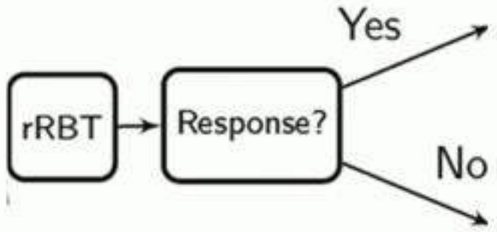
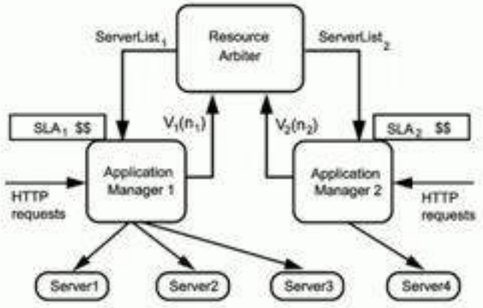
On the hardness of RL with value-function approximation

Nan Jiang
University of Illinois at Urbana-Champaign
May 30, 2019
@ MSR Redmond

Based on: *Information-Theoretic Considerations in Batch Reinforcement Learning.*
to appear at ICML-19, with Jinglin Chen (UIUC)

Reinforcement Learning (RL) Applications





[Levine et al'16] [Ng et al'03] [Singh et al'02] [Lei et al'12]
[Mandel et al'16] [Tesauro et al'07] [Mnih et al'15] [Silver et al'16]

Overview

- Value-function approximation

Overview

- Value-function approximation
  - Use a restricted class of functions to approximate the optimal value function Q^*

Overview

- Value-function approximation
  - Use a restricted class of functions to approximate the optimal value function Q^*
  - Batch mode: given exploratory data, no access to env

Overview

- Value-function approximation
  - Use a restricted class of functions to approximate the optimal value function Q^*
  - Batch mode: given exploratory data, no access to env
  - Provides understanding to popular deep RL algorithms, e.g., DQN

Overview

- Value-function approximation
  - Use a restricted class of functions to approximate the optimal value function Q^*
  - Batch mode: given exploratory data, no access to env
  - Provides understanding to popular deep RL algorithms, e.g., DQN
- When can we guarantee sample-efficient learning?

Comparison between SL and RL

- Supervised learning
  - Data: $(x, y) \sim P_{X,Y}$
  - A class of predictors F (assume finite), one of which is good
  - Can find a good predictor w/ $O(\log|F|)$ samples (info-theoretic)

Comparison between SL and RL

- Supervised learning
  - Data: $(x, y) \sim P_{X,Y}$
  - A class of predictors F (assume finite), one of which is good
  - Can find a good predictor w/ $O(\log|F|)$ samples (info-theoretic)
    - ERM, Hoeffding, Union Bound

Comparison between SL and RL

- Supervised learning
  - Data: $(x, y) \sim P_{X,Y}$
  - A class of predictors F (assume finite), one of which is good
  - Can find a good predictor w/ $O(\log|F|)$ samples (info-theoretic)
    - ERM, Hoeffding, Union Bound
- Reinforcement learning (batch-mode, VFA)

Comparison between SL and RL

- Supervised learning
  - Data: $(x, y) \sim P_{X,Y}$
  - A class of predictors F (assume finite), one of which is good
  - Can find a good predictor w/ $O(\log|F|)$ samples (info-theoretic)
    - ERM, Hoeffding, Union Bound
- Reinforcement learning (batch-mode, VFA)
  - Data: (s, a, r, s') from MDP (to be defined)

Comparison between SL and RL

- Supervised learning
  - Data: $(x, y) \sim P_{X,Y}$
  - A class of predictors F (assume finite), one of which is good
  - Can find a good predictor w/ $O(\log|F|)$ samples (info-theoretic)
    - ERM, Hoeffding, Union Bound

- Reinforcement learning (batch-mode, VFA)
  - Data: (s, a, r, s') from MDP (to be defined)
    - Needs to be exploratory (to be formalized)

Comparison between SL and RL

- Supervised learning
  - Data: $(x, y) \sim P_{X,Y}$
  - A class of predictors F (assume finite), one of which is good
  - Can find a good predictor w/ $O(\log|F|)$ samples (info-theoretic)
    - ERM, Hoeffding, Union Bound

- Reinforcement learning (batch-mode, VFA)
  - Data: (s, a, r, s') from MDP (to be defined)
    - Needs to be exploratory (to be formalized)
  - Function class F (assume finite), one of which is Q^*

Comparison between SL and RL

- Supervised learning
  - Data: $(x, y) \sim P_{X,Y}$
  - A class of predictors F (assume finite), one of which is good
  - Can find a good predictor w/ $O(\log|F|)$ samples (info-theoretic)
    - ERM, Hoeffding, Union Bound

- Reinforcement learning (batch-mode, VFA)
  - Data: (s, a, r, s') from MDP (to be defined)
    - Needs to be exploratory (to be formalized)
  - Function class F (assume finite), one of which is Q^*
  - Can we find a near-optimal policy using $O(\log|F|)$ samples?

Comparison between SL and RL

- Supervised learning
  - Data: $(x, y) \sim P_{X,Y}$
  - A class of predictors F (assume finite), one of which is good
  - Can find a good predictor w/ $O(\log|F|)$ samples (info-theoretic)
    - ERM, Hoeffding, Union Bound

- Reinforcement learning (batch-mode, VFA)
  - Data: (s, a, r, s') from MDP (to be defined)
    - Needs to be exploratory (to be formalized)
  - Function class F (assume finite), one of which is Q^*
  - Can we find a near-optimal policy using $O(\log|F|)$ samples?

We don't know
… but we should.

Markov Decision Process (MDP)

- For $h = 1, 2, \ldots,$ the agent
  - observes state $s_h \in S$ (very large)

Markov Decision Process (MDP)

- For $h = 1, 2, \ldots,$ the agent
  - observes state $s_h \in S$ (very large)
  - chooses action $a_h \in A$ (finite & small)

Markov Decision Process (MDP)

- For $h = 1, 2, \ldots,$ the agent
  - observes state $s_h \in S$ (very large)
  - chooses action $a_h \in A$ (finite & small)
  - receives reward $r_h \sim R(s_h, a_h)$

reward function
$R: S \times A \to \Delta([0,1])$

Markov Decision Process (MDP)

- For $h = 1, 2, \ldots,$ the agent
  - observes state $s_h \in S$ (very large)
    - $s_h \sim P(s_{h-1}, a_{h-1})$ for $h \geq 2$
  - chooses action $a_h \in A$ (finite & small)
  - receives reward $r_h \sim R(s_h, a_h)$

transition dynamics
$P: S \times A \to \Delta(S)$

reward function
$R: S \times A \to \Delta([0,1])$

Markov Decision Process (MDP)

- For $h = 1, 2, \ldots,$ the agent
  - observes state $s_h \in S$ (very large)
    - $s_h \sim P(s_{h-1}, a_{h-1})$ for $h \geq 2$
  - chooses action $a_h \in A$ (finite & small)
  - receives reward $r_h \sim R(s_h, a_h)$
- Policy $\pi: S \to A$

transition dynamics
$P: S \times A \to \Delta(S)$

reward function
$R: S \times A \to \Delta([0,1])$

Markov Decision Process (MDP)

- For $h = 1, 2, \ldots$, the agent
  - observes state $s_h \in S$ (very large)
    - $s_h \sim P(s_{h-1}, a_{h-1})$ for $h \geq 2$
  - chooses action $a_h \in A$ (finite & small)
  - receives reward $r_h \sim R(s_h, a_h)$

transition dynamics
$P: S \times A \to \Delta(S)$

reward function
$R: S \times A \to \Delta([0,1])$

- Policy $\pi: S \to A$
  - Ultimate measure of goodness: $v^\pi := \mathbb{E}[\sum_{h=1}^{\infty} \gamma^{h-1} r_h | s_1 \sim \nu_0; \pi]$

Markov Decision Process (MDP)

- For $h = 1, 2, \ldots$, the agent
  - observes state $s_h \in S$ (very large)
    - $s_h \sim P(s_{h-1}, a_{h-1})$ for $h \geq 2$
  - chooses action $a_h \in A$ (finite & small)
  - receives reward $r_h \sim R(s_h, a_h)$

transition dynamics
$P: S \times A \to \Delta(S)$

reward function
$R: S \times A \to \Delta([0,1])$

- Policy $\pi: S \to A$
  - Ultimate measure of goodness: $v^\pi := \mathbb{E}[\sum_{h=1}^{\infty} \gamma^{h-1} r_h | s_1 \sim \nu_0; \pi]$
  - $\gamma \in (0,1)$ is discount factor; translates to finite horizon $H = 1/(1-\gamma)$

Markov Decision Process (MDP)

- For $h = 1, 2, \ldots$, the agent
  - observes state $s_h \in S$ (very large)
    - $s_h \sim P(s_{h-1}, a_{h-1})$ for $h \geq 2$
  - chooses action $a_h \in A$ (finite & small)
  - receives reward $r_h \sim R(s_h, a_h)$

transition dynamics
$P: S \times A \to \Delta(S)$

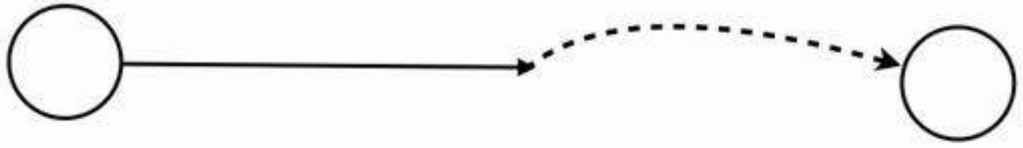
reward function
$R: S \times A \to \Delta([0,1])$

- Policy $\pi: S \to A$
  - Ultimate measure of goodness: $v^\pi := \mathbb{E}[\sum_{h=1}^{\infty} \gamma^{h-1} r_h | s_1 \sim \nu_0; \pi]$
  - $\gamma \in (0,1)$ is discount factor; translates to finite horizon $H = 1/(1-\gamma)$
  - ν_0 is the initial state distribution (e.g., empty board for Go)

Video game playing

state

$s_h \in S$



Video game playing

reward $r_h = R(s_h, a_h)$

state $s_h \in S$

+20

next state $s_{h+1} \in S$

action $a_h \in A$

Video game playing

reward $r_h = R(s_h, a_h)$

state $s_h \in S$

+20

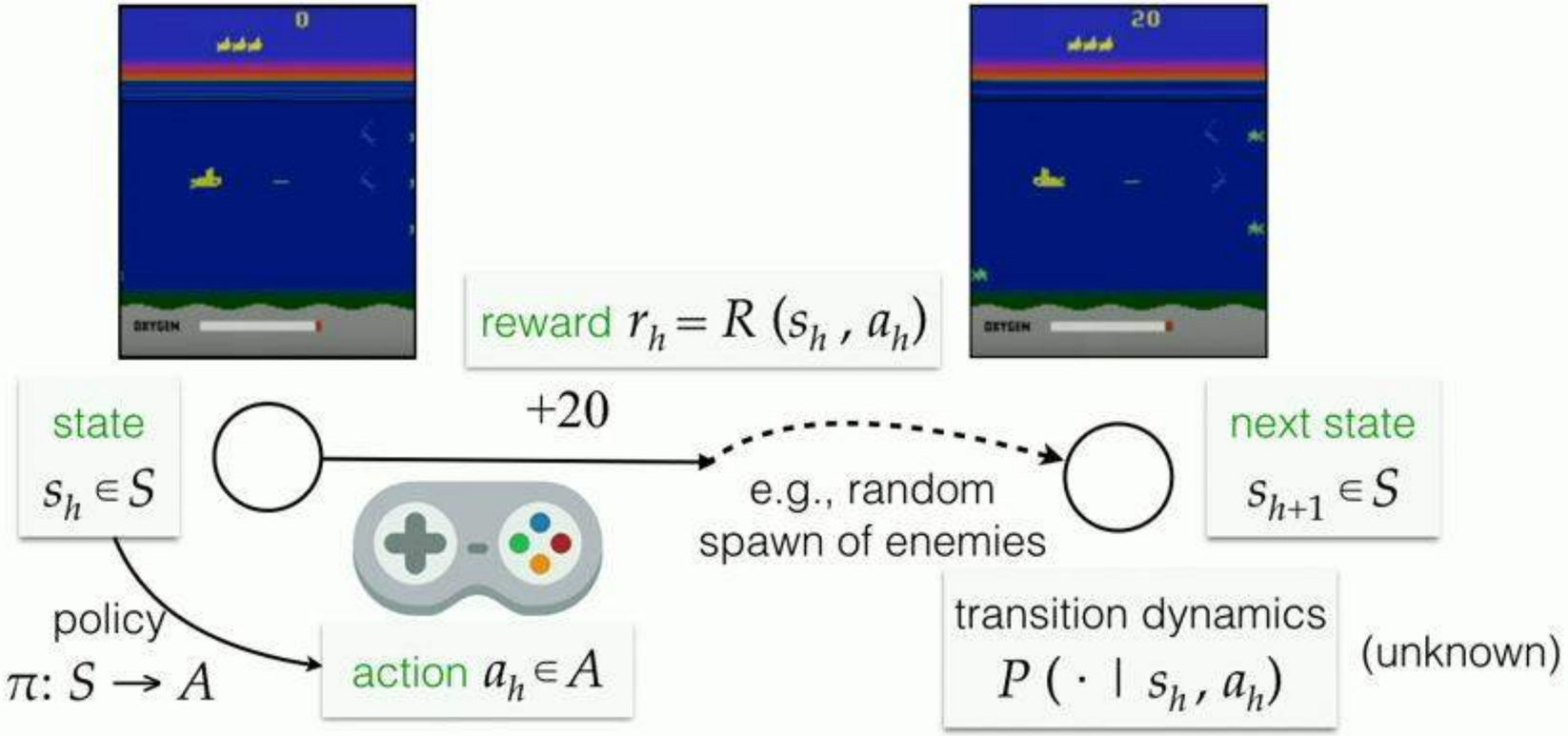
action $a_h \in A$

e.g., random spawn of enemies

next state $s_{h+1} \in S$

transition dynamics $P(\cdot \mid s_h, a_h)$ (unknown)

Video game playing



Markov Decision Process (MDP)

- For $h = 1, 2, \ldots$, the agent
  - observes state $s_h \in S$ (very large)
    - $s_h \sim P(s_{h-1}, a_{h-1})$ for $h \geq 2$
  - chooses action $a_h \in A$ (finite & small)
  - receives reward $r_h \sim R(s_h, a_h)$

transition dynamics
$P: S \times A \to \Delta(S)$

reward function
$R: S \times A \to \Delta([0,1])$

- Policy $\pi: S \to A$
  - Ultimate measure of goodness: $v^\pi := \mathbb{E}[\sum_{h=1}^{\infty} \gamma^{h-1} r_h | s_1 \sim \nu_0; \pi]$
  - $\gamma \in (0,1)$ is discount factor; translates to finite horizon $H = 1/(1-\gamma)$
  - ν_0 is the initial state distribution (e.g., empty board for Go)

Markov Decision Process (MDP)

- For $h = 1, 2, \ldots$, the agent
  - observes state $s_h \in S$ (very large)
    - $s_h \sim P(s_{h-1}, a_{h-1})$ for $h \geq 2$
  - chooses action $a_h \in A$ (finite & small)
  - receives reward $r_h \sim R(s_h, a_h)$

transition dynamics
$P: S \times A \to \Delta(S)$

reward function
$R: S \times A \to \Delta([0,1])$

- Policy $\pi: S \to A$
  - Ultimate measure of goodness: $v^\pi := \mathbb{E}[\sum_{h=1}^{\infty} \gamma^{h-1} r_h | s_1 \sim \nu_0; \pi]$
  - $\gamma \in (0,1)$ is discount factor; translates to finite horizon $H = 1/(1-\gamma)$
  - ν_0 is the initial state distribution (e.g., empty board for Go)
- Key solution concepts:
  - $Q^\star(s,a) := \max_\pi \mathbb{E}[\sum_{h=1}^{\infty} \gamma^{h-1} r_h \,| s_1 = s, a_1 = a; \pi]$

Markov Decision Process (MDP)

- For $h = 1, 2, \ldots$, the agent
  - observes state $s_h \in S$ (very large)
    - $s_h \sim P(s_{h-1}, a_{h-1})$ for $h \geq 2$
  - chooses action $a_h \in A$ (finite & small)
  - receives reward $r_h \sim R(s_h, a_h)$

transition dynamics
$P: S \times A \to \Delta(S)$

reward function
$R: S \times A \to \Delta([0,1])$

- Policy $\pi: S \to A$
  - Ultimate measure of goodness: $v^\pi := \mathbb{E}[\sum_{h=1}^{\infty} \gamma^{h-1} r_h | s_1 \sim \nu_0; \pi]$
  - $\gamma \in (0,1)$ is discount factor; translates to finite horizon $H = 1/(1-\gamma)$
  - ν_0 is the initial state distribution (e.g., empty board for Go)
- Key solution concepts:
  - $Q^\star(s,a) := \max_\pi \mathbb{E}[\sum_{h=1}^{\infty} \gamma^{h-1} r_h \,|s_1 = s, a_1 = a; \pi]$
  - Bellman eq: $Q^\star(s,a) = R(s,a) + \gamma \mathbb{E}_{s' \sim P(s,a)}[\max_{a'} Q^\star(s', a')]$
  - Optimal policy π^* is greedy w.r.t. Q^*

Batch learning in MDPs

- Dataset $D = \{(s, a, r, s')\}$

Batch learning in MDPs

- Dataset $D = \{(s, a, r, s')\}$
  - $(s, a) \sim \mu$ ("data distribution"), $r \sim R(s, a)$, $s' \sim P(s, a)$

Batch learning in MDPs

- Dataset $D = \{(s, a, r, s')\}$
  - $(s, a) \sim \mu$ ("data distribution"), $r \sim R(s, a)$, $s' \sim P(s, a)$
- Function class F s.t. $Q^* \in F$ (finite)

