MLSS Summer School
Aug 2, 2021

Large scale learning and planning in reinforcement learning

Csaba Szepesvári
DeepMind & University of Alberta

- Goal: Understanding of what RL algorithms can and cannot do.
- How? Theoretical insights

Contents

1. Big picture recap
2. Foundations: Planning in finite MDPs
3. Online learning in finite MDPs
4. Planning in large MDPs
5. Batch RL
6. What now?

BIG PICTURE RECAP

Getting the big picture right

RL⊆ML⊆CS

Goal: Algorithm design

What do we want from our algorithms?

- generality
- soundness or effectiveness
- efficiency

RL algos:
past data → actions

Algo+problem instance
→ (did it work? resource use?)

What makes an algorithm a good one?
“Best on **all** instances!?”

Hyperparameters?

Algorithm + hyperparameters $\neq$ algorithm!

"Algorithm family"

Questions studied

- Does the family have a "good member"?
- How to choose the hyperparameters? (To get an algorithm!)

Theoretical vs. empirical work

Empirical work (in RL/CS)

- Use benchmarks to evaluate/compare/analyze algorithms
- Modify algorithms to get better performance on benchmarks
- Modify benchmarks to create new challenges

Theoretical work

- Replace benchmarks with problem classes described by their properties
- Advantage: Infinitely many instances!
- Brain vs. computation

Application work is separate from these. Instance is fixed!!

Empirical work

Unique limitations/problems

- Limited by compute power
- Limited scope
- Reproducibility, soft claims

Unique merits

- Accessibility
- Problem choice often more obvious

Theory work

Unique limitations/problems

- Limited by brainpower,
- Lost in beauty
- Lost in detail

Unique merits

- Truths values are absolute and permanent
- Can prove impossibility
- Transparency, clarity

Common issues

- “overfitting” to the problem class/benchmark sets

Activities

- Problem oriented analysis
- Algorithm oriented analysis

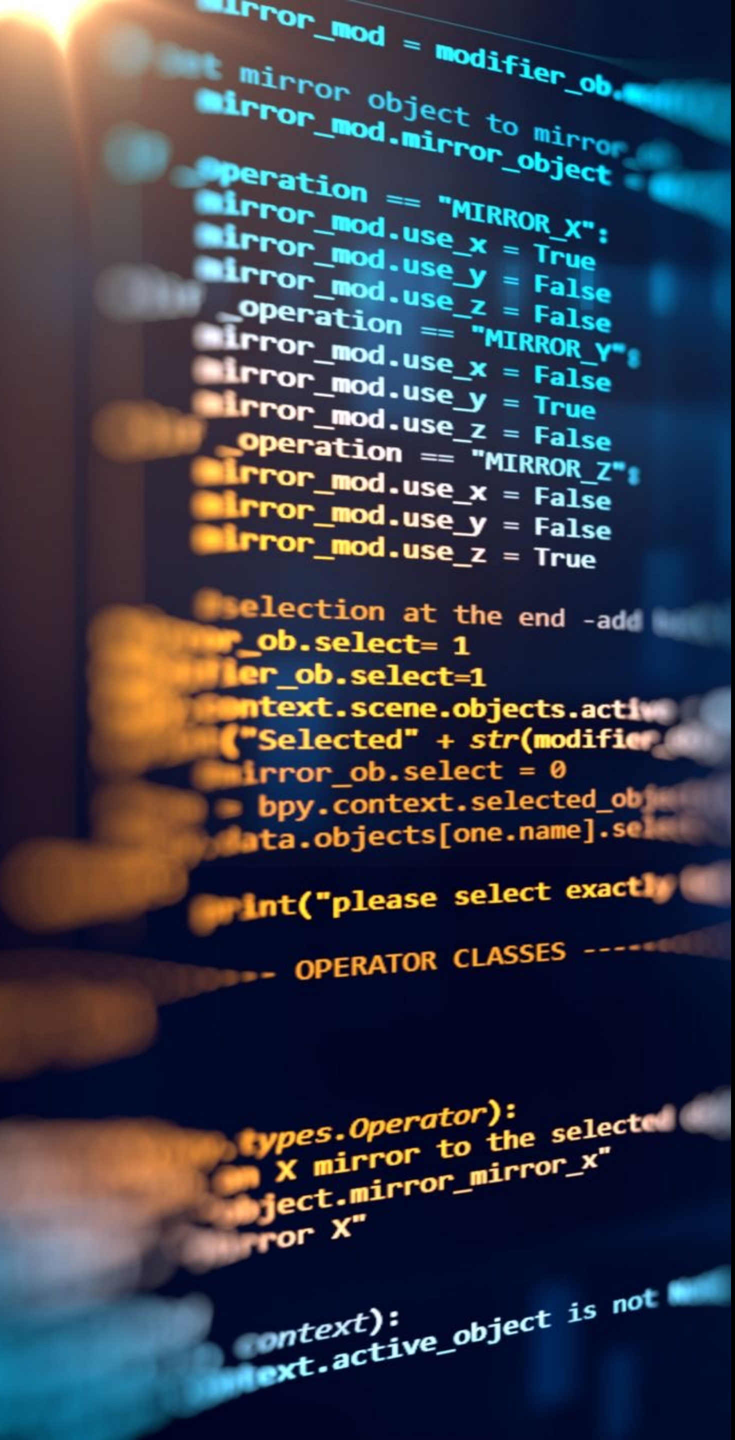
Problem oriented analysis

Descriptive

Complexity: Characterize resource needs to solve instances in a problem class

Prescriptive

Given problem class and metrics, find an algorithm that is “best”/”good enough”



Algorithm oriented analysis

- Just descriptive, never prescriptive
- Starts with an algorithm

- ➢ Does A work at all?
  - On which instances?
- ➢ Analyze resource needs

Why do theory?

Theory as sanity check

- Are there any conditions when your shiny new, greatest and latest alg A provably works?
- Does it work on tabular (simple) problems?

Benchmarking with theory

- How well does alg A do relative to the competition?

Understanding theoretical works

What is the problem considered?

(some "RL theory" papers are guilty of skipping this)

What is the result?
(theorem!)

Conditions/hypothesis/antecedent

Conclusion/consequent

What is the context?

Why was the theorem produced?

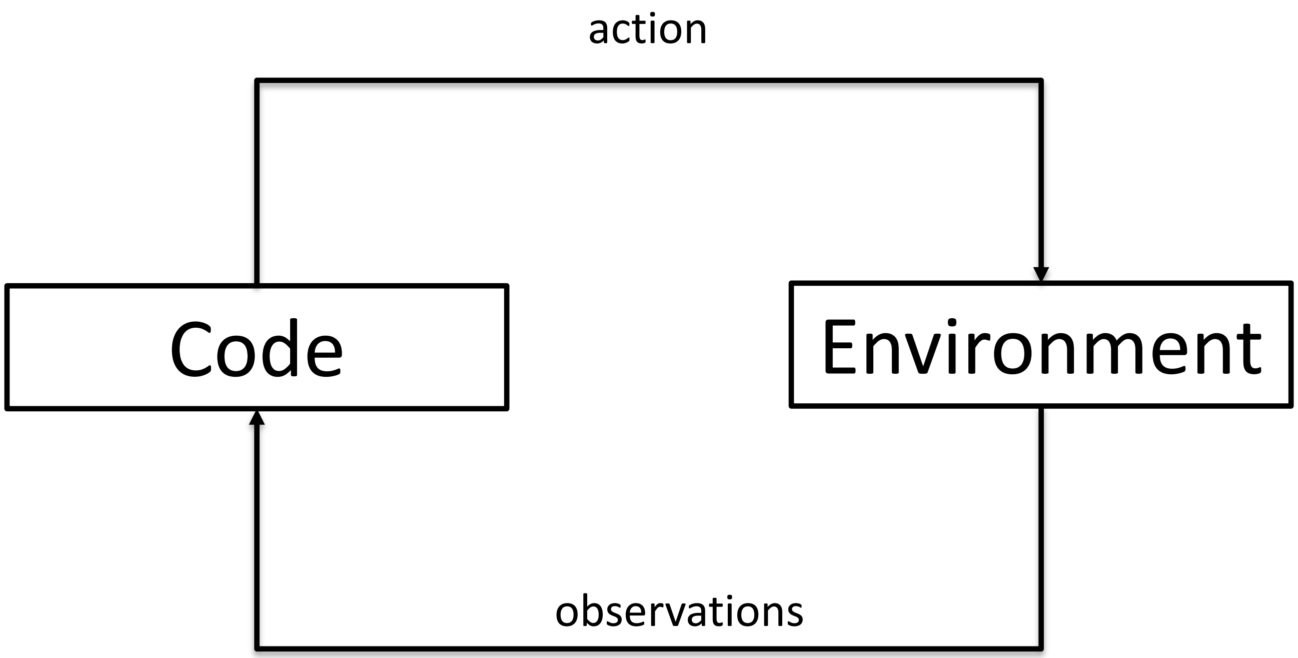
Produced in this form?

Could it hold more generally?

foundations for RL

- General lessons:
    - At the heart of RL is search helped by structure
    - With no or little structure, algorithms need to work hard
    - MDPs give some structure, but more structure is needed for scaling up

Control problems



Markov Decision Processes:

- Stochastic state transitions
- Control goal is to maximize total (discounted/undiscounted) reward
- State (and rewards) are available for measurement

Markov Decision Processes & Planning

$M = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$

$\mathcal{S} = \{1,2,\dots,S\}, \mathcal{A} = \{1,2,\dots,A\}$

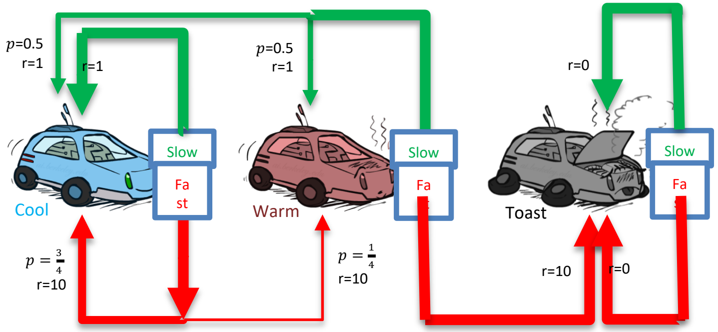
$P = \big(P(s,a)\big)_{s,a}, r = \big(r(s,a)\big)_{s,a}$

$\pi : \mathcal{S} \to \Delta(\mathcal{A})$ (feedback) policies

$v^\pi(s) = \mathbb{E}_s^\pi[\Sigma_{t=0}^\infty \gamma^t\, r(S_t, A_t)]$

$v^*(s) = \max_\pi v^\pi(s)$

Objective: find π s.t. $v^\pi \approx v^*$



Why discounting?

