Reinforcement Learning via an Optimization Lens

Lihong Li

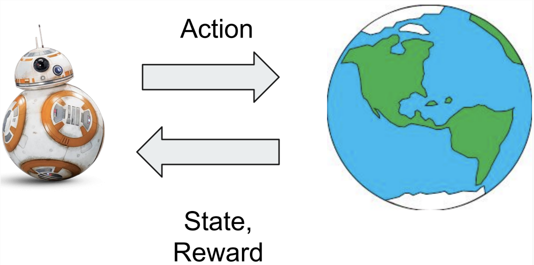
August 7, 2019

Google Brain

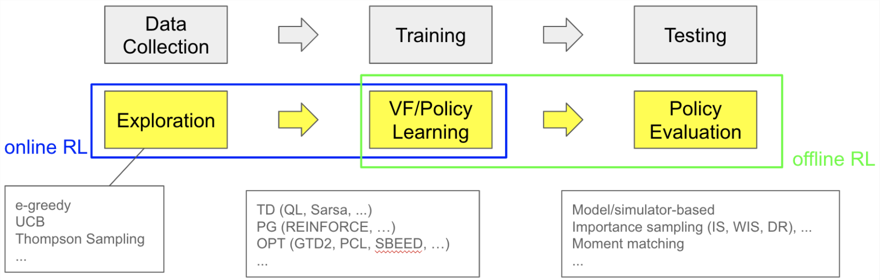
Simons Institute Workshop on Emerging Challenges in Deep Learning

Reinforcement learning: Learning to make decisions

- Video & board games
- Inventory management
- Robotics and control
- Medical treatment
- Web recommendation
- Conversational systems
- Education systems
- ...



Online vs. Offline (Batch) RL: A Basic View



This talk: batch value function learning

- Separate and focus on individual technical challenges
- Many common use cases in practice
- Will be used as a component in online RL (w/ exploration)

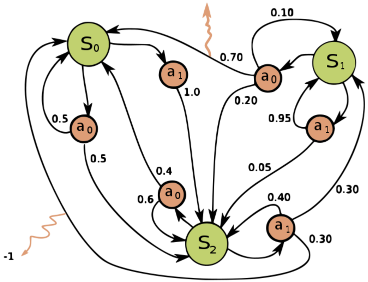
Outline

- **Background**
- How things may go wrong
- A primal-dual formulation
- A new kernel loss
- Conclusions

Markov Decision Process (MDP)

$M = \langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$

- Set of states $\mathcal{S}$
- Set of actions $\mathcal{A}$
- Transition probabilities $P(s'|s, a)$
- Immediate expected reward $R(s, a)$
- Discount factor $\gamma \in (0, 1)$



(from Wikipedia)

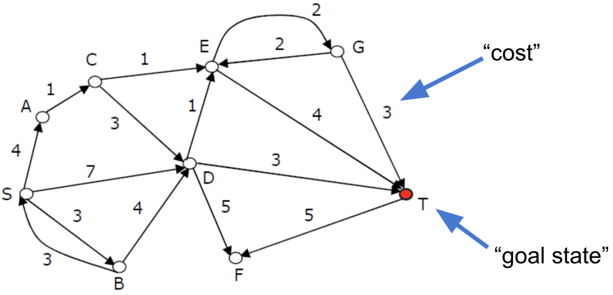
Goal: find $\pi^* : \mathcal{S} \to \mathcal{A}$ to maximize

$$\forall s \in \mathcal{S} : \quad \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, a_t \sim \pi(s_t), s_{t+1} \sim P(\cdot|s_t, a_t)\right]$$

MDP Example: Deterministic Shortest Path

Problem

Find a source→goal path with minimum cost.