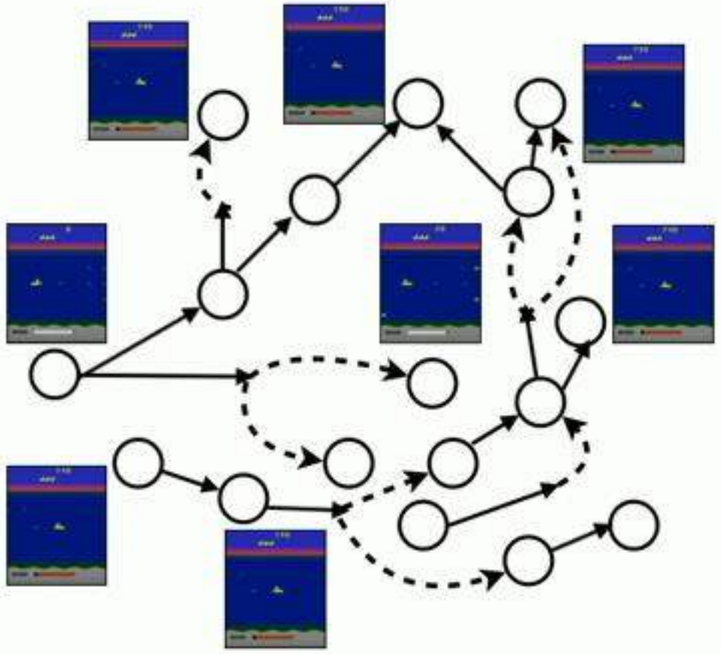
Batch learning in MDPs

- Dataset $D = \{(s, a, r, s')\}$
  - $(s, a) \sim \mu$ ("data distribution"), $r \sim R(s, a)$, $s' \sim P(s, a)$
- Function class F s.t. $Q^* \in F$ (finite)
- Goal: find $f \approx Q^*$ s.t. its greedy policy is ε-optimal

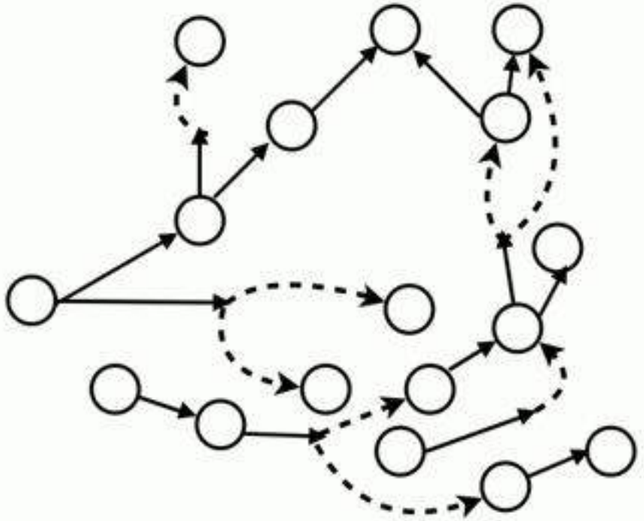
Batch learning in MDPs

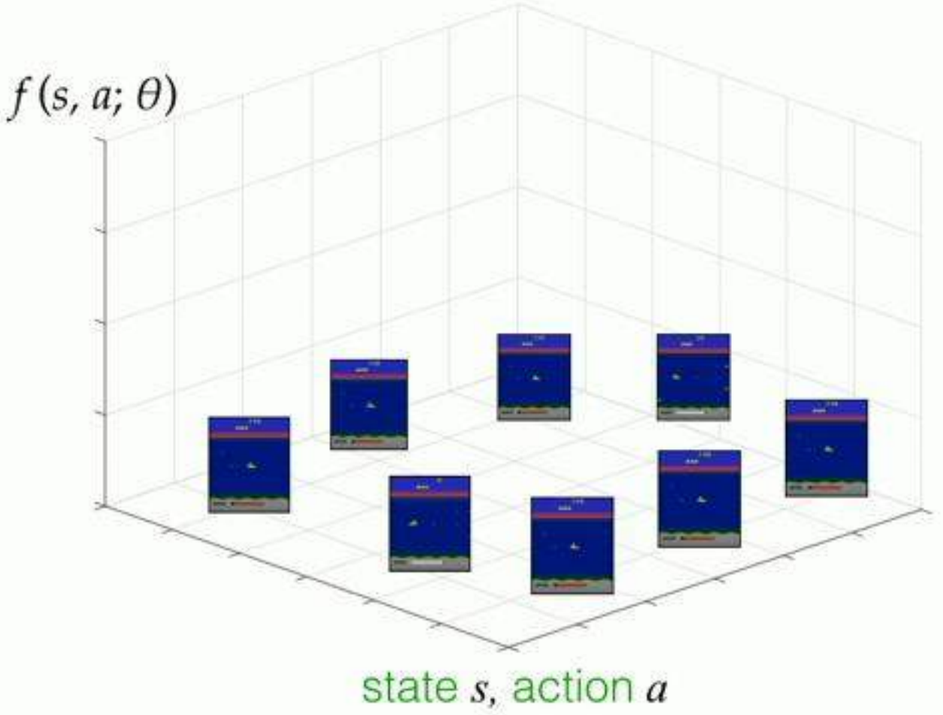
- Dataset $D = \{(s, a, r, s')\}$
  - $(s, a) \sim \mu$ ("data distribution"), $r \sim R(s, a)$, $s' \sim P(s, a)$
- Function class F s.t. $Q^* \in F$ (finite)
- Goal: find $f \approx Q^*$ s.t. its greedy policy is ε-optimal
  - Recall $v^\pi := \mathbb{E}[\sum_{h=1}^{\infty} \gamma^{h-1} r_h | s_1 \sim \nu_0; \pi]$

Example: Video game playing

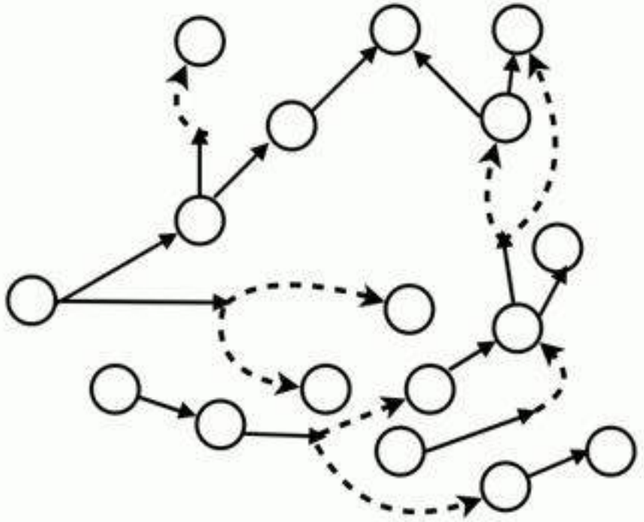


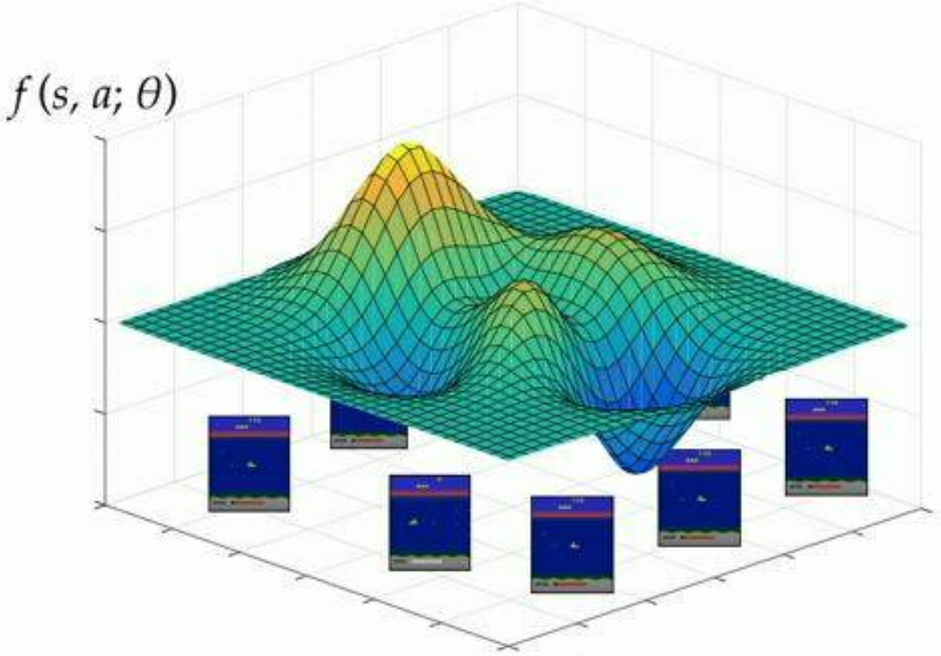
Example: Video game playing

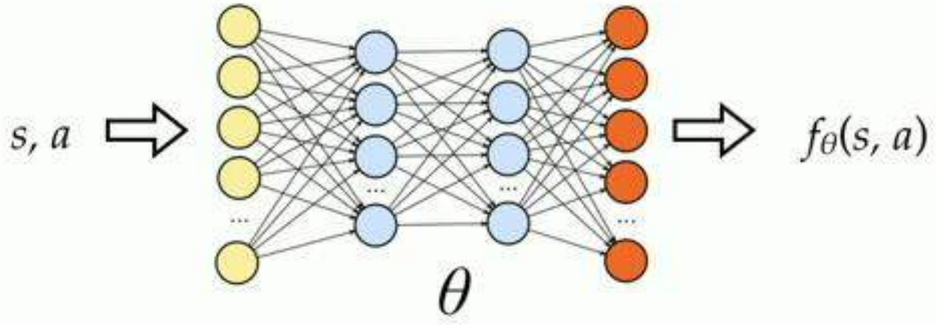




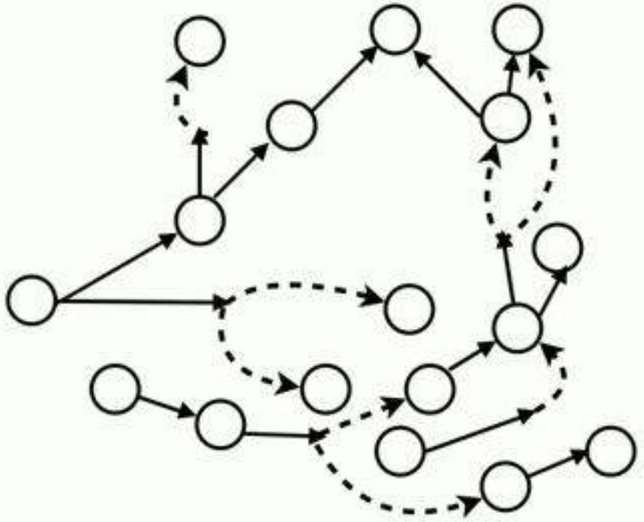
Example: Video game playing

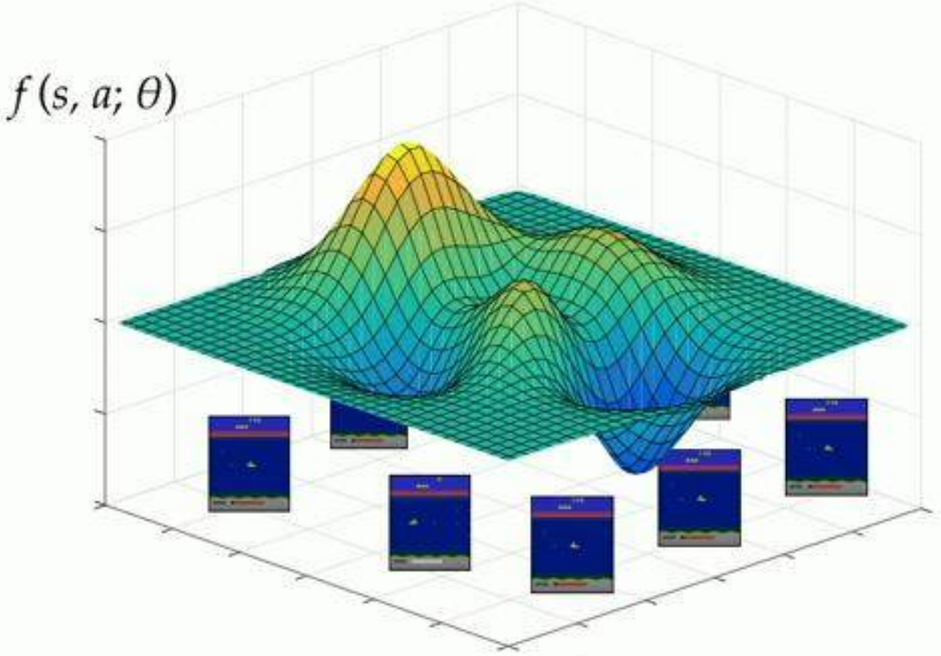


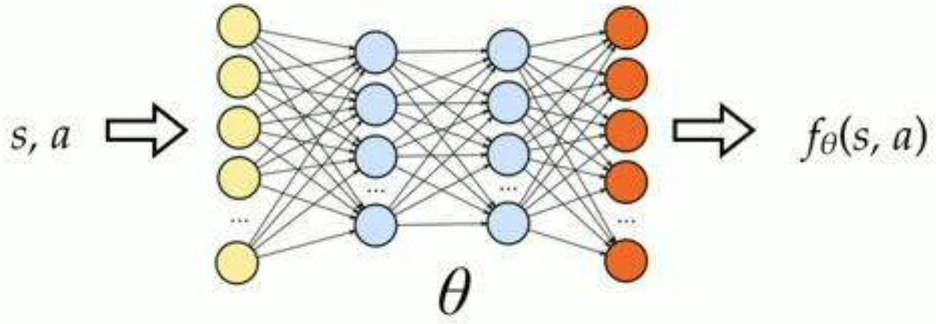




Example: Video game playing



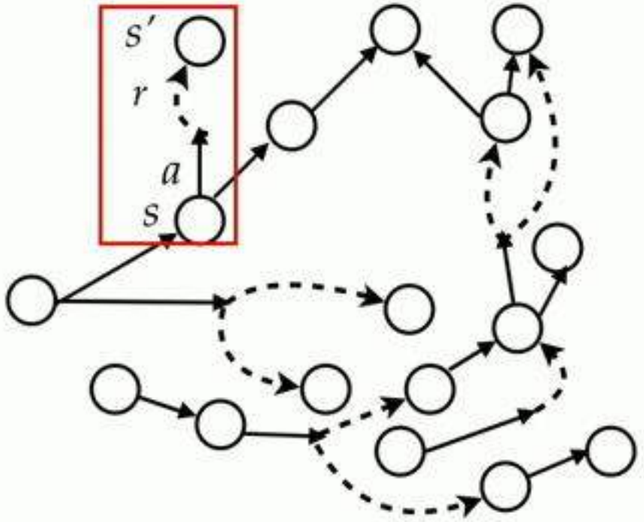


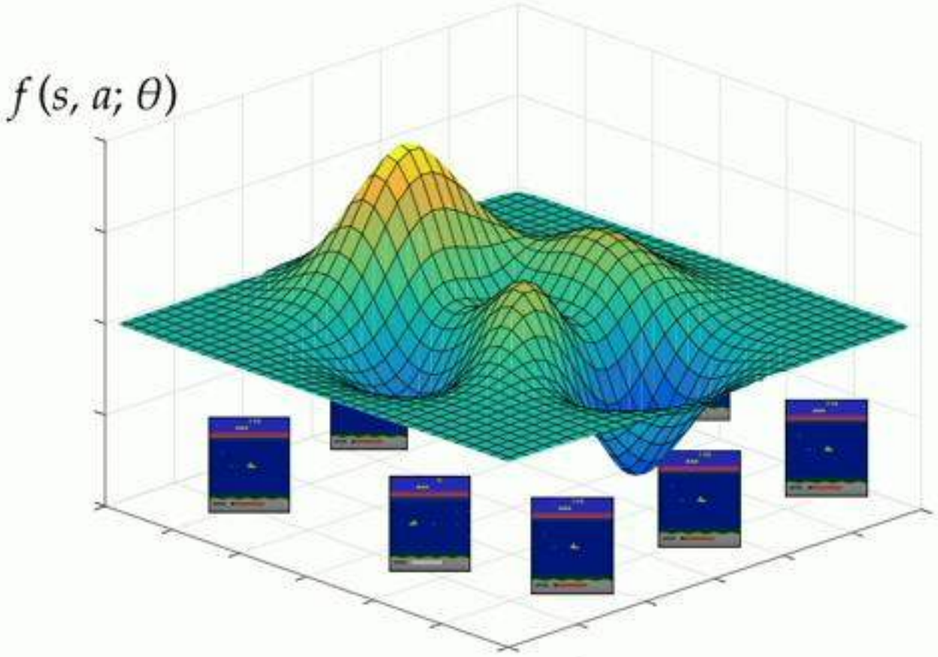


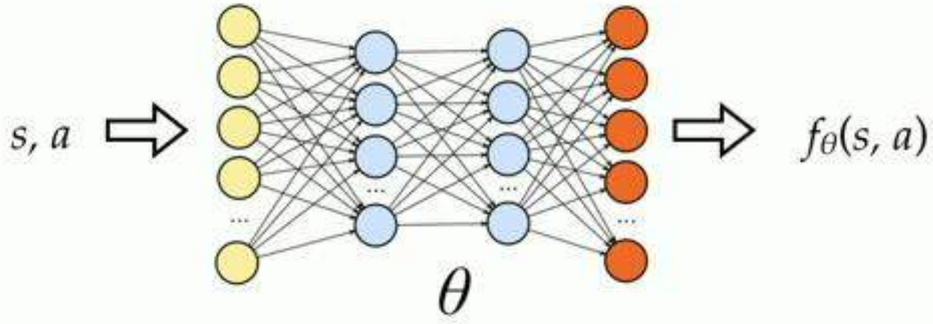
Find θ s.t.