“Solve” MDP: find $v \approx v^*$

and use $\pi(s) = \mathrm{argmax}_a\, r(s,a) + \gamma\langle P(s,a), v\,\rangle$

Example from ai.berkeley.edu

Why care about tabular MDP results?

1. Don’t build on sand (definitions, ..)
2. Clever abstractions may give finite MDPs
3. Results demonstrate fundamental, algorithm independent limitations

Problem settings

offline planning
table-based input

```
def getpolicy( P, r, $\delta$ ):
    …
  return $\pi$ # $\pi \in [A]^{[S]}, v^{\pi} \geq v^{*} - \delta \mathbf{1}$.
```

offline planning
random access simulator

```
def getpolicy( simulator, $\delta$, $\xi$ ):
  (S,A) := simulator.problemsize()
  …
    (s’,r’) := simulator.gen(s,a) # $s \in [S], a \in [A], s' \sim P(s,a), r' = r(s,a)$
  …
  return $\pi$ # $\pi \in [A]^{[S]}, v^{\pi} \geq v^{*} - \delta \mathbf{1}$ with probability of at least $1 - \xi$.
```

Problem settings

online planning
random access simulator

```
def getaction( simulator, $s_0$, $\delta$, $\xi$ ):
  (S,A) := simulator.problemsize()
  ...
    (s’,r’) := simulator.gen(s,a) # $s \in [S], a \in [A]$
  ...
  return $a$   # $a \in [A]$ s.t. for the policy $\pi$ induced, $v^\pi \geq v^* - \delta\mathbf{1}$ w.p. $\geq \xi$
```

online planning
local access simulator

```
def getaction( simulator, $s_0$, $\delta$, $\xi$ ):
  A := simulator.num_actions()
  ...
    (s’,r’) := simulator.gen(s,a) # $s$: state previously seen, $a \in [A]$
  ...
  return $a$   # $a \in [A]$ s.t. for the policy $\pi$ induced, $v^\pi \geq v^* - \delta\mathbf{1}$ w.p. $\geq \xi$
```

s

Settings

Planning is

- offline or online

2

MDP is specified with

- matrices (=tables)
- simulator with
  - Random access to state-actions
  - Local access to state-actions

3

Computation model:

- Turing (#bits matter)
- **Real RAM model**

2

```
def policy_iteration($P, r, \delta$): # getpolicy fn
$\pi$ :=arbitrary, $k := 0, H := 1/(1-\gamma)$
while $\gamma^k H > \delta$:
    for all $s \in [S]$:
        $\pi'(s) := \mathrm{argmax}_a\, r(s,a) + \gamma\langle P(s,a), v^\pi \rangle$
    $\pi := \pi', k := k+1$
return $\pi$
```

Shorthand: $\pi' := \Gamma v^\pi$

$$\begin{aligned} v^\pi &= r_\pi + \gamma P_\pi v^\pi \\ &=: T_\pi v^\pi \end{aligned}$$

```
def value_iteration($P, r, \delta$): # getpolicy fn
$v := 0, k := 0, H := 1/(1 - \gamma)$
while $\gamma^k H > \delta$:
    for all $s \in [S]$:
        $v'(s) := \max_a r(s, a) + \gamma \langle P(s, a), v \rangle$
    $v := v'$
return $\Gamma v$
```

Shorthand: $v' := Tv$

Fundamental theorem

Let $\|v\| = \max_{s} |v(s)|$

Theorem (contractions). The following hold:

1. $\forall u, v, \pi \;\; \|T_\pi u \, - T_\pi v \, \| \leq \gamma \|u - v \, \|$
2. $\forall u, v \qquad \|Tu \, - Tv \, \| \leq \gamma \|u - v \, \|$

Banach's fixed point theorem.

For any contraction map T over a $\| \cdot \|$-complete vector space, $T^n u \rightarrow \; v^*$, the unique fixed point of T.

Fundamental theorem

Theorem

The following holds true in any finite MDP:

1. Any policy that is greedy with respect to v^* is optimal
2. It holds that $v^* = Tv^*$

Proof sketch:

Step 1: From $v^\pi \leq v^*$, $v^\pi = T_\pi v^\pi \leq T_\pi v^* \Rightarrow v^* \leq Tv^*$

Step 2: For $\pi = \Gamma v^*$, $T_\pi v^* = Tv^* \geq v^*$

$$\Rightarrow v^* \geq v^\pi \leftarrow T_\pi^n v^* \geq v^*$$

$$\Rightarrow v^* = v^\pi = T_\pi v^\pi = T_\pi v^* = Tv^*$$

Qu.e.d

Global planning when MDP given with matrices

$$H = \frac{1}{1-\gamma}$$

Policy iteration [Ye, 2011; Scherrer, 2016]

$H \cdot \{SA \wedge \log(H^2/\delta)\} \cdot (SA + S^2 + S^{2.373})$ operations are sufficient to produce a δ-optimal policy

Value iteration [folklore]

$H \log(H^2/\delta) S^2 A$ operations are sufficient to produce a δ-optimal policy

$a \wedge b \coloneqq \min(a, b)$

Global planning when MDP given wi matrices

$$H = \frac{1}{1-\gamma}$$

Algorithm oriented analysis

Policy iteration [Ye, 2011; Scherrer

$H \cdot \{S A \wedge \log(H^2/\delta)\}$ $S^2 + S^{2.373})$
operations are su o produce a δ-
optimal polic

Value ite olklore]

H $\delta)S^2 A$ operations are sufficient to
pr e a δ-optimal policy

$a \wedge b \ := \min(a, b)$

Tractability of planning in MDPs

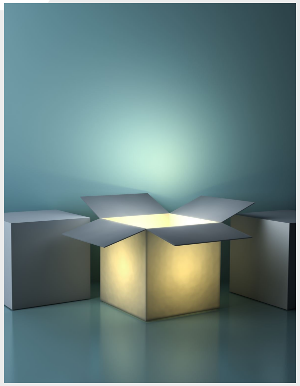
MDP given with a table

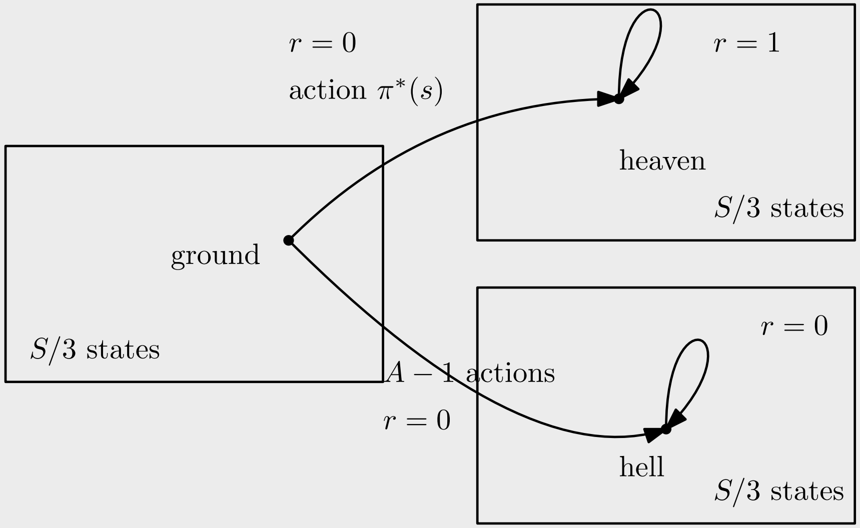
Query complexity: $\Omega(S^2A)$

Any algorithm that can find $\delta = 1$ suboptimal policies needs $\Omega(S^2A)$ steps on some MDP [Chen & Wang'17]

P(s'|s,a) (s,a)↓ s' →	3	4	5	6
(1,1)	1	0	0	0
(1,2)	1	0	0	0
(1,3)	0	0	0	1
(2,1)	1	0	0	0
(2,2)	0	0	1	0
(2,3)	1	0	0	0

$$\pi^*(1) = 3, \pi^*(2) = 2$$





$r = 0$

action $\pi^*(s)$

$r = 1$

heaven

$S/3$ states

ground

$S/3$ states

$r = 0$

$A - 1$ actions

$r = 0$

hell

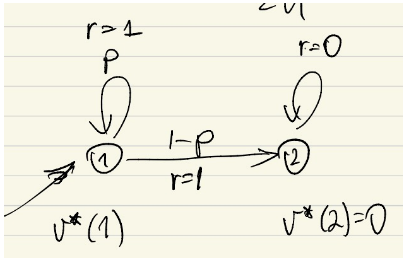
$S/3$ states

Simulation optimization

$$H \approx \frac{\log(1/\delta)}{1-\gamma}$$

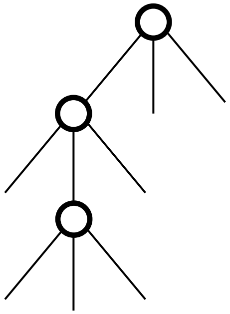
- Global planning, MDP given with a random-access simulator

  Query complexity: $\widetilde{\Theta}(SAH^3/\delta^2)$ [Azar et al '13]



- Local planning, MDP given with a local-access simulator

  Query complexity: $\Omega(A^H)$, $O((H^7A/\delta^2)^H)$ [Kearns et al., '02]
  No dependence on S, but exponential dependence on H.



Statistical uncertainty

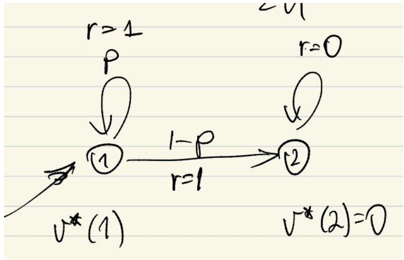
$H \approx \frac{\log(1/\delta)}{1-\gamma}$

Simulation optimization

Problem oriented analysis

- Global planning, MDP given with a random ulator

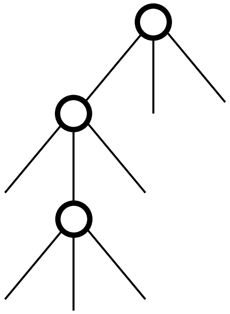
  Query complexity: $\tilde{\Theta}(SAH^3/\delta^2)$ [A 3]



- Local planning, MDP gi ocal-access simulator

  Query complexi $O((H^7A/\delta^2)^H)$ [Kearns et al., ’02]

  No depende ut exponential dependence on H.



What did we learn so far?

