Algorithmic Aspects of Reinforcement Learning II

Ankur Moitra (MIT)

February 24th, ITA+ALT Tutorial

OUTLINE

Part V: Linear MDPs

- Least Squares and the Elliptic Potential
- Feature Selection and Sparsity

Part VII: Sparse Linear MDPs

- Sparse Linear Regression
- Greedy Covers
- Emulators and their Algorithmic Applications

Part VI: Block MDPs

- The View from Supervised Learning

OUTLINE

Part V: Linear MDPs

- Least Squares and the Elliptic Potential
- Feature Selection and Sparsity

Part VII: Sparse Linear MDPs

- Sparse Linear Regression
- Greedy Covers
- Emulators and their Algorithmic Applications

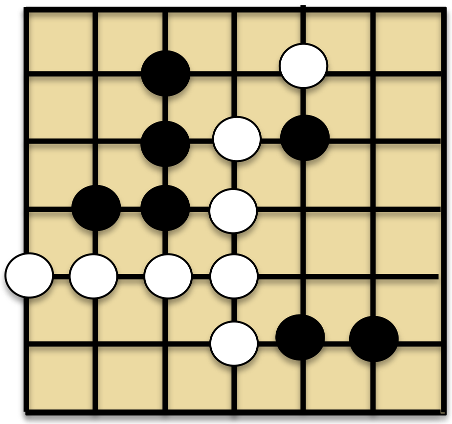
Part VI: Block MDPs

- The View from Supervised Learning

FUNCTION APPROXIMATION

Suppose there are too many states to write down/visit

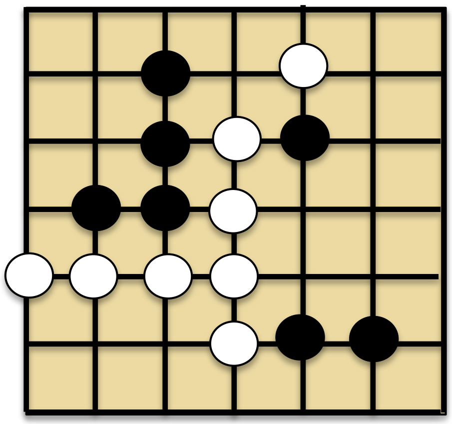
e.g.



FUNCTION APPROXIMATION

Suppose there are too many states to write down/visit

e.g.



Can we distill the relevant properties of the state into a high-dimensional feature vector?

LINEAR MARKOV DECISION PROCESSES

Assumption: There is a known **feature mapping**

$$\phi_h : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}^d$$

and rewards/transitions are linear in this representation, i.e.

LINEAR MARKOV DECISION PROCESSES

Assumption: There is a known **feature mapping**

$$\phi_h : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}^d$$

and rewards/transitions are linear in this representation, i.e.

(1) $R_h(s,a) = \langle \phi_h(s,a), \theta_h \rangle$ and

(2) $\mathbb{T}_h(s'|s,a) = \langle \phi_h(s,a), \mu_{h+1}(s') \rangle$

LINEAR MARKOV DECISION PROCESSES

Assumption: There is a known **feature mapping**

$$\phi_h : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$$

and rewards/transitions are linear in this representation, i.e.

(1) $R_h(s,a) = \langle \phi_h(s,a), \theta_h \rangle$ and

(2) $\mathbb{T}_h(s'|s,a) = \langle \phi_h(s,a), \mu_{h+1}(s') \rangle$

Note: There can be an infinite number of states and parameters defining the model

LINEAR MARKOV DECISION PROCESSES

Assumption: There is a known **feature mapping**

$$\phi_h : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}^d$$

and rewards/transitions are linear in this representation, i.e.

(1) $R_h(s,a) = \langle \phi_h(s,a), \theta_h \rangle$ and

(2) $\mathbb{T}_h(s'|s,a) = \langle \phi_h(s,a), \mu_{h+1}(s') \rangle$

Note: There can be an infinite number of states and parameters defining the model

So how do we even represent a policy? Or its value function?

VALUE FUNCTIONS, AGAIN

Key: The closure properties, e.g. of dynamic programming

VALUE FUNCTIONS, AGAIN

Proposition: An MDP is linear with respect to the feature mapping iff for any function $g : \mathcal{S} \to \mathbb{R}$ we have that

$$R_h(s,a) + \mathbb{E}_{s'|s,a}[g(s')] = \langle \phi_h(s,a), \theta \rangle$$

for some vector θ

VALUE FUNCTIONS, AGAIN

Proposition: An MDP is linear with respect to the feature mapping iff for any function $g : \mathcal{S} \to \mathbb{R}$ we have that

$$R_h(s,a) + \mathbb{E}_{s'|s,a}[g(s')] = \langle \phi_h(s,a), \theta \rangle$$

for some vector θ

Now consider the **Q-function** and **V-function** for any policy:

$$Q_h^\pi(s,a) = R_h(s,a) + \mathbb{E}_{s'|s,a}[V_{h+1}^\pi(s')]$$

$$V_h^\pi(s) = \mathbb{E}_{a \sim \pi}[Q_h^\pi(s,a)]$$

VALUE FUNCTIONS, AGAIN

Proposition: An MDP is linear with respect to the feature mapping iff for any function $g : \mathcal{S} \to \mathbb{R}$ we have that

$$R_h(s,a) + \mathbb{E}_{s'|s,a}[g(s')] = \langle \phi_h(s,a), \theta \rangle$$

for some vector θ

Now consider the **Q-function** and **V-function** for any policy:

$$Q_h^\pi(s,a) = R_h(s,a) + \mathbb{E}_{s'|s,a}[V_{h+1}^\pi(s')]$$

$$V_h^\pi(s) = \mathbb{E}_{a\sim\pi}[Q_h^\pi(s,a)]$$

Corollary: The Q-function of **any** policy is linear

FINDING AN OPTIMAL POLICY

Consider policies of the form

$$\pi(s) = \arg\max_a \langle \phi_h(s, a), \theta \rangle$$

FINDING AN OPTIMAL POLICY

Consider policies of the form

$$\pi(s) = \arg\max_a \langle \phi_h(s,a), \theta \rangle$$

Now use linear regression to fit the Q-function to the observed rewards

$$\widehat{\theta} = \arg\min_\theta \sum_i (\phi(s_i, a_i)^\top \theta - \underbrace{y_i}_{}\,)^2 + \|\theta\|^2$$

immediate reward + estimated opt. future reward + expl. bonus

FINDING AN OPTIMAL POLICY

Consider policies of the form

$$\pi(s) = \arg\max_a \langle \phi_h(s,a), \theta \rangle$$

Now use linear regression to fit the Q-function to the observed rewards

$$\widehat{\theta} = \arg\min_\theta \sum_i (\phi(s_i, a_i)^\top \theta - y_i)^2 + \|\theta\|^2$$

