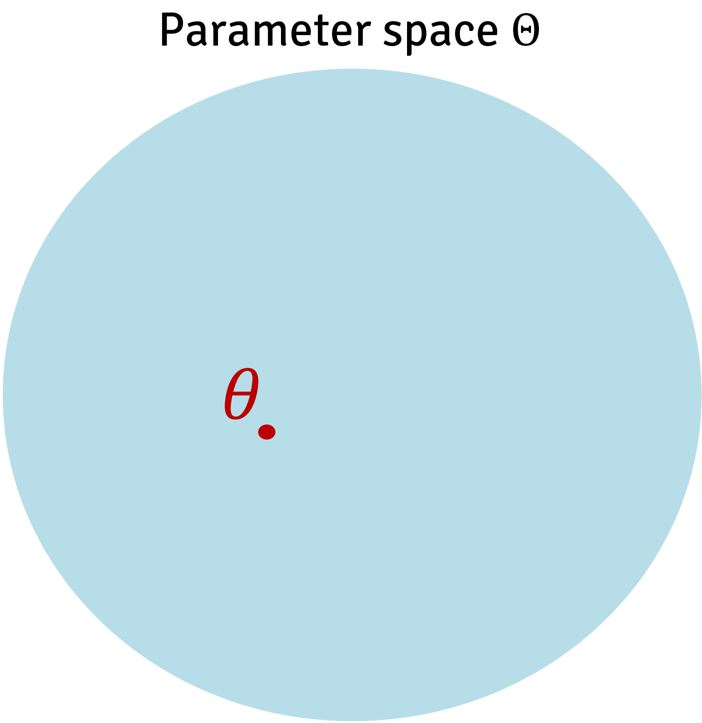


- Construct mapping

$$\theta \mapsto \pi_\theta$$

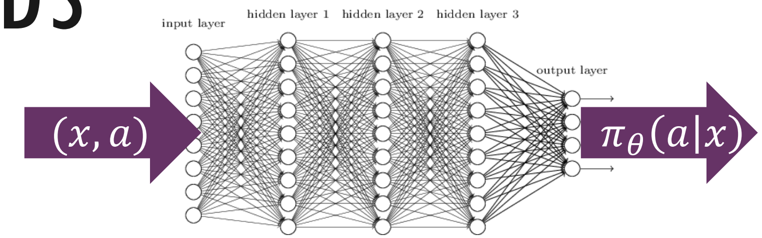
POLICY GRADIENT METHODS



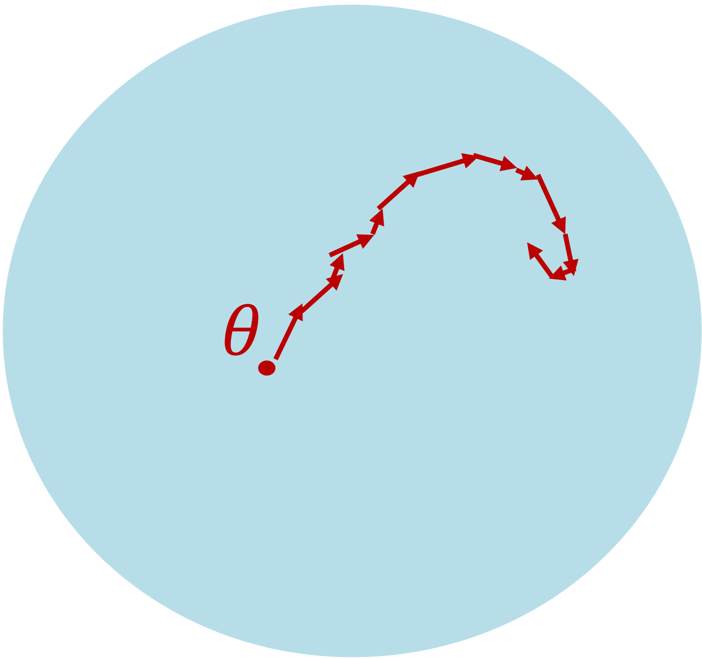


- Construct mapping
$$\theta \mapsto \pi_\theta$$
- Define objective function:
$$\rho(\theta) = R_\gamma^{\pi_\theta}$$

POLICY GRADIENT METHODS

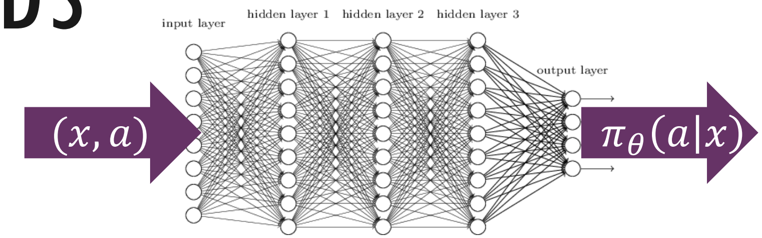


Parameter space Θ

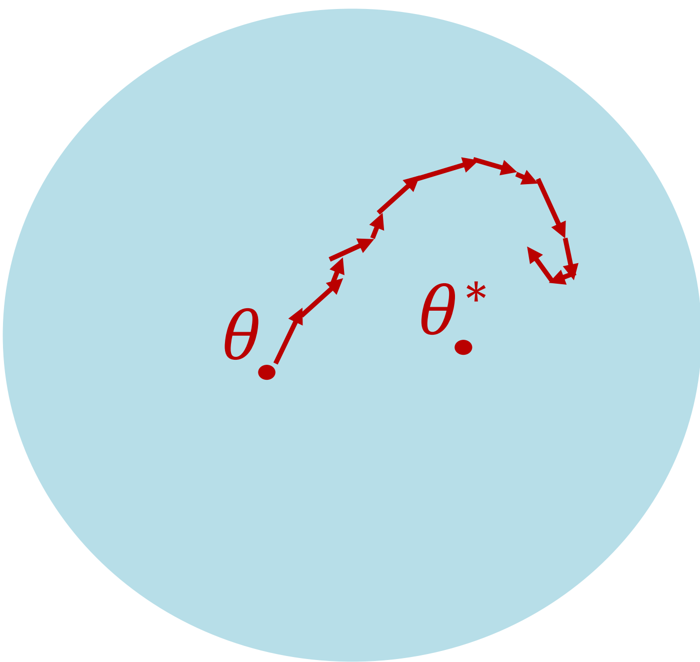


- Construct mapping
$$\theta \mapsto \pi_\theta$$
- Define objective function:
$$\rho(\theta) = R_\gamma^{\pi_\theta}$$
- Update parameters by gradient ascent:
$$\theta_{k+1} = \theta_k + \alpha_k \nabla_\theta \rho(\theta_k)$$

