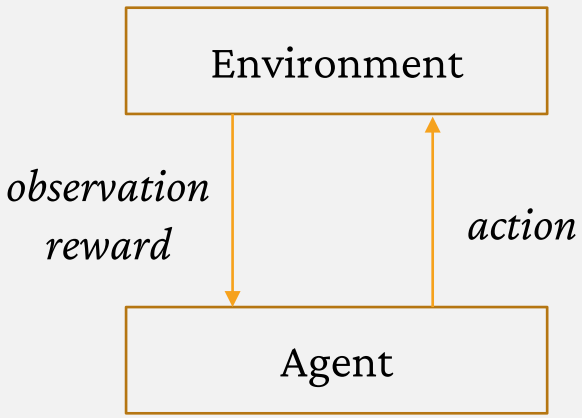
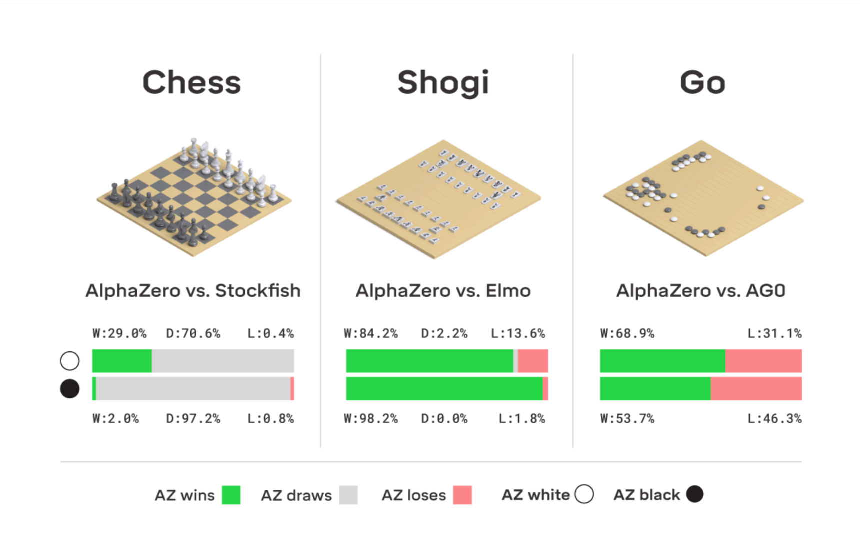
Learning Self-Correctable Policies and Value Functions from Demonstrations with Negative Sampling

Yuping Luo[1], Huazhe (Harry) Xu[2], Tengyu Ma[3]

[1]Princeton University, [2]UC Berkeley, [3]Stanford University

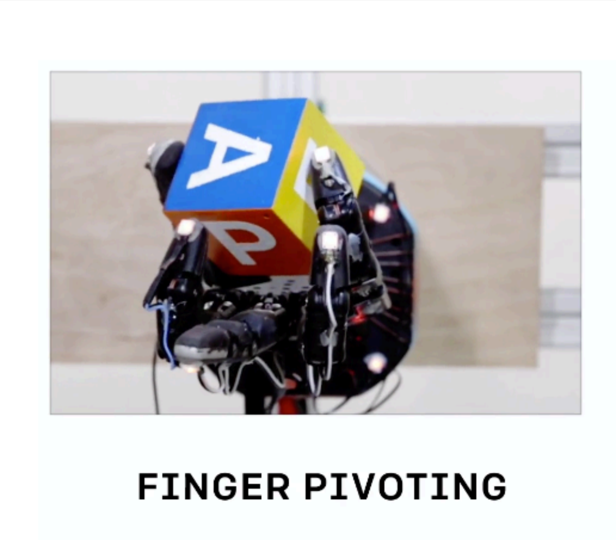
Reinforcement Learning (RL)





From [DeepMind]

Games



From [OpenAI]

Robotics

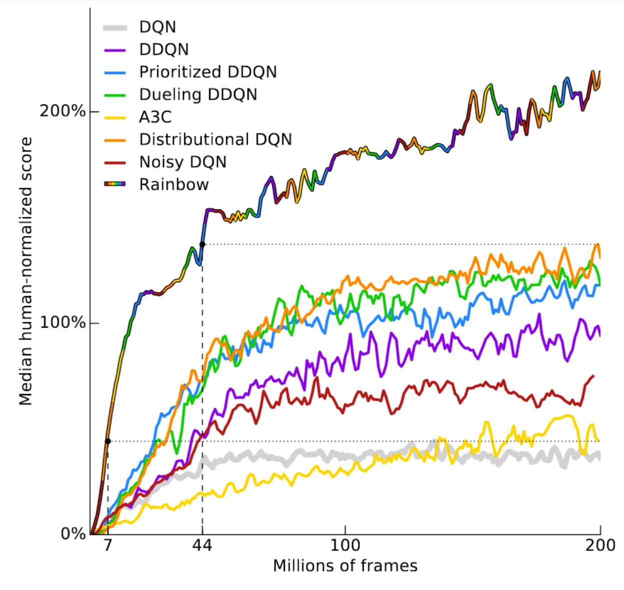
Imitation Learning (IL)

- Given demonstrations from experts.
- Learn a policy from demonstrations from *implicit reward function*.
- Two settings: w/ or w/o environment interactions

From [Yisong Yue]

Why Imitation Learning?

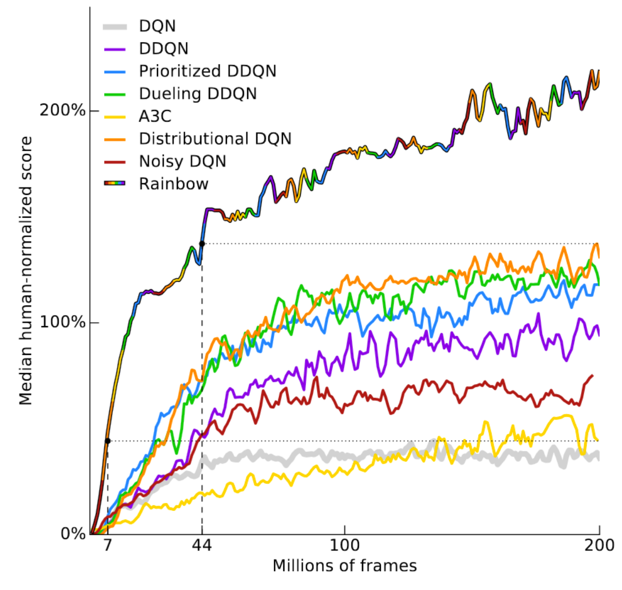
Sample efficiency



From [Hessal et. al]