- state: node
- action: edge
- reward: negative cost
- transition: landing state of directed edge

$$\mathrm{CostToGoal}(i) = \min_{j \in \mathrm{Neighbor}(i)} \left\{ \mathrm{cost}(i \to j) + \mathrm{CostToGoal}(j) \right\}$$

"Principle of Dynamic Programming" (Bellman, 1957)

More General Case: Bellman Equation

Deterministic Shortest Path

$$\mathrm{CostToGoal}(i) = \min_{j \in \mathrm{Neighbor}(i)} \{\mathrm{cost}(i \to j) + \mathrm{CostToGoal}(j)\}$$

MDP $M = \langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$

$$V^*(s) = \max_{a \in \mathcal{A}} \left\{ R(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)}[V^*(s')] \right\}$$

- "Bellman equation"
  (discrete-time Hamilton-Jacobi-Bellman equation)
- In this talk, RL $\approx$ solving for V^*

Bellman Operator

Bellman equation succinctly re-expressed as

$$V^* = \mathcal{T} V^*$$

where

$$\mathcal{T} V^*(s) := \max_{a \in \mathcal{A}} \Big\{ R(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} [V^*(s')] \Big\}$$

Well-known facts of Bellman operator $\mathcal{T}$:

- $\mathcal{T}$ is monotonic: $V_1 \preceq V_2$ implies $\mathcal{T} V_1 \preceq \mathcal{T} V_2$
- $\mathcal{T}$ is γ-contraction: $\|\mathcal{T} V_1 - \mathcal{T} V_2\|_\infty \leq \gamma \|V_1 - V_2\|_\infty$
- Hence, $V, \mathcal{T} V, \mathcal{T}^2 V, \mathcal{T}^3 V, \cdots \to V^*$ ("fixed point")
- Mathematical foundation of value iteration, TD(λ), Q-learning, etc. in the exact ($\approx$ finite-MDP) case

Outline

- Background
- **How things may go wrong**
- A primal-dual formulation
- A new kernel loss
- Conclusions

When Bellman Meets Gauss: Approximate DP

In practice, V^* is often approximated

- Eg: least-squares fit on linear models or neural networks, ...

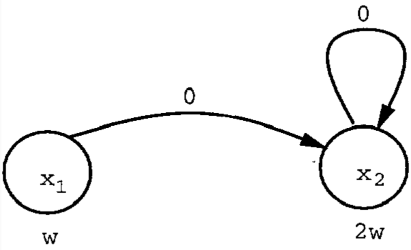
$$k = 0, 1, 2, \ldots : \quad V_{k+1} \leftarrow (\Pi_{\mathcal{V}} \circ \mathcal{T}) V_k$$

- Composing $\mathcal{T}$ and $\Pi_{\mathcal{V}}$ often loses contraction
- Many known divergent examples

  Baird (93), Boyan & Moore (95), Tsitsiklis & Van Roy (96), ...
- Limited positive theory or algorithms

  Gordon (96), Tsitsiklis & Van Roy (97), Lagoudakis & Parr (03), Sutton et al. (08, 09), Maei et al. (10), ...

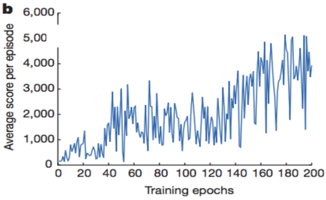
Divergence Example of Tsitsiklis & Van Roy (96)



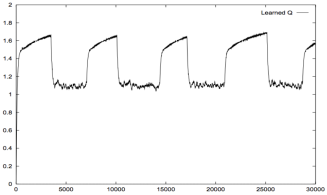
Starting with $w^{(0)} \neq 0$,
least-squares value iteration diverges when $\gamma > 5/6$,
although V^* may be exactly represented (with $w^* = 0$).

Does It Matter in Practice?