POLICY GRADIENT METHODS



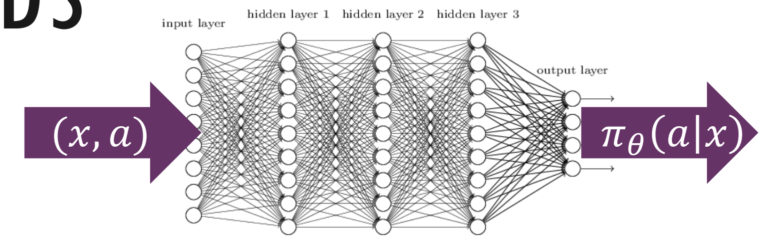
Parameter space Θ



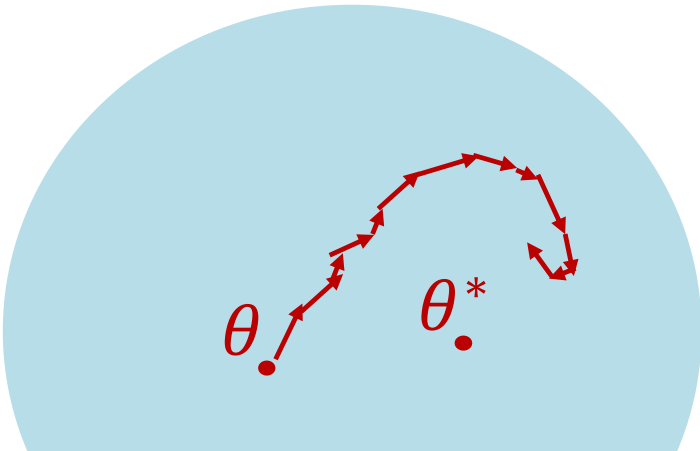
- Construct mapping
$$\theta \mapsto \pi_\theta$$
- Define objective function:
$$\rho(\theta) = R_\gamma^{\pi_\theta}$$
- Update parameters by gradient ascent:
$$\theta_{k+1} = \theta_k + \alpha_k \nabla_\theta \rho(\theta_k)$$

… and hope for convergence

POLICY GRADIENT METHODS



Parameter space Θ



How can we estimate the gradients?

- Construct mapping
$$\theta \mapsto \pi_\theta$$
- Define objective function:
$$\rho(\theta) = R_\gamma^{\pi_\theta}$$
- Update parameters by gradient ascent:
$$\theta_{k+1} = \theta_k + \alpha_k \nabla_\theta \rho(\theta_k)$$

… and hope for convergence

THE POLICY GRADIENT THEOREM

Theorem

$$\nabla_\theta \rho(\theta) = \sum_x \mu_\theta(x) \sum_a \nabla_\theta \pi_\theta(a|x) Q^{\pi_\theta}(x,a)$$

THE POLICY GRADIENT THEOREM

Theorem

$$\nabla_\theta \rho(\theta) = \sum_x \mu_\theta(x) \sum_a \nabla_\theta \pi_\theta(a|x) Q^{\pi_\theta}(x, a)$$

Corollary

Assuming that $\pi_\theta(a|x) > 0$ for all x, a,

$$\nabla_\theta \rho(\theta) = \sum_{x,a} \mu_\theta(x) \pi_\theta(a|x) \left(\nabla_\theta \log \pi_\theta(a|x)\, Q^{\pi_\theta}(x, a)\right)$$

THE POLICY GRADIENT THEOREM

Theorem

$$\nabla_\theta \rho(\theta) = \sum_x \mu_\theta(x) \sum_a \nabla_\theta \pi_\theta(a|x) Q^{\pi_\theta}(x,a)$$

Corollary

Assuming that $\pi_\theta(a|x) > 0$ for all x, a,

$$\nabla_\theta \rho(\theta) = \mathbf{E}_{(\tilde{x},\tilde{a})\sim\mu_\theta\pi_\theta}[\nabla_\theta \log \pi_\theta(\tilde{a}|\tilde{x})\, Q^{\pi_\theta}(\tilde{x},\tilde{a})]$$

THE POLICY GRADIENT THEOREM

Theorem

$$\nabla_\theta \rho(\theta) = \sum \mu_\theta(x) \sum \nabla_\theta \pi_\theta(a|x) Q^{\pi_\theta}(x, a)$$

Gradient can be written as an expectation!!!!

Corollary

Assuming that $\pi_\theta(a|x) > 0$ for all x, a,

$$\nabla_\theta \rho(\theta) = \mathbf{E}_{(\tilde{x},\tilde{a})\sim\mu_\theta\pi_\theta}[\nabla_\theta \log \pi_\theta(\tilde{a}|\tilde{x})\, Q^{\pi_\theta}(\tilde{x}, \tilde{a})]$$

REINFORCE: A STOCHASTIC POLICY GRADIENT ALGORITHM

Idea: replace expectation by a sample mean ⇒ stochastic gradient algorithm

REINFORCE: A STOCHASTIC POLICY GRADIENT ALGORITHM

Idea: replace expectation by a sample mean ⇒ stochastic gradient algorithm

REINFORCE

Input: arbitrary initial θ_0

For $k = 0,1, \ldots$

- Obtain sample trajectory $(x_t, a_t, r_t)_{t=1}^{T} \sim \pi_{\theta_k}$
- Estimate $\hat{Q}_k \approx Q^{\pi_{\theta_k}}$ by Monte Carlo
- Estimate $g_k \approx \nabla_\theta \rho(\theta_k)$ by the average of