immediate reward + estimated opt. future reward + expl. bonus

Need a bonus to maintain optimism

CHOOSING A BONUS

Natural bonus function depends on form of error bounds for linear regression; called the **elliptic potential**

$$\sqrt{\phi(s,a)^\top \Lambda^{-1} \phi(s,a)}$$

which accounts for errors in **unexplored directions**, where

$$\Lambda = I + \sum_i \phi(s_i, a_i)\phi(s_i, a_i)^\top$$

THE LSVI ALGORITHM

Theorem [Jin, Yang, Wang, Jordan '19]: LSVI has

running time: $\mathrm{poly}(d, |\mathcal{A}|, H)$

sample complexity: $\mathrm{poly}(d, H)$

No dependence on the # of states

and returns a near optimal policy in a linear MDP

THE LSVI ALGORITHM

Theorem [Jin, Yang, Wang, Jordan '19]: LSVI has

$$\left.\begin{array}{l}\text{running time: } \mathrm{poly}(d, |\mathcal{A}|, H) \\ \text{sample complexity: } \mathrm{poly}(d, H)\end{array}\right\}$$

No dependence on the # of states

and returns a near optimal policy in a linear MDP

Notes: Does not attempt to estimate model parameters

model-free vs. model-based

THE LSVI ALGORITHM

Theorem [Jin, Yang, Wang, Jordan '19]: LSVI has

running time: $\mathrm{poly}(d, |\mathcal{A}|, H)$

sample complexity: $\mathrm{poly}(d, H)$

No dependence on the # of states

and returns a near optimal policy in a linear MDP

Notes: Does not attempt to estimate model parameters

model-free vs. model-based

Later improvements get **optimal sample complexity**

OUTLINE

Part V: Linear MDPs

- Least Squares and the Elliptic Potential
- Feature Selection and Sparsity

Part VII: Sparse Linear MDPs

- Sparse Linear Regression
- Greedy Covers
- Emulators and their Algorithmic Applications

Part VI: Block MDPs

- The View from Supervised Learning

OUTLINE

Part V: Linear MDPs

- Least Squares and the Elliptic Potential
- **Feature Selection and Sparsity**

Part VII: Sparse Linear MDPs

- Sparse Linear Regression
- Greedy Covers
- Emulators and their Algorithmic Applications

Part VI: Block MDPs

- The View from Supervised Learning

FEATURE SELECTION?

How do we find a good feature mapping in the first place?

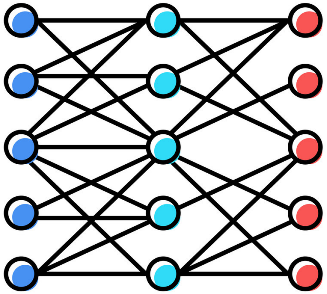
FEATURE SELECTION?

How do we find a good feature mapping in the first place?

Natural approach is to throw in the kitchen sink

+

+



domain expertise **heuristics** **learned features**

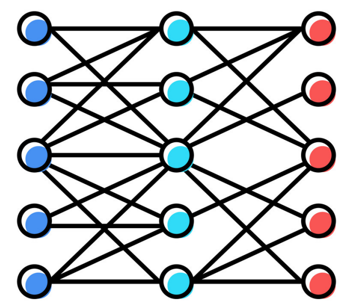
FEATURE SELECTION?

How do we find a good feature mapping in the first place?

Natural approach is to throw in the kitchen sink

+

+



domain expertise **heuristics** **learned features**

But we would pay a steep price in terms of sample complexity for using more features than we truly need

FEATURE SELECTION?

Can we automatically discover the relevant dimensions?

Sparsity Assumption: There is an unknown $S \subseteq [d]$ of size k and the rewards/transitions are linear functions of

$$\phi(s,a)\Big|_S$$

which is a k-dimensional representation

FEATURE SELECTION?

Can we automatically discover the relevant dimensions?

Sparsity Assumption: There is an unknown $S \subseteq [d]$ of size k and the rewards/transitions are linear functions of

$$\phi(s,a)\Big|_S$$

which is a k-dimensional representation

Are there efficient algorithms for learning sparse linear MDPs?

FEATURE SELECTION?

Can we automatically discover the relevant dimensions?

Sparsity Assumption: There is an unknown $S \subseteq [d]$ of size k and the rewards/transitions are linear functions of

$$\phi(s,a)\Big|_S$$

which is a k-dimensional representation

Are there efficient algorithms for learning sparse linear MDPs?

i.e. $\mathrm{poly}(k, \log d)$ sample complexity

Prior work relies on assumptions that obviate the need for exploration

(1) **[Hao, Lattimore, Szepesvari, Wang '21]:** assumes we're given an exploratory policy up front

Prior work relies on assumptions that obviate the need for exploration

(1) **[Hao, Lattimore, Szepesvari, Wang ‘21]:** assumes we’re given an exploratory policy up front

(2) **[Zhu, Wang, Lee ‘23]:** assumes every policy induces a well-conditioned feature distribution

Prior work relies on assumptions that obviate the need for exploration

(1) **[Hao, Lattimore, Szepesvari, Wang '21]:** assumes we're given an exploratory policy up front

(2) **[Zhu, Wang, Lee '23]:** assumes every policy induces a well-conditioned feature distribution

But ignoring computational efficiency, we know such assumptions are unnecessary

(3) **[Jiang, Krishnamurthy, Agarwal, Langford, Schapire '16]:** statistically efficient algorithm under low Bellman rank, **but oracle is known to be NP-hard to implement**

COMPUTATIONALLY EFFICIENT LEARNING

First end-to-end algorithmic guarantees in general

Theorem [Golowich, Moitra, Rohatgi '24]: An algorithm with

running time: $\mathrm{poly}(d, |\mathcal{A}|, H)$

sample complexity: $\mathrm{poly}(k, |\mathcal{A}|, H, \log d)$

that returns a near optimal policy in a sparse linear MDP

COMPUTATIONALLY EFFICIENT LEARNING

First end-to-end algorithmic guarantees in general

Theorem [Golowich, Moitra, Rohatgi '24]: An algorithm with

running time: $\mathrm{poly}(d, |\mathcal{A}|, H)$

sample complexity: $\mathrm{poly}(k, |\mathcal{A}|, H, \log d)$

that returns a near optimal policy in a sparse linear MDP

Based on a new abstraction we call an **emulator**

"Even though a linear MDP is non-parametric, can we find a parametric approximation to it?"

