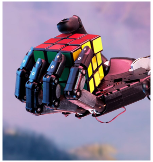
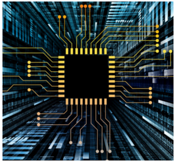
Computational-Statistical Gaps in Reinforcement Learning

Talk By: Gaurav Mahajan (UCSD)

Progress of RL in practice (And It Ain't Cheap)

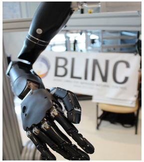


Robotics



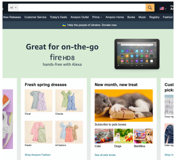
Chip Design

Dota2



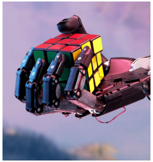
Prosthetics

Loon

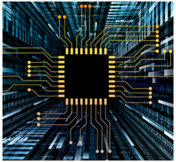


Search

Progress of RL in practice (And It Ain't Cheap)

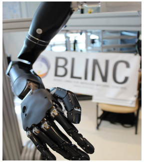


Robotics



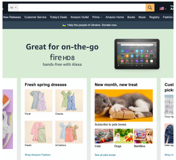
Chip Design

Dota2



Prosthetics

Loon

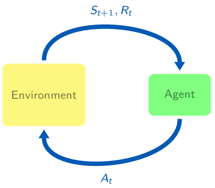


Search

- Huge computational and statistical demands.
  - Computational: OpenAI Five trained for 10 months.
  - Statistical: Played 10,000 years of games.

Goal: design statistically and computationally "efficient" algorithms in RL

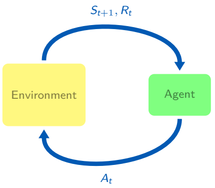
Framework for RL: MDPs and Trees

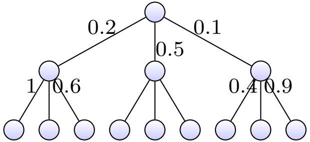


- Stochastic Transition $S_{t+1} \sim T(S_t, A_t)$
  Next state given current state and action
- Stochastic Reward $R_t \sim R(S_t, A_t)$
  Next reward given current state and action

- **Goal**: Find a policy π which maximizes the expected sum of rewards $V(\pi) = \mathbb{E}\left[\sum_{t=0}^{H} R_t \,\middle|\, \pi\right]$

Framework for RL: MDPs and Trees





- Stochastic Transition $S_{t+1} \sim T(S_t, A_t)$
  Next state given current state and action
- Stochastic Reward $R_t \sim R(S_t, A_t)$
  Next reward given current state and action

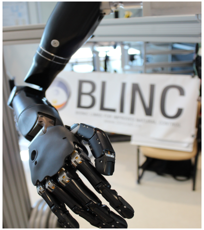
- **Goal**: Find a policy π which maximizes the expected sum of rewards $V(\pi) = \mathbb{E}\left[\sum_{t=0}^{H} R_t \;\middle|\; \pi\right]$

- Deterministic Transition

- Stochastic Reward
  Each edge e is associated with a noisy reward R_e.

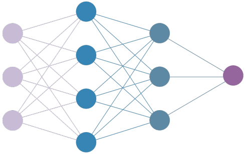
- **Goal**: Find a path π which maximizes the expected sum of rewards. $V(\pi) = \mathbb{E}\left[\sum_{e\in\pi}^{H} R_e\right]$,

This Talk: Interaction And Compute



Environment

Interaction



Compute

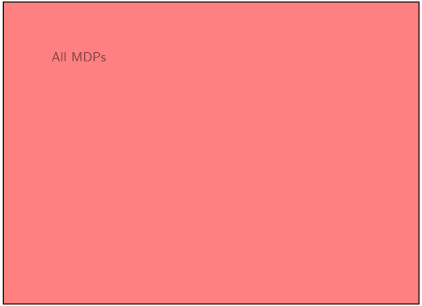
CPUs and GPUs

Statistical: amount of interaction with the environment to find near optimal policy

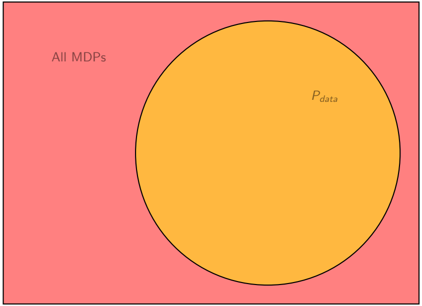
Computational: amount of compute to find near optimal policy

*near optimal policy π: $V^{\pi} > V^{*} - \epsilon$

This Talk: Goal

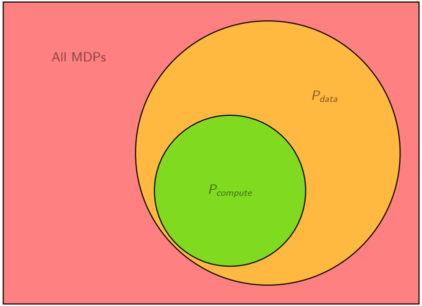


This Talk: Goal



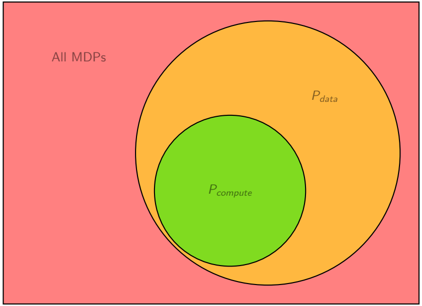
- MDPs with sample efficient algorithms (P_{data})

This Talk: Goal



- MDPs with sample efficient algorithms (P_{data})
- MDPs with computationally efficient algorithms ($P_{compute}$)

This Talk: Goal

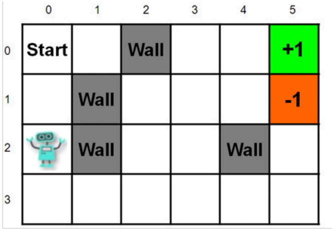


- MDPs with sample efficient algorithms (P_{data})
- MDPs with computationally efficient algorithms ($P_{compute}$)

Goal: characterize these classes of MDPs

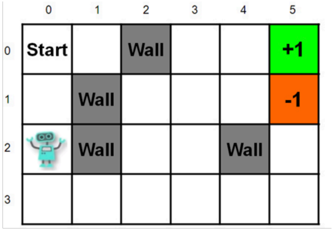
Classical Theory: Dependence on S or A^H

Q1: How many samples/compute do we need to find a near optimal policy?
for S states, A actions and H horizon



Classical Theory: Dependence on S or A^H

Q1: How many samples/compute do we need to find a near optimal policy?
for S states, A actions and H horizon



- **Theorem** (Kearns & Singh '98; ..., Kearns, Mansour, & Ng '00)
  min(**poly**$(S), A^H)$ samples/compute are sufficient and necessary to find a near optimal policy.
  - Algorithmic Ideas: Optimism + Dynamic Programming + Bonus
  - Hard Instance: Tree with reward only at a special leaf node.