$$g_{k,t} = \nabla_\theta \log \pi_{\theta_k}(a_t|x_t)\hat{Q}_k(x_t, a_t)$$

- Update $\theta_{k+1} = \theta_k + \alpha_k g_k$

REINFORCE: A STOCHASTIC POLICY GRADIENT ALGORITHM

Idea: replace expectation by a sample mean ⇒ stochastic gradient algorithm

REINFORCE

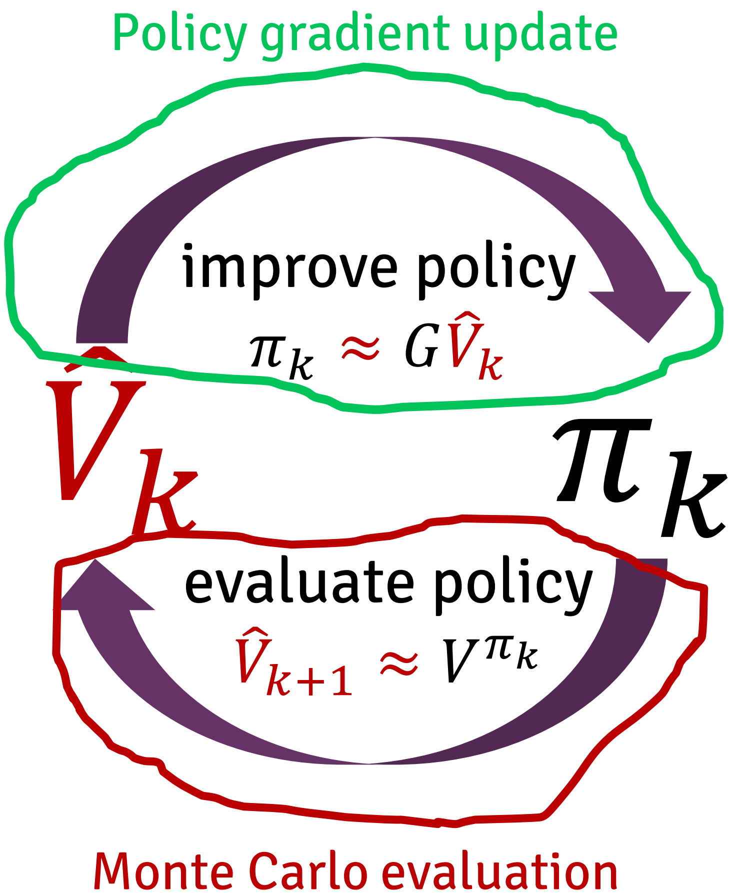
Input: arbitrary initial θ_0

For $k = 0,1, \ldots$

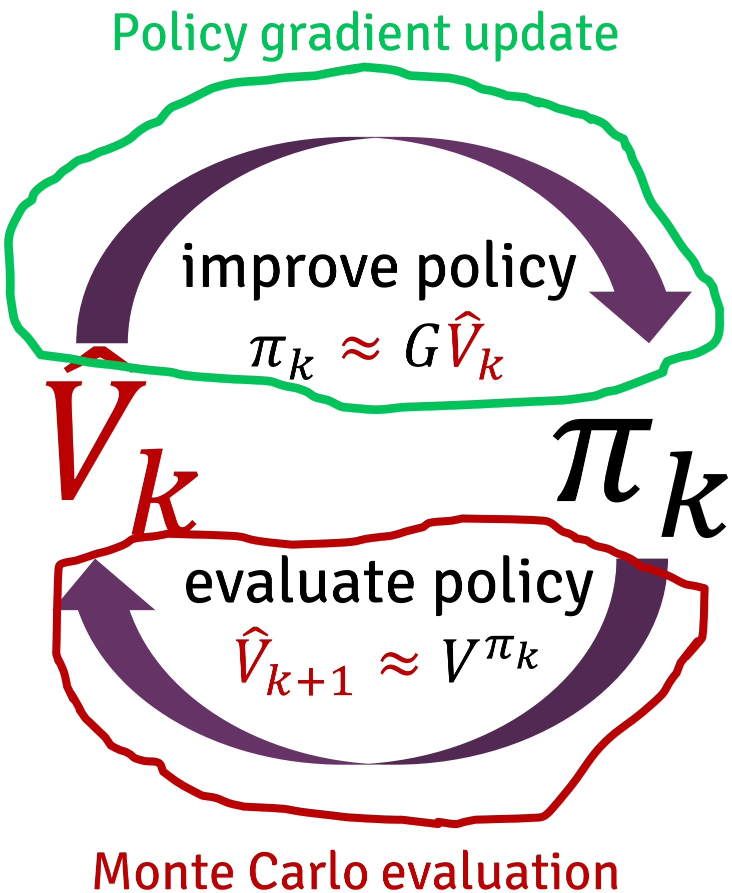
- Obtain sample trajectory $(x_t, a_t, r_t)_{t=1}^{T} \sim \pi_{\theta_k}$
- Estimate $\hat{Q}_k \approx Q^{\pi_{\theta_k}}$ by Monte Carlo
- Estimate $g_k \approx \nabla_\theta \rho(\theta_k)$ by the average of
$$g_{k,t} = \nabla_\theta \log \pi_{\theta_k}(a_t | x_t) \hat{Q}_k(x_t, a_t)$$
- Update $\theta_{k+1} = \theta_k + \alpha_k g_k$

$$\mathbf{E}[g_k] = \nabla_\theta \rho(\theta_k)$$

REINFORCE AS DIRECT POLICY SEARCH



REINFORCE AS DIRECT POLICY SEARCH



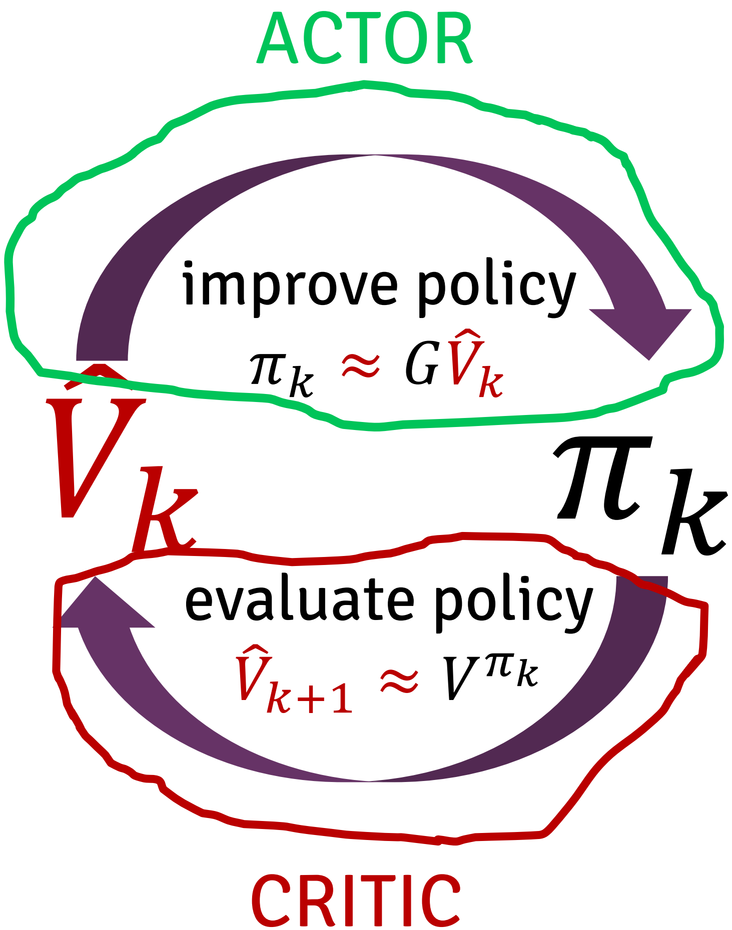
☺ direct method: no explicit approximation of V^{π} ☺