$f_\theta \approx Q^*$

Example: Video game playing







Find θ s.t. $f_\theta(s, a) \approx \mathrm{E}[\, r + \gamma \max_{a'} f_\theta(s', a')\,] \quad \Rightarrow \quad f_\theta \approx Q^*$

...

Batch learning in large MDPs

- Dataset $D = \{(s, a, r, s')\}$
  - $(s, a) \sim \mu$ ("data distribution"), $r \sim R(s, a)$, $s' \sim P(s, a)$
- Function class F s.t. $Q^* \in F$
- Goal: find $f \approx Q^*$ s.t. its greedy policy is ε-optimal
  - Recall $v^\pi := \mathbb{E}[\sum_{h=1}^{\infty} \gamma^{h-1} r_h | s_1 \sim \nu_0; \pi]$

Batch learning in large MDPs

- Dataset $D = \{(s, a, r, s')\}$
  - $(s, a) \sim \mu$ ("data distribution"), $r \sim R(s, a)$, $s' \sim P(s, a)$
- Function class F s.t. $Q^* \in F$
- Goal: find $f \approx Q^*$ s.t. its greedy policy is ε-optimal
  - Recall $v^\pi := \mathbb{E}[\sum_{h=1}^{\infty} \gamma^{h-1} r_h | s_1 \sim \nu_0; \pi]$

Back to the earlier question:

Can we learn such f with a sample complexity poly($\log|F|, H, 1/\varepsilon, 1/\delta$)?

Batch learning in large MDPs

- Dataset $D = \{(s, a, r, s')\}$
  - $(s, a) \sim \mu$ ("data distribution"), $r \sim R(s, a)$, $s' \sim P(s, a)$
- Function class F s.t. $Q^* \in F$
- Goal: find $f \approx Q^*$ s.t. its greedy policy is ε-optimal
  - Recall $v^\pi := \mathbb{E}[\sum_{h=1}^{\infty} \gamma^{h-1} r_h | s_1 \sim \nu_0; \pi]$

Back to the earlier question:

> Can we learn such f with a sample complexity poly($\log|F|, H, 1/\varepsilon, 1/\delta$)?

Yes—with two more assumptions

Batch learning in large MDPs

- Dataset $D = \{(s, a, r, s')\}$
  - $(s, a) \sim \mu$ ("data distribution"), $r \sim R(s, a)$, $s' \sim P(s, a)$
- Function class F s.t. $Q^* \in F$
- Goal: find $f \approx Q^*$ s.t. its greedy policy is ε-optimal
  - Recall $v^\pi := \mathbb{E}[\sum_{h=1}^{\infty} \gamma^{h-1} r_h | s_1 \sim \nu_0; \pi]$

Back to the earlier question:

Can we learn such f with a sample complexity poly($\log|F|, H, 1/\varepsilon, 1/\delta$)?

Yes—with two more assumptions

- On data: No policy induces distribution "too different" from μ

Batch learning in large MDPs

- Dataset $D = \{(s, a, r, s')\}$
  - $(s, a) \sim \mu$ ("data distribution"), $r \sim R(s, a)$, $s' \sim P(s, a)$
- Function class F s.t. $Q^* \in F$
- Goal: find $f \approx Q^*$ s.t. its greedy policy is ε-optimal
  - Recall $v^\pi := \mathbb{E}[\sum_{h=1}^{\infty} \gamma^{h-1} r_h | s_1 \sim \nu_0; \pi]$

Back to the earlier question:

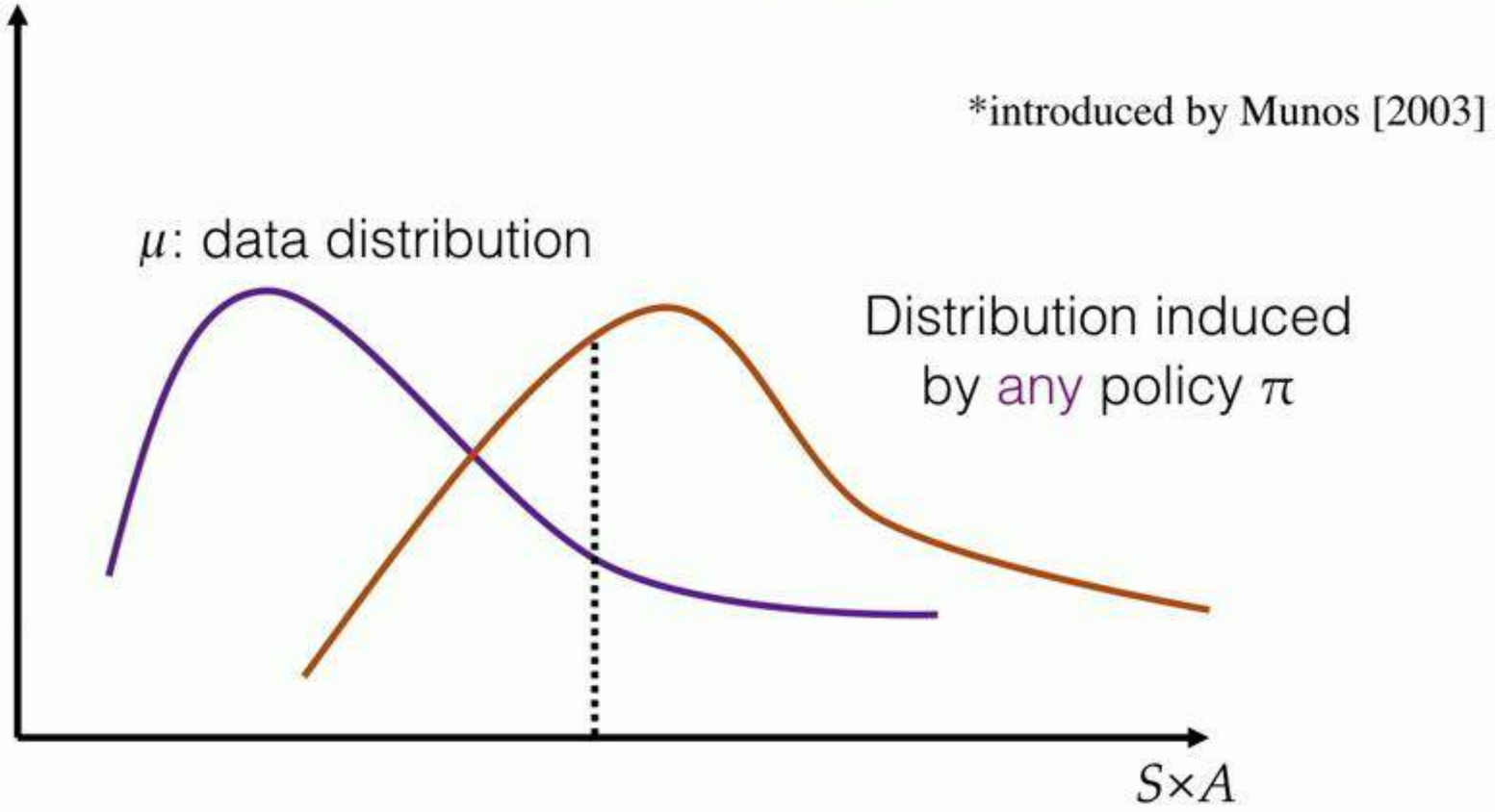
Can we learn such f with a sample complexity poly($\log|F|, H, 1/\varepsilon, 1/\delta$)?

Yes—with two more assumptions

- On data: No policy induces distribution "too different" from μ
- On function class: F is closed under Bellman update

Assumption on data

*introduced by Munos [2003]



Batch learning in large MDPs

- Dataset $D = \{(s, a, r, s')\}$
  - $(s, a) \sim \mu$ ("data distribution"), $r \sim R(s, a)$, $s' \sim P(s, a)$
- Function class F s.t. $Q^* \in F$
- Goal: find $f \approx Q^*$ s.t. its greedy policy is ε-optimal
  - Recall $v^\pi := \mathbb{E}[\sum_{h=1}^{\infty} \gamma^{h-1} r_h | s_1 \sim \nu_0; \pi]$

Back to the earlier question:

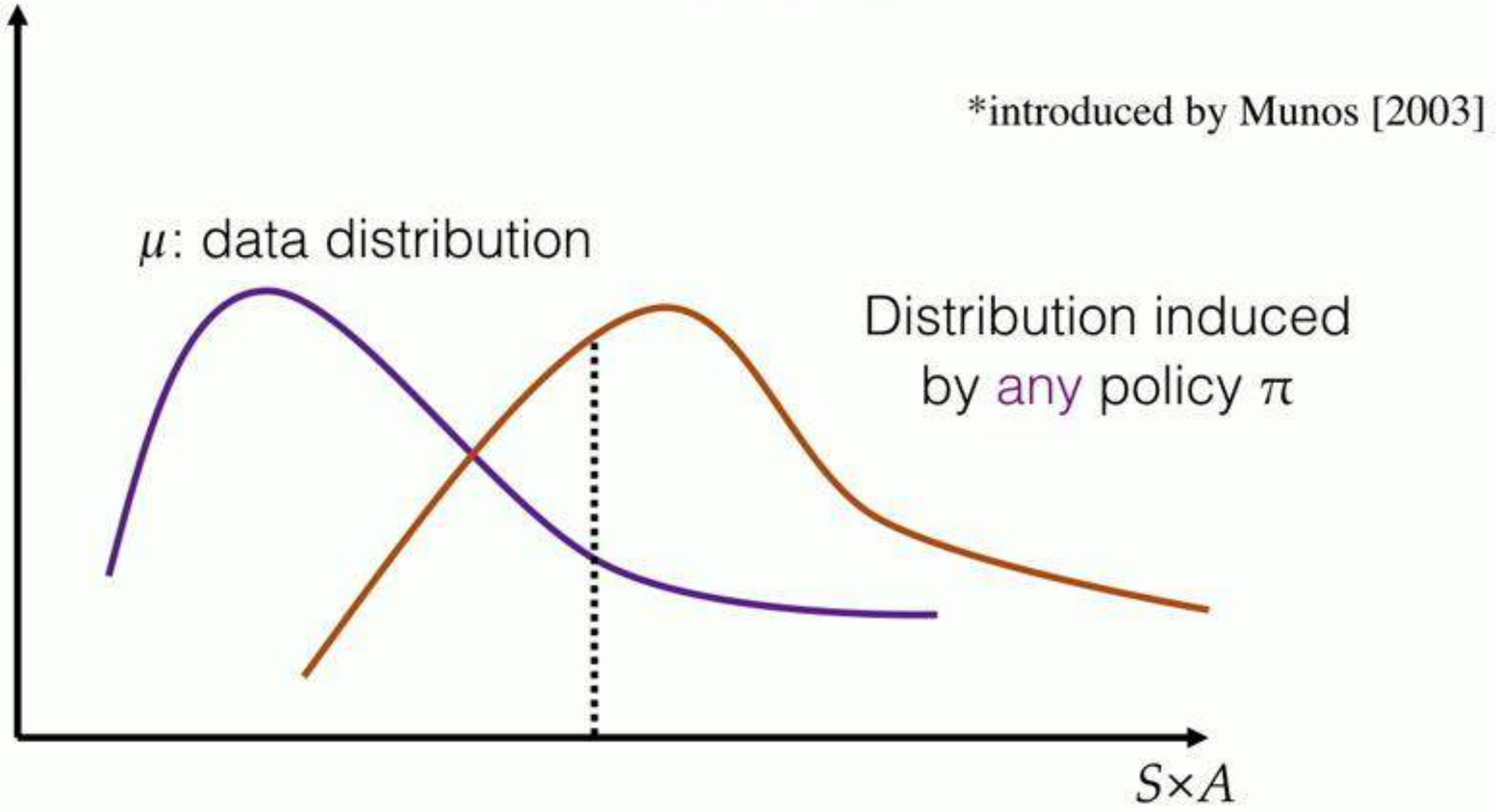
Can we learn such f with a sample complexity poly($\log|F|, H, 1/\varepsilon, 1/\delta$)?

Yes—with two more assumptions

- On data: No policy induces distribution "too different" from μ

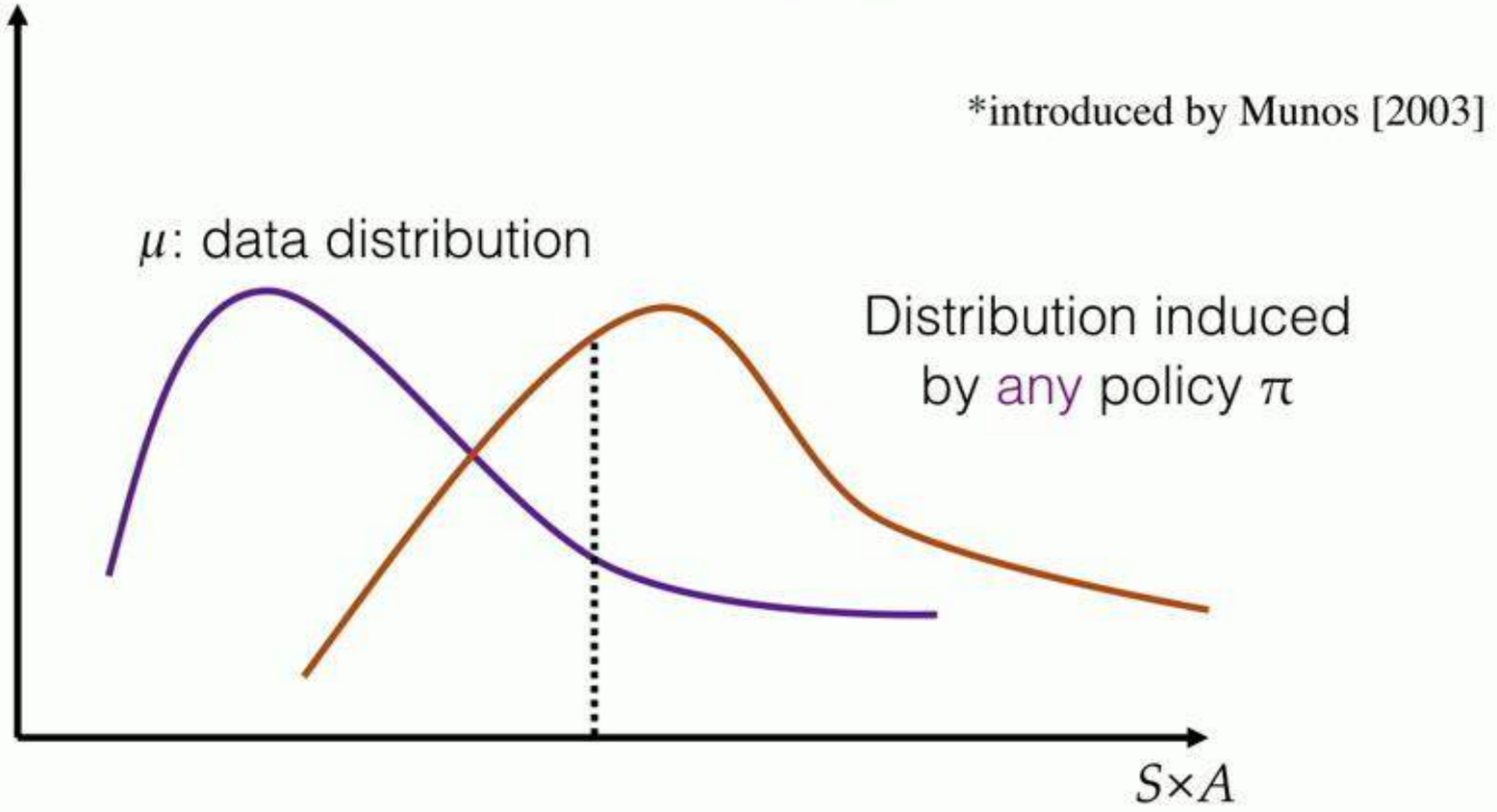
Assumption on data

*introduced by Munos [2003]



Assumption on data

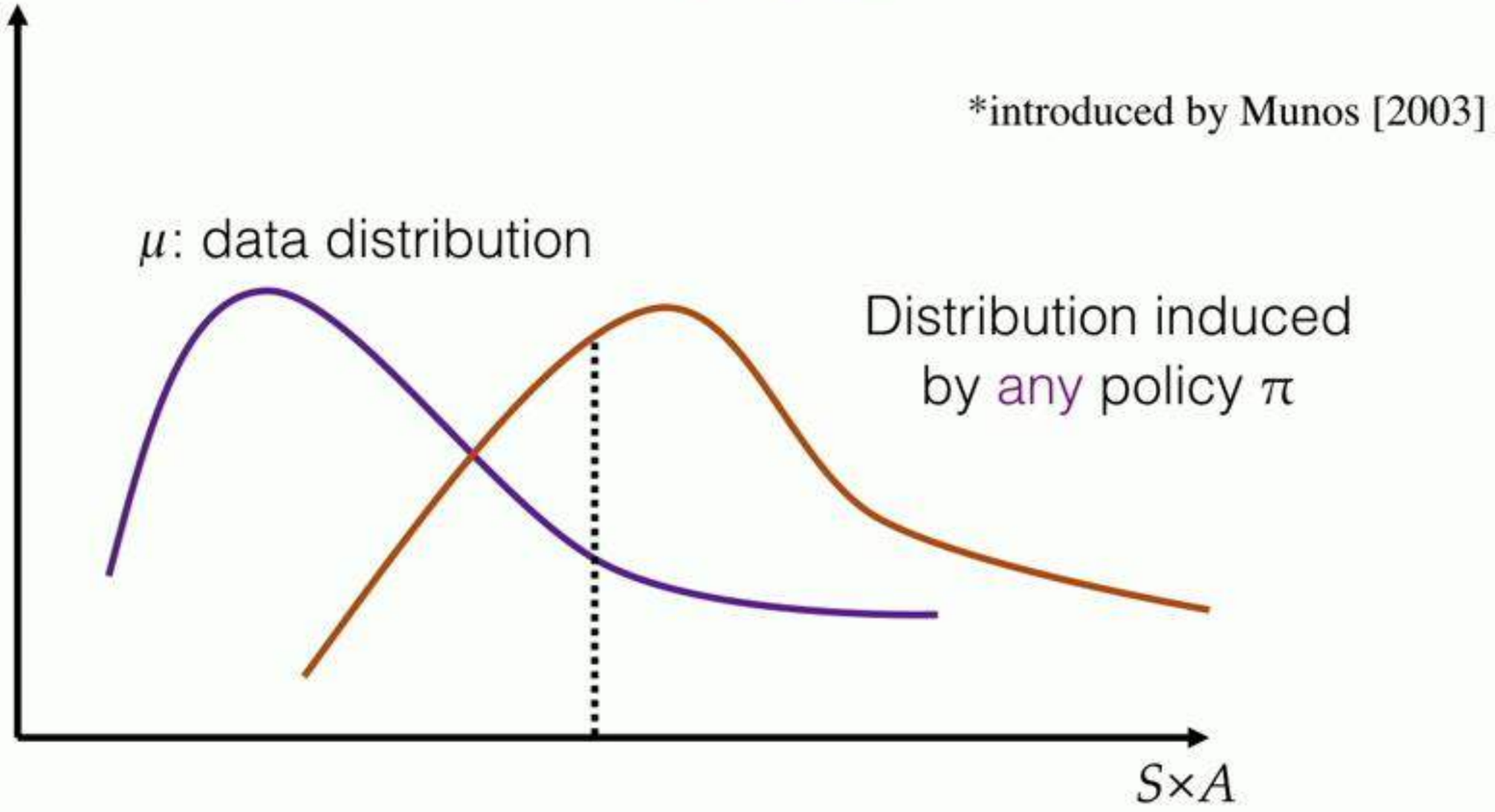
*introduced by Munos [2003]