With Assumptions: Independent of S.

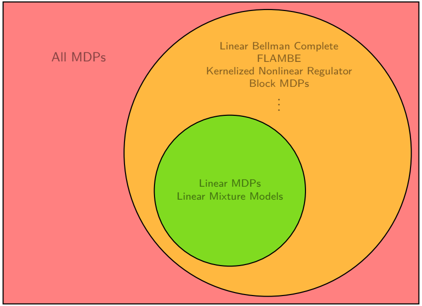
But Dota2 has $S \subset \mathbb{R}^{16000}$, Horizon $H \approx 20000!!!$

Q2: Can we find a near optimal policy with no $|S|$, $|A|$ dependence and **poly**$(H,$ "complexity measure"$)$?

- Polynomial Sample Complexity
  - Bellman Rank: [Jiang+ '17]
  - Linear MDPs: [Wang & Yang '18; Jin+ '19]
  - Linear Bellman Completion: [Zanette+ '19, Wang+ '2019]
  - Block MDPs [Du+ '19]
  - Factored MDPs [Sun+ '19]
  - Kernelized Nonlinear Regulator [Kakade+ '20]
  - FLAMBE / Feature Selection: [Agarwal+ '20]
  - Linear Mixture MDPs: [Modi+ '20, Ayoub+ '20, Zhou+ '21]
  - And more...

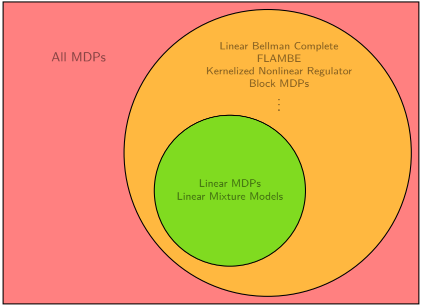
This Talk

- Too strong. Unlikely to be necessary. Want to exploit understanding of neural networks.



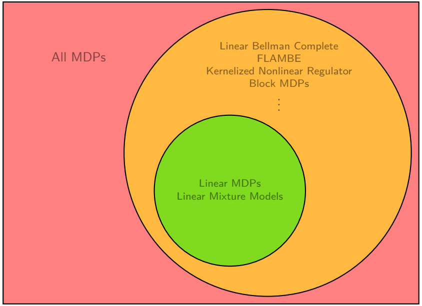
This Talk

- Too strong. Unlikely to be necessary. Want to exploit understanding of neural networks.
- Different proofs, algorithms. Structural property like VC dimension in supervised learning.



This Talk

- Too strong. Unlikely to be necessary. Want to exploit understanding of neural networks.
- Different proofs, algorithms. Structural property like VC dimension in supervised learning.
- Few computational results. When can we design computationally efficient algorithms?



This Talk

- Too strong. Unlikely to be necessary. Want to exploit understanding of neural networks.
- Different proofs, algorithms. Structural property like VC dimension in supervised learning.
- Few computational results. When can we design computationally efficient algorithms?

This Talk

- Introduce fundamental and natural setting: Linear Function Approximation.
  - Boundary of necessary vs sufficient
- Sample efficiency under Linear Function Approximation
  - Unifying Sufficient Structural Assumption for Sample Efficiency [DKLL**M**SW '21]
- Computational World: Different from Statistical World under Linear Function Approximation [KLL**M** '22]

Overview

Part 0: Natural Assumptions
- RL with Linear Function Approximation

Part 1: Why is RL hard?
- Baseline: Regression Chaining

Part 2: Sample Efficiency
- Algorithmic Ideas in Theory

Part 3: Computational Efficiency
- Different from sample efficiency
- Hard Instances

Part 0: RL with Linear Function Approximation

natural assumptions in RL

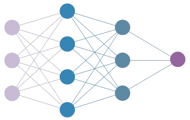
Linear Function Approximation

- Fundamental in theory: A lot of algorithms try to learn optimal value functions

$$V^*(s) = \max_{\pi} \mathbb{E}\left[\sum_{t=0}^{H} R_t \mid s_0 = s, \pi\right], \quad Q^*(s,a) = \max_{\pi} \mathbb{E}\left[\sum_{t=0}^{H} R_t \mid s_0 = s, A_0 = a, \pi\right]$$

- Fundamental in practice: A lot of model free algorithms used in practice try to learn the optimal value functions
  - Trains a neural network to predict optimal V^* and Q^* functions

(s, a)



$V^*(s), Q^*(s,a)$

$$\underset{\text{Learn features}}{\underline{\text{Representation Learning}}} + \underset{\text{Learn optimal value functions linear in these features}}{\underline{\text{Learning Linear Functions}}}$$

Linear Function Approximation: Linear $Q^\star$&$V^\star$

- Basic idea: Assume our neural networks learned "good" representations (features) $\phi(s,a), \psi(s) \in \mathbb{R}^d$ (where $d \ll \#states, \#actions$).

Linear Function Approximation

- Linear Q^*: There exists unknown $w^\star \in \mathbb{R}^d$ and known features $\phi : S \times A \to \mathbb{R}^d$ s.t.

$$Q^\star(s,a) = \langle w^\star,\ \phi(s,a)\rangle$$

- Linear V^*: There exists unknown $\theta^\star \in \mathbb{R}^d$ and known features $\psi : S \to \mathbb{R}^d$ s.t.

$$V^\star(s) = \langle \theta^\star,\ \psi(s)\rangle$$

- Lots of interesting variants: Linear Q^*, Linear V^*, Linear Q^*&V^* (reachable states).