☺ converges to local optimum ☺

☺ less aggressive updates ☺

☹ large variance of g_k ☹

ACTOR-CRITIC METHODS



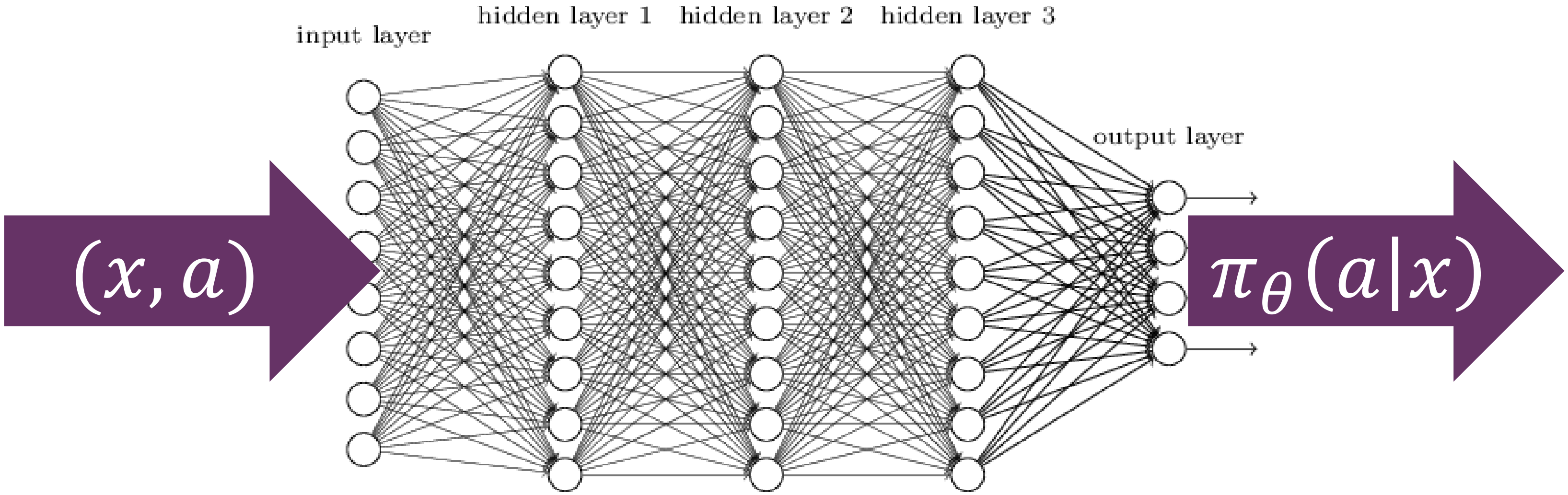
Typical actor:
policy gradient updates

Critic:

- Monte Carlo ⇒ REINFORCE
- TD(λ)
- LSTD(λ)
- DQN, …

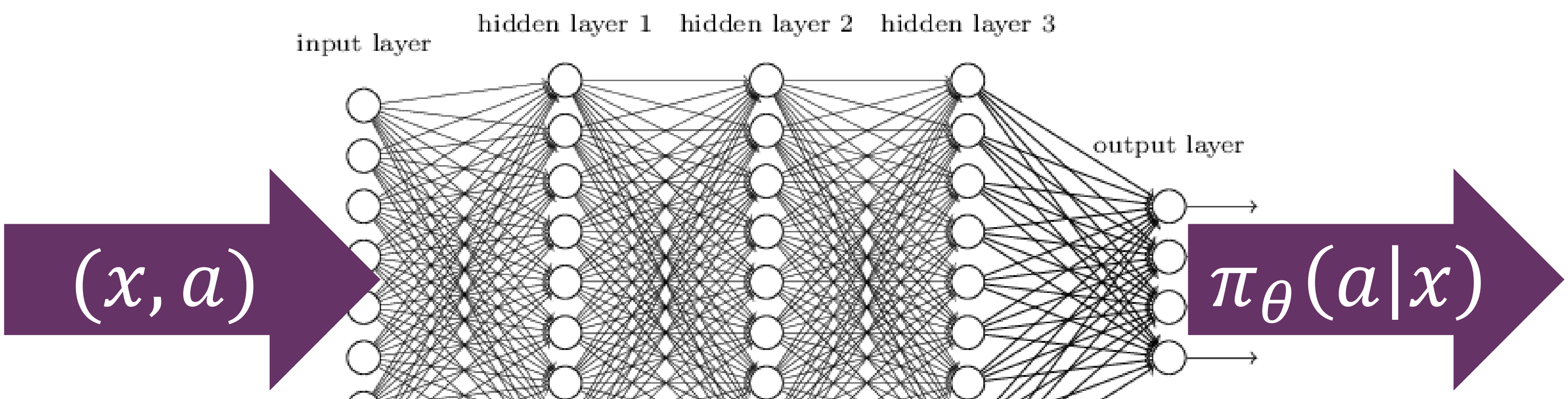
A TYPICAL DEEP RL ARCHITECTURE: A3C

Parametrize policy by a deep neural net



A TYPICAL DEEP RL ARCHITECTURE: A3C

Parametrize policy by a deep neural net



+ another neural net to estimate V^{π_θ} and to estimate Q^{π_θ} by "bootstrapped" Monte Carlo
+ asynchronous updates
+ entropy-regularization of the objective
+…

POLICY GRADIENTS: THE FINAL ANSWER?

Policy gradient update

$$\theta_{t+1} = \arg\max_{\theta} \left\{ \langle \theta, \nabla \rho(\theta_t) \rangle - \frac{1}{\alpha_t} \|\theta - \theta_t\|_2^2 \right\}$$

POLICY GRADIENTS: THE FINAL ANSWER?

Policy gradient update

$$\theta_{t+1} = \arg\max_{\theta} \left\{ \langle \theta, \nabla\rho(\theta_t) \rangle - \frac{1}{\alpha_t} \|\theta - \theta_t\|_2^2 \right\}$$

Issue #1:
Euclidean norm may be unnatural way to measure distance between μ_θ and μ_{θ_t}?

POLICY GRADIENTS: THE FINAL ANSWER?

Policy gradient update

$$\theta_{t+1} = \arg\max_{\theta} \left\{ \langle \theta, \nabla \rho(\theta_t) \rangle - \frac{1}{\alpha_t} \|\theta - \theta_t\|_2^2 \right\}$$

Issue #2:
Linearizing ρ at θ_t may lead to instability?

Issue #1:
Euclidean norm may be unnatural way to measure distance between μ_θ and μ_{θ_t}?

POLICY GRADIENTS: THE FINAL ANSWER?

Policy gradient update

$$\theta_{t+1} = \arg\max_{\theta} \left\{ \langle \theta, \nabla \rho(\theta_t) \rangle - \frac{1}{\alpha_t} \|\theta - \theta_t\|_2^2 \right\}$$

Issue #2:
Linearizing ρ at θ_t may lead to instability?

Issue #1:
Euclidean norm may be unnatural way to measure distance between μ_θ and μ_{θ_t}?