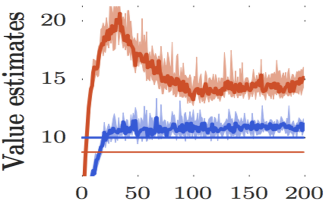
Many empirical successes of (double, dueling) DQN, A3C, ...
in video games, AlphaGo, robotics, dialogue management, ...
but often with surprises:



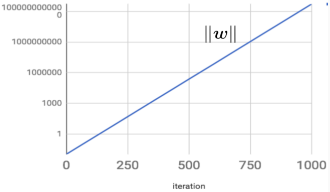
Mnih et al (2015)



Gordon (1996)



van Hasselt et al. (2016)



Baird (1993); Tsitsiklis & Van Roy (1996)

A Long-standing Open Problem

Ensuring convergent approximate dynamic programming

A major, decades-old open problem:

Functional Approximations and Dynamic Programming

By Richard Bellman and Stuart Dreyfus

Math. Tables & Other Aids Comp. (1959)

Essentially "deadly triad" (Sutton)

Unclear if solved by emphatic TD (Sutton et al., 2016)

Not solved by removing delusional bias (Lu et al., 2018)

Outline

- Background
- How things may go wrong
- **A primal-dual formulation**
- A new kernel loss
- Conclusions

Linear Programming Reformulation

Solving $V = \mathcal{T} V$ is equivalent to

$$\begin{aligned} \min_V \quad & \langle c, V \rangle \\ \text{s.t.} \quad & V \geq \mathcal{T} V \,, \end{aligned}$$

with some $c > 0$.

Schweitzer & Seidman (85), De Farias & Van Roy (06), Wang & co. (15–), Dai+ (17), Lakshminarayanan+ (18), ...

See Mengdi Wang's talk yesterday.

This talk focuses on a different approach.

Why Solving for Fixed Point Directly is Hard?

A natural objective function for solving $V = \mathcal{T}V$:

$$\min_V \quad \underbrace{\|V - \mathcal{T}V\|^2}_{\text{"Bellman error/residual"}}$$

$$= \quad \min_V \mathbb{E}_s \left[\left(V(s) - \max_{\mathbf{a}} (R(s,a) + \gamma \mathbb{E}_{\mathbf{s'|s,a}}[\mathbf{V(s')}]\right)^2 \right]$$

- Difficulty #1: breaks smoothness and continuity
- Difficulty #2: typical SGD gives biased gradient, known as "double sample" issue (Baird 95):

$$\underbrace{\left(\cdots + \gamma \mathbb{E}_{s'|s,a}[V_w(s')]\right)^2}_{\text{what we need}} \quad \neq \quad \underbrace{\mathbb{E}_{s'|s,a}\left[(\cdots + \gamma V_w(s'))^2\right]}_{\text{what empirical square loss approxiamtes}}$$

Addressing Difficulty #1: Nesterov Smoothing

$$V(s) \quad = \quad \max_a \left(R(s,a) + \gamma \mathbb{E}_{s'|s,a}[V(s')]\right)$$

$$\Downarrow$$

$$V(s) \quad = \quad \underbrace{\max_{\pi(\cdot|s)} \sum_a \pi(a|s) \left(R(s,a) + \gamma \mathbb{E}_{s'|s,a}[V(s')]\right) \quad \underbrace{+\lambda H(\pi(\cdot|s))}_{\text{entropic regularization}}}_{:= \mathcal{T}_\lambda V(s)}$$

The smoothed Bellman operator $\mathcal{T}_\lambda$ may be derived differently

Rawlik+ (12), Fox+ (16), Neu+ (17), Nachum+ (17), Asadi & Littman (17), Haarnoja+ (18), ...

Properties of $\mathcal{T}_\lambda$

$$\boxed{\mathcal{T}_\lambda V(s) := \max_{\pi(\cdot|s)} \sum_a \pi(a|s) \left(R(s,a) + \gamma \mathbb{E}_{s'|s,a}[V(s')]\right) + \lambda H(\pi(\cdot|s))}$$

- Still a γ-contraction
- Existence and uniqueness of fixed point V_λ^*
- Controlled bias: $\|V_\lambda^* - V^*\|_\infty = \mathcal{O}(\lambda/(1-\gamma))$
- Temporal consistency (as in PCL of Nachum+ (17))

$$\forall s, a : \quad V_\lambda^*(s) = R(s,a) + \gamma \mathbb{E}_{s'|s,a}[V_\lambda^*(s')] - \lambda \log \pi_\lambda^*(a|s)$$