Linear Function Approximation: Linear $Q^\star$&$V^\star$

- Basic idea: Assume our neural networks learned "good" representations (features) $\phi(s, a), \psi(s) \in \mathbb{R}^d$ (where $d \ll \#states, \#actions$).

Linear Function Approximation

- Linear Q^*: There exists unknown $w^\star \in \mathbb{R}^d$ and known features $\phi : S \times A \to \mathbb{R}^d$ s.t.

$$Q^\star(s, a) = \langle w^\star,\ \phi(s, a)\rangle$$

- Linear V^*: There exists unknown $\theta^\star \in \mathbb{R}^d$ and known features $\psi : S \to \mathbb{R}^d$ s.t.

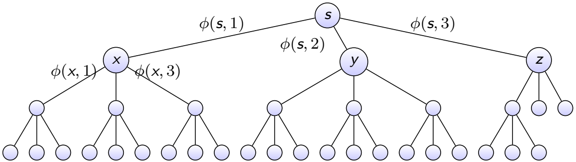
$$V^\star(s) = \langle \theta^\star,\ \psi(s)\rangle$$

- Lots of interesting variants: Linear Q^*, Linear V^*, Linear Q^*&V^* (reachable states).
- Weak Assumption. Implied by a lot of previous assumptions: Linear MDP, Linear Bellman Complete, ...
- Counterpart in supervised learning is well understood
  - What's efficiently possible in RL compared to supervised learning.

Part 1: Why is RL hard?

Connections to Bandits and Trees

Why is RL hard: From Bandits Theory



- Recall Linear Q^* means

$$Q^*(s,a) = \langle w^*, \phi(s,a) \rangle \text{ where } \phi(s,a) \in \mathbb{R}^d.$$

- Approach: Learn the linear function $Q^*(s,a)$ uniformally over all state action pairs!
  - Need something stronger than regression
    to learn w^* from estimates of the value function on different state action pairs $\{\hat{Q}^*(s,a)\}_{s,a}$

Why is RL hard: From Bandits Theory

- Proposition (John '48)
  There exists $O(d)$ state-action pairs $\{(s_1, a_1), (s_2, a_2), \dots, (s_d, a_d)\}$ such that every $\phi(s, a)$ can be written in terms of $\phi(s_1, a_1), \dots, \phi(s_d, a_d)$ with small coefficients.

$$\phi(s, a) = \sum_i \alpha_i \phi(s_i, a_i) \quad \text{and } \|\alpha\|_2 \leq \sqrt{d}$$

- This implies $Q^*(s, a)$ can be written in terms of $\{Q^*(s_i, a_i)\}$ with small coefficients

$$\phi(s, a) = \sum_i \alpha_i \phi(s_i, a_i) \implies Q^*(s, a) = \sum_i \alpha_i Q^*(s_i, a_i)$$

- Using good estimates $\hat{Q}(s_i, a_i)$ on these "landmark" state action pairs, we can build good estimates $\hat{Q}(s, a) = \sum_i \alpha_i \hat{Q}(s_i, a_i)$ for any state-action pair (s, a).

- How much does the error grow?

$$|Q^*(s, a) - \hat{Q}(s, a)| \leq \|\alpha\|_1 \max_i |Q^*(s_i, a_i) - \hat{Q}(s_i, a_i)| \leq O(d) \max_i |Q^*(s_i, a_i) - \hat{Q}(s_i, a_i)|$$

Attempt 1: Regression Chaining

We now have a different "landmark" set (John's basis) J_h for every level h.

Attempt 1: Regression Chaining

We now have a different "landmark" set (John's basis) J_h for every level h.

- Error for John's basis at last layer:

$$\max_{(s,a)\in J_H} \left| Q^*(s,a) - \hat{Q}(s,a) \right| \leq O(\epsilon)$$

Attempt 1: Regression Chaining

We now have a different "landmark" set (John's basis) J_h for every level h.

- Error for John's basis at last layer:

$$\max_{(s,a)\in J_H} \left| Q^*(s,a) - \hat{Q}(s,a) \right| \leq O(\epsilon)$$

- Error for every (s_H, a) at last level:

$$\left| Q^*(s_H,a) - \hat{Q}(s_H,a) \right| \leq \|\alpha\|_1 \epsilon \leq O(d\epsilon)$$

Attempt 1: Regression Chaining

We now have a different "landmark" set (John's basis) J_h for every level h.

- Error for John's basis at last layer:

$$\max_{(s,a)\in J_H} \left| Q^*(s,a) - \hat{Q}(s,a) \right| \leq O(\epsilon)$$

- Error for every (s_H, a) at last level:

$$\left| Q^*(s_H,a) - \hat{Q}(s_H,a) \right| \leq \|\alpha\|_1 \epsilon \leq O(d\epsilon)$$

- Error for John's basis at second last layer:

$$\max_{(s,a)\in J_{H-1}} \left| Q^*(s,a) - \hat{Q}(s,a) \right| \leq \mathbb{E}_{s_H \sim T(s,a)} \left[\left| V^*(s_H) - \hat{V}(s_H) \right| \right] \leq O(d\epsilon)$$

Attempt 1: Regression Chaining

We now have a different "landmark" set (John's basis) J_h for every level h.

- Error for John's basis at last layer:

$$\max_{(s,a)\in J_H} \left| Q^*(s,a) - \hat{Q}(s,a) \right| \le O(\epsilon)$$

- Error for every (s_H, a) at last level:

$$\left| Q^*(s_H,a) - \hat{Q}(s_H,a) \right| \le \|\alpha\|_1 \epsilon \le O(d\epsilon)$$

- Error for John's basis at second last layer:

$$\max_{(s,a)\in J_{H-1}} \left| Q^*(s,a) - \hat{Q}(s,a) \right| \le \mathbb{E}_{s_H \sim T(s,a)} \left[\left| V^*(s_H) - \hat{V}(s_H) \right| \right] \le O(d\epsilon)$$

- Error for every (s_{H-1}, a) at second last level:

$$|Q^*(s_{H-1},a) - \hat{Q}(s_{H-1},a)| \le \|\alpha\|_1 d\epsilon \le O(d^2\epsilon)$$

Attempt 1: Regression Chaining

We now have a different "landmark" set (John's basis) J_h for every level h.