Value iteration and policy iteration have complementary strengths

Sampling can help reduce complexity

All algorithms suffer from one of the following “curses”:

- exponential in H complexity
- linear complexity in SA

Large problems: $SA \wedge H$ is large

online learning

- Not all exploration strategies are born equal

- General lessons:
  - Exploration is separate from optimization
  - But everything is just optimization
  - If adaptivity is the goal, optimism is the answer, while epsilon-greedy falls short
  - why regret is more meaningful than reward for comparing algorithms

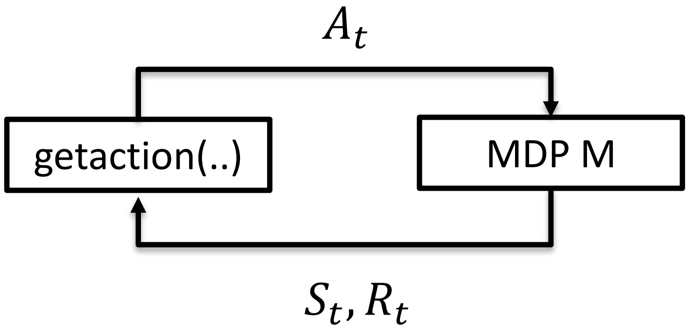
Problem settings

def getaction(s,r,S,A):

MDP with S states, A actions

…

return a # $a \in [A]$

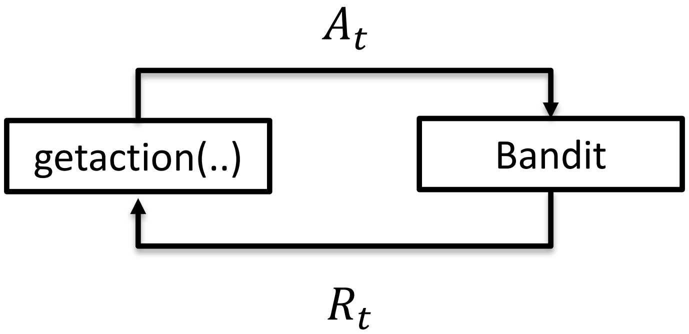


def bandit_getaction(r,A):

Bandit problem (S=1)

…

return a # $a \in [A]$



Performance metric

Total (expected) reward

$$V_T(\mathcal{A}, M) \coloneqq \mathbb{E}_{\mathcal{A},M}[\Sigma_{t=1}^{T} r(S_t, A_t)]$$

Goal: Find a single algorithm $\mathcal{A}$ that achieves as much reward as it is possible no matter the MDP M that the algorithm interacts with.

Say, M∈ $\mathcal{M}$ is an MDP with S states, A actions, rewards in $[0,1]$.

Perhaps this?

$$\mathrm{V}_{\mathrm{T}}^{*} \coloneqq \max_{\mathcal{A}} \min_{M \in \mathcal{M}} V_T(A, M)$$

Does this make sense? Why or why not?

Performance metric: Second attempt

$$\forall \mathcal{A}: \min_{M \in \mathcal{M}} V_T(\mathcal{A}, M) = 0 = V_T^*$$

$$r \equiv 0 \Rightarrow V_T(\mathcal{A}, M) = 0$$

Uninteresting M!

Big idea (Savage, Wald, 1950-51): **Metric should express how much an algorithm loses compared to the best specialized algorithm**!

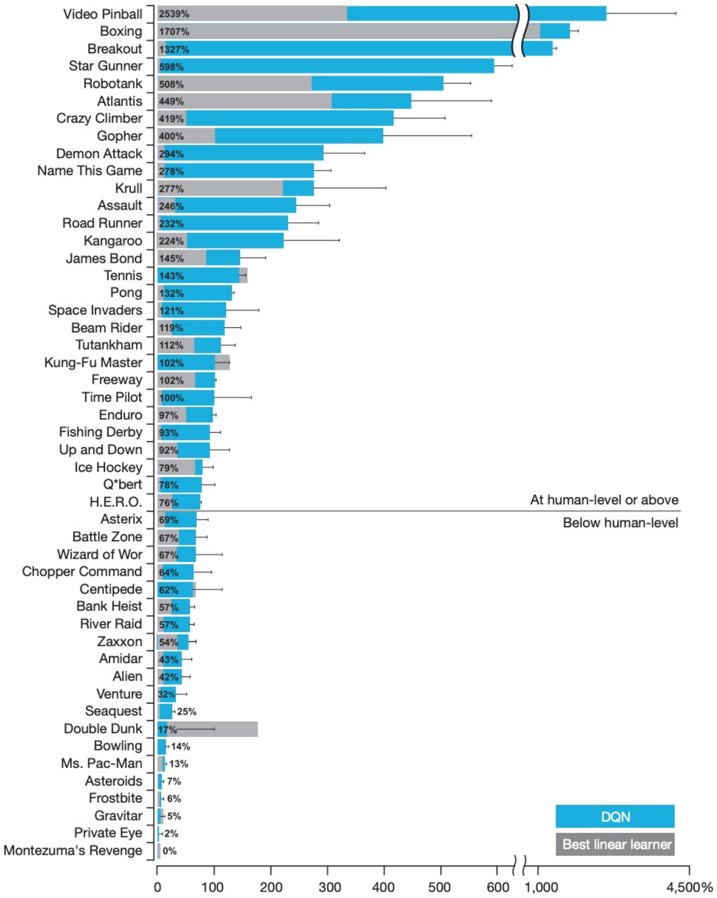
Regret:

$$R_T(\mathcal{A}, M) \coloneqq \max_{\mathcal{A}'} V_T(\mathcal{A}', M) - V_T(\mathcal{A}, M)$$

Minimax regret: $R_T^* = \min_{\mathcal{A}} \max_{M \in \mathcal{M}} R_T(\mathcal{A}, M)$

Algorithm design = multiobjective optimization

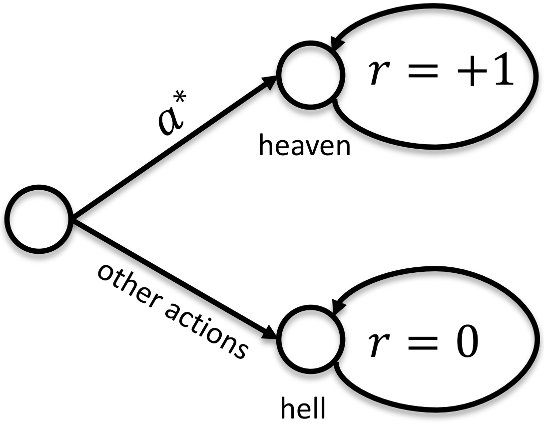
Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, et al. 2015. “Human-Level Control through Deep Reinforcement Learning.” *Nature* 518 (7540): 529–33.



Our troubles continue: Fix $S \geq 3, A \geq 2$

$\forall \mathcal{A}\ \exists M \in \mathcal{M}_{S,A}$ s.t.

$T\left(1-\frac{1}{A}\right) \leq R_T(\mathcal{A}, M) \leq T\ \ \forall \mathcal{A}$



oops.. Why? Traps!

Commute time/diameter:

$$\mathrm{diam}(M) = \max_{s,s' \in [S]} \min_{\pi} d_{\pi,M}(s,s')$$

Suggestion: Swap $\mathcal{M}_{S,A}$ with $\mathcal{M}_{D,S,A}$ where

$$\mathcal{M}_{D,S,A} = \{\, M \in \mathcal{M}_{S,A} : \mathrm{diam}(M) \leq D \}$$

Theorem (Jaksch, Ortner, Auer, 2010): For some $0 < c \le c'$, for any $T \ge 1, D \ge 6 + 2\log_A S, S \ge 3, A \ge 2$,

$$c\left(\sqrt{DSAT} \wedge T\right) \le R_T^*\left(\mathcal{M}_{D,S,A}\right) \le c' DS\sqrt{AT \log(SAT)}$$

How to read this result?

Problem oriented analysis

What next?

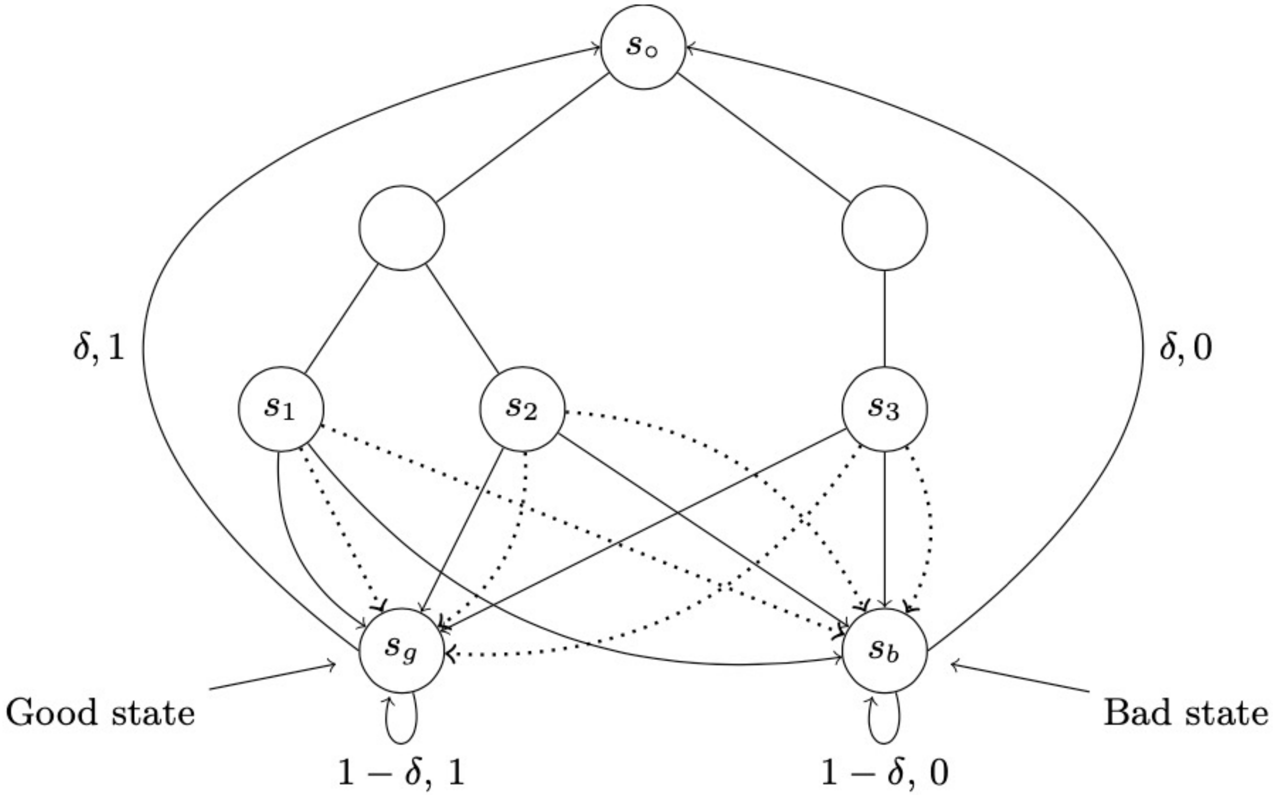
Remove log?