+ Issue #3:
Policy gradient estimator has huge variance ☹

A BETTER APPROACH: SMOOTHED LINEAR PROGRAMS

Dual LP

$$R_\gamma^* = \max_{\mu \in \Delta} \langle \mu, r \rangle$$

A BETTER APPROACH: SMOOTHED LINEAR PROGRAMS

Dual convex program

$$\tilde{R}_{\gamma}^{*} = \max_{\mu \in \Delta} \left\{ \langle \mu, r \rangle + \frac{1}{\eta} \Phi(\mu) \right\}$$

A BETTER APPROACH: SMOOTHED LINEAR PROGRAMS

Dual convex program

$$\tilde{R}_\gamma^* = \max_{\mu \in \Delta} \left\{ \langle \mu, r \rangle + \frac{1}{\eta} \Phi(\mu) \right\}$$

Φ: strongly convex function of μ:

- smooth optimum

$$\mu^* = \arg\max_{\mu} \left\{ \langle \mu, r \rangle + \frac{1}{\eta} \Phi(\mu) \right\} = \frac{1}{\eta} \nabla_r \Phi^*(\eta r)$$

- regularization effect ⇒ better generalization?

BETTER PROXIMAL REGULARIZATION: MIRROR DESCENT

Policy gradient update

$$\theta_{t+1} = \arg\max_{\theta} \left\{ \langle \theta, \nabla \rho(\theta_t) \rangle - \frac{1}{\alpha_t} \|\theta - \theta_t\|_2^2 \right\}$$

BETTER PROXIMAL REGULARIZATION: MIRROR DESCENT

Policy gradient update

$$\theta_{t+1} = \arg\max_{\theta} \left\{ \langle \theta, \nabla \rho(\theta_t) \rangle - \frac{1}{\alpha_t} \|\theta - \theta_t\|_2^2 \right\}$$

Mirror descent update

$$\mu_{t+1} = \arg\max_{\mu \in \Delta} \left\{ \langle \mu, r \rangle - \frac{1}{\eta_t} D(\mu | \mu_t) \right\}$$

BETTER PROXIMAL REGULARIZATION: MIRROR DESCENT

Policy gradient update

$$\ldots \theta_t)\rangle - \frac{1}{\alpha_t}\|\theta - \theta_t\|_2^2 \Big\}$$

No need for local linearization

Mirror descent update

$$\mu_{t+1} = \arg\max_{\mu \in \Delta} \left\{ \langle \mu, r \rangle - \frac{1}{\eta_t} D(\mu | \mu_t) \right\}$$

Proximal regularization through Bregman divergence $D(\mu|\mu')$
(strongly convex in μ)

DIRECT POLICY OPTIMIZATION

Idea: derive algorithms by thinking of $\mu \in \Delta$ as the decision variable!

Examples

- Policy gradient methods
  $=$ gradient descent on $-R_\gamma^\pi$
- Relative Entropy Policy Search (REPS)
  $=$ mirror descent on $-R_\gamma^\pi$
- Trust-region policy optimization (TRPO)
  $=$ mirror descent on (a surrogate of) $-R_\gamma^\pi$

RELATIVE ENTROPY POLICY SEARCH (REPS, PETERS ET AL., 2010)

Mirror descent update

$$\mu_{t+1} = \arg\max_{\mu \in \Delta} \left\{ \langle \mu, r \rangle - \frac{1}{\eta_t} D(\mu | \mu_t) \right\}$$

$$D(\mu | \mu') = \sum_{x,a} \mu(x,a) \log \frac{\mu(x,a)}{\mu'(x,a)}$$

RELATIVE ENTROPY POLICY SEARCH (REPS, PETERS ET AL., 2010)

Mirror descent update

$$\mu_{t+1} = \arg\max_{\mu \in \Delta} \left\{ \langle \mu, r \rangle - \frac{1}{\eta_t} D(\mu | \mu_t) \right\}$$

$$D(\mu|\mu') = \sum_{x,a} \mu(x,a) \log \frac{\mu(x,a)}{\mu'(x,a)}$$

Closed-form "policy update":

$$\mu_{t+1}(x,a) = \mu_t(x,a) e^{\eta_t \left(r(x,a) + \gamma \mathbf{E}_{y|x,a}[\widetilde{V}_t(y)] - \widetilde{V}_t(x) \right)}$$

RELATIVE ENTROPY POLICY SEARCH (REPS, PETERS ET AL., 2010)

Mirror descent update

$$\mu_{t+1} = \arg\max_{\mu \in \Delta} \left\{ \langle \mu, r \rangle - \frac{1}{\eta_t} D(\mu | \mu_t) \right\}$$

$$D(\mu|\mu') = \sum_{x,a} \mu(x,a) \log \frac{\mu(x,a)}{\mu'(x,a)}$$

Closed-form "policy update":

$$\mu_{t+1}(x,a) = \mu_t(x,a) e^{\eta_t \left(r(x,a) + \gamma \mathbf{E}_{y|x,a}[\tilde{V}_t(y)] - \tilde{V}_t(x) \right)}$$

"Value function"

$\tilde{V}_t = ???$

THE REPS VALUE FUNCTION

Theorem

The REPS value function $\tilde{V}_t$ is given as the minimizer of the loss function

$$\tilde{L}(V) = \log \mathbf{E}_{x\sim\mu_t}\left[e^{\eta_t\left(T^\pi V(x) - V(x)\right)}\right]$$

THE REPS VALUE FUNCTION

Theorem

The REPS value function $\widetilde{V}_t$ is given as the minimizer of the loss function

$$\widetilde{L}(V) = \log \mathbf{E}_{x\sim\mu_t}\left[e^{\eta_t\left(T^\pi V(x) - V(x)\right)}\right]$$

"Proof": Lagrangian duality.

THE REPS VALUE FUNCTION

Theorem
The REPS value function $\tilde{V}_t$ is given as the minimizer of the loss function

$$\tilde{L}(V) = \log \mathbf{E}_{x\sim\mu_t}\left[e^{\eta_t\left(T^\pi V(x) - V(x)\right)}\right]$$

"Proof": Lagrangian duality.

A natural competitor for the Bellman error

$$L(V) = \mathbf{E}_{x\sim\mu}\left[\left(T^\pi V(x) - V(x)\right)^2\right]???$$

Stay tuned for "deep REPS" results ☺

DIRECT POLICY OPTIMIZATION

Idea: derive algorithms by thinking of $\mu \in \Delta$ as the decision variable!