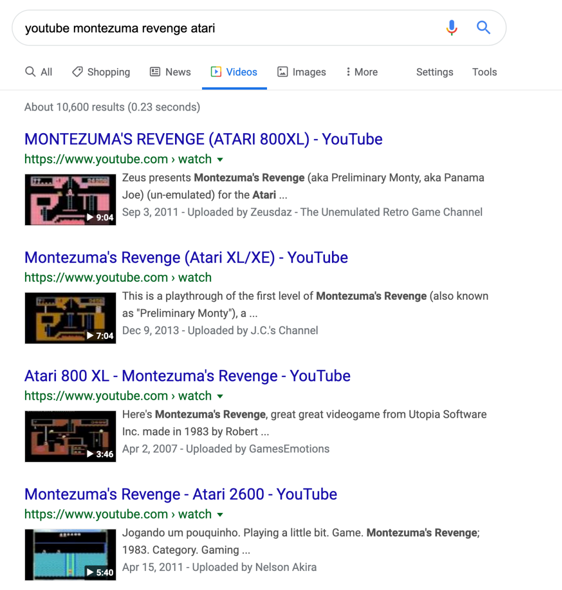
Why Imitation Learning?

Sample efficiency



From [Hessal et. al]

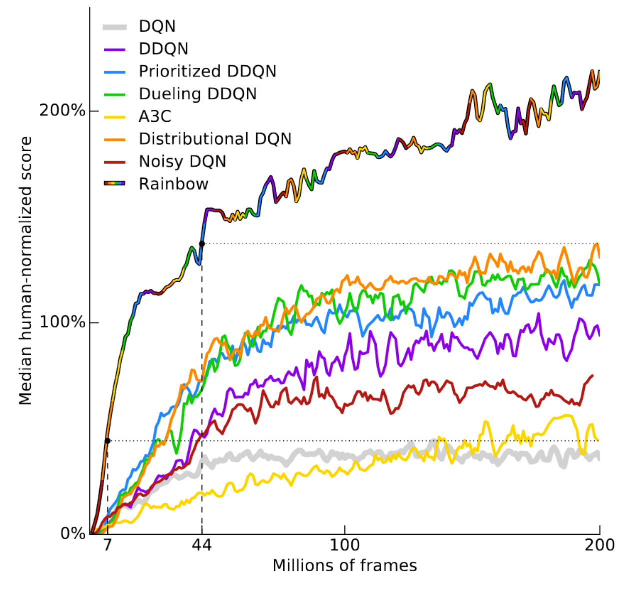
Use existing good demonstrations



From [Google]

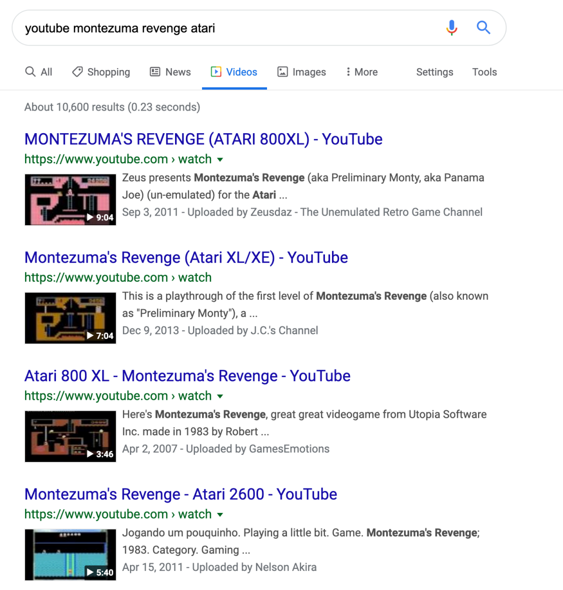
Why Imitation Learning?

Sample efficiency



From Hessal et. al

Use existing good demonstrations



From Google

Hard to design (good) reward function

$$r(b_z^{(1)}, s^P, s^{B1}, s^{B2}) = \begin{cases} 1 & \text{if stack}(b_z^{(1)}, s^P, s^{B1}, s^{B2}) \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

$$r(b_z^{(1)}, s^P, s^{B1}, s^{B2}) = \begin{cases} 1 & \text{if stack}(b_z^{(1)}, s^P, s^{B1}, s^{B2}) \\ 0.25 & \text{if } \neg\text{stack}(b_z^{(1)}, s^P, s^{B1}, s^{B2}) \wedge \text{grasp}(b_z^{(1)}, s^P, s^{B1}, s^{B2}) \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

$$r(b_z^{(1)}, s^P, s^{B1}, s^{B2}) = \begin{cases} 1 & \text{if stack}(b_z^{(1)}, s^P, s^{B1}, s^{B2}) \\ 0.25 & \text{if } \neg\text{stack}(b_z^{(1)}, s^P, s^{B1}, s^{B2}) \wedge \text{grasp}(b_z^{(1)}, s^P, s^{B1}, s^{B2}) \\ 0.125 & \text{if } \neg(\text{stack}(b_z^{(1)}, s^P, s^{B1}, s^{B2}) \vee \text{grasp}(b_z^{(1)}, s^P, s^{B1}, s^{B2})) \wedge \text{reach}(b_z^{(1)}, s^P, s^{B1}, s^{B2}) \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

$$r(b_z^{(1)}, s^P, s^{B1}, s^{B2}) = \begin{cases} 1 & \text{if stack}(b_z^{(1)}, s^P, s^{B1}, s^{B2}) \\ 0.25 + 0.25 r_{S2}(s^{B1}, s^P) & \text{if } \neg\text{stack}(b_z^{(1)}, s^P, s^{B1}, s^{B2}) \wedge \text{grasp}(b_z^{(1)}, s^P, s^{B1}, s^{B2}) \\ 0.125 & \text{if } \neg(\text{stack}(b_z^{(1)}, s^P, s^{B1}, s^{B2}) \vee \text{grasp}(b_z^{(1)}, s^P, s^{B1}, s^{B2})) \wedge \text{reach}(b_z^{(1)}, s^P, s^{B1}, s^{B2}) \\ 0 + 0.125 r_{S1}(s^{B1}, s^P) & \text{otherwise} \end{cases} \quad (6)$$