Addressing Difficulty #2: Legendre-Fenchel Transformation

$$\min_{V} \mathbb{E}_s \left[\left(V(s) - \max_{\mathbf{a}} (R(s,a) + \gamma \mathbb{E}_{\mathbf{s}'|\mathbf{s},\mathbf{a}}[\mathbf{V}(\mathbf{s}')]) \right)^2 \right]$$

$$\Downarrow \quad \text{(by Nesterov smoothing)}$$

$$\min_{V,\pi} \mathbb{E}_{s,a} \left[\left(\underbrace{R(s,a) + \gamma \mathbb{E}_{\mathbf{s}'|\mathbf{s},\mathbf{a}}[\mathbf{V}(\mathbf{s}')] - \lambda \log \pi(a|s) - V(s)}_{\text{denoted } x_{sa}} \right)^2 \right]$$

$$\Downarrow \quad \text{(L-F transform: } x_{sa}^2 = \max_{y \in \mathbb{R}} (2x_{sa}y - y^2))$$

$$\min_{V,\pi} \max_{\nu \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}} \mathbb{E}_{s,a} \left[(2\nu(s,a)x_{s,a} - \nu(s,a)^2) \right]$$

The last step also applies the interchangeability principle

(Rockafellar & Wets 88; Shapiro & Dentcheva 14; Dai+ 17)

Reformulation of Bellman Equation

We have now turned a fixed point into a saddle point:

$$\min_{V,\pi} \max_{\nu} \quad \mathbb{E}_{s,a}\left[2\nu(s,a) \cdot \mathcal{R}_{\pi,\lambda} V(s,a) - \nu(s,a)^2\right]$$

$$\text{where} \quad \mathcal{R}_{\pi,\lambda} V(s,a) := R(s,a) + \gamma V(s') - \lambda \log \pi(a|s) - V(s)$$

- Well-defined objective without requiring double samples
- May be optimized by gradient methods (SGD/BackProp, ...)
- Inner max achieved when $\nu = \mathcal{R}_{\pi,\lambda} V$
- Easily extended to other convex loss functions

SBEED: Smoothed Bellman Error EmbeDding [Dai et al., 18]

$$\min_{V,\pi} \max_{\nu} \quad \mathbb{E}_{s,a}\Big[2\nu(s,a) \cdot \mathcal{R}_{\pi,\lambda} V(s,a) - \nu(s,a)^2\Big]$$

$$\text{where} \quad \mathcal{R}_{\pi,\lambda} V(s,a) := R(s,a) + \gamma V(s') - \lambda \log \pi(a|s) - V(s)$$

Algorithmic ideas

- Parameterize $(V, \pi; \nu)$ by $(w_V, w_\pi; w_\nu)$
- Stochastic first-order updates on parameters
  - Two-time-scale updates for primal and dual variables; or
  - Exact maximization if concave in w_ν
- Our implementation uses stochastic mirror descent

SBEED Analysis

Error decomposition:

$$\left\|\hat{V}_w^N - V^*\right\| \leq \underbrace{\left\|\hat{V}_w^N - \hat{V}_w^*\right\|}_{\text{optimization}} + \underbrace{\left\|\hat{V}_w^* - V_w^*\right\|}_{\text{statistical}} + \underbrace{\|V_w^* - V_\lambda^*\|}_{\text{approximation}} + \underbrace{\|V_\lambda^* - V^*\|}_{\text{smoothing}}$$

- Optimization error: run N iterations to find an empirically near-optimal solution
- Statistical error: use a sample of size T to approximate underlying (unknown) MDP
- Approximation error: use of parametric families to represent (V, π, ν)
- Smoothing error: from Nesterov smoothing

SBEED: Optimization

Define $\bar{\ell}(V,\pi) := \max_{\nu} L(V,\pi,\nu)$, and assume

- $\nabla\bar{\ell}$ is Lipschitz-continuous
- the stochastic gradient has finite variance
- stepsizes are properly set

Theorem. SBEED solution satisfies $\mathbb{E}[\left\|\nabla\bar{\ell}(V_{\hat{w}},\pi_{\hat{w}})\right\|^2] \to \mathbf{0}$