Close $\sqrt{DS}$ gap?

Lower bound

S=8 A=2

$\delta \approx 1/D$



```
Input $\mathcal{S}$, $\mathcal{A}$, $r$, $\delta \in (0,1)$
$t = 0$
for $k = 1, 2, \dots$ do
    $\tau_k = t + 1$
    Find $\pi_k$ as the greedy policy with respect to $v_k$ satisfying Eq. (38.16)
    do
        $t \leftarrow t + 1$, observe $S_t$ and take action $A_t = \pi_k(S_t)$
    while $T_t(S_t, A_t) < 2T_{\tau_k - 1}(S_t, A_t)$
end for
```

$$\begin{aligned} \rho_k + v_k(s) &= \max_{a \in \mathcal{A}} r_a(s) + \langle P_{k,a}(s), v_k \rangle \text{ for all } s \in \mathcal{S} \text{ and } a \in \mathcal{A}\,, \\ \rho_k &= \max_{s \in \mathcal{S}} \max_{\pi \in \Pi_{\mathrm{DM}}} \max_{P \in \mathcal{C}_{\tau_k}} \rho_s^\pi(P)\,, \end{aligned} \tag{38.16}$$

$$\hat{P}_{t,a}(s, s') = \frac{\sum_{u=1}^t \mathbb{I}\{S_u = s, A_u = a, S_{u+1} = s'\}}{1 \vee T_t(s,a)} \qquad T_t(s,a) = \sum_{u=1}^t \mathbb{I}\{S_u = s, A_u = a\}$$

$$\mathcal{C}_t(s,a) = \left\{ P \in \mathcal{P}(\mathcal{S}) : \|P - \hat{P}_{t-1,a}(s)\|_1 \leq \sqrt{\frac{\mathrm{S}L_{t-1}(s,a)}{1 \vee T_{t-1}(s,a)}} \right\} \qquad L_t(s,a) = 2\log\left(\frac{4\mathrm{SA}T_t(s,a)(1 + T_t(s,a))}{\delta}\right)$$

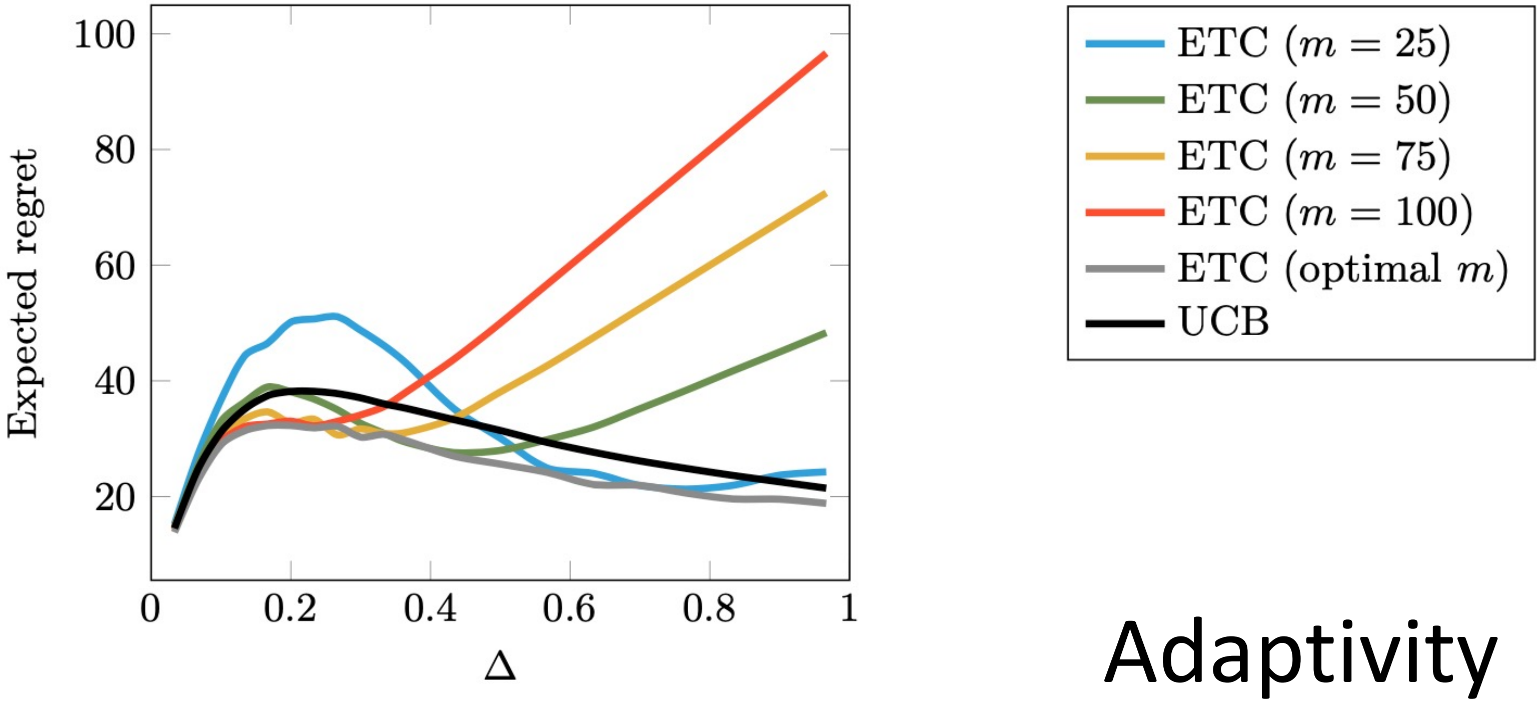
Optimism vs. forced exploration

Forced exploration

Systematically or randomly explore the actions for some portion of time

Optimism

Act as if the environment was the best among those that are plausible given the data so far



Adaptivity

Scaling up?

- Lower bound is clear: Without further assumptions, we hit a wall

- One possible avenue: Value function approximation

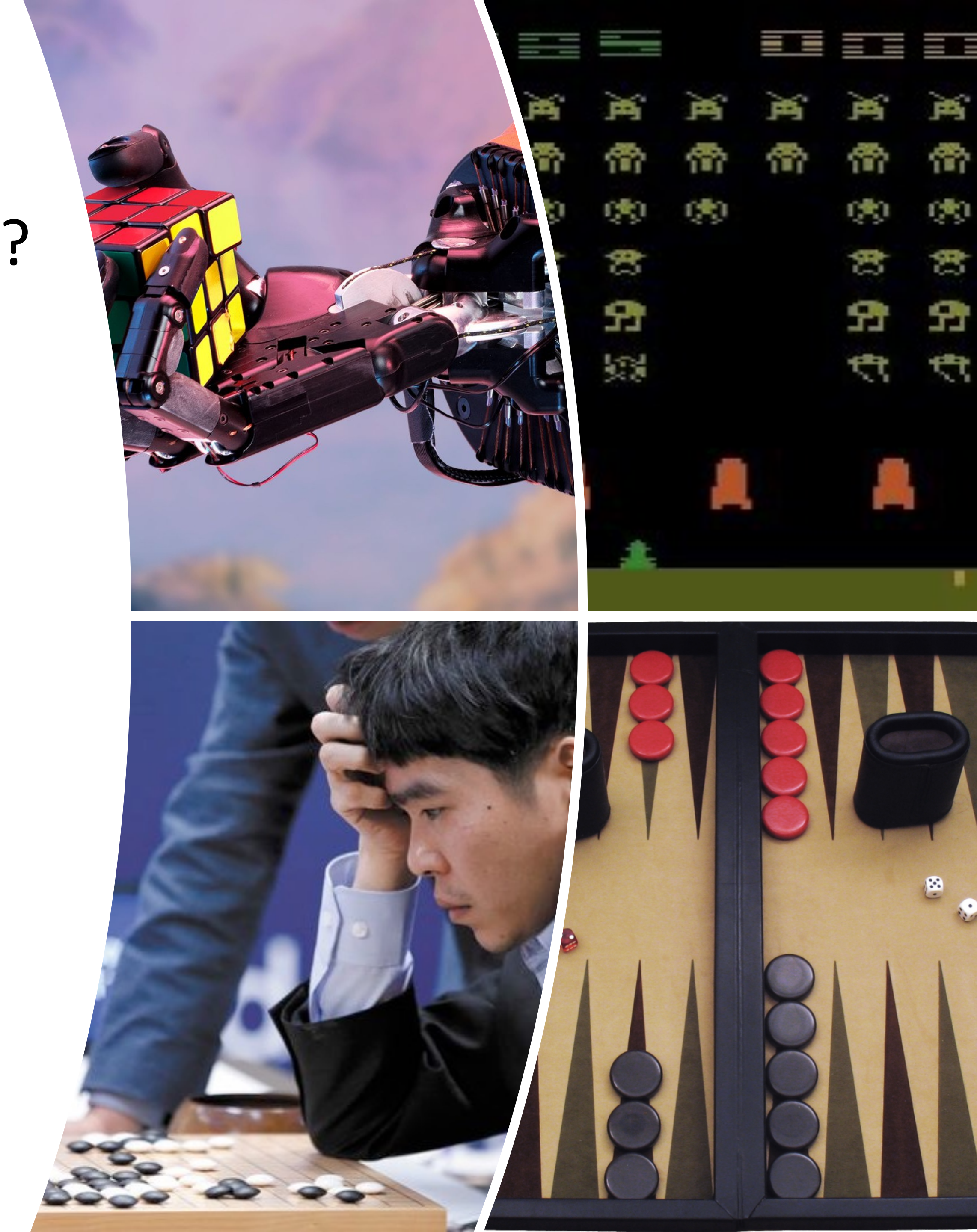
- We do it in planning first!

function approximation

- Function approximation is why RL algorithms can scale to large problems
- RL with function approximation = computation with compressed representations
- Classic DP algorithms can be made to work but have high demand for the function approximator
- Misspecification error inflation is unavoidable with poly-time algorithms
- Algorithms must control extrapolation error unlike in supervised learning
- Interesting case: only the optimal value function is compressible

How big is your MDP?

- Horizon: 100-1000+ steps
- Backgammon
  10^{20}
- Atari 2600 games:
  $2^{128} \approx 10^{38}$
- Game of Go
  10^{172} board positions
- Dexterous arm:
  60-dimensional,
  continuous state-space



Why/when does RL succeed?