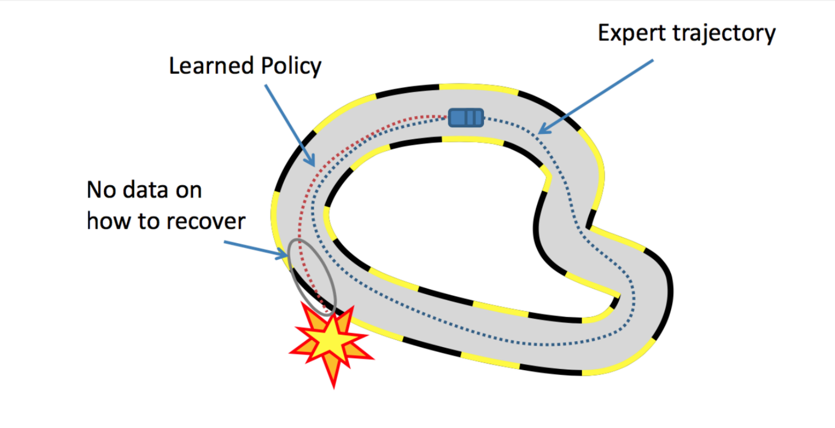
From Popov et. al

Basic Notations

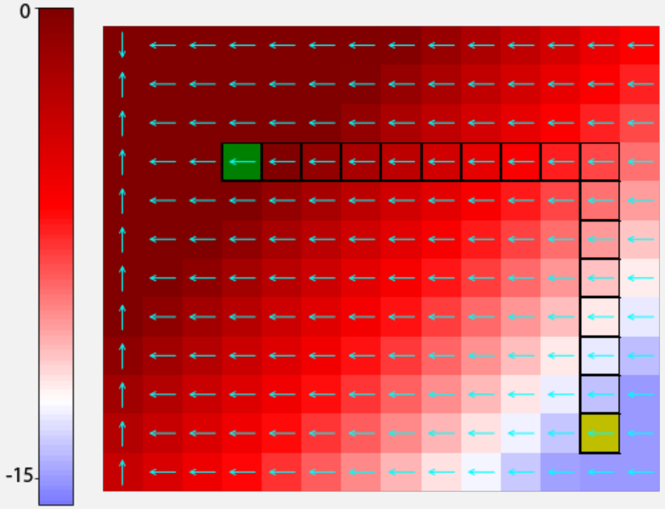
- For simplicity, we assume a deterministic MDP with:
  - State space $\mathcal{S} = \mathbb{R}^d$, action space $\mathcal{A} = \mathbb{R}^k$
  - Dynamics model $M^*: \mathcal{S} \times \mathcal{A} \to \mathcal{S}$
  - Reward function $r: \mathcal{S} \times \mathcal{A} \to \mathbb{R}$
  - Discount factor: $\gamma \in (0, 1]$
- (*Value function*) $V^\pi(s) = \mathbb{E}\left[\sum_{i=0} \gamma^i r_i | s_0 = s\right]$ is the value at state s
- Expert policy π_e
- Goal: find a policy $\pi = \arg\max_{\pi} \mathbb{E}_{s \sim D_{s_0}}[V^\pi(s)]$

Main Concern in Imitation Learning

- Covariate shift: Different state distributions in demonstration and testing.



From Emma Brunskill



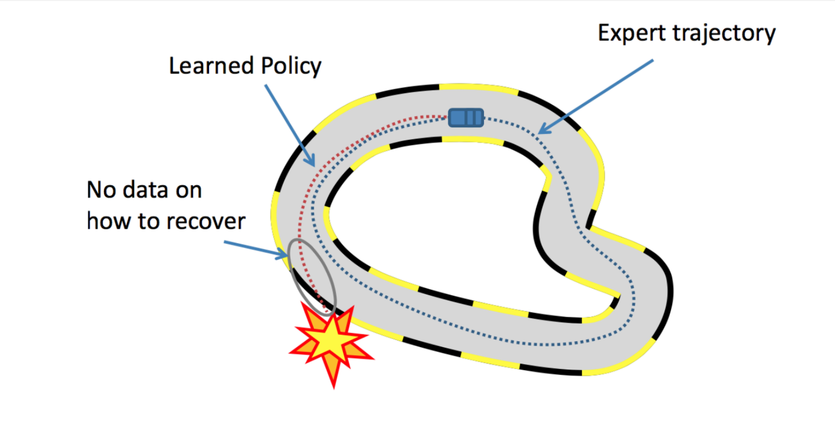
Grid world. Reward = -1 at each step and stops when reaching goal.

Color: learned value function.
Black path: demonstration.
Yellow: Starting point.
Green: Goal.
Arrows: Policy.

The learned value function via standard Bellman equation.

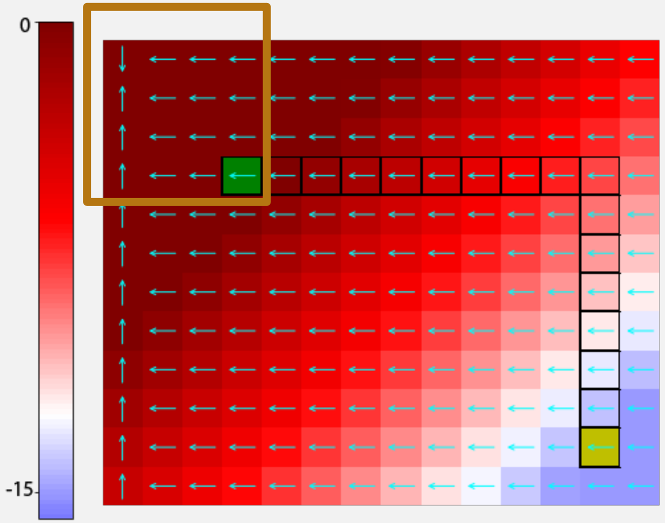
Main Concern in Imitation Learning

- Covariate shift: Different state distributions in demonstration and testing.



From Emma Brunskill

Wrong extrapolation of V



Grid world. Reward = -1 at each step and stops when reaching goal.

Color: learned value function.
Black path: demonstration.
Yellow: Starting point.
Green: Goal.
Arrows: Policy.

The learned value function via standard Bellman equation.

Challenges

Challenge 1. The value function V^{π_e} and Q^{π_e} are not unique outside of demonstration.