- Let C be a uniform upper bound on the density ratio

Assumption on data

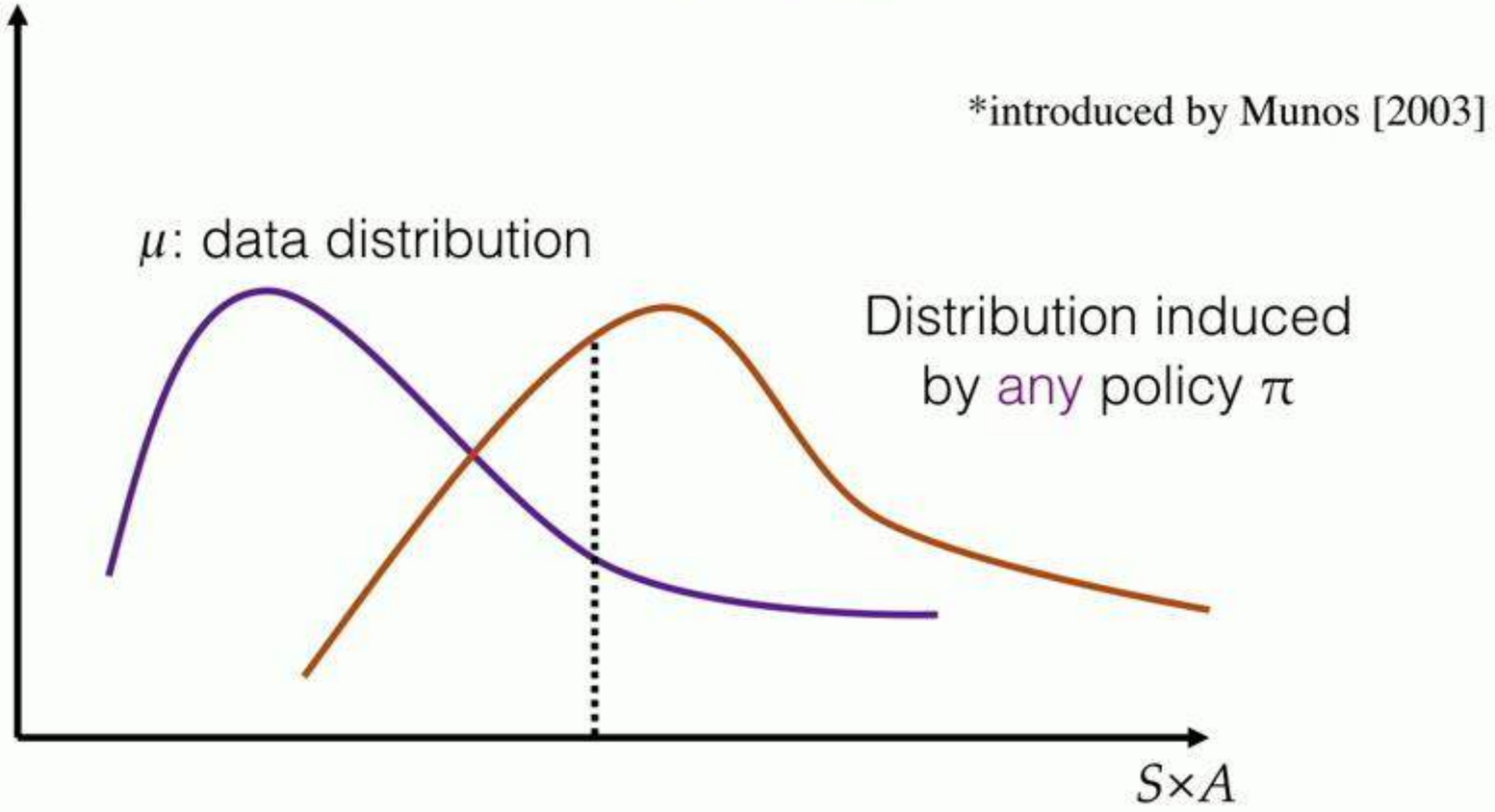
*introduced by Munos [2003]



- Let C be a uniform upper bound on the density ratio
- Assumption: C is small (= allow polynomial dependence on C)

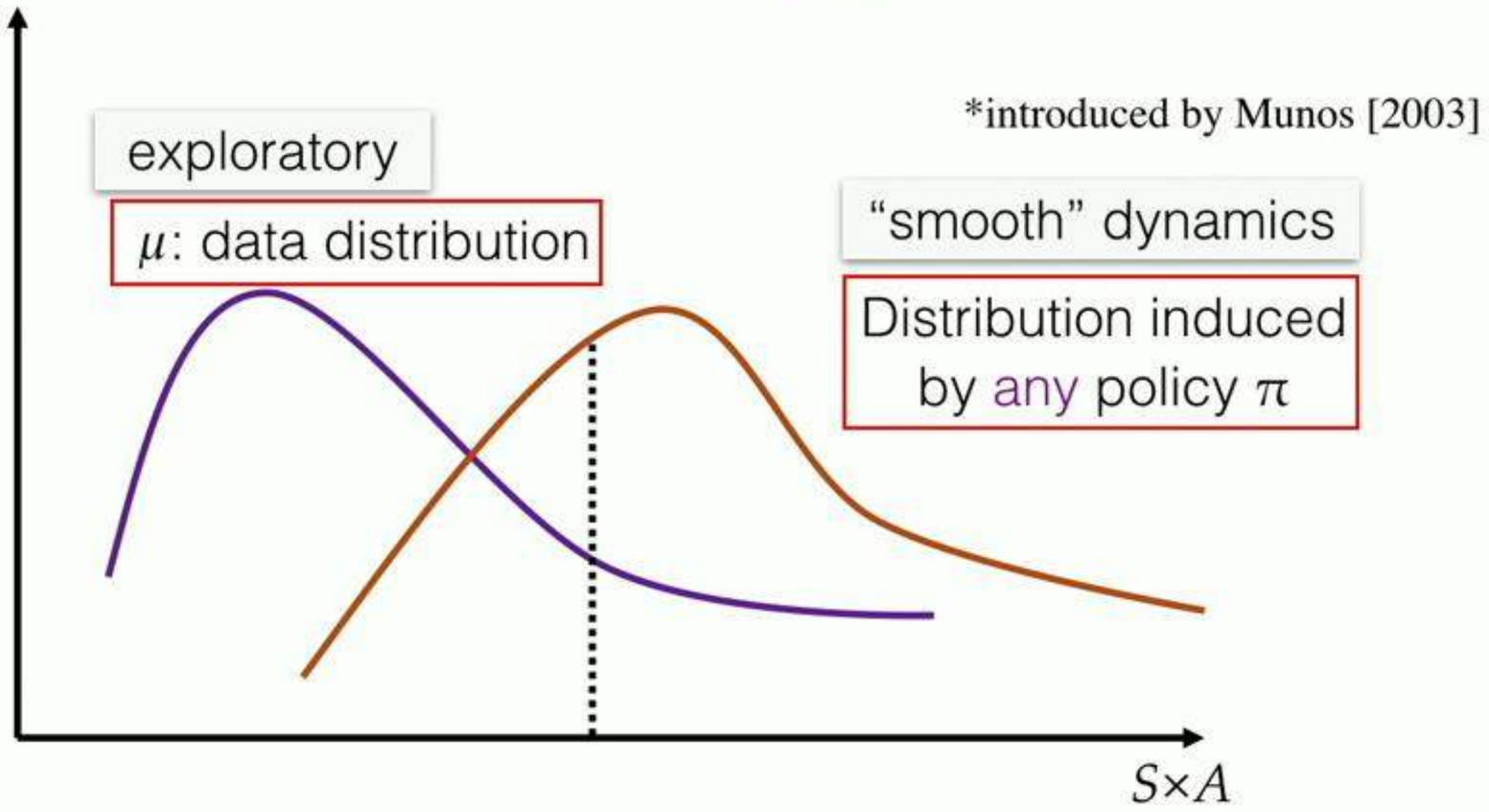
Assumption on data

*introduced by Munos [2003]



- Let C be a uniform upper bound on the density ratio
- Assumption: C is small (= allow polynomial dependence on C)
- When C is unbounded, exponential (in horizon) lower bound exists (see paper)

Assumption on data



*introduced by Munos [2003]

- Let C be a uniform upper bound on the density ratio
- Assumption: C is small (= allow polynomial dependence on C)
- When C is unbounded, exponential (in horizon) lower bound exists (see paper)

holds under the most exploratory data distribution!

Batch learning in large MDPs

- Dataset $D = \{(s, a, r, s')\}$
  - $(s, a) \sim \mu$ ("data distribution"), $r \sim R(s, a)$, $s' \sim P(s, a)$
- Function class F s.t. $Q^* \in F$ (finite)
- Goal: find $f \approx Q^*$ s.t. its greedy policy is ε-optimal
  - Recall $v^\pi := \mathbb{E}[\sum_{h=1}^{\infty} \gamma^{h-1} r_h | s_1 \sim \nu_0; \pi]$

Back to the earlier question:

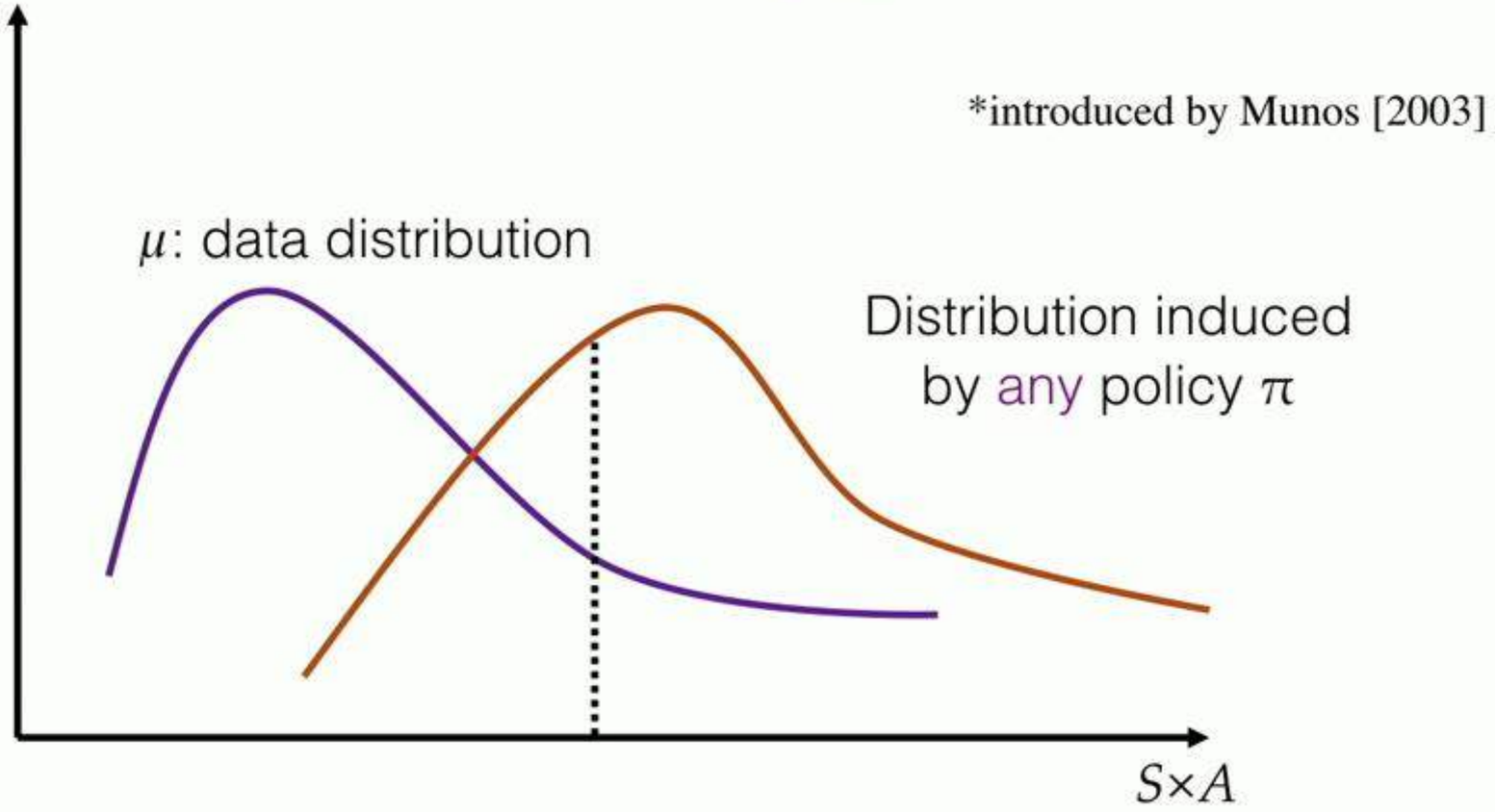
Can we learn such f with a sample complexity poly($\log|F|, H, 1/\varepsilon, 1/\delta$)?

Yes—with two more assumptions

- On data: No policy induces distribution "too different" from μ
- On function class: e.g., F is closed under Bellman update

Assumption on data

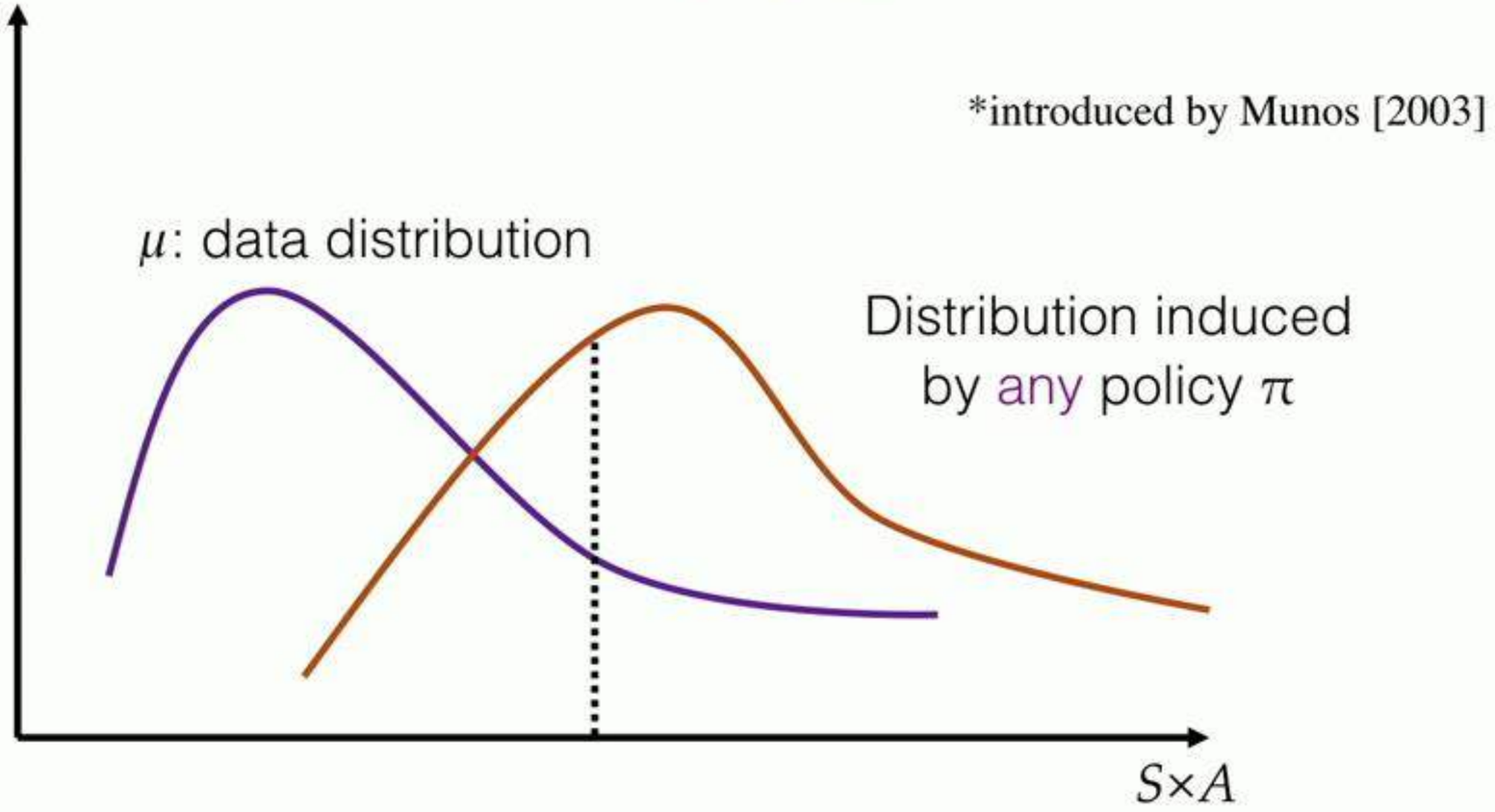
*introduced by Munos [2003]