OUTLINE

Part V: Linear MDPs

- Least Squares and the Elliptic Potential
- Feature Selection and Sparsity

Part VII: Sparse Linear MDPs

- Sparse Linear Regression
- Greedy Covers
- Emulators and their Algorithmic Applications

Part VI: Block MDPs

- The View from Supervised Learning

OUTLINE

Part V: Linear MDPs

- Least Squares and the Elliptic Potential
- Feature Selection and Sparsity

Part VII: Sparse Linear MDPs

- **Sparse Linear Regression**
- Greedy Covers
- Emulators and their Algorithmic Applications

Part VI: Block MDPs

- The View from Supervised Learning

SPARSE LINEAR REGRESSION

An aspirational example where sparsity helps:

Setup: We are given samples of the form

$$y = \langle x, \theta^* \rangle + \xi \text{ with } x \sim \nu, \mathbb{E}[\xi] = 0 \text{ and } |\xi| \leq \sigma$$

where $\|\theta^*\|_1 \leq k$

SPARSE LINEAR REGRESSION

An aspirational example where sparsity helps:

Setup: We are given samples of the form

$$y = \langle x, \theta^* \rangle + \xi \text{ with } x \sim \nu, \mathbb{E}[\xi] = 0 \text{ and } |\xi| \leq \sigma$$

where $\|\theta^*\|_1 \leq k$

Definition [Tibshirani]: The **Lasso estimator** is

$$\widehat{\theta} = \arg\min_{\|\theta\|_1 \leq k} \sum_{i=1}^{n} (\langle x_i, \theta \rangle - y_i)^2$$

SPARSE LINEAR REGRESSION

An aspirational example where sparsity helps:

Theorem: With probability $1 - \delta$, the Lasso satisfies

$$\underbrace{\mathbb{E}_{x \sim \nu}[\langle x, \theta^* - \widehat{\theta} \rangle^2]}_{\textbf{risk}} \leq \frac{C(k+\sigma)k\sqrt{\log d/\delta}}{\sqrt{n}}$$

SPARSE LINEAR REGRESSION

An aspirational example where sparsity helps:

Theorem: With probability $1-\delta$, the Lasso satisfies

$$\underbrace{\mathbb{E}_{x\sim\nu}[\langle x, \theta^* - \widehat{\theta}\rangle^2]}_{\textbf{risk}} \leq \frac{C(k+\sigma)k\sqrt{\log d/\delta}}{\sqrt{n}}$$

This is the kind of improvement we are hoping for in SLMDPs

SPARSE LINEAR REGRESSION

An aspirational example where sparsity helps:

Theorem: With probability $1-\delta$, the Lasso satisfies

$$\underbrace{\mathbb{E}_{x\sim\nu}[\langle x, \theta^* - \widehat{\theta}\rangle^2]}_{\text{risk}} \leq \frac{C(k+\sigma)k\sqrt{\log d/\delta}}{\sqrt{n}}$$

This is the kind of improvement we are hoping for in SLMDPs

But in RL, there is no one fixed distribution --- it depends on the policy we play

OUTLINE

Part V: Linear MDPs

- Least Squares and the Elliptic Potential
- Feature Selection and Sparsity

Part VII: Sparse Linear MDPs

- Sparse Linear Regression
- Greedy Covers
- Emulators and their Algorithmic Applications

Part VI: Block MDPs

- The View from Supervised Learning

OUTLINE

Part V: Linear MDPs

- Least Squares and the Elliptic Potential
- Feature Selection and Sparsity

Part VII: Sparse Linear MDPs

- Sparse Linear Regression
- **Greedy Covers**
- Emulators and their Algorithmic Applications

Part VI: Block MDPs

- The View from Supervised Learning

POLICY COVERS

How does sparsity help in reinforcement learning?

POLICY COVERS

How does sparsity help in reinforcement learning?

Definition: A collection of policies Ψ is an α-approximate policy cover at step h if for all states x

$$\frac{1}{|\Psi|} \sum_{\pi' \in \Psi} \mathbb{P}_{\pi'}[x_h = x] \ \geq \alpha \max_{\pi} \mathbb{P}_{\pi}[x_h = x]$$

POLICY COVERS

How does sparsity help in reinforcement learning?

Definition: A collection of policies Ψ is an α-approximate policy cover at step h if for all states x

$$\frac{1}{|\Psi|}\sum_{\pi'\in\Psi}\mathbb{P}_{\pi'}[x_h = x] \;\geq \alpha \max_{\pi}\mathbb{P}_{\pi}[x_h = x]$$

Can we show size of our policy cover improves with sparsity?

REACHABILITY

Assumption (normalization): For all s, a and h we have

$$\|\phi_h(s,a)\|_\infty \leq 1 \text{ and } \sum_{s \in \mathcal{S}} \|\mu_h(s)\|_1 \leq C$$

REACHABILITY

Assumption (normalization): For all s, a and h we have

$$\|\phi_h(s,a)\|_\infty \leq 1 \text{ and } \underbrace{\sum_{s \in \mathcal{S}} \|\mu_h(s)\|_1 \leq C}_{\text{analytic notion of sparsity}}$$

REACHABILITY

Assumption (normalization): For all s, a and h we have

$$\|\phi_h(s,a)\|_\infty \leq 1 \text{ and } \sum_{s \in \mathcal{S}} \|\mu_h(s)\|_1 \leq C$$

For now, we will assume all states are reachable:

Definition: A linear MDP is η-**reachable** if for all s and h we have

$$\max_\pi \mathbb{P}_\pi[x_h = x] \geq \eta \|\mu_h(s)\|_1$$

REACHABILITY

Assumption (normalization): For all s, a and h we have

$$\|\phi_h(s,a)\|_\infty \leq 1 \text{ and } \sum_{s\in\mathcal{S}} \|\mu_h(s)\|_1 \leq C$$

For now, we will assume all states are reachable:

Definition: A linear MDP is η-**reachable** if for all s and h we have

$$\max_\pi \mathbb{P}_\pi[x_h = x] \geq \eta \|\mu_h(s)\|_1$$

Assumption can be removed, but highly technical to do so

IDEALIZED GREEDY COVER

We will build a policy cover greedily

(0) **Initialize** policy cover $\Psi = \emptyset$ and uncovered states $\mathcal{B} = \mathcal{S}$

IDEALIZED GREEDY COVER

We will build a policy cover greedily

(0) **Initialize** policy cover $\Psi = \emptyset$ and uncovered states $\mathcal{B} = \mathcal{S}$

(1) In each step, find a policy that **maximizes the probability of reaching uncovered states**

IDEALIZED GREEDY COVER

We will build a policy cover greedily

(0) **Initialize** policy cover $\Psi = \emptyset$ and uncovered states $\mathcal{B} = \mathcal{S}$