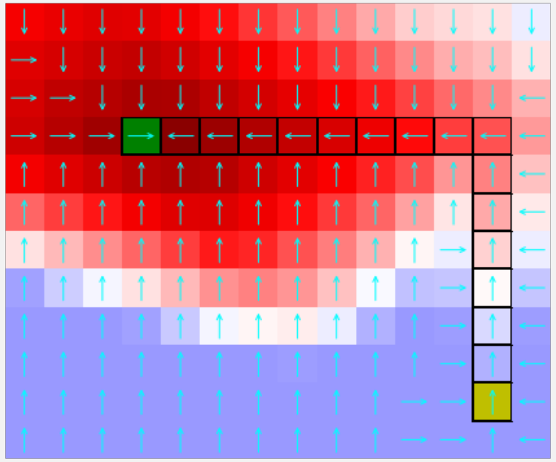
Challenge 2. Behavioral Cloning (BC) has cascading errors.

Challenge 3. Many RL algorithms use random initialized value function, which destroys a good initialized policy quickly.

Conservatively-Extrapolated Value Function

- Non-demonstration states should have lower value than demonstration states.
- Penalize non-demonstration states.

Color: learned value function.
Black path: demonstration.
Yellow: Starting point.
Green: Goal.
Arrows: Policy.

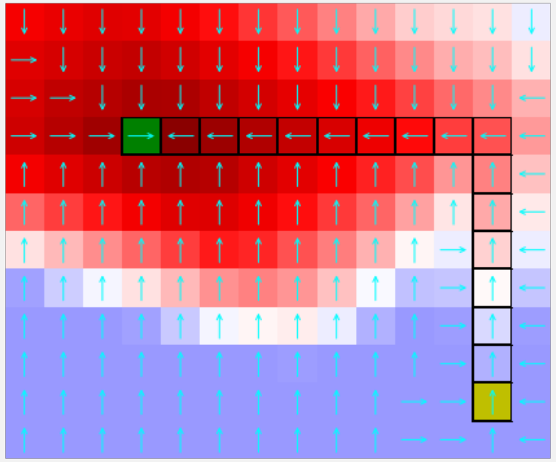


The Conservatively-Extrapolated Value Function

Conservatively-Extrapolated Value Function

- Non-demonstration states should have lower value than demonstration states.
- Penalize non-demonstration states.

Color: learned value function.
Black path: demonstration.
Yellow: Starting point.
Green: Goal.
Arrows: Policy.



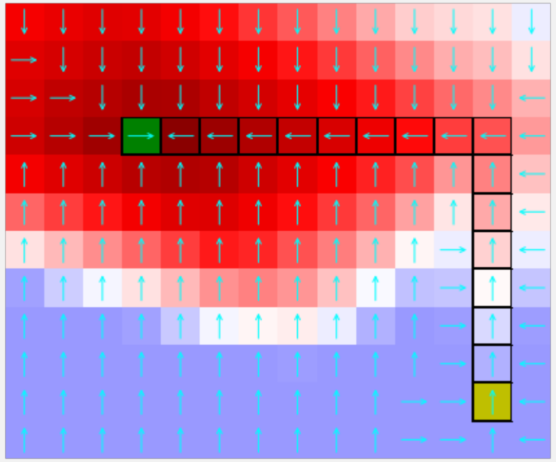
The Conservatively-Extrapolated Value Function

Algorithm

Conservatively-Extrapolated Value Function

- Non-demonstration states should have lower value than demonstration states.
- Penalize non-demonstration states.

Color: learned value function.
Black path: demonstration.
Yellow: Starting point.
Green: Goal.
Arrows: Policy.



The Conservatively-Extrapolated Value Function

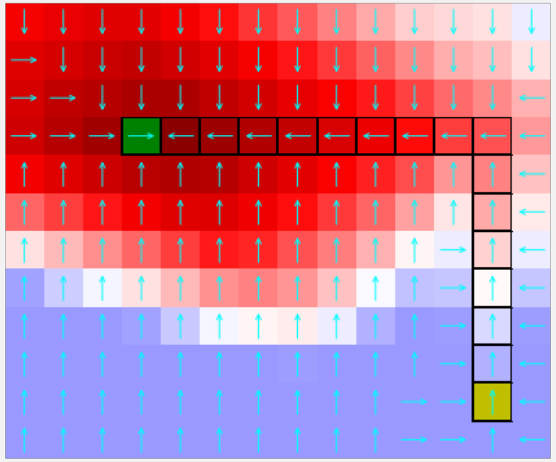
Algorithm

Learn a conservatively-extrapolated value function V

Conservatively-Extrapolated Value Function

- Non-demonstration states should have lower value than demonstration states.
- Penalize non-demonstration states.

Color: learned value function.
Black path: demonstration.
Yellow: Starting point.
Green: Goal.
Arrows: Policy.



The Conservatively-Extrapolated Value Function

Algorithm

Learn a conservatively-extrapolated value function V

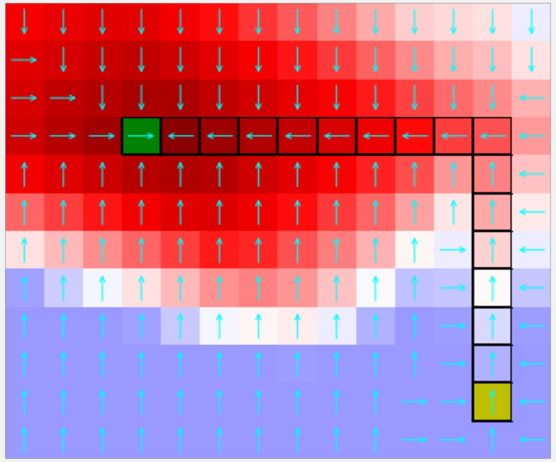
⇩

Learn the dynamics model M

Conservatively-Extrapolated Value Function

- Non-demonstration states should have lower value than demonstration states.
- Penalize non-demonstration states.

Color: learned value function.
Black path: demonstration.
Yellow: Starting point.
Green: Goal.
Arrows: Policy.



The Conservatively-Extrapolated Value Function

Algorithm

Learn a conservatively-extrapolated value function V

⇩

Learn the dynamics model M

⇩

$$\pi(s) = \arg\max_a V\big(M(s,a)\big)$$

Problem Setup

- Goal-reaching style: $r(s) = -\mathbb{I}[\|\text{loc} - \text{goal}\| \geq \varepsilon]$.
- For simplicity, $\gamma = 1$.
- Initial state distribution D_{s_0} has a low-dimensional bounded support
- $\mathcal{U}$ = the set of states which the expert policy π_e can visit w.p. > 0
  - **Assumption.** $\mathcal{U}$ is a low-dimensional manifold.

Theoretical Motivation

- BC can be correct in $\mathcal{U}$ but might not be correct outside of it.