- Let C be a uniform upper bound on the density ratio
- Assumption: C is small (= allow polynomial dependence on C)

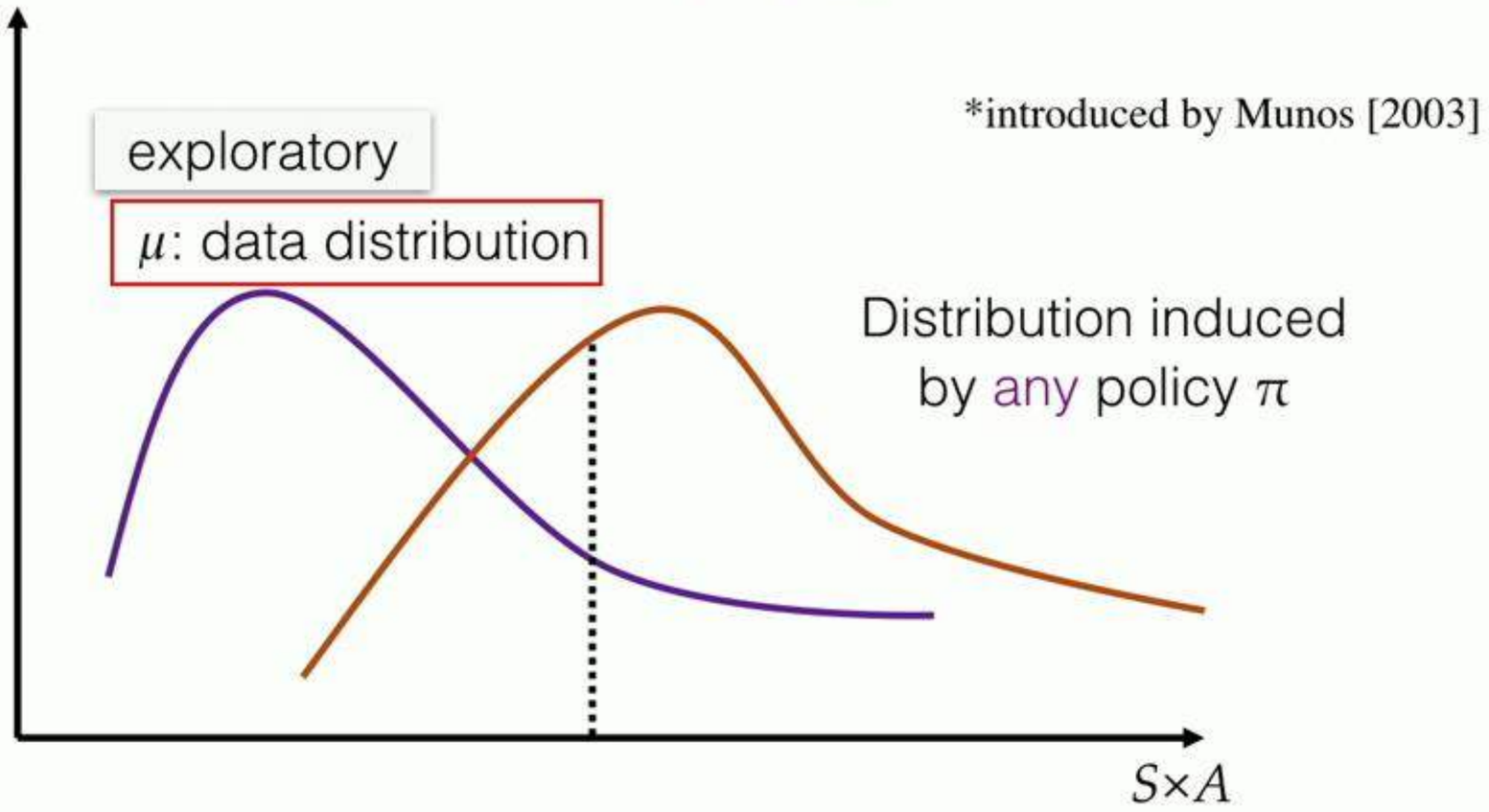
Assumption on data

*introduced by Munos [2003]



- Let C be a uniform upper bound on the density ratio
- Assumption: C is small (= allow polynomial dependence on C)
- When C is unbounded, exponential (in horizon) lower bound exists (see paper)

Assumption on data



*introduced by Munos [2003]

- Let C be a uniform upper bound on the density ratio
- Assumption: C is small (= allow polynomial dependence on C)
- When C is unbounded, exponential (in horizon) lower bound exists (see paper)

Batch learning in large MDPs

- Dataset $D = \{(s, a, r, s')\}$
  - $(s, a) \sim \mu$ ("data distribution"), $r \sim R(s, a)$, $s' \sim P(s, a)$
- Function class F s.t. $Q^* \in F$ (finite)
- Goal: find $f \approx Q^*$ s.t. its greedy policy is ε-optimal
  - Recall $v^\pi := \mathbb{E}[\sum_{h=1}^{\infty} \gamma^{h-1} r_h | s_1 \sim \nu_0; \pi]$

Back to the earlier question:

> Can we learn such f with a sample complexity poly($\log|F|, H, 1/\varepsilon, 1/\delta$)?

Yes—with two more assumptions

- On data: No policy induces distribution "too different" from μ
- On function class: e.g., F is closed under Bellman update

Algorithm for batch RL

Typical algorithm: Fitted Q-Iteration (FQI) [Ernst et al'05]

Algorithm for batch RL

Typical algorithm: Fitted Q-Iteration (FQI) [Ernst et al'05]

- Initialize $f_0 \in F$ arbitrarily

Algorithm for batch RL

Typical algorithm: Fitted Q-Iteration (FQI) [Ernst et al'05]

- Initialize $f_0 \in F$ arbitrarily
- In iteration t, convert D to least-square regression dataset

$$\left\{ \left((s,a), (r + \gamma \max_{a' \in \mathcal{A}} f_{t-1}(s', a')) \right) \right\}$$

Algorithm for batch RL

Typical algorithm: Fitted Q-Iteration (FQI) [Ernst et al'05]

- Initialize $f_0 \in F$ arbitrarily
- In iteration t, convert D to least-square regression dataset

$$\left\{ \left((s,a), (r + \gamma \max_{a' \in \mathcal{A}} f_{t-1}(s', a')) \right) \right\}$$

- ERM over F to obtain f_t , and repeat

Algorithm for batch RL

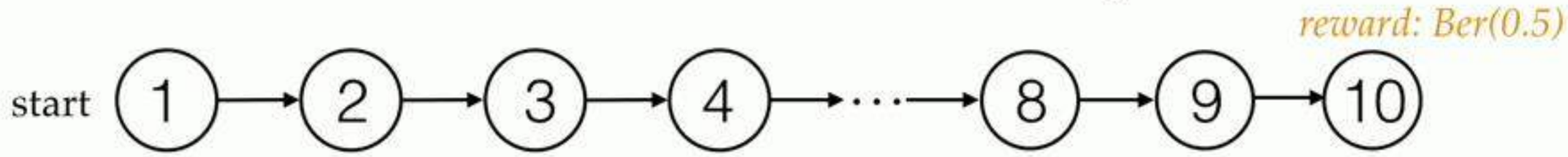
Typical algorithm: Fitted Q-Iteration (FQI) [Ernst et al'05]

- Initialize $f_0 \in F$ arbitrarily
- In iteration t, convert D to least-square regression dataset

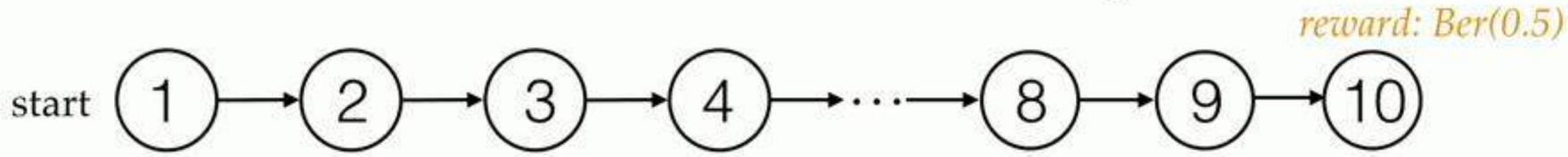
$$\left\{ \left((s,a), (r + \gamma \max_{a' \in \mathcal{A}} f_{t-1}(s', a')) \right) \right\}$$

- ERM over F to obtain f_t , and repeat
- That is, $f_t = \arg\min_{f \in \mathcal{F}} \sum_{(s,a,r,s') \in D} \left(f(s,a) - \left(r + \gamma \max_{a' \in \mathcal{A}} f_{t-1}(s', a') \right) \right)^2$

How FQI works (finite horizon, γ=1)

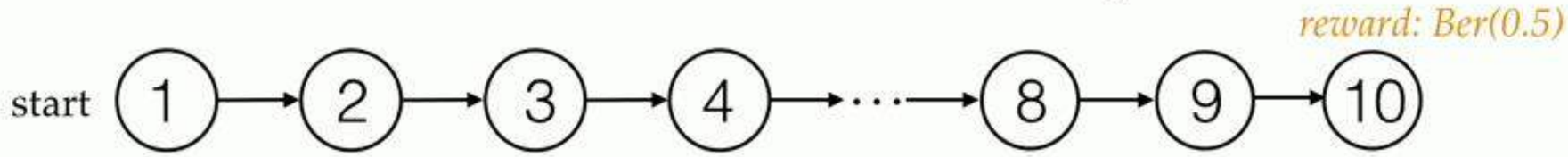


How FQI works (finite horizon, γ=1)



- Dataset $D = \{(s, r, s')\}$ looks like (action omitted):
  $\{(①, 0, ②), (②, 0, ③), \ldots, (⑩, 1, end), \ldots, (⑩, 0, end)\}$

How FQI works (finite horizon, γ=1)

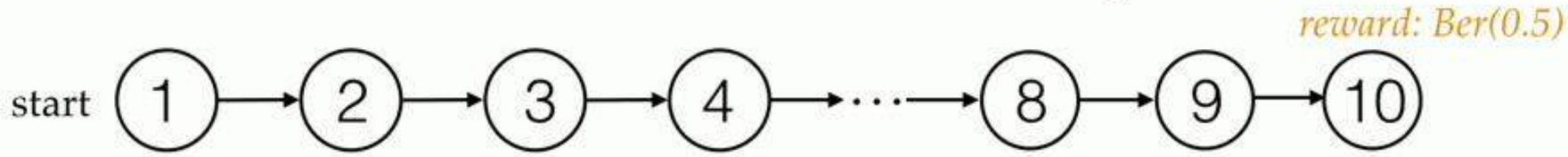


FQI Iter #1:

Data: (⑩, 1, *end*), …, (⑩, 0, *end*)

- Dataset $D = \{(s, r, s')\}$ looks like (action omitted):
  {(①, 0, ②), (②, 0,③), …, (⑩, 1, *end*), …, (⑩, 0, *end*)}

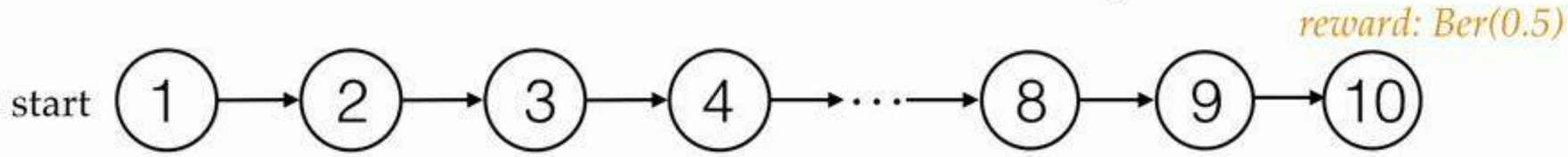
How FQI works (finite horizon, γ=1)



FQI Iter #1: Data: (⑩, 1, *end*), …, (⑩, 0, *end*) ⇨ 0.501

- Dataset $D = \{(s, r, s')\}$ looks like (action omitted):
  {(①, 0, ②), (②, 0,③), …, (⑩, 1, *end*), …, (⑩, 0, *end*)}

How FQI works (finite horizon, γ=1)

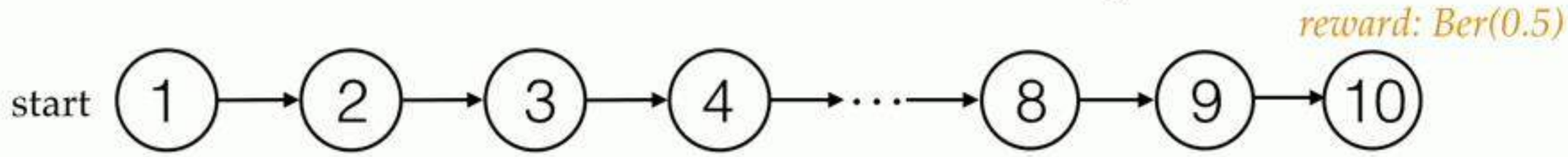


FQI Iter #1: Data: (⑩, 1, *end*), …, (⑩, 0, *end*) ⇨ 0.501

Iter #2: Data: (⑨, 0, ⑩)

- Dataset $D = \{(s, r, s')\}$ looks like (action omitted):
  {(①, 0, ②), (②, 0,③), …, (⑩, 1, *end*), …, (⑩, 0, *end*)}

How FQI works (finite horizon, γ=1)

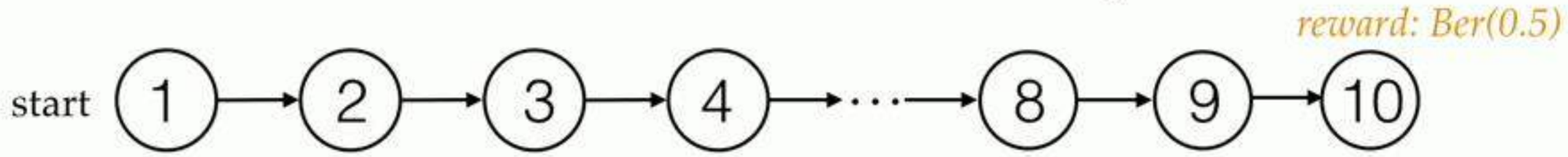


FQI Iter #1: Data: (⑩, 1, *end*), …, (⑩, 0, *end*) ⇨ 0.501

Iter #2: Data: (⑨, 0, ⑩) ⇨ (⑨, 0+0.501)

- Dataset $D = \{(s, r, s')\}$ looks like (action omitted):
  {(①, 0, ②), (②, 0,③), …, (⑩, 1, *end*), …, (⑩, 0, *end*)}

How FQI works (finite horizon, γ=1)

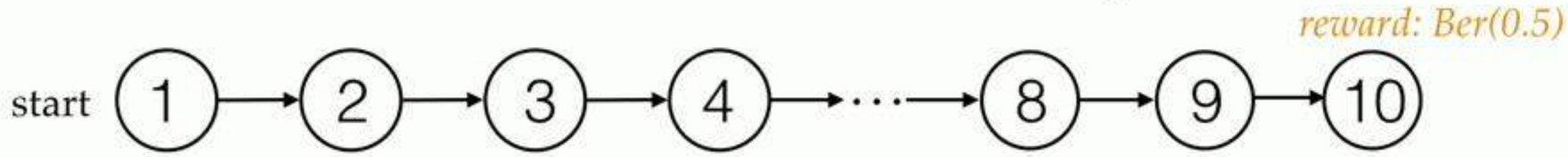


FQI Iter #1: Data: (⑩, 1, *end*), …, (⑩, 0, *end*) ⇨ 0.501

Iter #2: Data: (⑨, 0, ⑩) ⇨ (⑨, 0+0.501) ⇨ 0.501 0.501

- Dataset $D = \{(s, r, s')\}$ looks like (action omitted):
  {(①, 0, ②), (②, 0,③), …, (⑩, 1, *end*), …, (⑩, 0, *end*)}

How FQI works (finite horizon, γ=1)



FQI Iter #1: Data: (⑩, 1, *end*), …, (⑩, 0, *end*) ⇨ 0.501

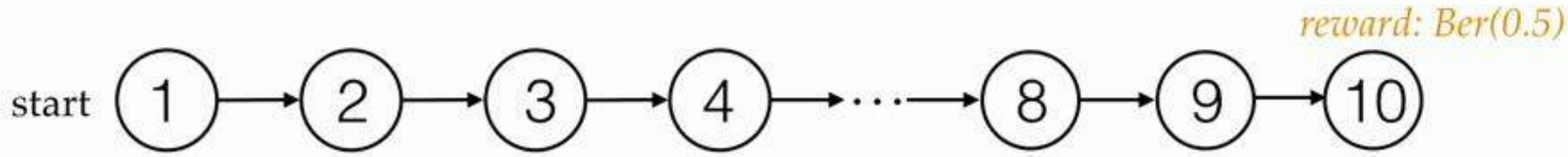
Iter #2: Data: (⑨, 0, ⑩) ⇨ (⑨, 0+0.501) ⇨ 0.501 0.501