(1) In each step, find a policy that **maximizes the probability of reaching uncovered states**

Set $\mu \leftarrow \sum_{x \in \mathcal{B}} \mu_{h+1}(s)$, find $\widehat{\pi} = \arg\max_{\pi} \mathbb{E}[\langle \phi_h(s_h, a_h), \mu \rangle]$

IDEALIZED GREEDY COVER

We will build a policy cover greedily

(0) **Initialize** policy cover $\Psi = \emptyset$ and uncovered states $\mathcal{B} = \mathcal{S}$

(1) In each step, find a policy that **maximizes the probability of reaching uncovered states**

Set $\mu \leftarrow \sum_{x \in \mathcal{B}} \mu_{h+1}(s)$, find $\widehat{\pi} = \arg\max_{\pi} \mathbb{E}[\langle \phi_h(s_h, a_h), \mu \rangle]$

(2) **Break** if total probability is small, i.e $\mathbb{E}[\langle \phi_h(s_h, a_h), \mu \rangle] < \xi$

IDEALIZED GREEDY COVER

We will build a policy cover greedily

(0) **Initialize** policy cover $\Psi = \emptyset$ and uncovered states $\mathcal{B} = \mathcal{S}$

(1) In each step, find a policy that **maximizes the probability of reaching uncovered states**

Set $\mu \leftarrow \sum_{x \in \mathcal{B}} \mu_{h+1}(s)$, find $\widehat{\pi} = \arg\max_{\pi} \mathbb{E}[\langle \phi_h(s_h, a_h), \mu \rangle]$

(2) **Break** if total probability is small, i.e $\mathbb{E}[\langle \phi_h(s_h, a_h), \mu \rangle] < \xi$

Otherwise, update $\Psi \leftarrow \Psi \cup \{\widehat{\pi}\}$ and set

$$\mathcal{B} \leftarrow \mathcal{B} \backslash \left\{ s \middle| \mathbb{E}_{\widehat{\pi}}[\langle \phi_h(s_h, a_h), \mu_{h+1}(s) \rangle] \geq \frac{\xi}{2C} \|\mu_{h+1}(s)\|_1 \right\}$$

TERMINATION CONDITIONS

Claim 1 [informal]: No policy π reaches the set of uncovered states with nonnegligible probability

TERMINATION CONDITIONS

Claim 1: For any policy π, we have $\mathbb{P}_\pi[s_{h+1} \in \mathcal{B}] < \xi$

TERMINATION CONDITIONS

Claim 1: For any policy π, we have $\mathbb{P}_\pi[s_{h+1} \in \mathcal{B}] < \xi$

Proof: We choose $\hat{\pi}$ to maximize this probability, and we only break when it is small. ∎

TERMINATION CONDITIONS

Claim 1: For any policy π, we have $\mathbb{P}_\pi[s_{h+1} \in \mathcal{B}] < \xi$

Proof: We choose $\hat{\pi}$ to maximize this probability, and we only break when it is small. ■

Claim 2 [informal]: For any covered state s, no policy π reaches it with much larger probability than our cover does

TERMINATION CONDITIONS

Claim 1: For any policy π, we have $\mathbb{P}_\pi[s_{h+1} \in \mathcal{B}] < \xi$

Proof: We choose $\hat{\pi}$ to maximize this probability, and we only break when it is small. ■

Claim 2: For all $s \in \mathcal{S} \setminus \mathcal{B}$ there is some $\pi' \in \Psi$ so that

$$\mathbb{P}_{\pi'}[s_{h+1} = s] \geq \frac{\xi}{2C} \max_\pi \mathbb{P}_\pi[s_{h+1} = s]$$

TERMINATION CONDITIONS

Claim 1: For any policy π, we have $\mathbb{P}_\pi[s_{h+1} \in \mathcal{B}] < \xi$

Proof: We choose $\hat{\pi}$ to maximize this probability, and we only break when it is small. ■

Claim 2: For all $s \in \mathcal{S} \setminus \mathcal{B}$ there is some $\pi' \in \Psi$ so that

$$\mathbb{P}_{\pi'}[s_{h+1} = s] \geq \frac{\xi}{2C} \max_\pi \mathbb{P}_\pi[s_{h+1} = s]$$

Proof: First $\mathbb{P}_\pi[s_{h+1} = s] = \mathbb{E}_\pi[\langle \phi_h(s_h, a_h), \mu_{h+1}(s) \rangle]$

TERMINATION CONDITIONS

Claim 1: For any policy π, we have $\mathbb{P}_{\pi}[s_{h+1} \in \mathcal{B}] < \xi$

Proof: We choose $\hat{\pi}$ to maximize this probability, and we only break when it is small. ■

Claim 2: For all $s \in \mathcal{S} \setminus \mathcal{B}$ there is some $\pi' \in \Psi$ so that

$$\mathbb{P}_{\pi'}[s_{h+1} = s] \geq \frac{\xi}{2C} \max_{\pi} \mathbb{P}_{\pi}[s_{h+1} = s]$$

Proof: First $\mathbb{P}_{\pi}[s_{h+1} = s] = \mathbb{E}_{\pi}[\langle \phi_h(s_h, a_h), \mu_{h+1}(s) \rangle]$

$$\leq \|\mu_{h+1}(s)\|_1$$ **(by normalization)**

TERMINATION CONDITIONS

Claim 1: For any policy π, we have $\mathbb{P}_\pi[s_{h+1} \in \mathcal{B}] < \xi$

Proof: We choose $\widehat{\pi}$ to maximize this probability, and we only break when it is small. ■

Claim 2: For all $s \in \mathcal{S} \setminus \mathcal{B}$ there is some $\pi' \in \Psi$ so that

$$\mathbb{P}_{\pi'}[s_{h+1} = s] \geq \frac{\xi}{2C} \max_\pi \mathbb{P}_\pi[s_{h+1} = s]$$