Theoretical Motivation

- BC can be correct in $\mathcal{U}$ but might not be correct outside of it.

Question 1. How correct are BC policy and learned dynamics model?
Assumption 1. *(informally stated)* BC policy and learned dynamics model is locally (around $\mathcal{U}$) correct.

Theoretical Motivation

- BC can be correct in $\mathcal{U}$ but might not be correct outside of it.

Question 1. How correct are BC policy and learned dynamics model?
Assumption 1. *(informally stated)* BC policy and learned dynamics model is locally (around $\mathcal{U}$) correct.

Question 2. Does such a correction exist?
Assumption 2. *(informally stated)* There exists an action which makes a correction so that the resulting state is ε-close to $\mathcal{U}$.

Theoretical Motivation

- BC can be correct in $\mathcal{U}$ but might not be correct outside of it.

Question 1. How correct are BC policy and learned dynamics model?
Assumption 1. *(informally stated)* BC policy and learned dynamics model is locally (around $\mathcal{U}$) correct.

Question 2. Does such a correction exist?
Assumption 2. *(informally stated)* There exists an action which makes a correction so that the resulting state is ε-close to $\mathcal{U}$.

Question 3. Can the dynamics model/policy/value function change too fast?
Assumption 3. *(informally stated)* BC policy, dynamics model, value functions and projection function to $\mathcal{U}$ are Lipschitz.

Theoretical Motivation

Definition. A conservatively-extrapolated value function V satisfies

$$\begin{aligned} V(s) &= V^{\pi_e}(s) \pm \delta_V, && \text{if } s \in \mathcal{U} \\ V(s) &= V^{\pi_e}(\Pi_{\mathcal{U}}(s)) - \lambda \|s - \Pi_{\mathcal{U}}(s)\| \pm \delta_V && \text{if } s \notin \mathcal{U} \end{aligned}$$

The following induced policy is self-correctable:

$$\pi(s) \triangleq \arg \max_{a:\|a - \pi_{\mathrm{bc}}(s)\| \leq \zeta} V(M(s, a))$$

Theoretical Motivation

Definition. A conservatively-extrapolated value function V satisfies

$$
\begin{aligned}
V(s) &= V^{\pi_e}(s) \pm \delta_V, && \text{if } s \in \mathcal{U} \\
V(s) &= V^{\pi_e}(\Pi_{\mathcal{U}}(s)) - \lambda \| s - \Pi_{\mathcal{U}}(s) \| \pm \delta_V && \text{if } s \notin \mathcal{U}
\end{aligned}
$$

The following induced policy is self-correctable:

$$\pi(s) \triangleq \arg \max_{a : \|a - \pi_{\mathrm{bc}}(s)\| \leq \zeta} V(M(s, a))$$

Challenge 1:
The value function V^{π_e} (and Q^{π_e}) are not unique outside of $\mathcal{U}$.

Not a problem for a conservatively-extrapolated value function!

Theoretical Motivation

Main Theorem. *(informally stated)* Under assumptions listed above, starting from $s_0 \in \mathcal{U}$ and executing a self-correctable policy π for $T_0 \leq T$ steps,

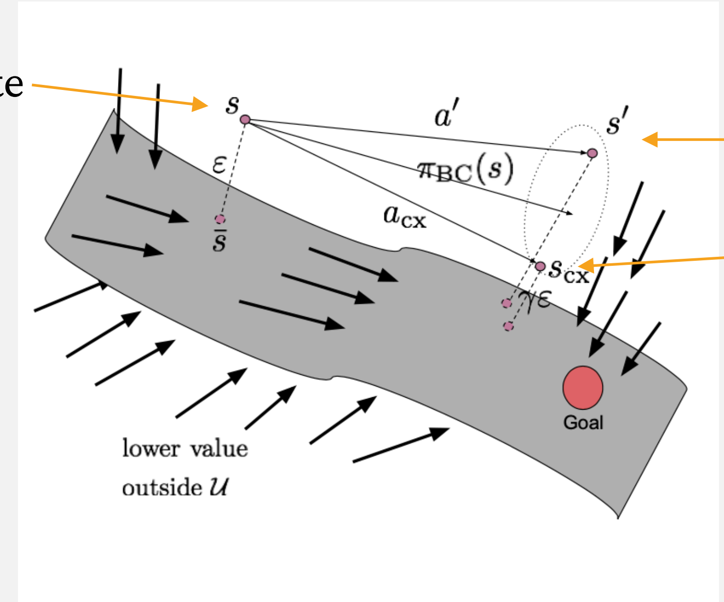
1. The resulting states $s_1, \ldots, s_{T_0}$ are all ε-close to the demonstrate states set $\mathcal{U}$.
2. If π_e improves V^{π_e} at every step, π also improves V^{π}.

Theoretical Motivation

Main Theorem. *(informally stated)* Under assumptions listed above, starting from $s_0 \in \mathcal{U}$ and executing a self-correctable policy π for $T_0 \leq T$ steps,

1. The resulting states $s_1, \dots, s_{T_0}$ are all ε-close to the demonstrate states set $\mathcal{U}$.
2. If π_e improves V^{π_e} at every step, π also improves V^{π}.

the current state
ε-close to $\mathcal{U}$



a random action

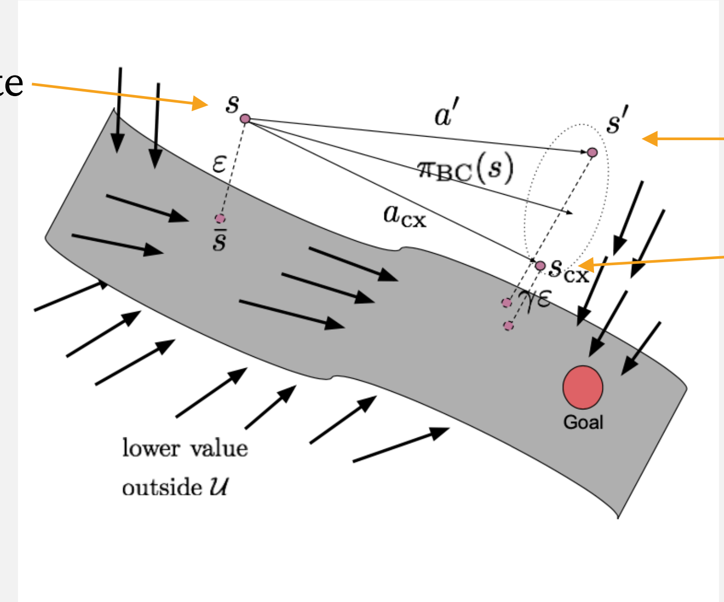
a possible correction

Theoretical Motivation

Main Theorem. *(informally stated)* Under assumptions listed above, starting from $s_0 \in \mathcal{U}$ and executing a self-correctable policy π for $T_0 \leq T$ steps,