- Error for John's basis at last layer:

$$\max_{(s,a)\in J_H} \left| Q^*(s,a) - \hat{Q}(s,a) \right| \le O(\epsilon)$$

- Error for every (s_H, a) at last level:

$$\left| Q^*(s_H,a) - \hat{Q}(s_H,a) \right| \le \|\alpha\|_1 \epsilon \le O(d\epsilon)$$

- Error for John's basis at second last layer:

$$\max_{(s,a)\in J_{H-1}} \left| Q^*(s,a) - \hat{Q}(s,a) \right| \le \mathbb{E}_{s_H \sim T(s,a)} \left[\left| V^*(s_H) - \hat{V}(s_H) \right| \right] \le O(d\epsilon)$$

- Error for every (s_{H-1}, a) at second last level:

$$|Q^*(s_{H-1},a) - \hat{Q}(s_{H-1},a)| \le \|\alpha\|_1 d\epsilon \le O(d^2\epsilon)$$

Issue: The error grows by a factor of d every level.
Leading to d^H sample and computational complexity. Can be improved to $d^{\sqrt{H}}$.

Part II: Sample Efficiency

Algorithmic Ideas

Goal: Improve upon the $O(d^{\sqrt{H}})$ sample complexity of regression chaining.

What more can we do?

Q: We do not observe Q^* and V^*. Then, what do we observe?

A: Local Consistency!

$$Q^*(s,a) = E_{r\sim R(s,a)}[r] + \mathbb{E}_{s'\sim T(s,a)}[V^*(s')] \tag{A}$$

$$V^*(s) = \max_a Q^*(s,a) \tag{B}$$

What more can we do?

Q: We do not observe Q^* and V^*. Then, what do we observe?

A: Local Consistency!

$$Q^*(s,a) = E_{r\sim R(s,a)}[r] + \mathbb{E}_{s'\sim T(s,a)}[V^*(s')] \quad \text{(A)}$$

$$V^*(s) = \max_a Q^*(s,a) \quad \text{(B)}$$

- Recall Linear Q^* and Linear V^* means

$$Q^*(s,a) = \langle w^*, \phi(s,a)\rangle \quad \text{and} \quad V^*(s) = \langle \theta^*, \psi(s)\rangle$$

What more can we do?

Q: We do not observe Q^* and V^*. Then, what do we observe?

A: Local Consistency!

$$Q^*(s,a) = E_{r\sim R(s,a)}[r] + \mathbb{E}_{s'\sim T(s,a)}[V^*(s')] \qquad \text{(A)}$$

$$V^*(s) = \max_a Q^*(s,a) \qquad \text{(B)}$$

- Recall Linear Q^* and Linear V^* means

$$Q^*(s,a) = \langle w^*, \phi(s,a)\rangle \quad \text{and} \quad V^*(s) = \langle \theta^*, \psi(s)\rangle$$

- **Equation (A) is a Linear Constraint**
  Equation (A) can be enforced by estimating "consistency feature" vector $[\phi(s,a), -\mathbb{E}_T(\psi(s')), -\mathbb{E}_R(r)]$.

$$\langle w^*, \phi(s,a)\rangle - \langle 1, E_{r\sim R(s,a)}[r]\rangle - \langle \theta^*, \mathbb{E}_{s'\sim T(s,a)}[\psi(s')]\rangle = 0$$

$$\langle [w^*, \theta^*, 1], [\phi(s,a), -\mathbb{E}_T(\psi(s')), -\mathbb{E}_R(r)]\rangle = 0$$

We should create John's basis for these "consistency feature" vectors.

What more can we do?

Q: We do not observe Q^* and V^*. Then, what do we observe?

A: Local Consistency!

$$Q^*(s,a) = E_{r\sim R(s,a)}[r] + \mathbb{E}_{s'\sim T(s,a)}[V^*(s')] \tag{A}$$

$$V^*(s) = \max_a Q^*(s,a) \tag{B}$$

- Recall Linear Q^* and Linear V^* means

$$Q^*(s,a) = \langle w^*, \phi(s,a)\rangle \quad \text{and} \quad V^*(s) = \langle \theta^*, \psi(s)\rangle$$

- **Equation (A) is a Linear Constraint**
  Equation (A) can be enforced by estimating "consistency feature" vector $[\phi(s,a), -\mathbb{E}_T(\psi(s')), -\mathbb{E}_R(r)]$.

$$\langle w^*, \phi(s,a)\rangle - \langle 1, E_{r\sim R(s,a)}[r]\rangle - \langle \theta^*, \mathbb{E}_{s'\sim T(s,a)}[\psi(s')]\rangle = 0$$

$$\langle [w^*, \theta^*, 1], [\phi(s,a), -\mathbb{E}_T(\psi(s')), -\mathbb{E}_R(r)]\rangle = 0$$

We should create John's basis for these "consistency feature" vectors.

- **Enforcing Equation (B) is free!**.
  Enforcing Equation (B) does *not* require any interaction with transition and rewards function.

$$V^*(s) = \langle \theta^*, \psi(s)\rangle = \max_a \langle w^*, \phi(s,a)\rangle = \max_a Q^*(s,a)$$

Polynomial Sample Complexity under Linear Function Approximation

Theorem (Du, Kakade, Lee, Lovett, M., Sun, Wang '21)

- Bilinear class: a special class of MDPs.
  set of MDPs where (a generalization of) Equation (A) is a low degree polynomial.
- All "named" models are bilinear classes.
- Sample efficient algorithm ($\mathbf{poly}(d, H)$) for all bilinear classes.

- Linear Q^*&V^* is a bilinear class.
  - Need only $\mathbf{poly}(d, H)$ samples when both Q^* and V^* are linear.
- For other variants, enforce Equation (A) by multiplying the constraint for all actions.[Weisz, Amortila, Janzer, Abbasi-Yadkori, Jiang, Szepesvári '21]
- Phase transition happens when only Q^* or V^* is linear.
  (series of works [Weisz, Amortila, Szepesvári '21; ...; Wang, Wang, Kakade '21])

	$\|A\| = 2$	$\|A\| = \Omega(d^{1/4})$	$\|A\| = \mathbf{exp}(d)$
linear Q^* and V^*	✓	✓	✓
linear Q^* and V^* (reachable)	✓	✗	✗
linear Q^*	✓	✗	✗
linear V^*	✓	✗	✗

Simple Global Algorithm: BiLin-UCB

- First we remove candidates which don't satisfy Equation (B) so only need to satisfy Equation (A).