Proof: First $\mathbb{P}_\pi[s_{h+1} = s] = \mathbb{E}_\pi[\langle \phi_h(s_h, a_h), \mu_{h+1}(s) \rangle]$

$$\leq \|\mu_{h+1}(s)\|_1 \quad \textbf{(by normalization)}$$

Claim now follows from rule for removing states from $\mathcal{B}$ ■

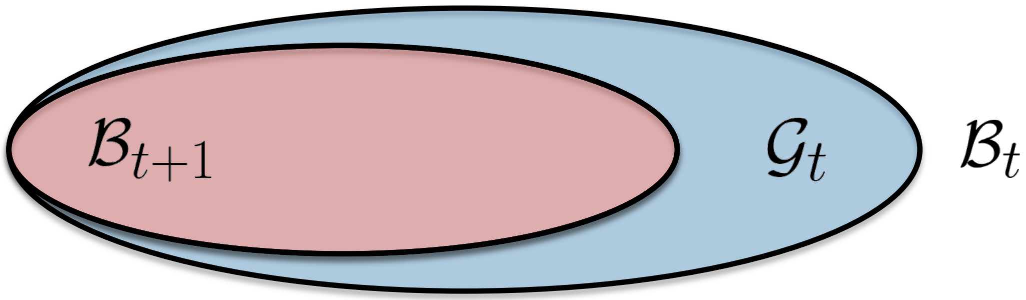
BOUNDING THE SIZE

Claim 3: Finally $|\Psi| \leq \dfrac{2C}{\xi}$

BOUNDING THE SIZE

Claim 3: Finally $|\Psi| \leq \frac{2C}{\xi}$

Proof: Suppose $\hat{\pi}$ was added to the policy cover in iteration t

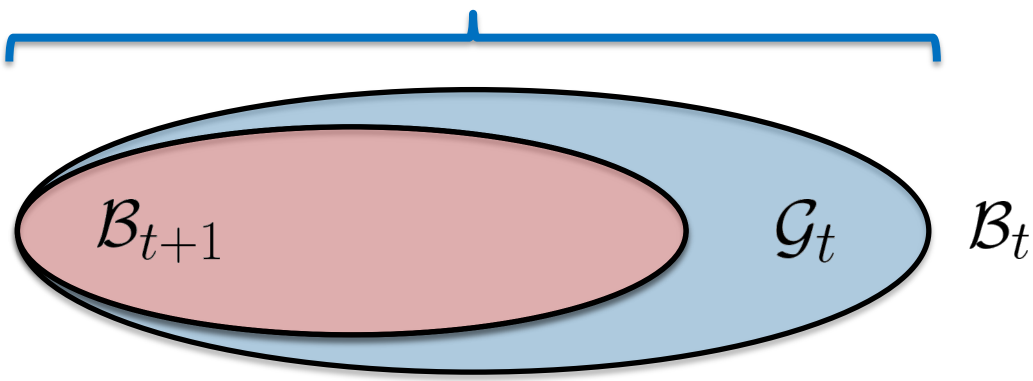


BOUNDING THE SIZE

Claim 3: Finally $|\Psi| \leq \dfrac{2C}{\xi}$

Proof: Suppose $\widehat{\pi}$ was added to the policy cover in iteration t

$\mathbb{P}_{\widehat{\pi}}[s_{h+1} \in \mathcal{B}_t] \geq \xi$ because we didn't break

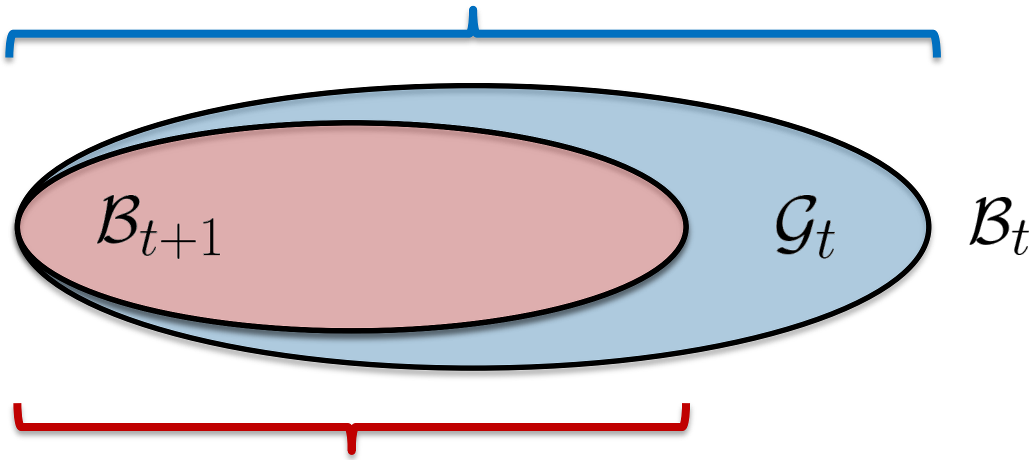


BOUNDING THE SIZE

Claim 3: Finally $|\Psi| \leq \frac{2C}{\xi}$

Proof: Suppose $\widehat{\pi}$ was added to the policy cover in iteration t

$\mathbb{P}_{\widehat{\pi}}[s_{h+1} \in \mathcal{B}_t] \geq \xi$ because we didn't break



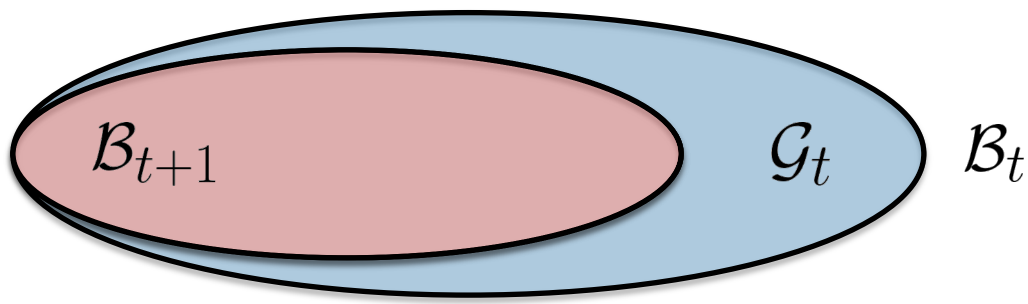
$\mathbb{P}_{\widehat{\pi}}[s_{h+1} \in \mathcal{B}_{t+1}]$, want to upper bound

BOUNDING THE SIZE

Claim 3: Finally $|\Psi| \le \frac{2C}{\xi}$

Proof: Suppose $\widehat{\pi}$ was added to the policy cover in iteration t

$\mathbb{P}_{\widehat{\pi}}[s_{h+1} \in \mathcal{B}_t] \ge \xi$ because we didn't break



(1) by rule for constructing $\mathcal{B}_{t+1}$

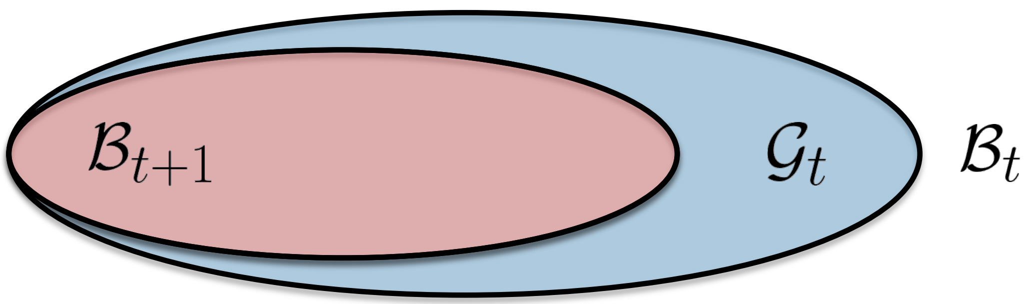
$$\mathbb{P}_{\widehat{\pi}}[s_{h+1} \in \mathcal{B}_{t+1}] \overset{(1)}{\le} \frac{\xi}{2C} \sum_{s \in \mathcal{B}_{t+1}} \|\mu_{h+1}(s)\|_1$$