1. The resulting states $s_1, \ldots, s_{T_0}$ are all ε-close to the demonstrate states set $\mathcal{U}$.
2. If π_e improves V^{π_e} at every step, π also improves V^{π}.

the current state ε-close to $\mathcal{U}$



a random action

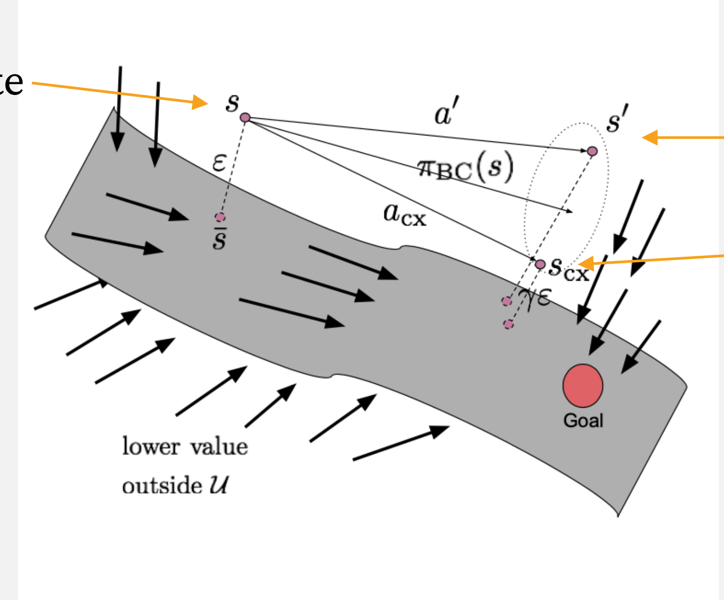
a possible correction

$V(s_{cx}) \geq V(s') \ \Rightarrow\ a_{cx}$ is preferred than a' by π

Theoretical Motivation

Main Theorem. *(informally stated)* Under assumptions listed above, starting from $s_0 \in \mathcal{U}$ and executing a self-correctable policy π for $T_0 \leq T$ steps,

1. The resulting states $s_1, \dots, s_{T_0}$ are all ε-close to the demonstrate states set $\mathcal{U}$.
2. If π_e improves V^{π_e} at every step, π also improves V^{π}.



Challenge 2:
Behavioral Cloning (BC) has cascading errors.

Not true for a self-correctable policy!

$V(s_{cx}) \geq V(s') \Rightarrow a_{cx}$ is preferred than a' by π

Value Iteration with Negative Sampling (VINS)

- Use negative sampling to learn a conservatively-extrapolated value function V.
  - For a demonstration state s, create a non-demonstration state $\tilde{s} = \text{perturb}(s)$
  - then minimize the following loss:

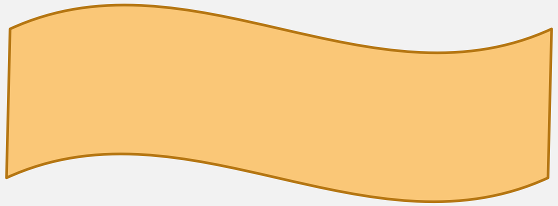
$$L(\phi) = \underbrace{\mathbb{E}_{(s,a,s')\sim\rho^{\pi_e}}\left[\left(r(s,a) + V_{\bar{\phi}}(s') - V_{\phi}(s)\right)^2\right]}_{\text{temporal difference loss}} + \mu \underbrace{\mathbb{E}_{s\sim\rho^{\pi_e},\tilde{s}\sim\text{perturb}(s)}\left[\left((V_{\bar{\phi}}(s) - \lambda\|s-\tilde{s}\|) - V_{\phi}(\tilde{s})\right)^2\right]}_{\text{negative sampling loss}}$$

Value Iteration with Negative Sampling (VINS)

- Use negative sampling to learn a conservatively-extrapolated value function V.
  - For a demonstration state s, create a non-demonstration state $\tilde{s} = \text{perturb}(s)$
  - then minimize the following loss:

$$L(\phi) = \underbrace{\mathbb{E}_{(s,a,s')\sim\rho^{\pi_e}}\left[\left(r(s,a) + V_{\bar{\phi}}(s') - V_\phi(s)\right)^2\right]}_{\text{temporal difference loss}} + \mu \underbrace{\mathbb{E}_{s\sim\rho^{\pi_e},\tilde{s}\sim\text{perturb}(s)}\left[\left((V_{\bar{\phi}}(s) - \lambda\|s-\tilde{s}\|) - V_\phi(\tilde{s})\right)^2\right]}_{\text{negative sampling loss}}$$

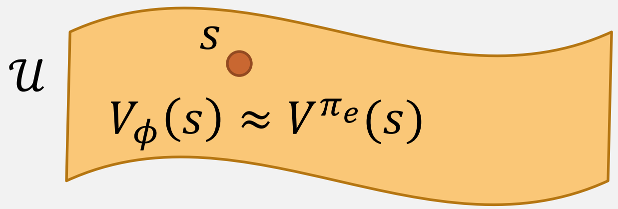
u



Value Iteration with Negative Sampling (VINS)

- Use negative sampling to learn a conservatively-extrapolated value function V.
  - For a demonstration state s, create a non-demonstration state $\tilde{s} = \text{perturb}(s)$
  - then minimize the following loss:

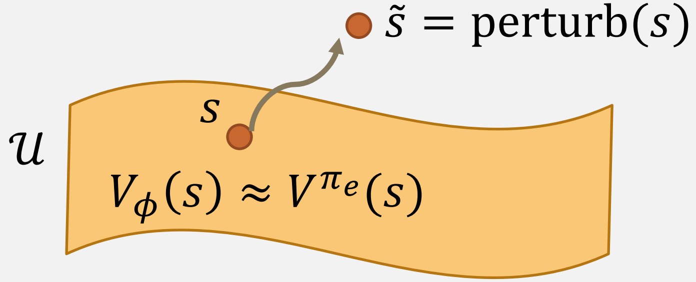
$$L(\phi) = \underbrace{\mathbb{E}_{(s,a,s')\sim\rho^{\pi_e}}\left[\left(r(s,a) + V_{\bar{\phi}}(s') - V_\phi(s)\right)^2\right]}_{\text{temporal difference loss}} + \mu \underbrace{\mathbb{E}_{s\sim\rho^{\pi_e},\tilde{s}\sim\text{perturb}(s)}\left[\left((V_{\bar{\phi}}(s) - \lambda\|s-\tilde{s}\|) - V_\phi(\tilde{s})\right)^2\right]}_{\text{negative sampling loss}}$$