…

Iter #10: 0.501 0.501 0.501 0.501 … 0.501 0.501 0.501

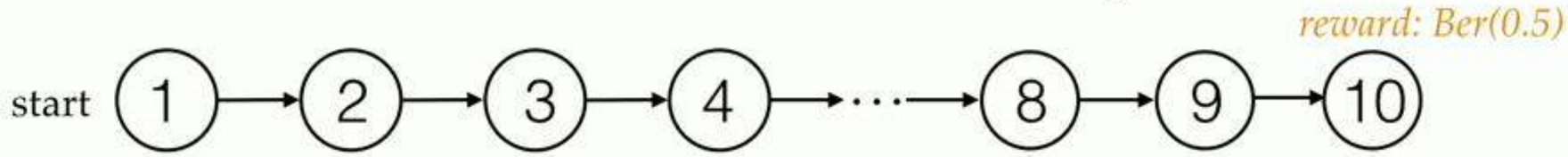
- Dataset $D = \{(s, r, s')\}$ looks like (action omitted):
  {(①, 0, ②), (②, 0,③), …, (⑩, 1, *end*), …, (⑩, 0, *end*)}

How things go wrong (w/ restricted class)

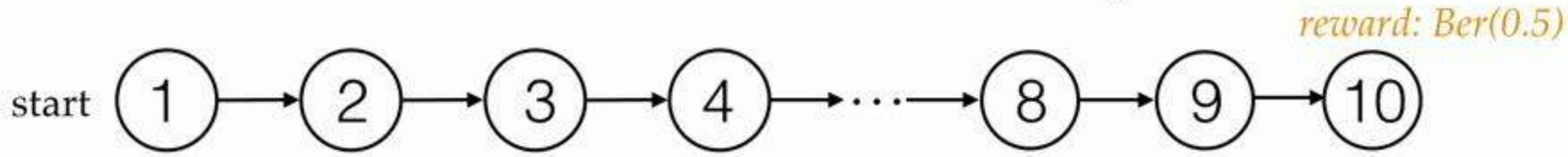


Function class

How FQI works (finite horizon, γ=1)



How FQI works (finite horizon, γ=1)



FQI Iter #1: Data: (⑩, 1, *end*), …, (⑩, 0, *end*) ⇨ 0.501

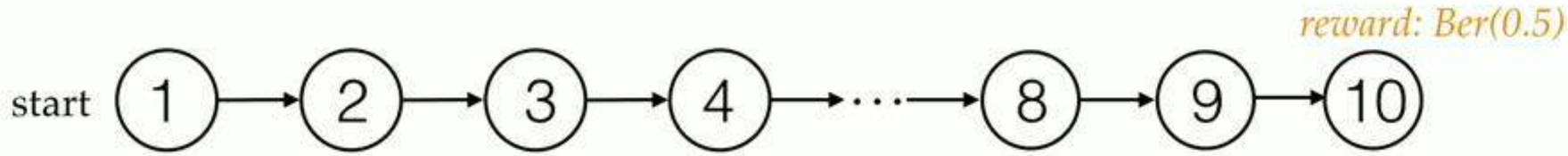
Iter #2: Data: (⑨, 0, ⑩) ⇨ (⑨, 0+0.501) ⇨ 0.501 0.501

…

Iter #10: 0.501 0.501 0.501 0.501 … 0.501 0.501 0.501

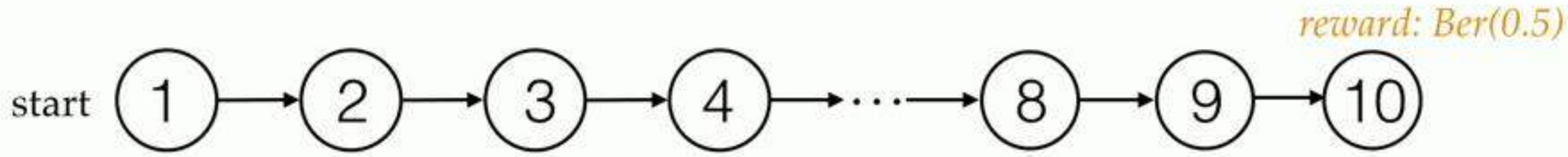
- Dataset $D = \{(s, r, s')\}$ looks like (action omitted):
  {(①, 0, ②), (②, 0,③), …, (⑩, 1, *end*), …, (⑩, 0, *end*)}

How things go wrong (w/ restricted class)



Function class

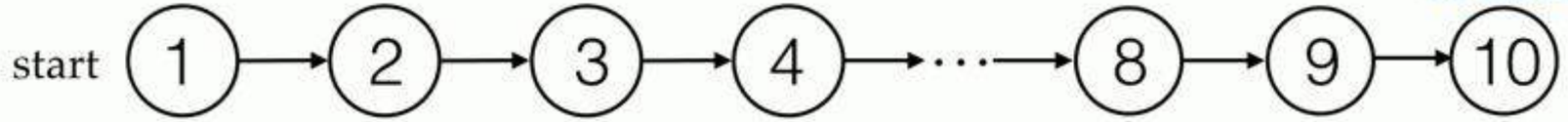
How things go wrong (w/ restricted class)



Function class	
	0.5
	0.501

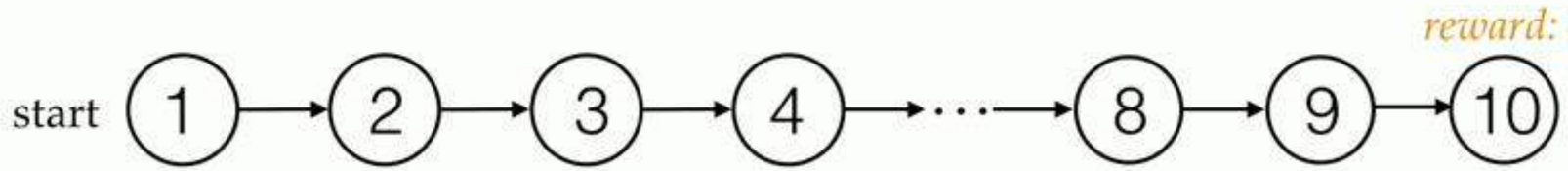
How things go wrong (w/ restricted class)

reward: Ber(0.5)



Function	0.5	×	0.5	×	0.5	×	…	×	0.5	×	0.5	×	0.5
class	1.012		0.756		0.628				0.504		0.502		0.501

How things go wrong (w/ restricted class)

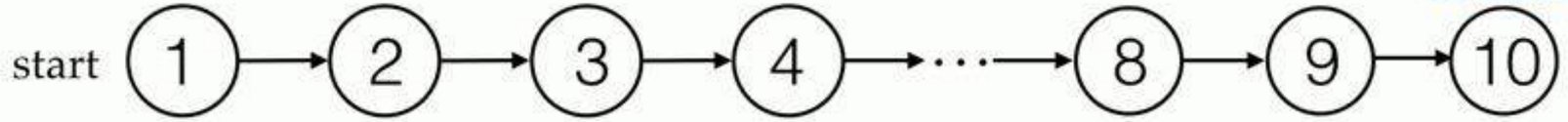


Function class	0.5	×	0.5	×	0.5	×	…	×	0.5	×	0.5	×	0.5
	1.012		0.756		0.628				0.504		0.502		0.501

FQI Iter #1: Data: (⑩, 1, *end*), …, (⑩, 0, *end*) ⇨ 0.501

How things go wrong (w/ restricted class)

reward: Ber(0.5)



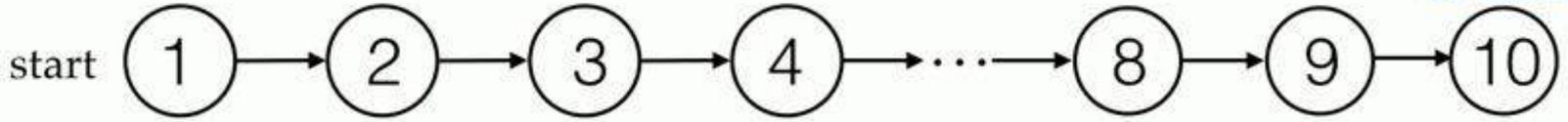
Function class	0.5 1.012	×	0.5 0.756	×	0.5 0.628	×	…	×	0.5 0.504	×	0.5 0.502	×	0.5 0.501

FQI Iter #1: Data: (⑩, 1, *end*), …, (⑩, 0, *end*) ⇨ 0.501

Iter #2: Data: (⑨, 0, ⑩) ⇨ (⑨, 0+0.501) ⇨ 0.501

How things go wrong (w/ restricted class)

reward: Ber(0.5)



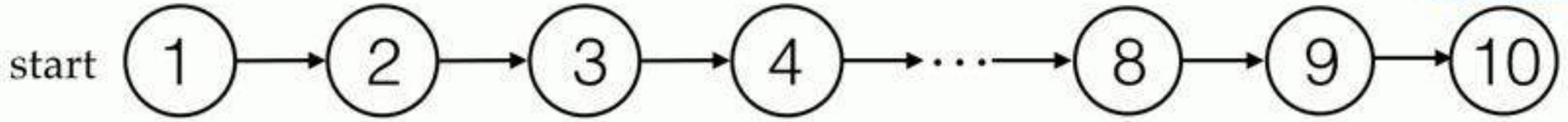
Function class	0.5 1.012	×	0.5 0.756	×	0.5 0.628	×	…	×	0.5 0.504	×	0.5 0.502	×	0.5 0.501

FQI Iter #1: Data: (⑩, 1, *end*), …, (⑩, 0, *end*) ⇨ 0.501

Iter #2: Data: (⑨, 0, ⑩) ⇨ (⑨, 0+0.501) ⇨ 0.502 0.501

How things go wrong (w/ restricted class)

reward: Ber(0.5)



Function class	0.5	×	0.5	×	0.5	×	…	×	0.5	×	0.5	×	0.5
	1.012		0.756		0.628				0.504		0.502		0.501

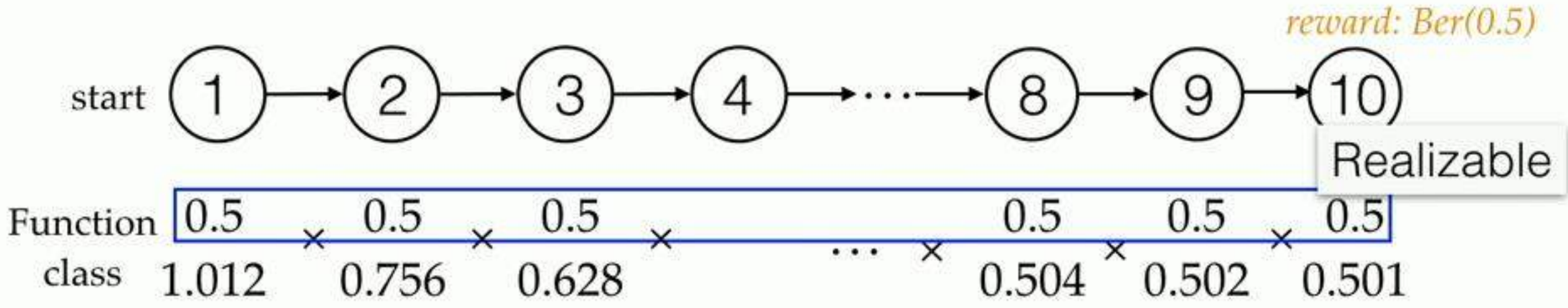
FQI Iter #1: Data: (⑩, 1, *end*), …, (⑩, 0, *end*) ⇨ 0.501

Iter #2: Data: (⑨, 0, ⑩) ⇨ (⑨, 0+0.501) ⇨ 0.502 0.501

…

Iter #10: 1.012 0.756 0.628 … 0.502 0.501

How things go wrong (w/ restricted class)



FQI Iter #1: Data: (⑩, 1, *end*), …, (⑩, 0, *end*) ⇨ 0.501

Iter #2: Data: (⑨, 0, ⑩) ⇨ (⑨, 0+0.501) ⇨ 0.502 0.501

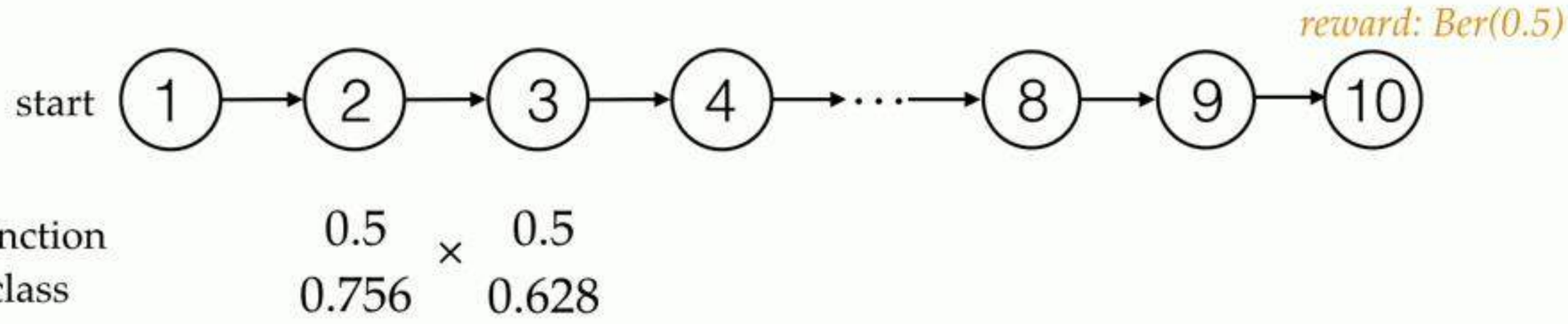
…

Iter #10: !!! 1.012 0.756 0.628 … 0.502 0.501

Example given in Dann et al'18

- similar conclusion known for decades (e.g., Van Roy'94)

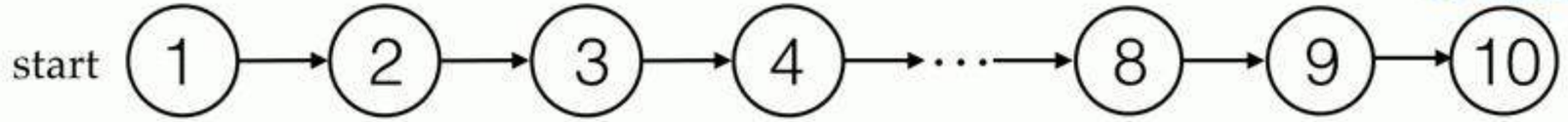
Fix using a strong assumption ("completeness")



Function class	0.5	×	0.5	
	0.756		0.628	

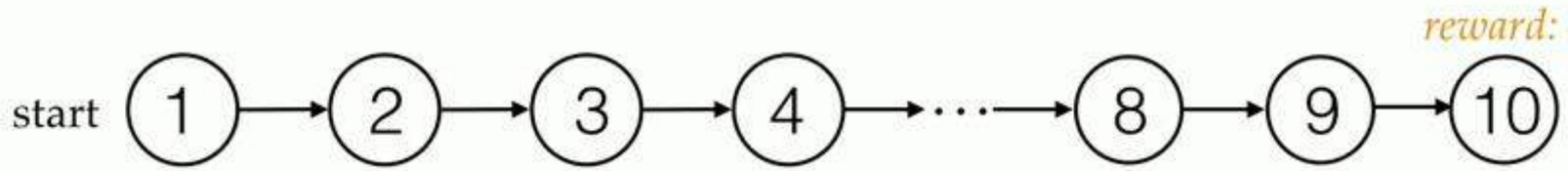
How things go wrong (w/ restricted class)

reward: Ber(0.5)



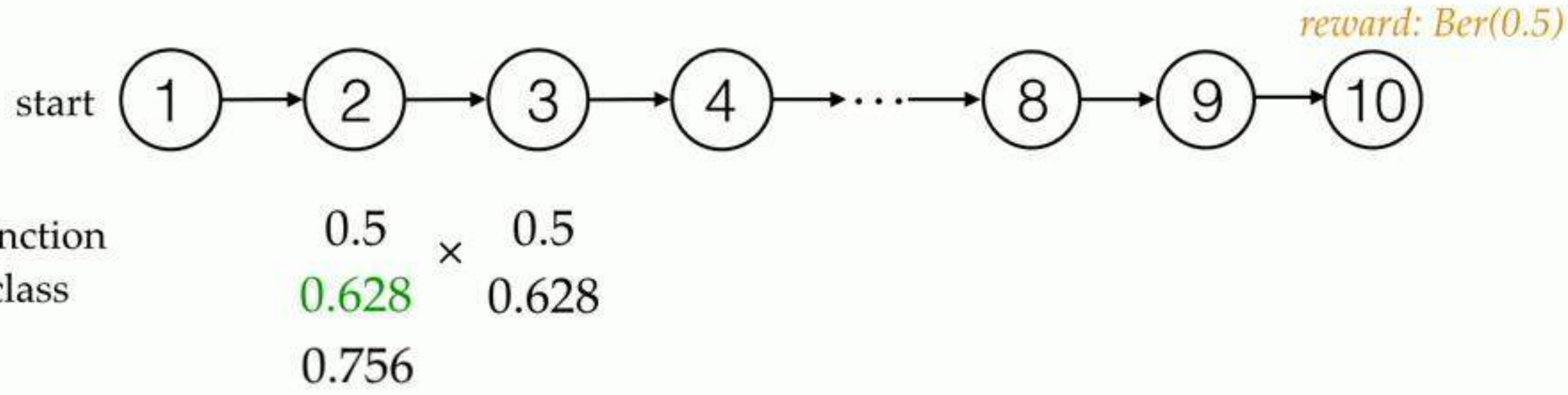
Function	0.5	×	0.5	×	0.5	×	…	×	0.5	×	0.5	×	0.5
class	1.012		0.756		0.628				0.504		0.502		0.501

Fix using a strong assumption ("completeness")

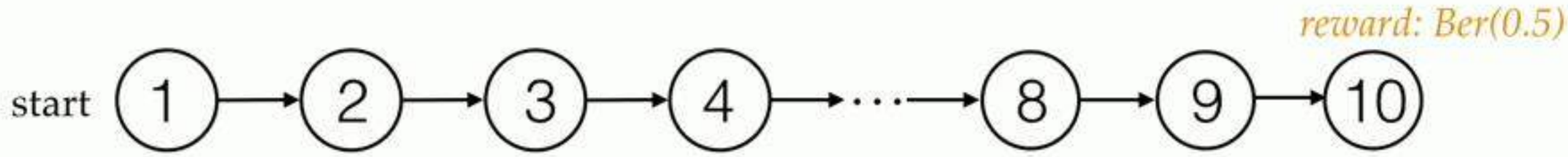


Function class	0.5	×	0.5
	0.756		0.628

Fix using a strong assumption ("completeness")