BOUNDING THE SIZE

Claim 3: Finally $|\Psi| \leq \dfrac{2C}{\xi}$

Proof: Suppose $\widehat{\pi}$ was added to the policy cover in iteration t

$$\mathbb{P}_{\widehat{\pi}}[s_{h+1} \in \mathcal{B}_t] \geq \xi$$ because we didn't break



(2) by sparsity bound

$$\mathbb{P}_{\widehat{\pi}}[s_{h+1} \in \mathcal{B}_{t+1}] \overset{(1)}{\leq} \frac{\xi}{2C} \sum_{s \in \mathcal{B}_{t+1}} \|\mu_{h+1}(s)\|_1 \overset{(2)}{\leq} \frac{\xi}{2}$$

BOUNDING THE SIZE

Claim 3: Finally $|\Psi| \leq \dfrac{2C}{\xi}$

Proof: Suppose $\widehat{\pi}$ was added to the policy cover in iteration t

BOUNDING THE SIZE

Claim 3: Finally $|\Psi| \leq \dfrac{2C}{\xi}$

Proof: Suppose $\widehat{\pi}$ was added to the policy cover in iteration t

$$\Rightarrow \sum_{s \in \mathcal{G}_t} \|\mu_{h+1}(s)\|_1 \geq \mathbb{P}_{\widehat{\pi}}[s_{h+1} \in \mathcal{G}_t] \geq \frac{\xi}{2}$$

BOUNDING THE SIZE

Claim 3: Finally $|\Psi| \leq \dfrac{2C}{\xi}$

Proof: Suppose $\widehat{\pi}$ was added to the policy cover in iteration t

$$\Rightarrow \sum_{s \in \mathcal{G}_t} \|\mu_{h+1}(s)\|_1 \geq \mathbb{P}_{\widehat{\pi}}[s_{h+1} \in \mathcal{G}_t] \geq \frac{\xi}{2}$$

But since the sets $\mathcal{G}_1, \mathcal{G}_2, \ldots$ are disjoint, we have

$$C \geq \sum_{s \in \mathcal{S}} \|\mu_{h+1}(s)\|_1 \geq \sum_{s \in \mathcal{G}_1 \cup \mathcal{G}_2 \ldots} \|\mu_{h+1}(s)\|_1 \geq \frac{t\xi}{2}$$

BOUNDING THE SIZE

Claim 3: Finally $|\Psi| \leq \dfrac{2C}{\xi}$

Proof: Suppose $\widehat{\pi}$ was added to the policy cover in iteration t

$$\Rightarrow \sum_{s \in \mathcal{G}_t} \|\mu_{h+1}(s)\|_1 \geq \mathbb{P}_{\widehat{\pi}}[s_{h+1} \in \mathcal{G}_t] \geq \frac{\xi}{2}$$

But since the sets $\mathcal{G}_1, \mathcal{G}_2, \ldots$ are disjoint, we have

$$C \geq \sum_{s \in \mathcal{S}} \|\mu_{h+1}(s)\|_1 \geq \sum_{s \in \mathcal{G}_1 \cup \mathcal{G}_2 \ldots} \|\mu_{h+1}(s)\|_1 \geq \frac{t\xi}{2}$$

Now rearranging completes the proof. ■

In general Ψ is not a policy cover but…

In general Ψ is not a policy cover but…

Lemma [informal]: Under reachability, the collection

$$\Psi \circ_{h+1} \mathrm{unif}(\mathcal{A})$$

is a policy cover at step h+2

MAKING IT EFFICIENT?

There are two main issues with our approach:

(1) We can’t afford to iterate over all the states

MAKING IT EFFICIENT?

There are two main issues with our approach:

(1) We can't afford to iterate over all the states

E.g. when we updated the set of uncovered states

MAKING IT EFFICIENT?

There are two main issues with our approach:

(1) We can't afford to iterate over all the states

E.g. when we updated the set of uncovered states

(2) We don't know the features $\mu_h(s)$

MAKING IT EFFICIENT?

There are two main issues with our approach:

(1) We can't afford to iterate over all the states

E.g. when we updated the set of uncovered states

(2) We don't know the features $\mu_h(s)$

Since the model is non-parametric, it's not even possible to estimate all these parameters

OUTLINE

Part V: Linear MDPs

- Least Squares and the Elliptic Potential
- Feature Selection and Sparsity

Part VII: Sparse Linear MDPs

- Sparse Linear Regression
- Greedy Covers
- Emulators and their Algorithmic Applications

Part VI: Block MDPs

- The View from Supervised Learning

OUTLINE

Part V: Linear MDPs

- Least Squares and the Elliptic Potential
- Feature Selection and Sparsity

Part VII: Sparse Linear MDPs

- Sparse Linear Regression
- Greedy Covers
- **Emulators and their Algorithmic Applications**

Part VI: Block MDPs

- The View from Supervised Learning

EMULATORS

Can we construct a tabular MDP that approximates a linear MDP well enough to be used in Greedy Cover instead?

EMULATORS

Definition: An **emulator at step h** for a sparse linear MDP is a collection of synthetic features that satisfy…

EMULATORS

Definition: An **emulator at step h** for a sparse linear MDP is a collection of synthetic features that satisfy

(1) they are **analytically sparse**

EMULATORS

Definition: An **emulator at step h** for a sparse linear MDP is a collection of vectors $\widehat{\mu}_{h+1}^1, \widehat{\mu}_{h+1}^2, \ldots, \widehat{\mu}_{h+1}^m$ that satisfy

(1) $$\sum_j \|\widehat{\mu}_{h+1}^j\|_1 \leq C'$$

EMULATORS

Definition: An **emulator at step h** for a sparse linear MDP is a collection of vectors $\widehat{\mu}^1_{h+1}, \widehat{\mu}^2_{h+1}, \ldots, \widehat{\mu}^m_{h+1}$ that satisfy