Examples

- Policy gradient methods
  $=$ gradient descent on $-R_\gamma^\pi$
- Relative Entropy Policy Search (REPS)
  $=$ mirror descent on $-R_\gamma^\pi$
- Trust-region policy optimization (TRPO)
  $=$ mirror descent on (a surrogate of) $-R_\gamma^\pi$

THE REGULARIZED BELLMAN EQUATIONS

The Bellman opt. equations

$$V^*(x) = \max_a \left\{ r(x,a) + \gamma \sum_y P(y|x,a) V^*(y) \right\}$$

THE REGULARIZED BELLMAN EQUATIONS

The **regularized** Bellman opt. equations

$$V^*(x) = \underset{a}{\text{softmax}}^{\eta}\left\{r(x,a) + \gamma \sum_y P(y|x,a)V^*(y)\right\}$$

THE REGULARIZED BELLMAN EQUATIONS

The regularized Bellman opt. equations

$$V^*(x) = \underset{a}{\text{softmax}}^{\eta}\left\{r(x,a) + \gamma \sum_y P(y|x,a) V^*(y)\right\}$$

Used almost exclusively since ~late 2016

- Better optimization properties: smooth gradients, less sensitive to errors
- Better exploration: optimal policy naturally stochastic, no need for ε −greedy trick

THE REGULARIZED BELLMAN EQUATIONS

Is there a natural "dual" explanation?

The regularized Bellman opt. equations

$$V^*(x) = \underset{a}{\text{softmax}}^{\eta}\left\{r(x,a) + \gamma \sum_y P(y|x,a)V^*(y)\right\}$$

Used almost exclusively since ~late 2016

- Better optimization properties:
  smooth gradients, less sensitive to errors
- Better exploration:
  optimal policy naturally stochastic, no need for ε −greedy trick

DUALITY THEORY FOR THE REGULARIZED BELLMAN EQUATIONS

The regularized Bellman opt. equations

$$V^*(x) = \underset{a}{\text{softmax}}^{\eta}\left\{r(x,a) + \gamma \sum_y P(y|x,a) V^*(y)\right\}$$

??? Dual convex program ???

$$\tilde{R}_\gamma^* = \max_{\mu \in \Delta} \left\{ \langle \mu, r \rangle - \frac{1}{\eta} \Phi(\mu) \right\}$$

DUALITY THEORY FOR THE REGULARIZED BELLMAN EQUATIONS

Theorem (Neu et al., 2017)
The two formulations are connected by Lagrangian duality with the choice

$$\Phi(\mu) = \sum_{x,a} \mu(x,a) \log \frac{\mu(x,a)}{\sum_b \mu(x,b)}$$

$$= \sum_{x} \mu(x) \sum_{a} \pi_\mu(a|x) \log \pi_\mu(a|x)$$

DUALITY THEORY FOR THE REGULARIZED BELLMAN EQUATIONS

Theorem (Neu et al., 2017)
The two formulations are connected by Lagrangian duality with the choice

$$\Phi(\mu) = \sum_{x,a} \mu(x,a) \log \frac{\mu(x,a)}{\sum_b \mu(x,b)}$$

$$= \sum_x \mu(x) \sum_a \pi_\mu(a|x) \log \pi_\mu(a|x)$$

The conditional entropy of $A|X$ under μ

DUALITY THEORY FOR THE REGULARIZED BELLMAN EQUATIONS

Theorem (Neu et al., 2017)
The two formulations are connected by Lagrangian duality with the choice

$$\Phi(\mu) = \sum_{x,a} \mu(x,a) \log \frac{\mu(x,a)}{\sum_b \mu(x,b)}$$

$$= \sum_x \mu(x) \sum_a \pi_\mu(a|x) \log \pi_\mu(a|x)$$

The conditional entropy of $A|X$ under μ

A convex function of μ!

DUALITY THEORY FOR THE REGULARIZED BELLMAN EQUATIONS

The regularized Bellman opt. equations

$$V^*(x) = \underset{a}{\text{softmax}}^{\eta}\left\{r(x,a) + \gamma \sum_y P(y|x,a)V^*(y)\right\}$$

Dual convex program

$$\widetilde{R}_\gamma^* = \max_{\mu \in \Delta}\left\{\langle \mu, r \rangle - \frac{1}{\eta}\Phi(\mu)\right\}$$

MIRROR DESCENT WITH CONDITIONAL ENTROPY (NEU ET AL., 2017)

Mirror descent update

$$\mu_{t+1} = \arg\max_{\mu \in \Delta} \left\{ \langle \mu, r \rangle - \frac{1}{\eta_t} D_\Phi(\mu | \mu_t) \right\}$$

$$D_\Phi(\mu | \mu_t) = \sum_{x,a} \mu(x, a) \log \frac{\pi_\mu(a|x)}{\pi_t(x,a)}$$

MIRROR DESCENT WITH CONDITIONAL ENTROPY (NEU ET AL., 2017)

Mirror descent update

$$\mu_{t+1} = \arg\max_{\mu \in \Delta} \left\{ \langle \mu, r \rangle - \frac{1}{\eta_t} D_\Phi(\mu | \mu_t) \right\}$$

$$D_\Phi(\mu|\mu_t) = \sum_{x,a} \mu(x,a) \log \frac{\pi_\mu(a|x)}{\pi_t(x,a)}$$

Closed-form policy update:

$$\pi_{t+1}(a|x) = \pi_t(a|x) e^{\eta_t \left(r(x,a) + \gamma \mathbf{E}_{y|x,a}[\widetilde{V}_t(y)] - \widetilde{V}_t(x) \right)}$$

MIRROR DESCENT WITH CONDITIONAL ENTROPY (NEU ET AL., 2017)

Mirror descent update

$$\mu_{t+1} = \arg\max_{\mu \in \Delta} \left\{ \langle \mu, r \rangle - \frac{1}{\eta_t} D_\Phi(\mu | \mu_t) \right\}$$

$$D_\Phi(\mu|\mu_t) = \sum_{x,a} \mu(x,a) \log \frac{\pi_\mu(a|x)}{\pi_t(x,a)}$$

Closed-form policy update:

$$\pi_{t+1}(a|x) = \pi_t(a|x) e^{\eta_t \left(r(x,a) + \gamma \mathbf{E}_{y|x,a}[\tilde{V}_t(y)] - \tilde{V}_t(x) \right)}$$

Value function $\tilde{V}_t$ = solution to proximally regularized BOE

TRUST-REGION POLICY OPTIMIZATION (TRPO, SCHULMAN ET AL., 2015)

Mirror descent update

$$\mu_{t+1} = \arg\max_{\mu \in \Delta} \left\{ \langle \mu, r \rangle - \frac{1}{\eta_t} D_\Phi(\mu | \mu_t) \right\}$$

$$D_\Phi(\mu | \mu_t) = \sum_{x,a} \mu(x,a) \log \frac{\pi_\mu(a|x)}{\pi_t(x,a)}$$

TRUST-REGION POLICY OPTIMIZATION (TRPO, SCHULMAN ET AL., 2015)