Helpful/critical:

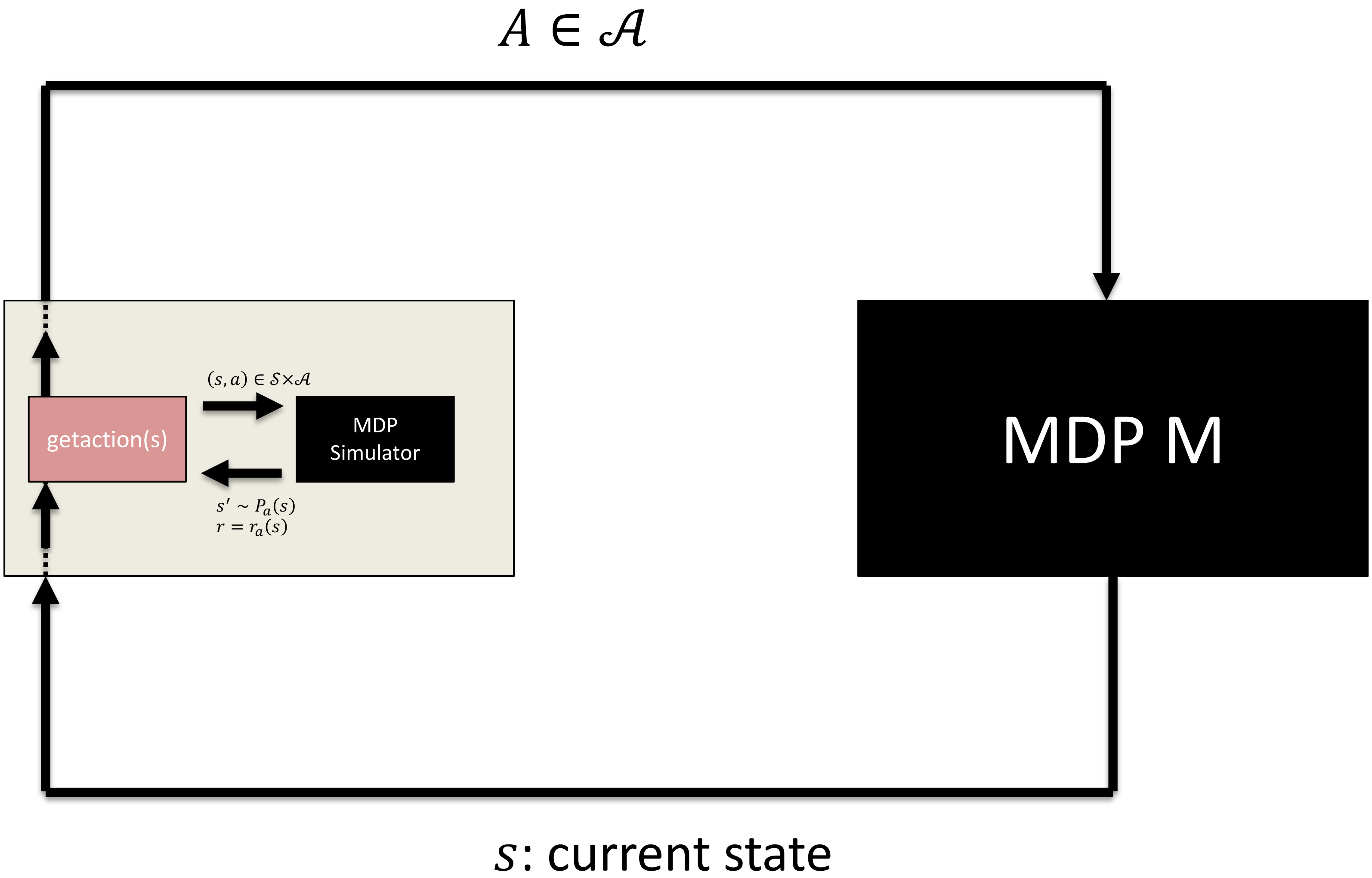
- Simulator
- Large compute
- Large neural networks
  ≈ function approximation

Planning

Generality
/flexibility

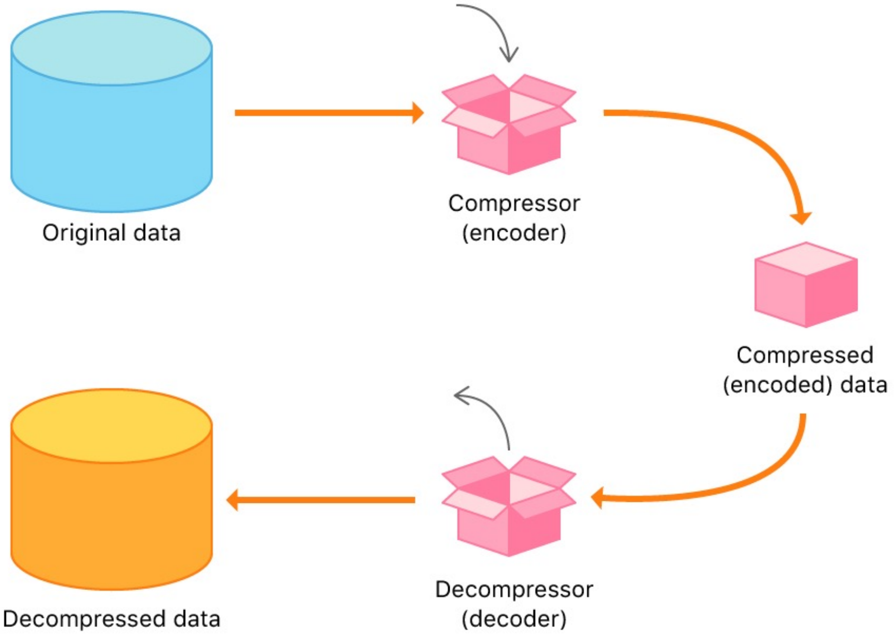
But are these sufficient? When?
Which algorithms will work?

Setting: Online planning (MPC)



Function approximation = compression!

(this can help!)



- Value functions

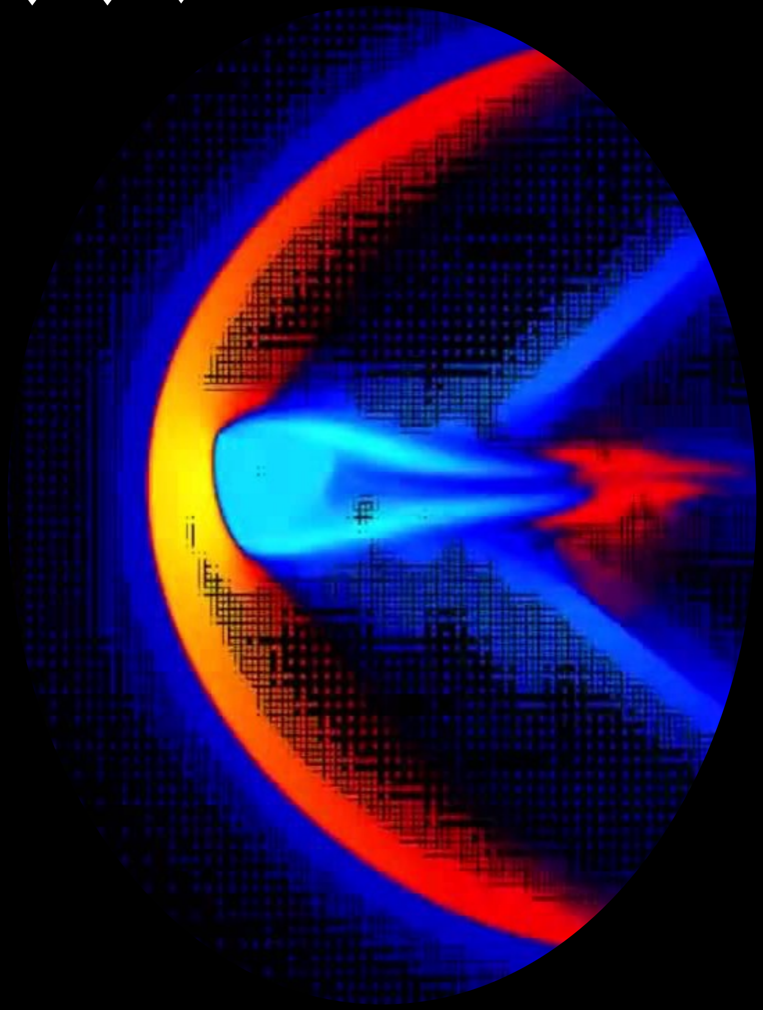
$$v^*, v^\pi : \mathcal{S} \to \mathbb{R}$$

$$q^*, q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$$

- Policies

$$\pi : \mathcal{S} \to \mathcal{A}$$

Image from: https://developer.apple.com/documentation/compression



Large scale RL $\equiv$ Computation in compressed form

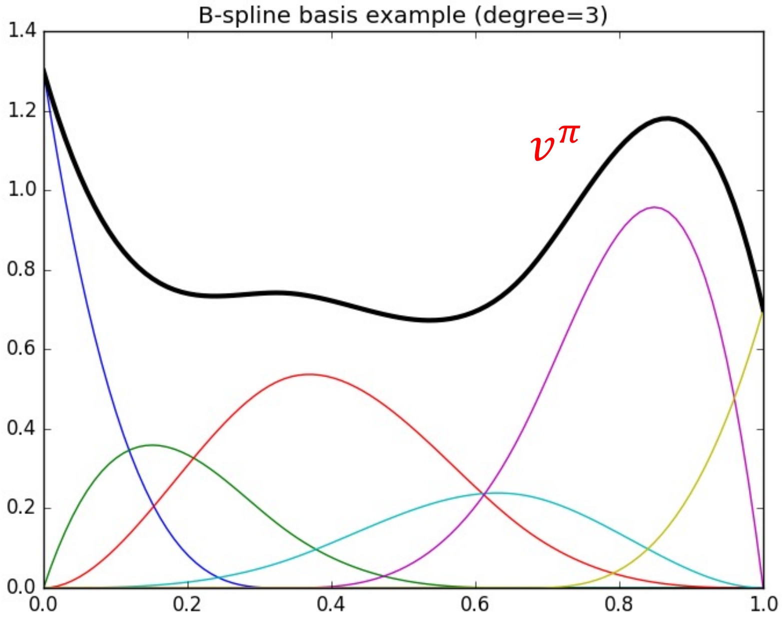
$\mathcal{S} = [0,1]$

$\# \text{ states} = \infty$

$d = 6$

$$v^\pi(s) \approx \Sigma_{i=1}^{d} \theta_i \phi_i(s)$$
$$\forall s \in \mathcal{S}$$

linear function approximation


B-spline basis example (degree=3)

$$d = 6 \ll \#\text{states} = \infty$$

https://patsy.readthedocs.io/en/latest/spline-regression.html

Problem settings

online planning
random access simulator

```
def getaction( simulator, $\delta, \xi$ ):
  (S,A) := simulator.problemsize()
  …
    (s’,r’) := simulator.gen(s,a) # $s \in [S], a \in [A]$
    f := simulator.getfeature(s,a) # $f = \phi(s,a)$
  …
  return $a$   # $a \in [A]$ s.t. for the policy $\pi$ induced, $v^{\pi} \geq v^{*} - \delta\mathbf{1}$ w.p. 1−ξ
```

online planning
local access simulator

```
def getaction( simulator, $s_0, f_0, \delta, \xi$ ):
  A := simulator.num_actions() # $f_0 = \phi(s_0)$
  …
    (s’,r’,f’) := simulator.gen(s,a) # $s$: state previously seen, $a \in [A]$, $f' = \phi(s')$
  …
  return $a$   # $a \in [A]$ s.t. for the policy $\pi$ induced, $v^{\pi} \geq v^{*} - \delta\mathbf{1}$ w.p. 1−ξ
```