(1) $\sum_j \|\widehat{\mu}^j_{h+1}\|_1 \leq C'$

(2) they have **nonnegative** inner-products with feature maps

EMULATORS

Definition: An **emulator at step h** for a sparse linear MDP is a collection of vectors $\widehat{\mu}^1_{h+1}, \widehat{\mu}^2_{h+1}, \ldots, \widehat{\mu}^m_{h+1}$ that satisfy

(1) $\sum_j \|\widehat{\mu}^j_{h+1}\|_1 \leq C'$

(2) $\langle \mathbb{E}_\pi[\phi_h(s_h, a_h)], \widehat{\mu}^j_{h+1} \rangle \geq 0$ for all π and j

EMULATORS

Definition: An **emulator at step h** for a sparse linear MDP is a collection of vectors $\widehat{\mu}^1_{h+1}, \widehat{\mu}^2_{h+1}, \ldots, \widehat{\mu}^m_{h+1}$ that satisfy

(1) $\sum_j \|\widehat{\mu}^j_{h+1}\|_1 \leq C'$

(2) $\langle \mathbb{E}_\pi[\phi_h(s_h, a_h)], \widehat{\mu}^j_{h+1} \rangle \geq 0$ for all π and j

(3) they determine the same **input-output behavior** as the true MDP, for any policy π

EMULATORS

Definition: An **emulator at step h** for a sparse linear MDP is a collection of vectors $\widehat{\mu}^1_{h+1}, \widehat{\mu}^2_{h+1}, \ldots, \widehat{\mu}^m_{h+1}$ that satisfy

(1) $\sum_j \|\widehat{\mu}^j_{h+1}\|_1 \leq C'$

(2) $\langle \mathbb{E}_\pi[\phi_h(s_h, a_h)], \widehat{\mu}^j_{h+1} \rangle \geq 0$ for all π and j

(3) There are states $\widetilde{s}^1, \widetilde{s}^2, \ldots, \widetilde{s}^m$ so that for all π

true input-output

$$\sum_{s \in \mathcal{S}} \langle \mathbb{E}_\pi[\phi_h(s_h, a_h)], \mu_{h+1}(s) \rangle \phi^{\text{avg}}_{h+1}(s)$$

$$\approx \sum_{j=1}^{m} \langle \mathbb{E}_\pi[\phi_h(s_h, a_h)], \widehat{\mu}^j_{h+1} \rangle \phi^{\text{avg}}_{h+1}(\widetilde{s}^j)$$

synthetic input-output

PUTTING IT TOGETHER

Let's run Greedy Cover on the $\hat{\mu}$'s instead!

PUTTING IT TOGETHER

Let's run Greedy Cover on the $\widehat{\mu}$'s instead!

uncovered states: $\mathcal{B} = [m] \longleftrightarrow \widetilde{s}^1, \widetilde{s}^2, \ldots, \widetilde{s}^m$

target vector: $\mu \leftarrow \sum_{j \in \mathcal{B}} \widehat{\mu}^j_{h+1}$

PUTTING IT TOGETHER

Let's run Greedy Cover on the $\widehat{\mu}$'s instead!

uncovered states: $\mathcal{B} = [m] \longleftrightarrow \widetilde{s}^1, \widetilde{s}^2, \ldots, \widetilde{s}^m$

target vector: $\mu \leftarrow \sum_{j \in \mathcal{B}} \widehat{\mu}^j_{h+1}$

Essentially, the emulator gives us a tabular approximation, now **Greedy Cover** is efficient

PUTTING IT TOGETHER

In the analysis of **Idealized Greedy Cover** we needed to show

$$\sum_{s \in \mathcal{S} \setminus \mathcal{B}} \langle \mathbb{E}_{\pi}[\phi_h(s_h, a_h)], \mu_{h+1}(s) \rangle \langle \phi^{\text{avg}}_{h+1}(s), \mu_{h+2}(s') \rangle$$

$$\leq \frac{4C^2}{\xi^2} \mathbb{E}_{\widehat{\pi} \sim \Psi} \Big[\sum_{s \in \mathcal{S} \setminus \mathcal{B}} \langle \mathbb{E}_{\widehat{\pi}}[\phi_h(s_h, a_h)], \mu_{h+1}(s) \rangle \langle \phi^{\text{avg}}_{h+1}(s), \mu_{h+2}(s') \rangle \Big]$$

i.e. can't reach any state s' much more often than the cover does

PUTTING IT TOGETHER

In the analysis of **Idealized Greedy Cover** we needed to show

$$\sum_{s \in \mathcal{S} \setminus \mathcal{B}} \langle \mathbb{E}_{\pi}[\phi_h(s_h, a_h)], \mu_{h+1}(s) \rangle \langle \phi_{h+1}^{\text{avg}}(s), \mu_{h+2}(s') \rangle$$

$$\leq \frac{4C^2}{\xi^2} \mathbb{E}_{\widehat{\pi} \sim \Psi}\Big[\sum_{s \in \mathcal{S} \setminus \mathcal{B}} \langle \mathbb{E}_{\widehat{\pi}}[\phi_h(s_h, a_h)], \mu_{h+1}(s) \rangle \langle \phi_{h+1}^{\text{avg}}(s), \mu_{h+2}(s') \rangle \Big]$$

i.e. can't reach any state s' much more often than the cover does

The properties of an emulator allow us to replace the μ's with $\widehat{\mu}$'s in the expressions above, and the analysis goes through

Why do small emulators exist? And how do we find them?

Why do small emulators exist? And how do we find them?

We use convex programming

Why do small emulators exist? And how do we find them?

We use convex programming

Main Idea: Instead of a constraint for each policy, i.e. for all π

true input-output $\approx$ **synthetic input-output**

Why do small emulators exist? And how do we find them?

We use convex programming

Main Idea: Instead of a constraint for each policy, i.e. for all π

true input-output $\approx$ **synthetic input-output**

We will draw samples from a policy cover at step h, and ask for synthetic features that are good predictors for the output

A CONVEX PROGRAM

Input: Samples drawn from a policy cover Ψ at step h

A CONVEX PROGRAM

Input: Samples drawn from a policy cover Ψ at step h

(1) Solve a linear regression problem for all $\ell \in [d]$

$$\widehat{w}_\ell = \arg\min_{\|w\|_1 \leq C} \sum_{i=1}^{n} \Big(\langle \phi_h(s_h^i, a_h^i), w \rangle - \phi_{h+1}^{\text{avg}}(s_{h+1}^i)_\ell \Big)^2$$

A CONVEX PROGRAM

Input: Samples drawn from a policy cover Ψ at step h

(1) Solve a linear regression problem for all $\ell \in [d]$

$$\widehat{w}_\ell = \underset{\|w\|_1 \leq C}{\arg\min} \sum_{i=1}^{n} \Big(\langle \phi_h(s_h^i, a_h^i), w \rangle - \phi_{h+1}^{\text{avg}}(s_{h+1}^i)_\ell \Big)^2$$

(2) Find synthetic features that satisfy **ℓ_1 boundedness**, **nonnegativity**, and are also **good predictors** i.e.