- Decay rate $\sim O(N^{-1/2})$ after N iterations
- Building on results of Ghadimi & Lan (13)
- See paper for variants of convergence results
- Still hard to quantify optimization error

SBEED: Statistical Analysis

Assumptions

- MDP regularity: $\|R\|_\infty \leq C_R$, $\|\log \pi^*_\lambda(a|s)\|_\infty \leq C_\pi$.
- Data collection is exponentially β-mixing with a unique stationary distribution over $\mathcal{S}$

Theorem.

$$\epsilon_{\mathsf{stat}}(T) = O(T^{-1/2})$$

Further Remarks on Related Work

- Bellman residual minimization for (fixed) policy evaluation (Antos+08, Farahmand+08)
- Specializes to certain gradient TD algorithms with linear approximation (Sutton, Maei & co.; Liu+15; Macua+15)
- Can be analyzed using well-established techniques (Antos+08, Farahmand+08, Liu+15)
- Can benefit from advanced optimization techniques such as SVRG/SAGA (Du+17)

Outline

- Background
- How things may go wrong
- A primal-dual formulation
- **A new kernel loss**
- Conclusions

Primal-dual Problems are Hard to Solve

Consider simplified case $|\mathcal{A}| = 1$ (can be extended to $|\mathcal{A}| > 1$)
(π and λ play no role now)
Restricting dual ν to Reproducing Kernel Hilbert Space (RKHS):

$$\min_{V} \max_{\nu \in \mathcal{H}_k} \quad \mathbb{E}_s\left[2\nu(s) \cdot \mathcal{R}V(s) - \nu(s)^2\right]$$

$$\Downarrow$$

$$\min_{V} \max_{\nu \in \mathcal{H}_k : \|\nu\|_{\mathcal{H}_k} \leq 1} \quad \mathbb{E}_s\left[2\nu(s) \cdot \mathcal{R}V(s)\right]$$

$$\Downarrow$$

$$\min_{V} \quad 2 \cdot \mathbb{E}_{s,\bar{s}}\left[K(s,\bar{s}) \cdot \mathcal{R}V(s) \cdot \mathcal{R}V(\bar{s})\right]$$

A New Loss for Solving Bellman Equation

$$\boxed{\min_V L_K(V) := \mathbb{E}_{s,\bar{s}}\left[K(s,\bar{s}) \cdot \mathcal{R}V(s) \cdot \mathcal{R}V(\bar{s})\right]}$$

- Well-defined objective without requiring double samples
- May be optimized by gradient methods w/ mini-batches (SGD/BackProp, ...)
- May be extended to the controlled case $|\mathcal{A}| > 1$

[Feng et al. 2019] https://arxiv.org/abs/1905.10506

Eigenfunction Interpretation

By Mercer's theorem

$$K(s, \bar{s}) = \sum_i \lambda_i e_i(s) e_i(\bar{s})$$

Proposition

$$L_K(V) = \sum_i \lambda_i \big(\mathbb{E}_s[\mathcal{R}V(s) \cdot e_i(s)] \big)^2$$

Thus, $L_K(V)$ is λ_i-weighted ℓ_2-norm in space spanned by $\{e_i\}$.
Difference choices of K lead to different $(\{\lambda_i\}, \{e_i\})$.
Example: RBF kernel favors *smooth* eigenfunctions.

Extension to $|\mathcal{A}| > 1$

Option 1

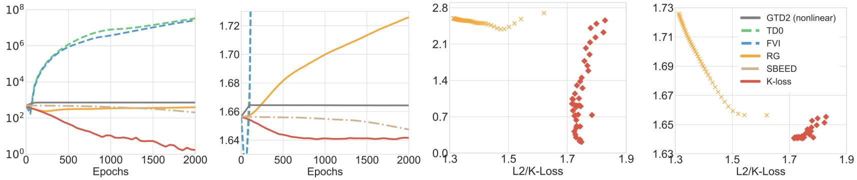
$$L_K(V, \pi) = \mathbb{E}_{s,\bar{s},a\sim\pi(s),\bar{a}\sim\bar{s}}[K([s,a],[\bar{s},\bar{a}]) \cdot \mathcal{R}_{\pi,\lambda} V(s,a) \cdot \mathcal{R}_{\pi,\lambda} V(\bar{s},\bar{a})]$$

Option 2

Use previous $L_K(V)$ for (fixed) policy evaluation within other algorithms (e.g., API, actor-critic, ...)