Fix using a strong assumption ("completeness")



Function class			
	0.5	×	0.5
	0.628		0.628
	0.756		

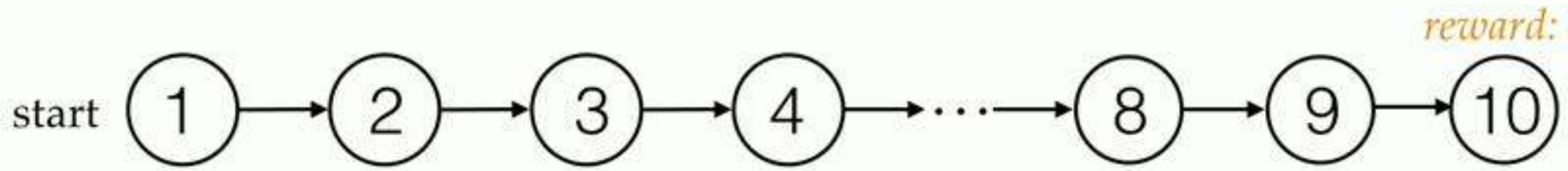
*introduced by Szepesvari & Munos [2005]

- More generally: the regression problem

$$\left\{\left((s,a),(r+\gamma\max_{a'\in\mathcal{A}} f_{t-1}(s',a'))\right)\right\}$$

should be realizable with F, for any $f_{t\text{-}1} \in F$

Fix using a strong assumption ("completeness")



Function class			
	0.5	×	0.5
	0.628		0.628
	0.756		

*introduced by Szepesvari & Munos [2005]

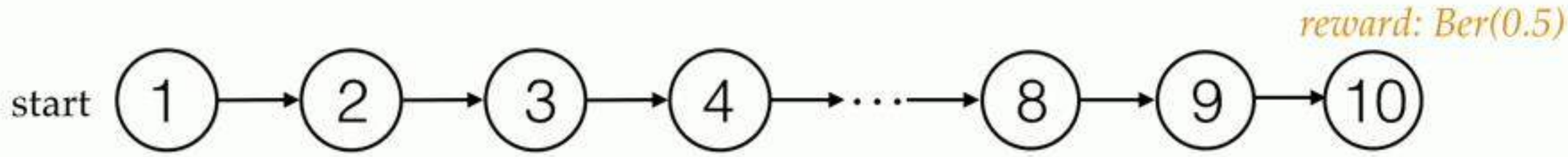
- More generally: the regression problem

$$\left\{\left((s,a),(r+\gamma\max_{a'\in\mathcal{A}} f_{t-1}(s',a'))\right)\right\}$$

  should be realizable with F, for any $f_{t\text{-}1} \in F$

- In finite-horizon setting: the richer function class you use at a lower level, the more difficult to satisfy realizability at higher level

Fix using a strong assumption ("completeness")



Function class

$$\begin{matrix} 0.5 \\ 0.628 \\ 0.756 \end{matrix} \times \begin{matrix} 0.5 \\ 0.628 \end{matrix}$$

*introduced by Szepesvari & Munos [2005]

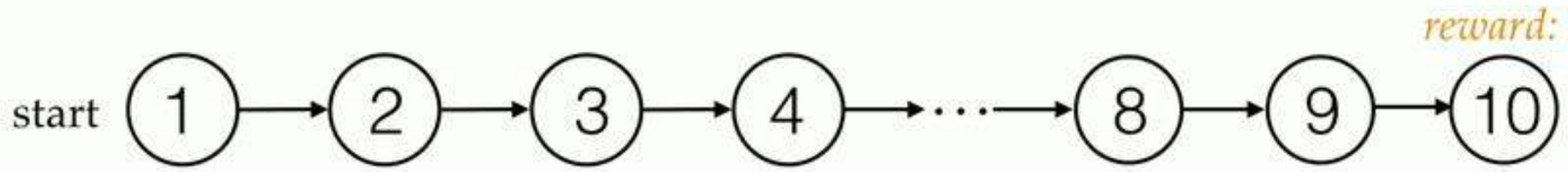
- More generally: the regression problem

$$\left\{\left((s,a),(r+\gamma \max_{a' \in \mathcal{A}} f_{t-1}(s',a'))\right)\right\}$$

  should be realizable with F, for any $f_{t\text{-}1} \in F$

- In finite-horizon setting: the richer function class you use at a lower level, the more difficult to satisfy realizability at higher level
- In discounted setting: F closed under Bellman update—adding functions can hurt approximation power

Fix using a strong assumption ("completeness")



Function class

0.5×0.5

$0.628 \quad 0.628$

0.756

*introduced by Szepesvari & Munos [2005]

- More generally: the regression problem

$$\left\{\left((s,a),(r+\gamma \max_{a' \in \mathcal{A}} f_{t-1}(s',a'))\right)\right\}$$

  should be realizable with F, for any $f_{t\text{-}1} \in F$

- In finite-horizon setting: the richer function class you use at a lower level, the more difficult to satisfy realizability at higher level
- In discounted setting: F closed under Bellman update—adding functions can hurt approximation power
- With this assumption, sample complexity *poly*$(\log|F|, H, 1/\varepsilon, 1/\delta, C)$

Realizability alone is insufficient?

- Most known alg: dynamic programming + least-sq regression —subject to the counterexample

Realizability alone is insufficient?

- Most known alg: dynamic programming + least-sq regression —subject to the counterexample
- Additional assumptions: F is closed under Bellman update, projected Bellman update is a contraction, density models for state distributions, 2 samples from exactly the same (s,a), etc.

Realizability alone is insufficient?

- Most known alg: dynamic programming + least-sq regression —subject to the counterexample
- Additional assumptions: F is closed under Bellman update, projected Bellman update is a contraction, density models for state distributions, 2 samples from exactly the same (s,a), etc.
  - (all of them satisfied in tabular setting)

Realizability alone is insufficient?

- Most known alg: dynamic programming + least-sq regression —subject to the counterexample
- Additional assumptions: F is closed under Bellman update, projected Bellman update is a contraction, density models for state distributions, 2 samples from exactly the same (s,a), etc.
  - (all of them satisfied in tabular setting)
- Suggest: realizability is not enough.

Realizability alone is insufficient?

- Most known alg: dynamic programming + least-sq regression —subject to the counterexample
- Additional assumptions: F is closed under Bellman update, projected Bellman update is a contraction, density models for state distributions, 2 samples from exactly the same (s,a), etc.
  - (all of them satisfied in tabular setting)
- Suggest: realizability is not enough.
- Obvious, right? info-theoretic lower bound?

Realizability alone is insufficient?

- Most known alg: dynamic programming + least-sq regression —subject to the counterexample
- Additional assumptions: F is closed under Bellman update, projected Bellman update is a contraction, density models for state distributions, 2 samples from exactly the same (s,a), etc.
  - (all of them satisfied in tabular setting)
- Suggest: realizability is not enough.
- Obvious, right? info-theoretic lower bound?
- Conjecture: given F that is realizable, there exists (a family of) problems where the worst-case sample complexity of any algorithm cannot be $poly(\log|F|, H, 1/\varepsilon, 1/\delta, C)$

Realizability alone is insufficient?

- Most known alg: dynamic programming + least-sq regression —subject to the counterexample
- Additional assumptions: F is closed under Bellman update, projected Bellman update is a contraction, density models for state distributions, 2 samples from exactly the same (s,a), etc.
  - (all of them satisfied in tabular setting)
- Suggest: realizability is not enough.
- Obvious, right? info-theoretic lower bound?
- Conjecture: given F that is realizable, there exists (a family of) problems where the worst-case sample complexity of any algorithm cannot be $poly(\log|F|, H, 1/\varepsilon, 1/\delta, C)$
  - Will get back to its importance at the end

Proving the conjecture: Attempt 1

- Let $\mathcal{M}$ be a family of MDP instances
  - will choose one for any given algorithm adversarially

Realizability alone is insufficient?

- Most known alg: dynamic programming + least-sq regression —subject to the counterexample
- Additional assumptions: F is closed under Bellman update, projected Bellman update is a contraction, density models for state distributions, 2 samples from exactly the same (s,a), etc.
  - (all of them satisfied in tabular setting)
- Suggest: realizability is not enough.
- Obvious, right? info-theoretic lower bound?
- Conjecture: given F that is realizable, there exists (a family of) problems where the worst-case sample complexity of any algorithm cannot be $poly(\log|F|, H, 1/\varepsilon, 1/\delta, C)$

Proving the conjecture: Attempt 1

- Let $\mathcal{M}$ be a family of MDP instances
  - will choose one for any given algorithm adversarially

Proving the conjecture: Attempt 1

- Let $\mathcal{M}$ be a family of MDP instances
  - will choose one for any given algorithm adversarially
- Natural F: $\{Q^*_M : M \in \mathcal{M}\}$
  - Assume no collapse: $|F| = |\mathcal{M}|$

Proving the conjecture: Attempt 1

- Let $\mathcal{M}$ be a family of MDP instances
  - will choose one for any given algorithm adversarially
- Natural F: $\{Q^*_M : M \in \mathcal{M}\}$
  - Assume no collapse: $|F| = |\mathcal{M}|$
- w.t.s. $poly(\log|F|, H, 1/\varepsilon, 1/\delta, C)$ is not achievable

Proving the conjecture: Attempt 1

- Let $\mathcal{M}$ be a family of MDP instances
  - will choose one for any given algorithm adversarially
- Natural F: $\{Q^*_M : M \in \mathcal{M}\}$
  - Assume no collapse: $|F| = |\mathcal{M}|$
- w.t.s. $poly(\log|F|, H, 1/\varepsilon, 1/\delta, C)$ is not achievable
- Already failed!

Proving the conjecture: Attempt 1

- Let $\mathcal{M}$ be a family of MDP instances
  - will choose one for any given algorithm adversarially
- Natural F: $\{Q^*_M : M \in \mathcal{M}\}$
  - Assume no collapse: $|F| = |\mathcal{M}|$
- w.t.s. $poly(\log|F|, H, 1/\varepsilon, 1/\delta, C)$ is not achievable
- Already failed!

There exists info-theoretic algorithm that achieves $poly(\log|\mathcal{M}|, H, 1/\varepsilon, 1/\delta, C)$

Proving the conjecture: Attempt 1

- Let $\mathcal{M}$ be a family of MDP instances
  - will choose one for any given algorithm adversarially
- Natural F: $\{Q^*_M : M \in \mathcal{M}\}$
  - Assume no collapse: $|F| = |\mathcal{M}|$
- w.t.s. $poly(\log|F|, H, 1/\varepsilon, 1/\delta, C)$ is not achievable
- Already failed!

  There exists info-theoretic algorithm that achieves $poly(\log|\mathcal{M}|, H, 1/\varepsilon, 1/\delta, C)$

  - Roughly: can create F' that is "slightly bigger" than F yet closed under Bellman update (of any MDP in family)

Proving the conjecture: Attempt 1

- Let $\mathcal{M}$ be a family of MDP instances
  - will choose one for any given algorithm adversarially
- Natural F: $\{Q^*_M : M \in \mathcal{M}\}$
  - Assume no collapse: $|F| = |\mathcal{M}|$
- w.t.s. $poly(\log|F|, H, 1/\varepsilon, 1/\delta, C)$ is not achievable
- Already failed!

  There exists info-theoretic algorithm that achieves $poly(\log|\mathcal{M}|, H, 1/\varepsilon, 1/\delta, C)$

  - Roughly: can create F' that is "slightly bigger" than F yet closed under Bellman update (of any MDP in family)
- Caveat: cannot prevent an info-theoretic learner from being model-based

Proving the conjecture: Attempt 2

- Let $\mathcal{M}$ be a family of MDP instances
- Natural F: $\{Q_M^*: M \in \mathcal{M}\}$

Proving the conjecture: Attempt 2

- Let $\mathcal{M}$ be a family of MDP instances
- Natural F: $\{Q_M^*: M \in \mathcal{M}\}$
- Want to prevent the learner from fully leveraging $\mathcal{M}$
- Mask the data [Sutton & Barto'18; Wen et al'19]
  - Original data: $\{(s, a, r, s')\}$

Proving the conjecture: Attempt 2

- Let $\mathcal{M}$ be a family of MDP instances
- Natural F: $\{Q_M^*: M \in \mathcal{M}\}$
- Want to prevent the learner from fully leveraging $\mathcal{M}$
- Mask the data [Sutton & Barto'18; Wen et al'19]
  - Original data: $\{(s, a, r, s')\}$
  - Now: learner cannot see s (or s'), but instead $\{f(s,a)\}_{f\in\mathcal{F},a\in\mathcal{A}}$

"value-profile"

Proving the conjecture: Attempt 2

- Let $\mathcal{M}$ be a family of MDP instances
- Natural F: $\{Q_M^*: M \in \mathcal{M}\}$
- Want to prevent the learner from fully leveraging $\mathcal{M}$
- Mask the data [Sutton & Barto'18; Wen et al'19]
  - Original data: $\{(s, a, r, s')\}$
  - Now: learner cannot see s (or s'), but instead $\{f(s,a)\}_{f\in\mathcal{F},a\in\mathcal{A}}$
  - All known value-based algs implementable thru this

"value-profile"

Proving the conjecture: Attempt 2

- Let $\mathcal{M}$ be a family of MDP instances
- Natural F: $\{Q_M^*: M \in \mathcal{M}\}$
- Want to prevent the learner from fully leveraging $\mathcal{M}$
- Mask the data [Sutton & Barto'18; Wen et al'19]
  - Original data: $\{(s, a, r, s')\}$
  - Now: learner cannot see s (or s'), but instead $\{f(s,a)\}_{f\in\mathcal{F},a\in\mathcal{A}}$
  - All known value-based algs implementable thru this

"value-profile"

- Sutton & Barto: Even with small state space ("tabular") & inf data, cannot learn Bellman error if function class not realizable

Proving the conjecture: Attempt 2

- Let $\mathcal{M}$ be a family of MDP instances
- Natural F: $\{Q_M^*: M \in \mathcal{M}\}$
- Want to prevent the learner from fully leveraging $\mathcal{M}$
- Mask the data [Sutton & Barto'18; Wen et al'19]
  - Original data: $\{(s, a, r, s')\}$
  - Now: learner cannot see s (or s'), but instead $\{f(s,a)\}_{f\in\mathcal{F},a\in\mathcal{A}}$
  - All known value-based algs implementable thru this

"value-profile"

- Sutton & Barto: Even with small state space ("tabular") & inf data, cannot learn Bellman error if function class not realizable
- We proved: in tabular setting, poly sample complexity under realizability

Checklist for a plausible construction

- Cannot be tabular

Checklist for a plausible construction

- Cannot be tabular
- Cannot be uncontrolled (Monte-Carlo works)
- Cannot be deterministic (Bellman Residual Minimization works)

Checklist for a plausible construction

- Cannot be tabular
- Cannot be uncontrolled (Monte-Carlo works)
- Cannot be deterministic (Bellman Residual Minimization works)

$$\arg\min_{f \in \mathcal{F}} \sum_{(s,a,r,s') \in D} \Big(f(s,a) - (r + \gamma \max_{a'} f(s',a')) \Big)^2$$

don't block gradient here;
1 line code change from DQN

Checklist for a plausible construction

- Cannot be tabular
- Cannot be uncontrolled (Monte-Carlo works)
- Cannot be deterministic (Bellman Residual Minimization works)

$$\arg\min_{f \in \mathcal{F}} \sum_{(s,a,r,s') \in D} \left(f(s,a) - (r + \gamma \max_{a'} f(s',a')) \right)^2$$

don't block gradient here;
1 line code change from DQN

DQN

BRM

Work in submission with Ehsan Saleh (UIUC)

Checklist for a plausible construction

- Cannot be tabular
- Cannot be uncontrolled (Monte-Carlo works)
- Cannot be deterministic (Bellman Residual Minimization works)

$$\arg\min_{f \in \mathcal{F}} \sum_{(s,a,r,s') \in D} \left(f(s,a) - (r + \gamma \max_{a'} f(s',a')) \right)^2$$

don't block gradient here;
1 line code change from DQN

DQN

BRM

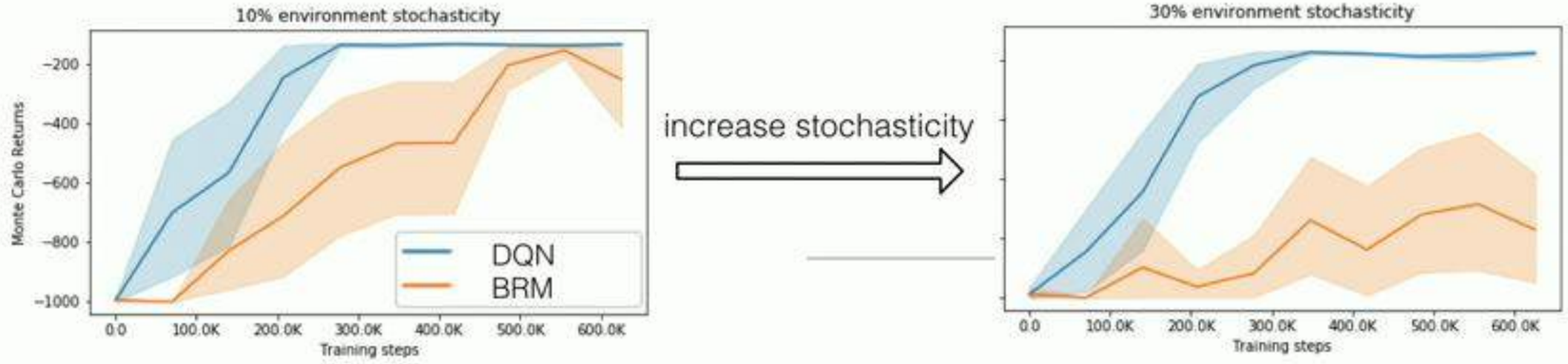
Work in submission with Ehsan Saleh (UIUC)