Algorithm requirements

- Flexibility

Accept any feature map ϕ and MDP simulator

- Effectiveness

Policy induced should improve
with MDP-feature-map "fitness"

- Efficiency

poly(H,A,d,1/δ) runtime, regardless of #states

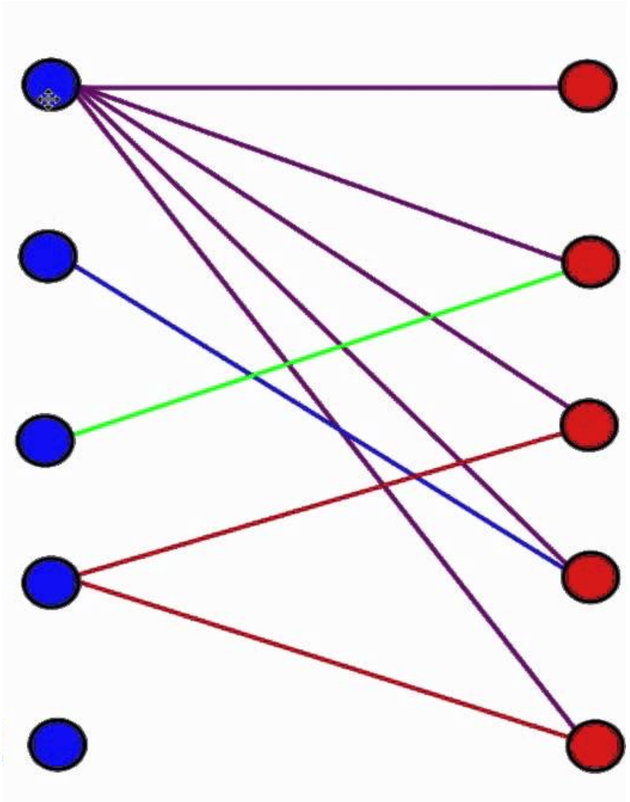
Fitness between MDPs and features

State-action feature-map $\phi: \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$

Def:
$f_\theta(s,a) \coloneqq \theta^\top \phi(s,a) \qquad (s,a) \in \mathcal{S} \times \mathcal{A}$

- $\varepsilon_{\text{saopt}}(M,\phi) = \inf_\theta \|q^* - f_\theta\|_\infty$
- $\varepsilon_{\text{sapol}}(M,\phi) = \sup_\pi \inf_\theta \|q^\pi - f_\theta\|_\infty$
- $\varepsilon_{\text{sabe}}(M,\phi) = \inf_\theta \|T f_\theta - f_\theta\|_\infty$
- …

Can do the same with state-features



Effectiveness

$$v^* - v^\pi = \mathrm{g}\big(\, \varepsilon(\mathrm{MDP}, \phi)\big) + \mathrm{O}(\mathrm{poly}(\mathrm{H}, \mathrm{A}, \mathrm{d})/N^p)$$

N: effort, $p > 0$

Examples for g:

$$g(\varepsilon) = 0.1\, \varepsilon$$
$$g(\varepsilon) = H\, \varepsilon$$
$$g(\varepsilon) = \sqrt{d} H\, \varepsilon$$

Results for DP-style methods

- Policy iteration + function approximation
  - All $q^\pi \in \mathcal{F}$
  - Rollout from core set found by solving G-optimal design, least-squares fit to return
  - Fully polytime
  - Approximation error inflation by $\sqrt{d}H$ (slower optimizer, conservative updates ≈NPG,PPO,..) or $\sqrt{d}H^2$ (least-squares policy iteration)
  - Lower inflation comes at exponential increase of compute cost (sphere packing)

- Value iteration + function approximation
  - $\mathrm{Img}(T) \subset \mathcal{F}$ (or $T\mathcal{F} \subset \mathcal{F}$)
  - Tf approximated by least-squares + sampling, from core set found by solving G-optimal design
  - Fully polytime, same as above
  - Similar to DQN

Results for DP-style method

- Policy iteration + function approximation
  - All $q^{\pi} \in \mathcal{F}$
  - Rollout from core set found by solving G-[illegible] least-squares fit to return
  - Fully polytime
  - Approximation error inflation [illegible] er optimizer, conservative updates ≈NPG,PPO,..) or [illegible] uares policy iteration)
  - Lower inflation comes [illegible] increase of compute cost (sphere packing)

- Value iterati[illegible] approximation
  - $\mathrm{Img}($[illegible]$\subset \mathcal{F})$
  - T[illegible] ed by least-squares + sampling, from core set found by [illegible] timal design
  - [illegible] ytime, same as above
  - Si[illegible] ar to DQN

Algorithm oriented analysis?

```
def policy_iteration($P, r, \delta$): # getpolicy fn
$\pi$ :=arbitrary, $k \coloneqq 0, H \coloneqq 1/(1-\gamma)$
while $\gamma^k H > \delta$:
    for all $s \in [S]$:
        $\pi'(s) \coloneqq \mathrm{argmax}_a \overbrace{r(s,a) + \gamma\langle P(s,a), v^\pi \rangle}^{q^\pi(s,a)}$
    $\pi \coloneqq \pi', k \coloneqq k + 1$
return $\pi$
```

$$\begin{aligned} v^\pi &= r_\pi + \gamma P_\pi v^\pi \\ &=: T_\pi v^\pi \end{aligned}$$

Shorthand: $\pi' \coloneqq \Gamma v^\pi$

```
def policy_iteration($P, r, \delta$): # getpolicy fn
    $q \coloneqq 0\ , k \coloneqq 0, H \coloneqq 1/(1-\gamma)$
    while $\gamma^k H > \delta$:
        $q' \coloneqq$ eval( gpolicy(q) ) # $q^{\mathrm{gpolicy}(q)}$
        $q \coloneqq q', k \coloneqq k+1$
    return $\mathrm{gpolicy}(\mathrm{q})$

def gpolicy(q)$(s) \coloneqq \mathrm{argmax}_a\, q(s,a)$
```

```
def fitted_policy_it(simulator, $\delta$): # getpolicy fn
    $\phi \coloneqq \mathrm{simulator.getfeatures}(s, a)\ \forall s, a$
    $C \coloneqq \mathrm{coreset}(\phi)$ # G-optimal design
    $\theta \coloneqq 0, k \coloneqq 0, H \coloneqq 1/(1-\gamma)$
    while $\gamma^k H + \cdots > \delta$:
        $\theta' \coloneqq \mathrm{eval}(\mathrm{simulator}, C, \phi, \mathrm{gpolicy}(\phi, \theta))$
        $\theta \coloneqq \theta', k \coloneqq k + 1$
    return $\mathrm{policy}(\phi, \theta)$
```

```
def eval(simulator, $C, \phi, \pi$):
    $M \coloneqq \ldots, T \coloneqq \ldots$ , data:=[]
    for $(s_0, a_0) \in C$:
        return := 0, $(s, a) \coloneqq (s_0, a_0)$
        for $m$ in $\{1, \ldots, M\}$:
            for $t$ in $\{0, \ldots, T\}$:
                (s’,r’) := simulator.gen(s,a), return += $\gamma^t r'$
                s:=s’, a$\coloneqq\pi$(s)
        return /= M
        data.append( simulator.getfeature(s,a), return )
    return leastsquaresfit(data)
```

Weighted least-squares extrapolation error control. Let $\mathcal{Z} \subset \mathbb{R}^d$.

Theorem

For any $\theta \in \mathbb{R}^d, \varepsilon: \mathcal{Z} \to \mathbb{R}, \rho \in \Delta_1(\mathcal{Z})$ such that

$$G_\rho := \Sigma_{z \in \mathcal{Z}} \rho(z) z\, z^\top$$

is nonsingular, for any $z \in \mathcal{Z}$:

$$\left| z^\top \hat{\theta} - z^\top \theta \right| \leq \|z\|_{G_\rho^{-1}} \max_{z' \in \mathcal{Z}} |\varepsilon(z)|$$

where

$$\hat{\theta} = G_\rho^{-1} \Sigma_{z \in \mathcal{Z}}\, \rho(z) \left(z^\top \theta + \varepsilon(z) \right) z.$$

G-optimal design

Theorem (Jack Kiefer-Jacob Wolfowitz)
Let $\mathcal{Z} \subset \mathbb{R}^d$ be an arbitrary finite set, spanning $\mathbb{R}^d$.
There exists $\mathcal{C} \subset \mathcal{Z}$ finite, $\rho \in \Delta_1(\mathcal{Z})$ supported on $\mathcal{C}$ such that

1. $|\mathcal{C}| \leq \frac{d(d+1)}{2}$
2. $\max_{z \in \mathcal{Z}} \|z\|_{G_\rho^{-1}} \leq \sqrt{d}$
3. In fact $\inf_{\rho} \max_{z \in \mathcal{Z}} \|z\|_{G_\rho^{-1}} = \sqrt{d}$.

Compressing optimal value functions

- Only q^* is realizable, there are many actions
  - $2^{\Omega(d \wedge H)}$ lower bound: No query efficient algorithm exists
  - Sphere packing: Can squeeze $k \approx \exp(\tau^2 d)$ vectors on the d-dimensional sphere such that any distinct two of them are τ- orthogonal. Needle in a haystack!
  - Build a tree with these being the actions. Rewards zero except at the end of the episode, where their SNR is "exponentially" poor.