Algorithm 1: BiLin-UCB

Parameters number of iterations T, batch size m, confidence radius R

Initialize constraint $\sigma : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ as $\sigma(w, \theta) = 0$

for *iteration* $t = 0, 1, \ldots, T-1$ **do**

  Find the optimistic (w_t, θ_t):

$$(w_t, \theta_t) := \operatorname*{\mathbf{arg\,max}}_{(w,\theta)} \langle \theta, \psi(s_0) \rangle \quad \text{subject to } \sigma^2(w, \theta) \le R$$

  Sample m trajectories using π_t and create a batch dataset of size mH:

$$D = \{(r_h, s_h, a_h, s_{h+1}) \in \textbf{trajectories}\}$$

  Update the constraint $\sigma^2(\cdot)$

$$\sigma^2(w, \theta) \leftarrow \sigma^2(w, \theta) + \mathbb{E}_D \Big[\langle w, \phi(s_h, a_h) \rangle - r_h - \langle \theta, \psi(s_{h+1}) \rangle \Big]^2$$

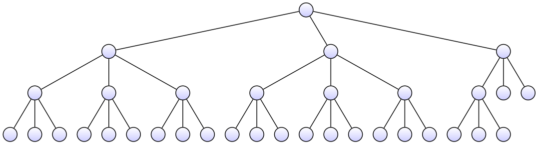
return: the best π_t found until now.

Part III: Computational Complexity of RL

Hard Instances

Goal: Improve upon the $O(d^{\sqrt{H}})$ computational complexity of regression chaining.

Global vs Local Algorithm



- A global algorithm exists! Computationally inefficient.
  - Regression Chaining (Du, Lee, M., Wang '19) Takes $\mathbf{exp}(H)$ time. Can be improved to $\mathbf{exp}(\sqrt{H})$.
  - BiLin-UCB (Du, Kakade, Lee, Lovett, M., Sun, Wang '21): Loops over all linear functions!
  - TensorPlan (Weisz, Szepesvari, Gyorgy '21) Takes $\mathbf{exp}(d)$ time.

- Open Question: Can we design polynomial time algorithms under linear function approximation?

Computational-Statistical Gap

- **Theorem** (Kane, Liu, Lovett, **M.**, 2022)
  Unless NP = RP, no polynomial time algorithm exists for RL with linear function approximation.

	$\lvert A\rvert = 2$	$\lvert A\rvert = \Omega(d^{1/4})$	$\lvert A\rvert = \mathbf{exp}(d)$
linear Q^* and V^*	✗	✗	✗
linear Q^* and V^* (reachable)	✗	✗	✗
linear Q^*	✗	✗	✗
linear V^*	✗	✗	✗

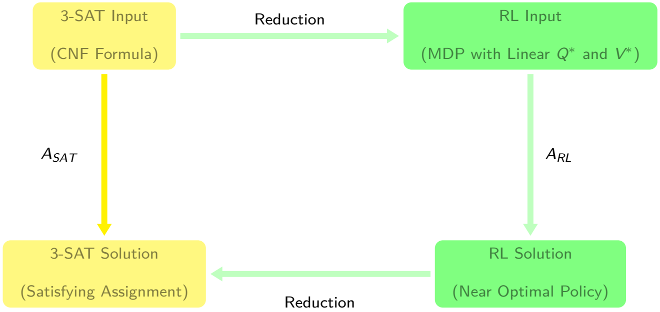
- Hardness for all the variants of linear function approximation
  even in the easiest case: deterministic transition + 2 actions.
- Unlike classical theory, computational and statistical worlds are really different.
- Reinforcement vs Supervised Learning.
  - Learning features which allow target functions to be linear are not enough.

Reduction: 3-SAT to MDP with Linear Q^* and V^*

Complexity problem 3-SAT

Input: a CNF formula φ with v variables, $O(v)$ clauses.
Goal: is φ satisfiable?

- CNF Formula: $(x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_3 \vee x_4)$
- Satisfying Assignment: $(1, 1, -1, 1)$



Hard Instance

- We need to embed a hard problem, 3SAT, in RL.
- Let's start with non-constant number of actions.

Hard Instance

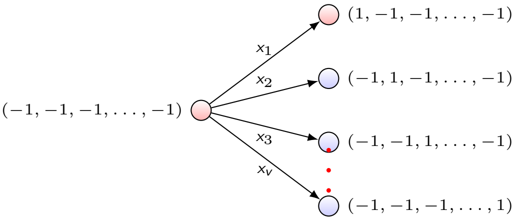
- We need to embed a hard problem, 3SAT, in RL.
- Let's start with non-constant number of actions.

$(-1, -1, -1, \ldots, -1)$

- Every state is some assignment to 3SAT variables.

Hard Instance

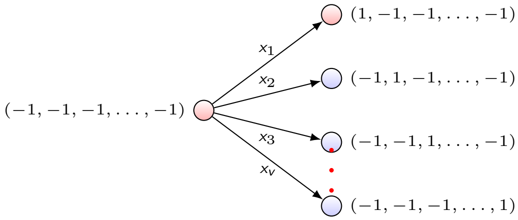
- We need to embed a hard problem, 3SAT, in RL.
- Let's start with non-constant number of actions.



- Every state is some assignment to 3SAT variables.
- v actions correspond to flipping the assignment for each of the v variables.

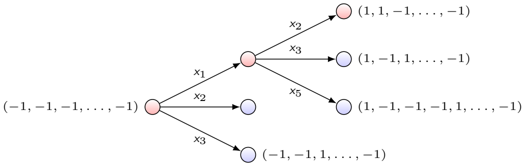
Hard Instance

- We need to embed a hard problem, 3SAT, in RL.
- Let's start with non-constant number of actions.



- Every state is some assignment to 3SAT variables.
- v actions correspond to flipping the assignment for each of the v variables.
- Reward $= 1$ only when you reach some fixed satisfying assignment w^*. Otherwise 0.

Hard Instance: Issues



- For horizon $H = v$, Solving RL $\implies$ solving 3SAT.
- Issue 1: We can not write Q^* or V^* linearly.
  Value of assignment w at level l (here $D(w, w^*)$ is hamming distance between w and w^*)

$$V^*(w, l) = \begin{cases} 1 & \text{if } D(w, w^*) < H - l \\ 0 & \text{otherwise} \end{cases}$$

- Issue 2: The rewards are deterministic.
  There exists polynomial time algorithms when rewards are deterministic (Gaussian Elimination).