A CONVEX PROGRAM

Input: Samples drawn from a policy cover Ψ at step h

(1) Solve a linear regression problem for all $\ell \in [d]$

$$\widehat{w}_\ell = \arg\min_{\|w\|_1 \leq C} \sum_{i=1}^{n} \left(\langle \phi_h(s_h^i, a_h^i), w \rangle - \phi_{h+1}^{\text{avg}}(s_{h+1}^i)_\ell \right)^2$$

(2) Find synthetic features that satisfy **ℓ_1 boundedness**, **nonnegativity**, and are also **good predictors** i.e.

$$\frac{1}{n} \sum_{i=1}^{n} \left(\langle \phi_h(s_h^i, a_h^i), \widehat{w}_\ell \rangle - \sum_{j=1}^{m} \langle \phi_h(s_h^i, a_h^i), \widehat{\mu}_{h+1}^j \rangle \phi_{h+1}^{\text{avg}}(\widetilde{s}_{h+1}^j)_\ell \right)^2 \leq \epsilon$$

A CONVEX PROGRAM

Can establish feasibility and correctness

A CONVEX PROGRAM

Can establish feasibility and correctness

Intuition: The true features, with importance-weighting are **feasible with high probability**

A CONVEX PROGRAM

Can establish feasibility and correctness

Intuition: The true features, with importance-weighting are **feasible with high probability**

Proof of correctness uses a net on policies, argues that checking constrains on a representative set (i.e. cover) of policies suffices

A CONVEX PROGRAM

Can establish feasibility and correctness

Intuition: The true features, with importance-weighting are **feasible with high probability**

Proof of correctness uses a net on policies, argues that checking constrains on a representative set (i.e. cover) of policies suffices

Ultimately we use policy covers at steps ≤ h to (1) build an emulator and (2) solve the optimization problem in Greedy Cover, together **which gives a small cover at step h+2**

OUTLINE

Part V: Linear MDPs

- Least Squares and the Elliptic Potential
- Feature Selection and Sparsity

Part VII: Sparse Linear MDPs

- Sparse Linear Regression
- Greedy Covers
- Emulators and their Algorithmic Applications

Part VI: Block MDPs

- The View from Supervised Learning

OUTLINE

Part V: Linear MDPs

- Least Squares and the Elliptic Potential
- Feature Selection and Sparsity

Part VII: Sparse Linear MDPs

- Sparse Linear Regression
- Greedy Covers
- Emulators and their Algorithmic Applications

Part VI: Block MDPs

- The View from Supervised Learning

BLOCK MDPS

Settings with rich observations ($\mathcal{X}$) but simpler latent state ($\mathcal{S}$)

e.g images
vs.
physical location

BLOCK MDPS

Settings with rich observations ($\mathcal{X}$) but simpler latent state ($\mathcal{S}$)

e.g images

vs.

physical location

Assumption: There is an unknown **decoding function**

$$\rho^* : \mathcal{X} \to \mathcal{S}$$

in some known class Φ of functions

BLOCK MDPS

Key result:

Theorem [Du, Krishnamurthy, Jiang, Agarwal, Dudik, Langford ‘19]:
There is a framework with

sample complexity: $\mathrm{poly}(|\mathcal{S}|, |\mathcal{A}|, H, \log|\Phi|)$

that returns a near optimal policy in a sparse linear MDP

BLOCK MDPS

Key result:

Theorem [Du, Krishnamurthy, Jiang, Agarwal, Dudik, Langford '19]:
There is a framework with

sample complexity: $\mathrm{poly}(|\mathcal{S}|, |\mathcal{A}|, H, \log|\Phi|)$

that returns a near optimal policy in a sparse linear MDP

But it relies on computationally hard oracles

BLOCK MDPS

Key result:

Theorem [Du, Krishnamurthy, Jiang, Agarwal, Dudik, Langford '19]:
There is a framework with

sample complexity: $\mathrm{poly}(|\mathcal{S}|, |\mathcal{A}|, H, \log|\Phi|)$

that returns a near optimal policy in a sparse linear MDP

But it relies on computationally hard oracles

When can we get algorithmic guarantees?

THE VIEW FROM SUPERVISED LEARNING

Not hard to show:

Observation: For some class Φ of decoding functions, if the associated noisy supervised learning problem is hard, then the RL problem is hard too

THE VIEW FROM SUPERVISED LEARNING

Not hard to show:

Observation: For some class Φ of decoding functions, if the associated noisy supervised learning problem is hard, then the RL problem is hard too

e.g. if the decoder is a parity function

THE VIEW FROM SUPERVISED LEARNING

Not hard to show:

Observation: For some class Φ of decoding functions, if the associated noisy supervised learning problem is hard, then the RL problem is hard too

e.g. if the decoder is a parity function

What if we start with a class Φ that can be PAC learned?

THE VIEW FROM SUPERVISED LEARNING

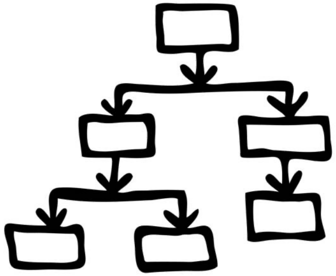
Not hard to show:

Observation: For some class Φ of decoding functions, if the associated noisy supervised learning problem is hard, then the RL problem is hard too

e.g. if the decoder is a parity function

What if we start with a class Φ that can be PAC learned?

e.g. bounded-depth decision trees



simple, interpretable

MORE ALGORITHMIC APPLICATIONS

By interpreting decision trees as sparse regressors:

Corollary: There is a quasi-polynomial time algorithm for learning a near optimal policy in any block MDP with a bounded depth decision tree decoder

MORE ALGORITHMIC APPLICATIONS

By interpreting decision trees as sparse regressors:

Corollary: There is a quasi-polynomial time algorithm for learning a near optimal policy in any block MDP with a bounded depth decision tree decoder

This gives an RL-style generalization of a classic result in supervised learning

Summary:

- **First computationally efficient algorithm for learning in sparse linear MDPs**
- Applications to Block MDPs, including an RL-style generalization of learning decision trees
- Meaningful way to approximate nonparametric models through **emulators**

Summary:

- **First computationally efficient algorithm for learning in sparse linear MDPs**
- Applications to Block MDPs, including an RL-style generalization of learning decision trees
- Meaningful way to approximate nonparametric models through **emulators**

Thanks! Any Questions?