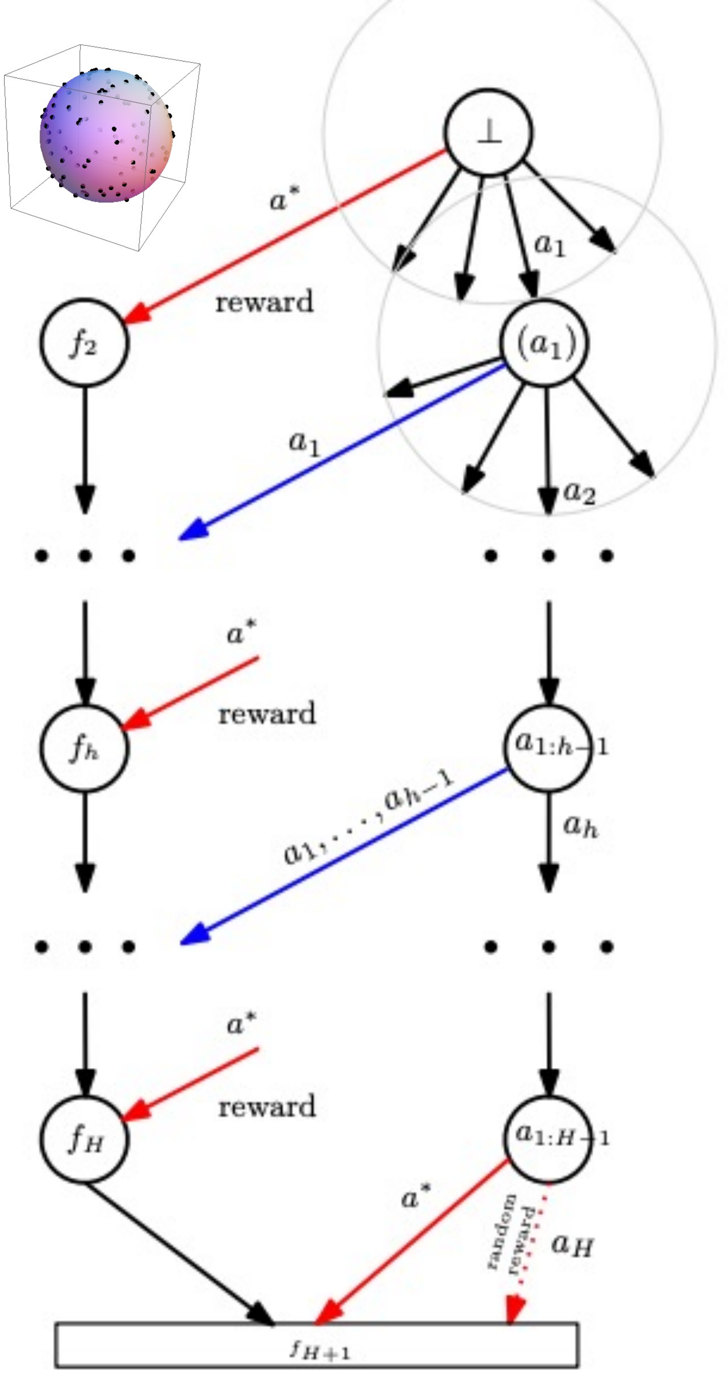
- Only v^* is realizable, there are only a few actions
  - There is a query efficient algorithm with query cost $\tilde{O}(\ (dH/\delta)^A\ )$
  - Algorithms: Optimistic parameter choice, rollouts to check for consistency (zero TD-error)
  - Poly compute time?

Compressing optimal value functions

- Only q^* is realizable, there are many action
  - $2^{\Omega(d \wedge H)}$ lower bound: No query efficient al
  - Sphere packing: Can squeeze $k \approx \exp($ on the d-dimensional sphere such that any di them are τ- orthogonal. Needle in a hayst
  - Build a tree with these bein Rewards zero except at the end of the episode, whe "exponentially" poor.
- Only v^* is reali e are only a few actions
  - There is a nt algorithm with query cost $\tilde{O}(\,(dH/\delta)^A\,)$
  - Algo istic parameter choice, rollouts to check for c o TD-error)
  - Po te time?

Problem oriented analysis

q^* lower bound



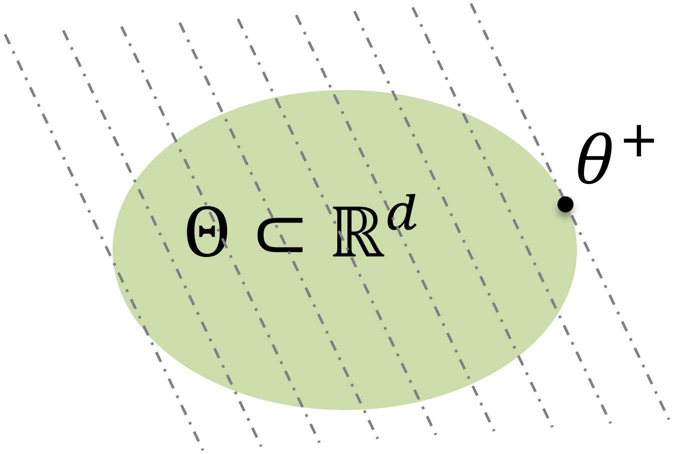
1. Maintain Bellman optimality equation
2. same a^* optimal @ every state, 2^d actions

$$q_h^*(s,a) = r_a(s) + q_{h+1}^*(s,a^*)$$

3. Features from sphere packing
4. Deterministic transitions
5. Last stage: Bernoulli rewards, scale 2^{-H}
6. Scale features by 2^{-h} in stage h
7. 2^d actions necessary for hiding large reward gap at $h = 1$
8. Suboptimal actions a should give no info: $r_a(s) = 0$
9. Feature-bias to maintain consistency
10. Exit lane to maintain consistency

TensorPlan: Optimism + test/rollouts

$v_h(s_0;\theta) \coloneqq \theta^\top \phi_h(s_0)$



Given θ^+, roll out with π_{θ^+} to check whether:

1. $v_1(s_0;\theta)$ is achieved by π_{θ^+}
2. (*) holds

$\theta \mapsto \pi_\theta$:

$\pi_\theta(s) = a$ for the action a such that

(*) $v_h(s;\theta) = r_a(s) + P_a(s)^\top v_{h+1}(\cdot;\theta)$

No max!!

Why will TensorPlan stop changing Θ?

$\theta \mapsto \pi_\theta$:

$\pi_\theta(s) = a$ for the action a such that

(*) $v_h(s;\theta) = r_a(s) + P_a(s)^\top v_{h+1}(\cdot;\theta)$

π_θ is well-defined if $\forall s$ $\exists$ a such that

$$\Delta(s,a,\theta) \coloneqq r_a(s) + P_a(s)^\top v_{h+1}(\cdot;\theta) - v_h(s;\theta) = 0$$

$$\Leftrightarrow \Pi_a \Delta(s,a,\theta) = 0$$

$$\Leftrightarrow \left\langle \otimes_a \overline{r_a(s)\left(P_a(s)^\top \phi_{h+1} - \phi_h(s)\right)}, \otimes_a \overline{1\ \theta} \right\rangle = 0$$

$\otimes_a \overline{1\ \theta} \in \mathbb{R}^{(d+1)^A} \Rightarrow$ must stop after $(d+1)^A$ constraints

batch RL

- Minimax regret with policy induced data scales exponentially even in tabular MDPs

Problem setting

MDP with S states, A actions

$D = ((S_i, A_i, R_i, S_i')_{i\in[n]})$

```
def getpolicy($s, S, A, D, \delta, \xi$):
    …
    return $\pi$   # $v^{\pi}(s) \geq v^{*}(s) - \delta \; w.p. \, 1 - \xi$
```

Policy induced data

Data is obtained by following some "logging" policy $\pi_{\log}$ for some episodes in the MDP.

$$A_i = \pi_{\log}(S_i)$$

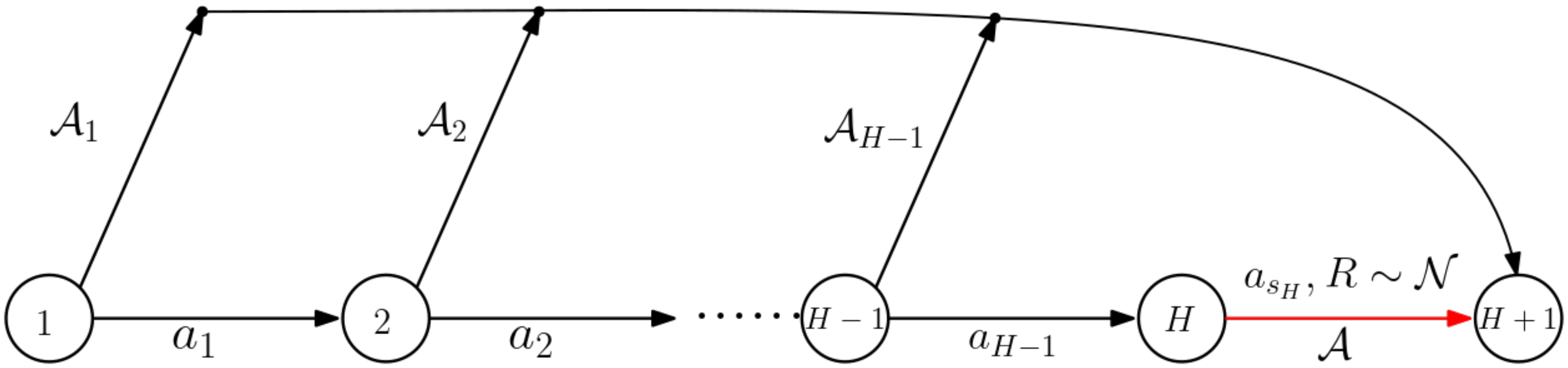
Theorem

With policy induced data, for any logging policy, any constant-probability δ-sound algorithm needs at least

$$c\ A^{\min(S-1,H)} / \ \delta^2$$

observations on some MDP with S states, A actions and horizon H.

Problem oriented analysis



Choose $a_i = \underset{a}{\operatorname{argmin}}\, \pi_{\log}(a|i)$
$R \sim \mathcal{N}(\mu, 1), \mu \in \{\pm 2\delta\}$

π_0: choose a_i in state i

If $\mu = 2\delta$: $v^{\pi_0}(1) = 2\delta, v^{\pi}(1) = 0$ for any other (deterministic) π.
If $\mu = -2\delta$: $v^{\pi_0}(1) = -2\delta, v^{\pi}(1) = 0$ for any other (deterministic) π.

If $H \to H+1$ transition not seen $1/\delta^2$ times, can't choose between π_0 and others and keep the error small.

Open question

- What is a good way of evaluating and comparing batch RL algorithms beyond worst-case?
- Instance optimality to the rescue? Nope.
- Pessimistic algorithm
  - Studied in many fields under many names
  - Addresses winner’s remorse
  - Weighted-minimax optimal
  - Samples need to provide coverage only where π^* goes

Summary

RL⊆CS

Algorithms! Instances!