Checklist for a plausible construction

- Cannot be tabular
- Cannot be uncontrolled (Monte-Carlo works)
- Cannot be deterministic (Bellman Residual Minimization works)

$$\arg\min_{f \in \mathcal{F}} \sum_{(s,a,r,s') \in D} \left(f(s,a) - (r + \gamma \max_{a'} f(s',a')) \right)^2$$

don't block gradient here;
1 line code change from DQN



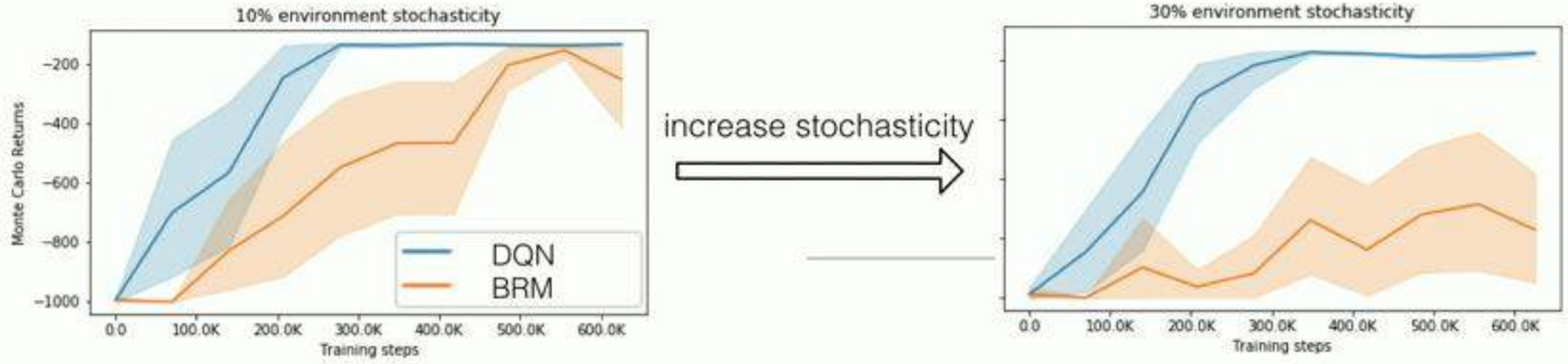
Work in submission with Ehsan Saleh (UIUC)

Checklist for a plausible construction

- Cannot be tabular
- Cannot be uncontrolled (Monte-Carlo works)
- Cannot be deterministic (Bellman Residual Minimization works)

$$\arg\min_{f \in \mathcal{F}} \sum_{(s,a,r,s') \in D} \left(f(s,a) - (r + \gamma \max_{a'} f(s',a')) \right)^2$$

don't block gradient here;
1 line code change from DQN



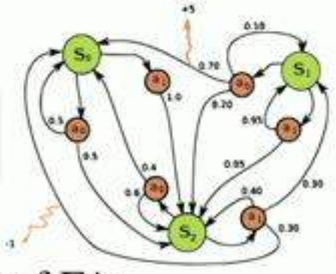
Work in submission with Ehsan Saleh (UIUC)

- For a finite-horizon construction, last level (reward fitting) cannot be solved exactly in poly samples

Importance of the conjecture

RL tractable & challenging

Tabular



Special case of FA:

- Function class closed under Bellman update
- Perfect density estimator
- Etc, etc

Batch

Nice data (small C)
+ realizability + ???

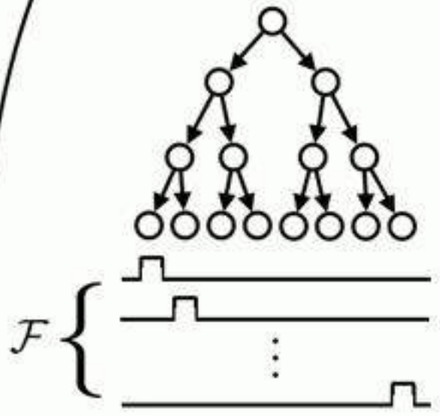
Nice data (small C)
+ realizability

Online (exploration)

Nice env
(low Bellman rank; Jiang et al'17)
+ realizability

Nice env
(low witness rank; Sun et al'18)
+ realizability

RL intractable



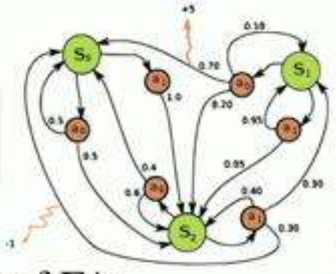
value-based

model-based

Importance of the conjecture

RL tractable & challenging

Tabular



Special case of FA:

- Function class closed under Bellman update
- Perfect density estimator
- Etc, etc

Batch

Nice data (small C)
+ realizability + ???

Nice data (small C)
+ realizability

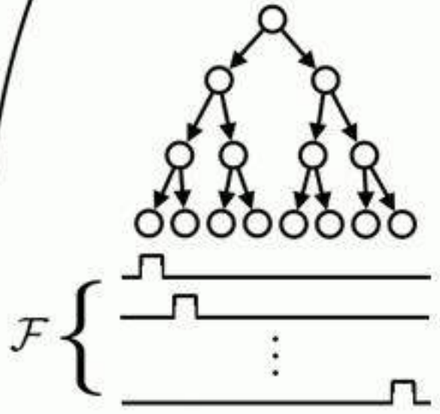
Online (exploration)

Nice env
(low Bellman rank; Jiang et al'17)
+ realizability

Gap confirmed

Nice env
(low witness rank; Sun et al'18)
+ realizability

RL intractable



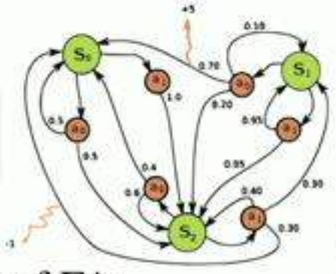
value-based

model-based

Importance of the conjecture

RL tractable & challenging

Tabular



Special case of FA:

- Function class closed under Bellman update
- Perfect density estimator
- Etc, etc

Batch

Nice data (small C)
+ realizability + ???

Nice data (small C)
+ realizability

Gap?

Gap?

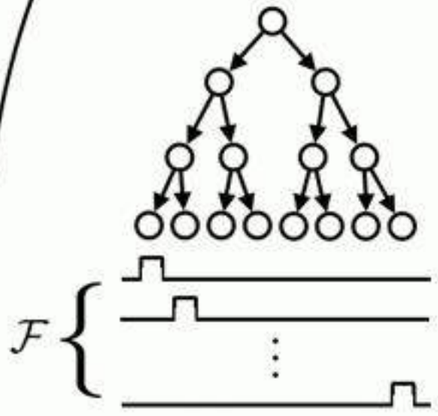
Online (exploration)

Gap confirmed

Nice env
(low Bellman rank; Jiang et al'17)
+ realizability

Nice env
(low witness rank; Sun et al'18)
+ realizability

RL intractable



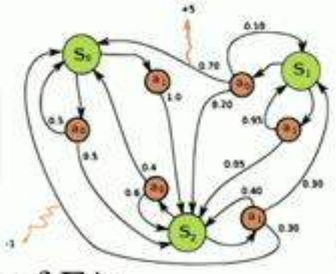
value-based

model-based

Importance of the conjecture

RL tractable & challenging

Tabular



Special case of FA:

- Function class closed under Bellman update
- Perfect density estimator
- Etc, etc

Batch

Nice data (small C)
+ realizability + ???

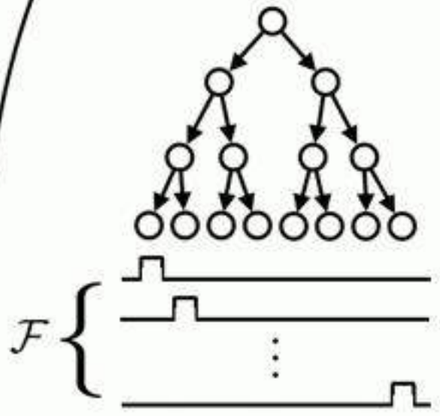
Nice data (small C)
+ realizability

Online (exploration)

Nice env
(low Bellman rank; Jiang et al'17)
+ realizability

Nice env
(low witness rank; Sun et al'18)
+ realizability

RL intractable



value-based

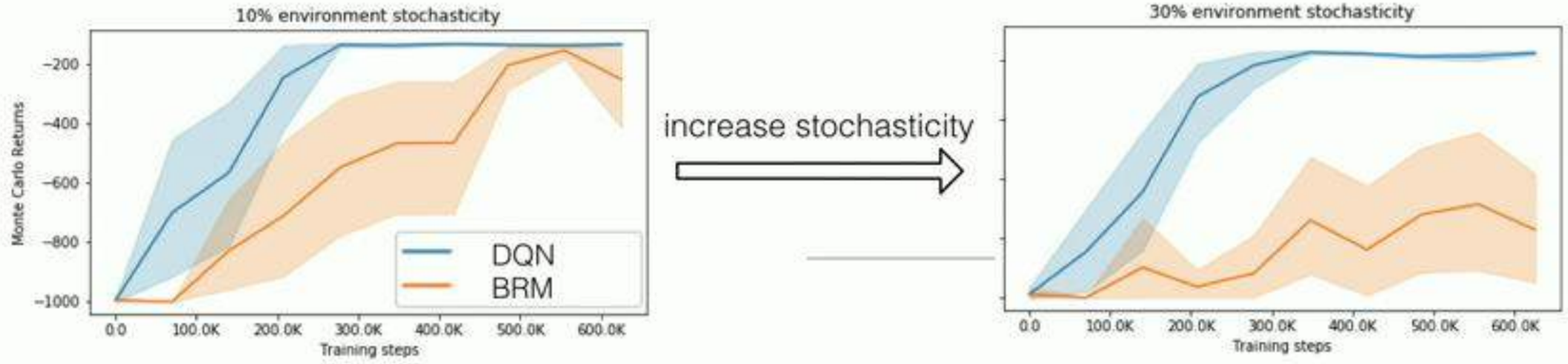
model-based

Checklist for a plausible construction

- Cannot be tabular
- Cannot be uncontrolled (Monte-Carlo works)
- Cannot be deterministic (Bellman Residual Minimization works)

$$\arg\min_{f \in \mathcal{F}} \sum_{(s,a,r,s') \in D} \left(f(s,a) - (r + \gamma \max_{a'} f(s',a')) \right)^2$$

don't block gradient here;
1 line code change from DQN



Work in submission with Ehsan Saleh (UIUC)

- For a finite-horizon construction, last level (reward fitting) cannot be solved exactly in poly samples

Checklist for a plausible construction

- Cannot be tabular
- Cannot be uncontrolled (Monte-Carlo works)
- Cannot be deterministic (Bellman Residual Minimization works)

$$\arg\min_{f \in \mathcal{F}} \sum_{(s,a,r,s') \in D} \left(f(s,a) - (r + \gamma \max_{a'} f(s',a')) \right)^2$$

don't block gradient here;
1 line code change from DQN

DQN

BRM

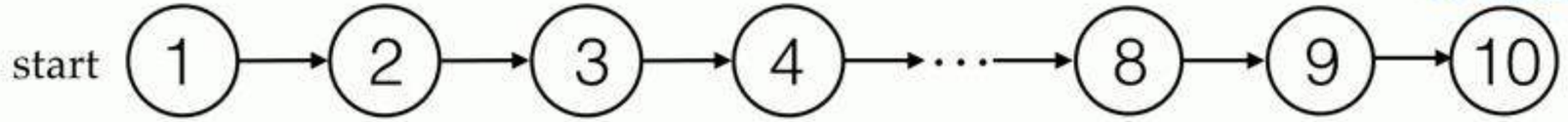
Work in submission with Ehsan Saleh (UIUC)

Proving the conjecture: Attempt 2

- Let $\mathcal{M}$ be a family of MDP instances
- Natural F: $\{Q_M^*: M \in \mathcal{M}\}$

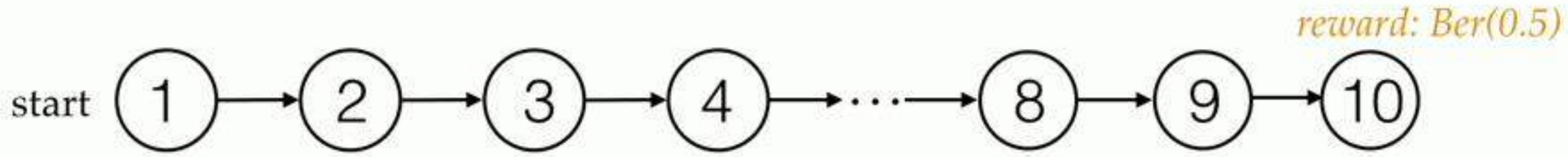
Fix using a strong assumption ("completeness")

reward: Ber(0.5)



Function class	0.5	×	0.5
	0.756		0.628

How things go wrong (w/ restricted class)



Function class	0.5	×	0.5	×	0.5	×	… ×	0.5	× 0.5 × 0.5
	1.012		0.756		0.628			0.504	0.502 0.501

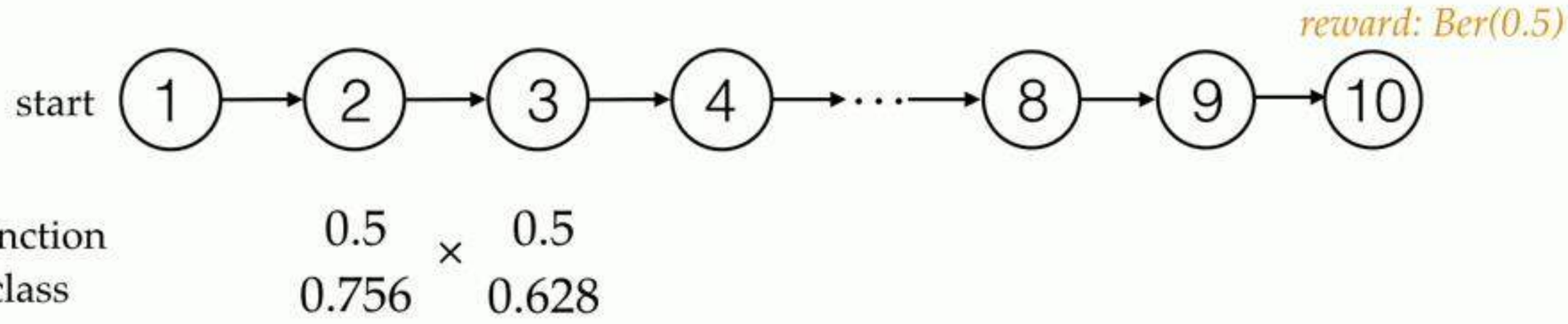
FQI Iter #1: Data: (⑩, 1, *end*), …, (⑩, 0, *end*) ⇨ 0.501

Iter #2: Data: (⑨, 0, ⑩) ⇨ (⑨, 0+0.501) ⇨ 0.502 0.501

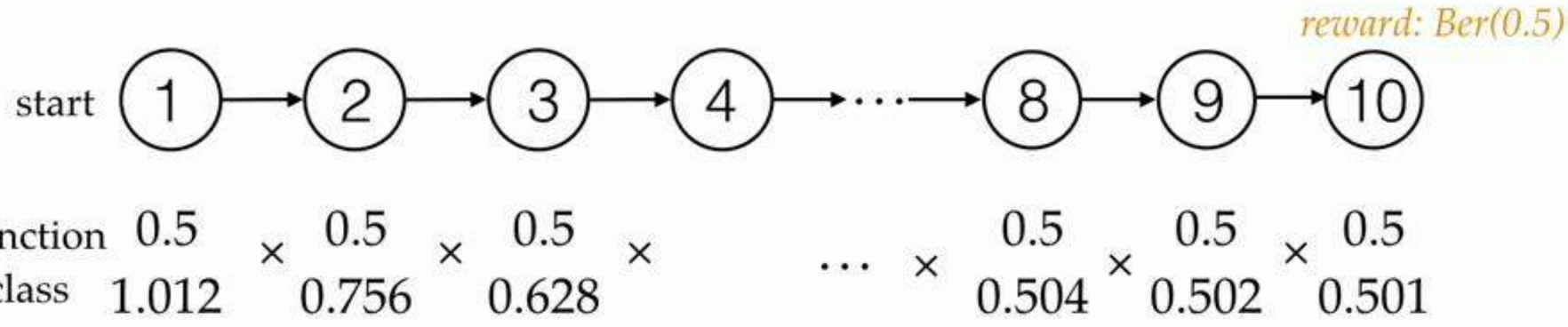
…

Iter #10: !!! 1.012 0.756 0.628 … 0.502 0.501

Fix using a strong assumption ("completeness")



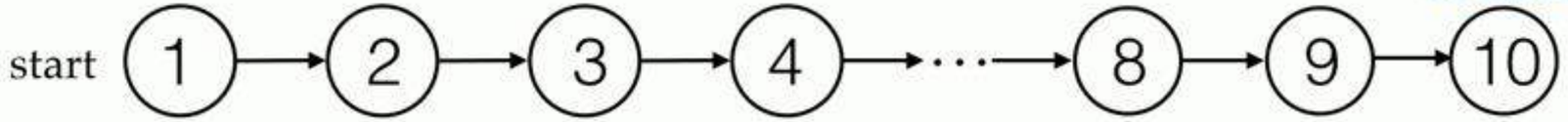
How things go wrong (w/ restricted class)



FQI Iter #1: Data: (⑩, 1, *end*), …, (⑩, 0, *end*) ⇨ 0.501

How things go wrong (w/ restricted class)

reward: Ber(0.5)



Function class	0.5	×	0.5	×	0.5	×	…	×	0.5	×	0.5	×	0.5
	1.012		0.756		0.628				0.504		0.502		0.501