Hard Instance: Issues

- Let's add randomness. Bernoulli reward.
  Expected reward on reaching w at level l

$$\mathbb{E}[R(w,l)] = \begin{cases} 1 - \frac{l+D(w,w^*)}{H+v} & \text{if } l = H \text{ or } w = w^* \\ 0 & \text{otherwise} \end{cases}$$

- We can show in this case the optimal policy is to go towards w^* as fast as possible.
  Therefore, value of assignment w at level l

$$V^*(w,l) = 1 - \frac{l + D(w,w^*)}{H+v}$$

  Since hamming distance is linear in w and w^*, V^* is also linear in w and w^*

Hard Instance: Issues

- Let's add randomness. Bernoulli reward.
  Expected reward on reaching w at level l

$$\mathbb{E}[R(w,l)] = \begin{cases} 1 - \frac{l+D(w,w^*)}{H+v} & \text{if } l = H \text{ or } w = w^* \\ 0 & \text{otherwise} \end{cases}$$

- We can show in this case the optimal policy is to go towards w^* as fast as possible.
  Therefore, value of assignment w at level l

$$V^*(w,l) = 1 - \frac{l + D(w,w^*)}{H + v}$$

  Since hamming distance is linear in w and w^*, V^* is also linear in w and w^*

- New issue: We leak a lot of reward information at the last layer.
  Can not simulate $D(w,w^*)$ efficiently.
  This can be fixed but a bit technical.

Hard Instance: Issues

- Expected reward on reaching w at level l

$$\mathbb{E}[R(w,l)] = \begin{cases} \left(1 - \frac{l+D(w,w^*)}{H+v}\right)^r & \text{if } l = H \text{ or } w = w^* \\ 0 & \text{otherwise} \end{cases}$$

- Same as before the optimal policy is to go towards w^* as fast as possible.
  Therefore, value of assignment w at level l

$$V^*(w,l) = \left(1 - \frac{l + D(w,w^*)}{H+v}\right)^r$$

  V^* is a polynomial of degree r in w and w^*.
  In terms of its monomials, V^* is linear in $d = v^r$ dimensional features.

Hard Instance: Issues

- Consider a polynomial time algorithm for RL.
  Since, our dimension d and horizon H are both $d = H = v^r$,
  the algorithm runs in **poly**(v^r) time.
- But the expected reward at the last layer is at most
$$\left(1 - \frac{H}{H+v}\right)^r \in O(v^{-r^2})$$
- Therefore, any polytime algorithm with high probability only sees 0 at the last layer.
- We can simulate this algorithm efficiently by always returning 0 on the last layer
  (and only slightly decreasing the success probability)!

Hard Instance: v actions to 3 actions

- There always exists a variable we can flip to get closer to the optimal solution.
- The three actions available are the variables in the unsatisfied clause
  (one such clause exists, because current assignment is not satisfying assignment.)

Consider the following CNF formula

$(x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee x_3 \vee x_4) \wedge (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_3 \vee \bar{x}_3 \vee \bar{x}_3) \wedge (x_1 \vee x_1 \vee x_1)$.

Hard Instance: v actions to 3 actions

- There always exists a variable we can flip to get closer to the optimal solution.
- The three actions available are the variables in the unsatisfied clause
  (one such clause exists, because current assignment is not satisfying assignment.)

Consider the following CNF formula

$(x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee x_3 \vee x_4) \wedge (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_3 \vee \bar{x}_3 \vee \bar{x}_3) \wedge (x_1 \vee x_1 \vee x_1)$.

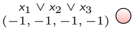
$(-1, -1, -1, -1)$

Hard Instance: v actions to 3 actions

- There always exists a variable we can flip to get closer to the optimal solution.
- The three actions available are the variables in the unsatisfied clause
  (one such clause exists, because current assignment is not satisfying assignment.)

Consider the following CNF formula

$(x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee x_3 \vee x_4) \wedge (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_3 \vee \bar{x}_3 \vee \bar{x}_3) \wedge (x_1 \vee x_1 \vee x_1)$.

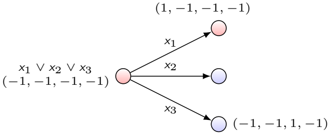


Hard Instance: v actions to 3 actions

- There always exists a variable we can flip to get closer to the optimal solution.
- The three actions available are the variables in the unsatisfied clause
  (one such clause exists, because current assignment is not satisfying assignment.)

Consider the following CNF formula

$(x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee x_3 \vee x_4) \wedge (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_3 \vee \bar{x}_3 \vee \bar{x}_3) \wedge (x_1 \vee x_1 \vee x_1)$.

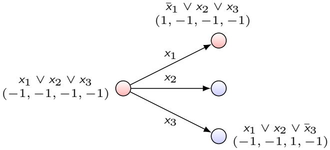


Hard Instance: v actions to 3 actions

- There always exists a variable we can flip to get closer to the optimal solution.
- The three actions available are the variables in the unsatisfied clause
  (one such clause exists, because current assignment is not satisfying assignment.)

Consider the following CNF formula

$(x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee x_3 \vee x_4) \wedge (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_3 \vee \bar{x}_3 \vee \bar{x}_3) \wedge (x_1 \vee x_1 \vee x_1)$.

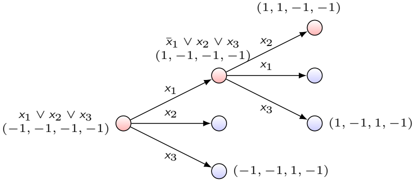


Hard Instance: v actions to 3 actions

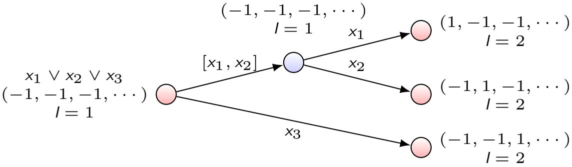
- There always exists a variable we can flip to get closer to the optimal solution.
- The three actions available are the variables in the unsatisfied clause
  (one such clause exists, because current assignment is not satisfying assignment.)