Mirror descent update

$$\mu_{t+1} = \arg\max_{\mu \in \Delta} \left\{ \langle \mu, \tilde{Q}_t - \tilde{V}_t \rangle - \frac{1}{\eta_t} D_\Phi(\mu | \mu_t) \right\}$$

$$D_\Phi(\mu|\mu_t) = \sum_x \mu_t(x) \sum_a \pi_\mu(a|x) \log \frac{\pi_\mu(a|x)}{\pi_t(x,a)}$$

TRUST-REGION POLICY OPTIMIZATION (TRPO, SCHULMAN ET AL., 2015)

Dense surrogate for $\langle \mu, r \rangle$

(works because $\langle \mu, r \rangle = \langle \mu, \tilde{Q}_t - \tilde{V}_t \rangle$ when $\mu \in \Delta$)

Mirror descent update

$$\mu_{t+1} = \arg\max_{\mu \in \Delta} \left\{ \langle \mu, \tilde{Q}_t - \tilde{V}_t \rangle - \frac{1}{\eta_t} D_\Phi(\mu | \mu_t) \right\}$$

$$D_\Phi(\mu|\mu_t) = \sum_x \mu_t(x) \sum_a \pi_\mu(a|x) \log \frac{\pi_\mu(a|x)}{\pi_t(x,a)}$$

$\mu_t \approx \mu_{t+1}$, but μ_t can be sampled from

TRUST-REGION POLICY OPTIMIZATION (TRPO, SCHULMAN ET AL., 2015)

Theorem (Neu et al., 2017)
TRPO is equivalent to the MDP-E algorithm of Even-Dar, Kakade and Mansour (2006)

$$\Rightarrow$$

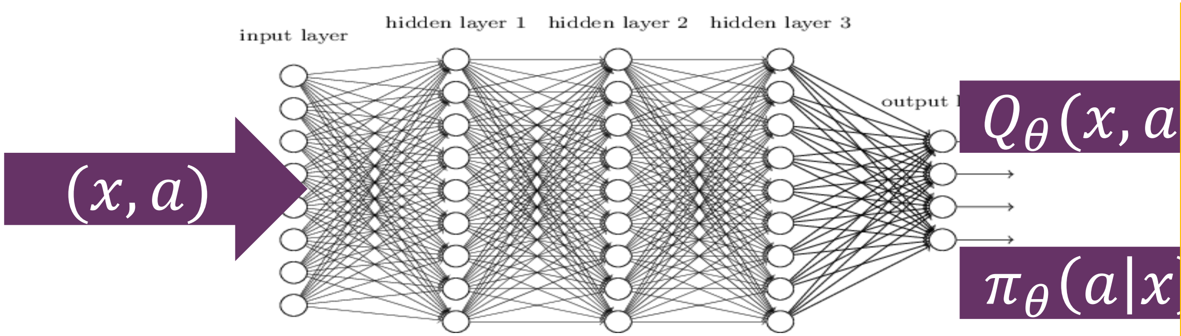
$$\lim_{t \to \infty} \langle \mu_t, r \rangle = \langle \mu^*, r \rangle$$

TRUST-REGION POLICY OPTIMIZATION (TRPO, SCHULMAN ET AL., 2015)

Theorem (Neu et al., 2017)
TRPO is equivalent to the MDP-E algorithm of Even-Dar, Kakade and Mansour (2006)

$$\Rightarrow$$

$$\lim_{t\to\infty} \langle \mu_t, r\rangle = \langle \mu^*, r\rangle$$



+ more tricks:
- Another surrogate for μ
- Truncation of objective
- Constraint vs. penalty
- Mini-batch SGD
- …

TRUST-REGION POLICY OPTIMIZATION (TRPO, SCHULMAN ET AL., 2015)

Theorem (Neu et al., 2017)
TRPO is equivalent to the MDP-E algorithm of Even-Dar, Kakade and Mansour (2006)

$$\Rightarrow$$

$$\lim_{t\to\infty} \langle \mu_t, r \rangle = \langle \mu^*, r \rangle$$

Literally the most broadly used deep RL algorithm!
(but reading the original paper is not recommended…)

+ more tricks:

- Another surrogate for μ
- Truncation of objective
- Constraint vs. penalty
- Mini-batch SGD
- …

BEYOND LINEAR PROGRAMMING: SADDLE-POINT OPTIMIZATION

Dual LP

$$R_\gamma^* = \max_{\mu \in \Delta} \langle \mu, r \rangle$$

Primal LP

$$R_\gamma^* = \min_{V \in \mathbb{R}^X} \langle \mu_0, V \rangle$$

s.t. $V(x) \geq r(x,a) + \gamma \sum_y P(y|x,a) V(y) \quad (\forall x, a)$

BEYOND LINEAR PROGRAMMING: SADDLE-POINT OPTIMIZATION

Bellman saddle point

$$\min_{V} \max_{\mu \in \Delta} \{ \langle \mu, r + \gamma P V - V \rangle + (1-\gamma) \langle \mu_0, V \rangle \}$$

BEYOND LINEAR PROGRAMMING: SADDLE-POINT OPTIMIZATION

Bellman saddle point

$$\min_V \max_{\mu\in\Delta}\{\langle\mu, r+\gamma PV - V\rangle + (1-\gamma)\langle\mu_0, V\rangle\}$$

≈ the Lagrangian of the two LPs

$\Rightarrow$

solution exists & optimal policy can be extracted under same conditions

PRIMAL-DUAL π-LEARNING (WANG ET AL., 2017-)

Bellman saddle point

$$\min_{V} \max_{\mu \in \Delta} \{ \langle \mu, r + \gamma P V - V \rangle + (1-\gamma) \langle \mu_0, V \rangle \}$$

PRIMAL-DUAL π-LEARNING (WANG ET AL., 2017-)

Bellman saddle point

$$\min_{V} \max_{\mu \in \Delta} \{ \langle \mu, r + \gamma P V - V \rangle + (1-\gamma) \langle \mu_0, V \rangle \}$$

Value update:

$$\tilde{V}_{t+1} = \tilde{V}_t + \alpha_t (\mu_t - \gamma \mu_t P)$$

Policy update:

$$\mu_{t+1}(x,a) = \mu_t(x,a) e^{\eta_t \left(r(x,a) + \gamma \mathbf{E}_{y|x,a}[\tilde{V}_t(y)] - \tilde{V}_t(x) \right)}$$

PRIMAL-DUAL π-LEARNING (WANG ET AL., 2017-)

Bellman saddle point

$$\min_V \max_{\mu\in\Delta}\{\langle \mu, r+\gamma PV - V\rangle + (1-\gamma)\langle \mu_0, V\rangle\}$$

Value update:

$$\tilde{V}_{t+1} = \tilde{V}_t + \alpha_t(\mu_t - \gamma\mu_t P)$$

Gradient step in primal

Policy update:

$$\mu_{t+1}(x,a) = \mu_t(x,a)e^{\eta_t\left(r(x,a)+\gamma\mathbf{E}_{y|x,a}[\tilde{V}_t(y)]-\tilde{V}_t(x)\right)}$$

Exponentiated gradient step in dual

PRIMAL-DUAL π-LEARNING (WANG ET AL., 2017-)

Bellman saddle point

$$\min_{V} \max_{\mu \in \Delta} \{\langle \mu, r + \gamma PV - V \rangle + (1-\gamma)\langle \mu_0, V \rangle\}$$

$\mu_t P)$

Gradient step in primal

≈ incremental REPS

state-of-the art sample complexity results for discounted & undiscounted MDPs!