Foundations by MDPs, planning

Online learning

Function approximation in planning

Batch RL

Questions?

References

- https://rltheory.github.io (RL Theory Lecture notes, CMPUT 652 @UofA)
- Agarwal, Jiang, Kakade, Sun. Reinforcement Learning: Theory and Algorithms https://rltheorybook.github.io/
- RL Theory virtual seminar series: https://sites.google.com/view/rltheoryseminars

Basics

- Martin L. Puterman. Markov Decision Processes: Discrete Stochastic Dynamic Programming, 1994
- Chen, Y., & Wang, M. (2017). Lower bound on the computational complexity of discounted markov decision problems. arXiv preprint arXiv:1705.07312. [link]
- Singh, S. P., & Yee, R. C. (1994). An upper bound on the loss from approximate optimal-value functions. Machine Learning, 16(3), 227-233.
- Feinberg, E. A., Huang, J., & Scherrer, B. (2014). Modified policy iteration algorithms are not strongly polynomial for discounted dynamic programming. Operations Research Letters, 42(6-7), 429-431. [link]
- Scherrer, B. (2016). Improved and generalized upper bounds on the complexity of policy iteration. Mathematics of Operations Research, 41(3), 758-774. [link]
- Ye, Y. (2011). The simplex and policy-iteration methods are strongly polynomial for the Markov decision problem with a fixed discount rate. Mathematics of Operations Research, 36(4), 593-603. [link]
- Kearns, M., Mansour, Y., & Ng, A. Y. (2002). A sparse sampling algorithm for near-optimal planning in large Markov decision processes. Machine learning, 49(2), 193-208. [link]
- Remi Munos (2014). From Bandits to Monte-Carlo Tree Search: The Optimistic Principle Applied to Optimization and Planning. Foundations and Trends in Machine Learning: Vol. 7: No. 1, pp 1-129.

Online Learning

- Thomas Jaksch, Ronald Ortner, and Peter Auer. Near-optimal regret bounds for reinforcement learning. The Journal of Machine Learning Research, 11:1563–1600, 2010.
- Zihan Zhang and Xiangyang Ji. Regret minimization for reinforcement learning by evaluating the optimal bias function. arXiv preprint arXiv:1906.05110, 2019.
- Bourel, Hippolyte, Odalric Maillard, and Mohammad Sadegh Talebi. 2020. “Tightening Exploration in Upper Confidence Reinforcement Learning.” Edited by Hal Daumé Iii and Aarti Singh, Proceedings of Machine Learning Research, 119: 1056–66.
- Fruit, Ronan, Matteo Pirotta, and Alessandro Lazaric. n.d. “Improved Analysis of UCRL2 with Empirical Bernstein Inequality.” https://rlgammazero.github.io/docs/ucrl2b_improved.pdf.
- Lattimore, T., & Szepesvári, C. (2020). Bandit algorithms. Cambridge University Press.
- L.J. Savage, The theory of statistical decision, J. Amer. Statist. Assoc. 46 (1951) 55-67.
- Wald, Statistical Decision Functions, Wiley, New York, 1950.

Function approximation

- Simon S. Du, Sham M. Kakade, Ruosong Wang, and Lin F. Yang. 2020. “Is a Good Representation Sufficient for Sample Efficient Reinforcement Learning?” ICLR and arXiv:1910.03016.
- Tor Lattimore, Csaba Szepesvári, and Gellért Weisz. 2020. “Learning with Good Feature Representations in Bandits and in RL with a Generative Model.” ICML and arXiv:1911.07676.
- Roshan Shariff and Csaba Szepesvári. 2020. “Efficient Planning in Large MDPs with Weak Linear Function Approximation”. In NeurIPS 2020 and arXiv:2007.06184
- Gellért Weisz, Philip Amortila, Csaba Szepesvári. 2020. Exponential Lower Bounds for Planning in MDPs With Linearly-Realizable Optimal Action-Value Functions, To appear at ALT and also arXiv:2010.01374
- Gellért Weisz, Philip Amortila, Barnabás Janzer, Yasin Abbasi-Yadkori, Nan Jiang, Csaba Szepesvári. 2021. On Query-efficient Planning in MDPs under Linear Realizability of the Optimal State-value Function, arXiv:2102.02049
- Z. Wen and B. Van Roy. 2017. "Efficient Reinforcement Learning in Deterministic Systems with Value Function Generalization", *Mathematics of Operations Research*, 42(3):762–782. [arXiv]
- Simon S Du, Yuping Luo, Ruosong Wang, and Hanrui Zhang. Provably efficient Q-learning with function approximation via distribution shift error checking oracle. In Advances in Neural Information Processing Systems, pages 8060–8070, 2019b.

Function approximation/2

- Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, John Langford, and Robert E Schapire. Contextual decision processes with low Bellman rank are PAC-learnable. In International Conference on Machine Learning, pages 1704–1713. PMLR, 2017.
- Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I Jordan. Provably efficient reinforcement learning with linear function approximation. In Conference on Learning Theory, pages 2137–2143, 2020.
- Lin Yang and Mengdi Wang. Sample-optimal parametric q-learning using linearly additive features. In ICML, pages 6995–7004, 2019.
- Richard Bellman, Robert Kalaba and Bella Kotkin. 1963. Polynomial Approximation--A New Computational Technique in Dynamic Programming: Allocation Processes. Mathematics of Computation, 17 (82): 155-161
- Daniel, James W. 1976. “Splines and Efficiency in Dynamic Programming.” *Journal of Mathematical Analysis and Applications* 54 (2): 402–7.
- Schweitzer, Paul J., and Abraham Seidmann. 1985. “Generalized Polynomial Approximations in Markovian Decision Processes.” *Journal of Mathematical Analysis and Applications* 110 (2): 568–82.

Batch RL

- Xiao, Chenjun, Ilbin Lee, Bo Dai, Dale Schuurmans, and Csaba Szepesvari. 2021. “On the Sample Complexity of Batch Reinforcement Learning with Policy-Induced Data.” *arXiv [cs.LG]*. arXiv. http://arxiv.org/abs/2106.09973
- Andrea Zanette. Exponential lower bounds for batch reinforcement learning: Batch RL can be exponentially harder than online RL. In ICML, 2021.
- Chenjun Xiao, Yifan Wu, Tor Lattimore, Bo Dai, Jincheng Mei, Lihong Li, Csaba Szepesvári, and Dale Schuurmans. On the optimality of batch policy optimization algorithms. In ICML, 2021.
- Philip Amortila, Nan Jiang, and Tengyang Xie. A variant of the Wang-Foster-Kakade lower bound for the discounted setting. arXiv preprint 2011.01075, 2020.
- Ruosong Wang, Dean P. Foster, and Sham M. Kakade. What are the statistical limits of offline RL with linear function approximation?, 2020.
- Jacob Buckman, Carles Gelada, and Marc G. Bellemare. The importance of pessimism in fixed-dataset policy optimization. In ICLR, 2021.
- Lin Chen, Bruno Scherrer, and Peter L. Bartlett. Infinite-horizon offline reinforcement learning with linear function approximation: Curse of dimensionality and algorithm. arXiv preprint 2103.09847, 2021.

Batch RL

- Nan Jiang and Lihong Li. Doubly robust off-policy value evaluation for reinforcement learning. In ICML, pages652–661, 2016.
- Ying Jin, Zhuoran Yang, and Zhaoran Wang. Is pessimism provably efficient for offline RL? In ICML, 2021.
- Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. Provably good batch off-policy reinforcement learning without great exploration. In NeurIPS, 2020.
- Masatoshi Uehara, Masaaki Imaizumi, Nan Jiang, Nathan Kallus, Wen Sun, and Tengyang Xie. Finite sample analysisof minimax offline reinforcement learning: Completeness, fast rates and first-order efficiency.arXiv preprintarXiv:2102.02981, 2021.
- Ming Yin and Yu-Xiang Wang. Asymptotically efficient off-policy evaluation for tabular reinforcement learning. In AISTATS, pages 3948–3958, 2020.
- Ming Yin, Yu Bai, and Yu-Xiang Wang. Near-optimal provable uniform convergence in offline policy evaluation for reinforcement learning. In AISTATS, pages 1567–1575, 2021a.
- Ming Yin, Yu Bai, and Yu-Xiang Wang. Near-optimal offline reinforcement learning via double variance reduction. arXiv preprint arXiv:2102.01748, 2021b.

Why so complicated?

How about policy search?

$\Pi = \{\, f\big(\, \phi(s,a)\big) : f \in \mathcal{F}\}, \mathcal{F} \subset (\Delta_{\mathcal{A}})^{\mathbb{R}^d}$

E.g. Boltzmann/softmax policies: Π_B

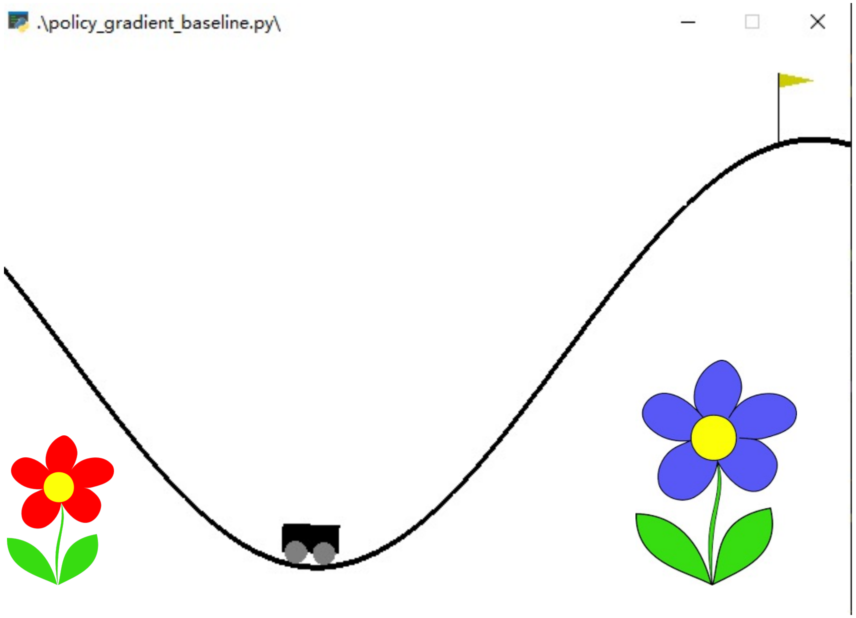
$$\operatorname*{argmax}_{\pi \in \Pi} J(\pi)$$

Theorem (Vlassis-Littman-Barber ‘12):

Policy search is NP-hard with $J(\pi) = \mu^\top v^\pi$, discounting, μ uniform, state-aggregation

Proof: MAX-INDSET

What can be compressed?



Visual mountain car