Consider the following CNF formula

$(x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee x_3 \vee x_4) \wedge (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_3 \vee \bar{x}_3 \vee \bar{x}_3) \wedge (x_1 \vee x_1 \vee x_1)$.



Hard Instance: 3 actions to 2 actions



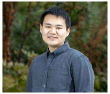
- We replace the three actions by 2 actions, grouping any two actions together.

What have we learned?

- Computational-Statistical Gap in RL with Linear Function Approximation.
  - Simple sample efficient algorithm works for all known "named" models.
  - Novel construction exposing computational hardness.
- Reinforcement vs Supervised Learning.
  - Learning features which allow target functions to be linear are not enough.
  - We need more assumptions for RL.
- Tight characterization of computational complexity of RL.
  - Best Upper Bounds: Takes $\mathbf{exp}(\sqrt{H}\,\mathbf{log}\, d)$ or $\mathbf{exp}(d)$ time.
  - Best Lower Bound: No polynomial time algorithm exists.
  - Can we close this gap? (maybe there exist a quasi-polynomial time algorithm)

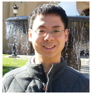
Thanks!

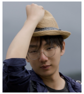
Joint work with:


Simon Du

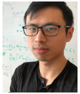

Sham Kakade


Daniel Kane


Jason Lee


Sihan Liu

Shachar Lovett


Wen Sun


Ruosong Wang

Proof intuition

- The proof follows from this lemma about existence of high quality policy.

Lemma (Existence of high quality policy)

Suppose we run the algorithm for $T \approx d$ iterations. Then, there exists $t \in [T]$ such that the following is true for hypothesis (w_t, θ_t):

$$V^\star - V^{\pi_t}(s_0) \leq \frac{\mathbf{poly}(d, H)}{\sqrt{m}}$$

Bilinear Regret Lemma

- Bilinear regret assumption and Optimism give an upper bound for sub-optimality.

Bilinear Regret Lemma

- Bilinear regret assumption and Optimism give an upper bound for sub-optimality.
- Define W_t as the collective parameters
$$W_t = [w_t, -\theta_t]$$

Bilinear Regret Lemma

- Bilinear regret assumption and Optimism give an upper bound for sub-optimality.
- Define W_t as the collective parameters

$$W_t = [w_t, -\theta_t]$$

- Define $X_{t,h}$ as the expected "consistency feature" vector seen at level h under policy π_t.

$$X_{t,h} = \mathbb{E}_{\pi_t}\left[\phi(s_h, a_h), \psi(s_{h+1})\right]$$

Bilinear Regret Lemma

- Bilinear regret assumption and Optimism give an upper bound for sub-optimality.
- Define W_t as the collective parameters

$$W_t = [w_t, -\theta_t]$$

- Define $X_{t,h}$ as the expected "consistency feature" vector seen at level h under policy π_t.

$$X_{t,h} = \mathbb{E}_{\pi_t}\left[\phi(s_h, a_h), \psi(s_{h+1})\right]$$

Lemma (Bilinear Regret Lemma)

The following holds for all $t \in [T]$ w.h.p.:

$$V^\star(s_0) - V^{\pi_t}(s_0) \leq \sum_{h=0}^{H-1} \left| \langle W_t - W_t^*, X_{t,h} \rangle \right| .$$

Proof of Bilinear Regret Lemma

Proof:

$$V^{\star}(s_0) - V^{\pi_t}(s_0)$$

Proof of Bilinear Regret Lemma

Proof:

$$
\begin{aligned}
& V^\star(s_0) - V^{\pi_t}(s_0) \\
& \leq \langle \theta_t, \psi(s_0) \rangle - V^{\pi_t}(s_0) && \text{(optimism)}
\end{aligned}
$$

Proof of Bilinear Regret Lemma

Proof:

$$
\begin{aligned}
& V^{\star}(s_0) - V^{\pi_t}(s_0) \\
& \leq \langle \theta_t, \psi(s_0) \rangle - V^{\pi_t}(s_0) && \text{(optimism)} \\
& = \langle w_t, \phi(s_0, a_0) \rangle - V^{\pi_t}(s_0) && \text{(Equation (B))}
\end{aligned}
$$

Proof of Bilinear Regret Lemma

Proof:

$$
\begin{aligned}
& V^\star(s_0) - V^{\pi_t}(s_0) \\
& \leq \langle \theta_t, \psi(s_0) \rangle - V^{\pi_t}(s_0) && \text{(optimism)} \\
& = \langle w_t, \phi(s_0, a_0) \rangle - V^{\pi_t}(s_0) && \text{(Equation (B))} \\
& = \langle w_t, \phi(s_0, a_0) \rangle - \mathbb{E}\left[\sum_{h=0}^{H} r_h \right] && \text{(definition of } V^{\pi_t}\text{)}
\end{aligned}
$$

Proof of Bilinear Regret Lemma

Proof:

$$
\begin{aligned}
& V^\star(s_0) - V^{\pi_t}(s_0) \\
& \leq \langle \theta_t, \psi(s_0) \rangle - V^{\pi_t}(s_0) && \text{(optimism)} \\
& = \langle w_t, \phi(s_0, a_0) \rangle - V^{\pi_t}(s_0) && \text{(Equation (B))} \\
& = \langle w_t, \phi(s_0, a_0) \rangle - \mathbb{E}\left[\sum_{h=0}^{H} r_h\right] && \text{(definition of } V^{\pi_t}\text{)} \\
& = \sum_{h=0}^{H-1} \mathbb{E}_{\pi_t}\left[\langle w_t, \phi(s_h, a_h)\rangle - r_h - \langle w_t, \phi(s_{h+1}, a_{h+1})\rangle\right] && \text{(telescoping sum)}
\end{aligned}
$$