Exponentiated gradient step in dual

$\mu_{t+1}(x,a) = \mu_t(x,a)e^{\eta(\ldots_{y|x,a}[\widetilde{V}_t(y)] - \widetilde{V}_t(x))}$

THIS SHORT COURSE: A PRIMAL-DUAL VIEW

part 1

- Markov decision processes
  - Value functions and optimal policies
- Primal view: Dynamic programming
  - Policy evaluation, value and policy iteration
  - Value-function-based methods
    - Temporal differences, Q-learning, LSTD, deep Q networks,…

part 2

- Dual view: Linear programming
  - LP duality in MDPs
  - Direct policy optimization methods
    - Policy gradients, REPS, TRPO,…

THIS SHORT COURSE: A PRIMAL-DUAL VIEW

part 1

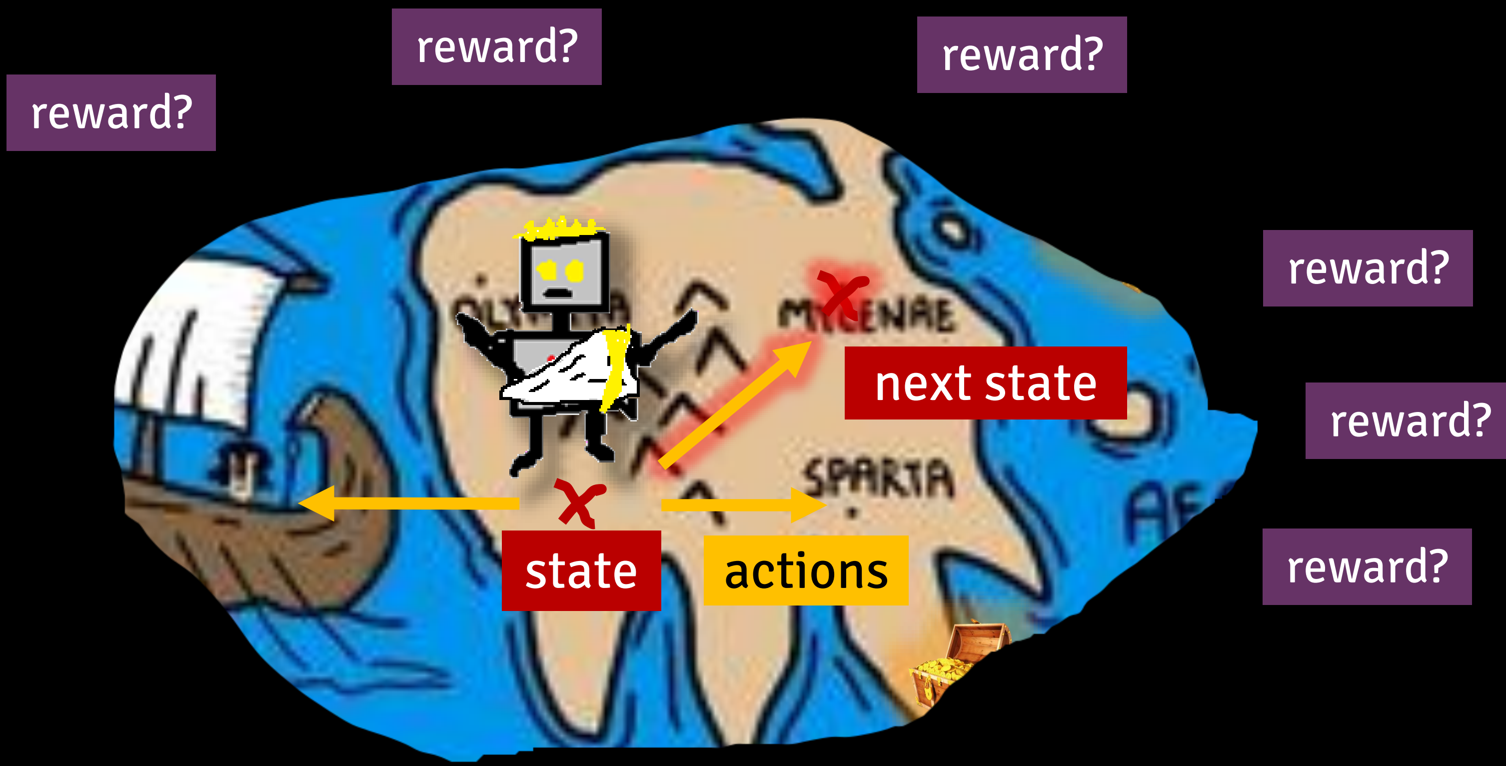
- Markov decision processes
  - Value functions and optimal policies
- Primal view: Dynamic programming
  - Policy evaluation, value and policy iteration
  - Value-functi
    - Temporal diffe … rks,…

what else?

part 2

- Dual view: L
  - LP duality in MDPs
  - Direct policy optimization methods
    - Policy gradients, REPS, TRPO,…

EXPLORATION VS. EXPLOITATION



EXPLORATION VS. EXPLOITATION

reward?

reward?

reward?

reward?

next state

reward?

reward?

- Multi-armed bandits
- Exploration bonuses
- Thompson sampling
- Monte Carlo tree search
- …

EXPLORATION VS. EXPLOITATION

Still no practical algorithms!

- Multi-armed bandits
- Exploration bonuses
- Thompson sampling
- Monte Carlo tree search
- …

CONCLUSION

RL is an insanely popular field with

- huge recent successes
- some beautiful fundamental theory
- unique algorithmic ideas

CONCLUSION

RL is an insanely popular field with

- huge recent successes
- some beautiful fundamental theory
- unique algorithmic ideas

BUT still fundamental challenges in

- understanding efficient exploration
- stability of algorithms
- generalizability of successes

CONCLUSION

Come and work on RL theory ;)

BUT still fundamental challenges in

- understanding efficient exploration
- stability of algorithms
- generalizability of successes

CONCLUSION

Come and work on RL theory ;)

+ also come see
PARADISE LOST
tonight!

Thanks!!!