Value Iteration with Negative Sampling (VINS)

- Use negative sampling to learn a conservatively-extrapolated value function V.
  - For a demonstration state s, create a non-demonstration state $\tilde{s} = \text{perturb}(s)$
  - then minimize the following loss:

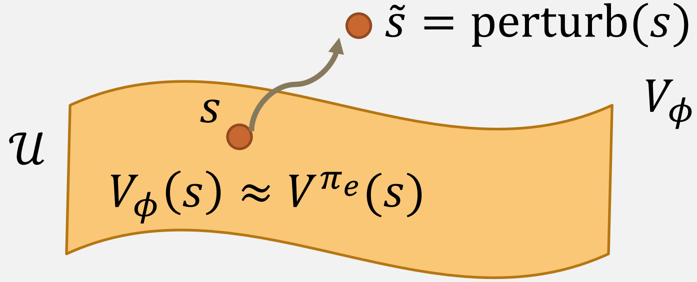
$$L(\phi) = \underbrace{\mathbb{E}_{(s,a,s')\sim\rho^{\pi_e}}\left[\left(r(s,a) + V_{\bar{\phi}}(s') - V_{\phi}(s)\right)^2\right]}_{\text{temporal difference loss}} + \mu \underbrace{\mathbb{E}_{s\sim\rho^{\pi_e},\tilde{s}\sim\text{perturb}(s)}\left[\left(\left(V_{\bar{\phi}}(s) - \lambda\|s-\tilde{s}\|\right) - V_{\phi}(\tilde{s})\right)^2\right]}_{\text{negative sampling loss}}$$



Value Iteration with Negative Sampling (VINS)

- Use negative sampling to learn a conservatively-extrapolated value function V.
  - For a demonstration state s, create a non-demonstration state $\tilde{s} = \text{perturb}(s)$
  - then minimize the following loss:

$$L(\phi) = \underbrace{\mathbb{E}_{(s,a,s')\sim\rho^{\pi_e}}\left[\left(r(s,a) + V_{\bar{\phi}}(s') - V_\phi(s)\right)^2\right]}_{\text{temporal difference loss}} + \mu \underbrace{\mathbb{E}_{s\sim\rho^{\pi_e},\tilde{s}\sim\text{perturb}(s)}\left[\left(\left(V_{\bar{\phi}}(s) - \lambda\|s-\tilde{s}\|\right) - V_\phi(\tilde{s})\right)^2\right]}_{\text{negative sampling loss}}$$



$$V_\phi(\tilde{s}) \leftarrow V_{\bar{\phi}}(s) - \lambda\|s - \tilde{s}\|$$

Value Iteration with Negative Sampling (VINS)

- Also learn the dynamics M_θ by minimizing prediction error.
- Use the induced policy from M_θ and V_ϕ, i.e., $\pi(s) = \arg\max_a V_\phi\big(M_\theta(s,a)\big)$.
  - Use Behavior Cloning policy for better optimization.

function POLICY(s)
    Option 1: $a = \pi_{\text{BC}}(s)$; Option 2: $a = 0$
    sample k noises $\xi_1, \ldots, \xi_k$ from Uniform$[-1, 1]^m$ ▷ m is the dimension of action space
    $i^* = \text{argmax}_i\, V_\phi(M_\theta(s, a + \alpha\xi_i))$ ▷ $\alpha > 0$ is a hyper-parameter
    return $a + \alpha\xi_{i^*}$

Value Iteration with Environment Interaction

- Initialize the model and value function by VINS.
- The policy is not destroyed as we have a reasonable value function.

Algorithm 3 Value Iteration with Environment Interactions Initialized by VINS (VINS+RL)

Require: Initialize parameters ϕ, θ from the result of VINS (Algorithm 2)

$\mathcal{R} \leftarrow$ demonstration trajectories;
for stage $t = 1, \ldots$ **do**
    collect n_1 samples using the induced policy π in Algorithm 2 (with Option 2 in Line 10) and add them to $\mathcal{R}$
    for $i = 1, \ldots, n_{\text{inner}}$ **do**
        sample mini-batch $\mathcal{B}$ of N transitions (s, a, r, s') from $\mathcal{R}$
        update ϕ to minimize $\mathcal{L}_{td}(\phi; \mathcal{B})$
        update target value network: $\bar{\phi} \leftarrow \bar{\phi} + \tau(\phi - \bar{\phi})$
        update θ to minimize loss $\mathcal{L}_{\text{model}}(\theta; \mathcal{B})$

Value Iteration with Environment Interaction

- Initialize the model and value function by VINS.
- The policy is not destroyed as we have a reasonable value function.

Algorithm 3 Value Iteration with Environment Interactions Initialized by VINS (VINS+RL)

Require: Initialize parameters ϕ, θ from the result of VINS (Algorithm 2)

```
R ← demonstration trajectories;
for stage t = 1, ... do
    collect n_1 samples using the induced policy π in Algorithm 2 (with Option 2 in Line 10) and
    add them to R
    for i = 1, ..., n_inner do
        sample mini-batch B of N transitions (s, a, r, s') from R
        update φ to minimize L_td(φ; B)
        update target value network: φ̄ ← φ̄ + τ(φ − φ̄)
        update θ to minimize loss L_model(θ; B)
```

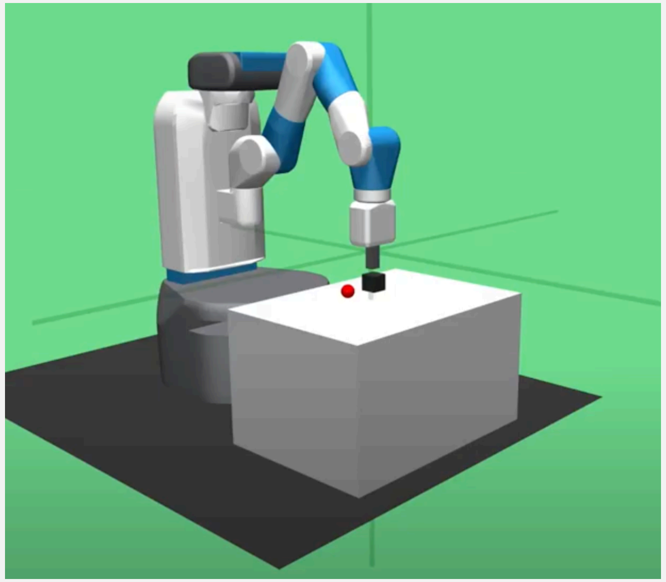
Challenge 3:

Make use of good initialization.