Proof of Bilinear Regret Lemma

Proof:

$$
\begin{aligned}
& V^\star(s_0) - V^{\pi_t}(s_0) \\
& \leq \langle \theta_t, \psi(s_0) \rangle - V^{\pi_t}(s_0) && \text{(optimism)} \\
& = \langle w_t, \phi(s_0, a_0) \rangle - V^{\pi_t}(s_0) && \text{(Equation (B))} \\
& = \langle w_t, \phi(s_0, a_0) \rangle - \mathbb{E}\left[\sum_{h=0}^{H} r_h\right] && \text{(definition of } V^{\pi_t}\text{)} \\
& = \sum_{h=0}^{H-1} \mathbb{E}_{\pi_t}\left[\langle w_t, \phi(s_h, a_h) \rangle - r_h - \langle w_t, \phi(s_{h+1}, a_{h+1}) \rangle\right] && \text{(telescoping sum)} \\
& = \sum_{h=0}^{H-1} \mathbb{E}_{\pi_t}\left[\langle w_t, \phi(s_h, a_h) \rangle - r_h - \langle \theta_t, \psi(s_{h+1}) \rangle\right] && \text{(Equation (B))}
\end{aligned}
$$

Proof of Bilinear Regret Lemma

Proof:

$$
\begin{aligned}
& V^\star(s_0) - V^{\pi_t}(s_0) \\
& \le \langle \theta_t, \psi(s_0) \rangle - V^{\pi_t}(s_0) && \text{(optimism)} \\
& = \langle w_t, \phi(s_0, a_0) \rangle - V^{\pi_t}(s_0) && \text{(Equation (B))} \\
& = \langle w_t, \phi(s_0, a_0) \rangle - \mathbb{E}\left[\sum_{h=0}^{H} r_h\right] && \text{(definition of } V^{\pi_t}\text{)} \\
& = \sum_{h=0}^{H-1} \mathbb{E}_{\pi_t}\left[\langle w_t, \phi(s_h, a_h) \rangle - r_h - \langle w_t, \phi(s_{h+1}, a_{h+1}) \rangle\right] && \text{(telescoping sum)} \\
& = \sum_{h=0}^{H-1} \mathbb{E}_{\pi_t}\left[\langle w_t, \phi(s_h, a_h) \rangle - r_h - \langle \theta_t, \psi(s_{h+1}) \rangle\right] && \text{(Equation (B))} \\
& = \sum_{h=0}^{H-1} \left|\langle W_t - W^*, X_{t,h} \rangle\right| && \text{(by definition)}
\end{aligned}
$$

Proof of main lemma

- Bilinear regret assumption and Optimism give an upper bound on sub-optimality for all iterations t.

$$V^{\star} - V^{\pi_t}(s_0) \leq \sum_{h=0}^{H-1} \left| \langle W_t - W^{*}, X_{t,h} \rangle \right| .$$

Proof of main lemma

- Bilinear regret assumption and Optimism give an upper bound on sub-optimality for all iterations t.

$$V^\star - V^{\pi_t}(s_0) \leq \sum_{h=0}^{H-1} \left| \langle W_t - W^*, X_{t,h} \rangle \right| \; .$$

- Our goal then is to show existence of iteration $t \in [T]$ such that

$$\sum_{h=0}^{H-1} \left| \langle W_t - W^*, X_{t,h} \rangle \right| \quad \text{is small}$$

Proof of main lemma

- Bilinear regret assumption and Optimism give an upper bound on sub-optimality for all iterations t.

$$V^\star - V^{\pi_t}(s_0) \le \sum_{h=0}^{H-1} \left| \langle W_t - W^*, X_{t,h} \rangle \right| \, .$$

- Our goal then is to show existence of iteration $t \in [T]$ such that

$$\sum_{h=0}^{H-1} \left| \langle W_t - W^*, X_{t,h} \rangle \right| \quad \text{is small}$$

- To that end, we will show existence of iteration $t \in [T]$ such that for $\Sigma_{0;h} = \lambda I$ and $\Sigma_{t;h} = \Sigma_{0;h} + \sum_{i=0}^{t-1} X_{i,h} X_{i,h}^\top$, the following is true

$$\| W_t - W^* \|_{\Sigma_{t;h}} \quad \left\| X_{t,h} \right\|_{\Sigma_{t;h}^{-1}} \quad \text{is small for all } h \in [H]$$

Proof of main lemma

- To that end, we will show existence of iteration $t \in [T]$ such that for $\Sigma_{0;h} = \lambda I$ and $\Sigma_{t;h} = \Sigma_{0;h} + \sum_{i=0}^{t-1} X_{i,h} X_{i,h}^{\top}$, the following is true

$$\|W_t - W^*\|_{\Sigma_{t;h}} \quad \|X_{t,h}\|_{\Sigma_{t;h}^{-1}} \quad \text{is small for all } h \in [H]$$

Proof of main lemma

- To that end, we will show existence of iteration $t \in [T]$ such that for $\Sigma_{0;h} = \lambda I$ and $\Sigma_{t;h} = \Sigma_{0;h} + \sum_{i=0}^{t-1} X_{i,h} X_{i,h}^\top$, the following is true

$$\|W_t - W^*\|_{\Sigma_{t;h}} \quad \|X_{t,h}\|_{\Sigma_{t;h}^{-1}} \quad \text{is small for all } h \in [H]$$

- From our optimization constraint and uniform convergence, we get that for all time t

$$\|W_t - W^*\|_{\Sigma_{t;h}} \leq \underbrace{\frac{\mathbf{poly}(d, H)}{\sqrt{m}}}_{\approx d \times \text{SL generalization error}} \quad \text{for all } h \in [H]$$

Proof of main lemma

- To that end, we will show existence of iteration $t \in [T]$ such that for $\Sigma_{0;h} = \lambda I$ and $\Sigma_{t;h} = \Sigma_{0;h} + \sum_{i=0}^{t-1} X_{i,h} X_{i,h}^\top$, the following is true

$$\|W_t - W^*\|_{\Sigma_{t;h}} \quad \|X_{t,h}\|_{\Sigma_{t;h}^{-1}} \quad \text{is small for all } h \in [H]$$

- From our optimization constraint and uniform convergence, we get that for all time t

$$\|W_t - W^*\|_{\Sigma_{t;h}} \leq \underbrace{\frac{\mathbf{poly}(d, H)}{\sqrt{m}}}_{\approx d \times \text{SL generalization error}} \qquad \text{for all } h \in [H]$$

- From Elliptical Potential Lemma, there exists $t \in [T]$ such that

$$\|X_{t,h}\|_{\Sigma_{t;h}^{-1}} = O(1) \quad \text{for all } h \in [H]$$

Basically, we can write $X_{t,h}$ in terms of $\{X_{t-1,h}, \ldots, X_{1,h}, X_{0,h}\}$ with small coefficients.