VINS+RL can do it!

Environments for Experiments

- Environments from OpenAI Gym: `FetchReach-v0`, `FetchPickAndPlace-v0`, `FetchPush-v0`.
- Observation: data from sensors
  - e.g., arm position/velocity, gripper position.
- Reward: of form $-\mathbb{I}[||\text{loc} - \text{goal}|| \geq \varepsilon]$.



Experimental Results

	VINS (ours)	BC
Reach 10	$\mathbf{99.3 \pm 0.1}\%$	$98.6 \pm 0.1\%$
Pick 100	$\mathbf{75.7 \pm 1.0}\%$	$66.8 \pm 1.1\%$
Pick 200	$\mathbf{84.0 \pm 0.5}\%$	$82.0 \pm 0.8\%$
Push 100	$\mathbf{44.0 \pm 1.5}\%$	$37.3 \pm 1.1\%$
Push 200	$\mathbf{55.2 \pm 0.7}\%$	$51.3 \pm 0.6\%$

Without environment interaction: <u>VINS achieves higher success rate than BC given the same demonstrations.</u>

Experimental Results

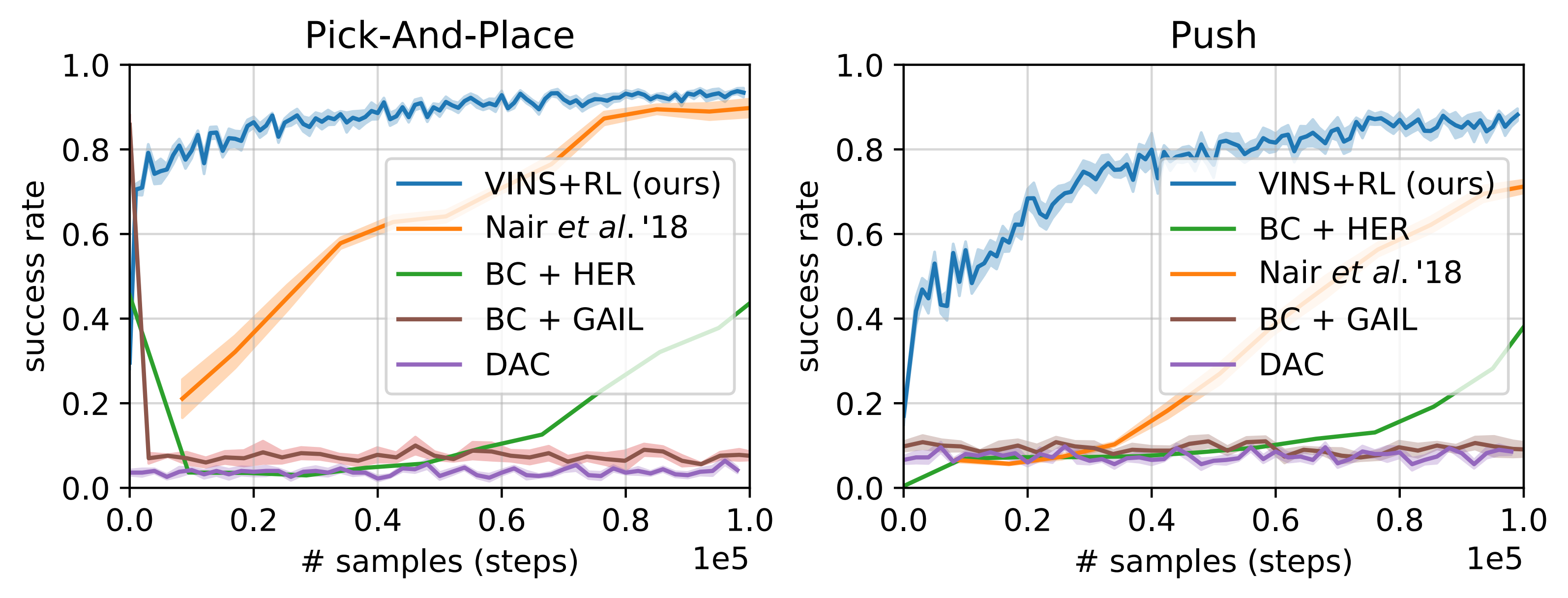


Figure: With environment interaction: VINS outperforms Nair *et al.* '18, HER, DAC, GAIL in terms of sample efficiency.

Ablation Study

- Three components in VINS: dynamics model, value function, optimization
  - *Dynamics Model*: learned model vs oracle model
  - *Value Function*: with negative sampling vs without negative sampling
  - *Optimization*: with BC vs without BC

Ablation Study

- Three components in VINS: dynamics model, value function, optimization
  - *Dynamics Model*: learned model vs oracle model
  - *Value Function*: with negative sampling vs without negative sampling
  - *Optimization*: with BC vs without BC

	Pick 100	Pick 200	Push 100	Push 200
BC	$66.8 \pm 1.1\%$	$82.0 \pm 0.8\%$	$37.3 \pm 1.1\%$	$51.3 \pm 0.6\%$
VINS	$75.7 \pm 1.0\%$	$84.0 \pm 0.5\%$	$44.0 \pm 0.8\%$	$55.2 \pm 0.7\%$
VINS w/o BC	$28.5 \pm 1.1\%$	$43.6 \pm 1.2\%$	$14.3 \pm 0.5\%$	$24.9 \pm 1.3\%$
VINS w/ oracle w/o BC	$51.4 \pm 1.4\%$	$62.3 \pm 1.1\%$	$40.7 \pm 1.4\%$	$42.9 \pm 1.3\%$
VINS w/ oracle	$76.3 \pm 1.4\%$	$87.0 \pm 0.7\%$	$48.7 \pm 1.2\%$	$63.8 \pm 1.3\%$
VINS w/o NS	$48.5 \pm 2.1\%$	$71.6 \pm 0.9\%$	$29.3 \pm 1.2\%$	$38.7 \pm 1.5\%$

Conclusion

- A conservatively-extrapolated value function leads to self-correction.
- VINS can be an alternative to BC and can also be combined with BC.
- The learned value function from demonstration helps initialization for faster convergence.