PuddleWorld with Neural Networks

PuddleWorld: classic divergence example (Boyan & Moore 95)



(a) MSE (b) Bellman Error (c) L2/K-Loss vs MSE (d) L2/K-Loss vs Bellman Error

MSE: $\left\|\hat{V} - V^*\right\|_2^2$

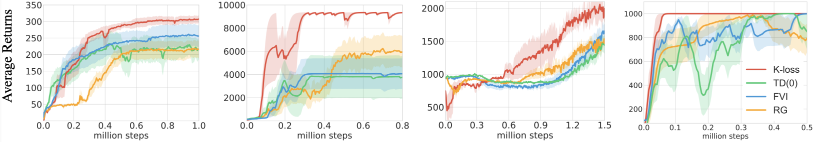
Bellman error: $\left\|\mathcal{R}\hat{V}\right\|_2^2$

L2 loss: squared TD error that suffers double-sample bias

Similar results in other classic problems: CartPole, MountainCar.

Results on Mujoco

Use our loss for learning V^π for fixed policy π, inside Trust-PCL (Nachum+ 18)



(a) Swimmer (b) InvertedDoublePendulum (c) Ant (d) InvertedPendulum

Figure 4: Results of various variants of Trust PCL on Mujoco Benchmark.

Outline

- Background
- How things may go wrong
- A primal-dual formulation
- A new kernel loss
- **Conclusions**

Conclusions

- Modern RL requires going beyond tabular/linear cases
- Convergence conditions for DP-based approaches often brittle
- Promising and fruitful direction: DP $\longrightarrow$ OPT
  - different formulations as optimization problems
  - new algorithms with provable convergence and stronger guarantees
  - more transparent behavior (using established stats/ML techniques)
  - potentially make RL easier to use in practice

An Upcoming Workshop

Optimization Foundation for Reinforcement Learning

Workshop at NeurIPS, Dec 13-14th. 2019, Vancouver, Canada

Background

Dynamic programming (DP) based algorithms, which apply various forms of the Bellman operator, dominate the literature on model-free reinforcement learning (RL). While DP is powerful, the value function estimate can oscillate or even diverge when function approximation is introduced with off-policy data, except in special cases [1-8]. This problem has been well-known for decades (referred to as the deadly triad in the literature), and has remained a critical open fundamental problem in RL.

More recently, the community witnessed a fast-growing trend that frames RL problems as well-posed optimization problems, in which a proper objective function is proposed whose minimization results in the optimal value function [9-28]. Such an optimization-based approach provides a promising perspective that brings mature mathematical tools to bear on integrating linear/nonlinear function approximation with off-policy data, while avoiding DP's inherent instability. Moreover, the optimization perspective is naturally extensible to incorporating constraints, sparsity regularization, distributed multi-agent scenarios, and other new settings.

In addition to being able to apply powerful optimization techniques to a variety of RL problems, the special recursive structure and restricted exploration sampling in RL also naturally raises the question of whether tailored algorithms can be developed to improve sample efficiency, convergence rates, and asymptotic performance, under the guidance of the established optimization techniques.

The goal of this workshop is to catalyze the collaboration between reinforcement learning and optimization communities, pushing the boundaries from both sides. It will provide a forum for establishing a mutually accessible introduction to current research on this integration, and allow exploration of recent advances in optimization for potential application in reinforcement learning. It will also be a window to identify and discuss existing challenges and forward-looking problems of interest in reinforcement learning to the optimization community.

https://optrl2019.github.io

THANK YOU !

APPENDIX

Online SBEED Learning with Experience Replay

Algorithm 1 Online SBEED learning with experience replay

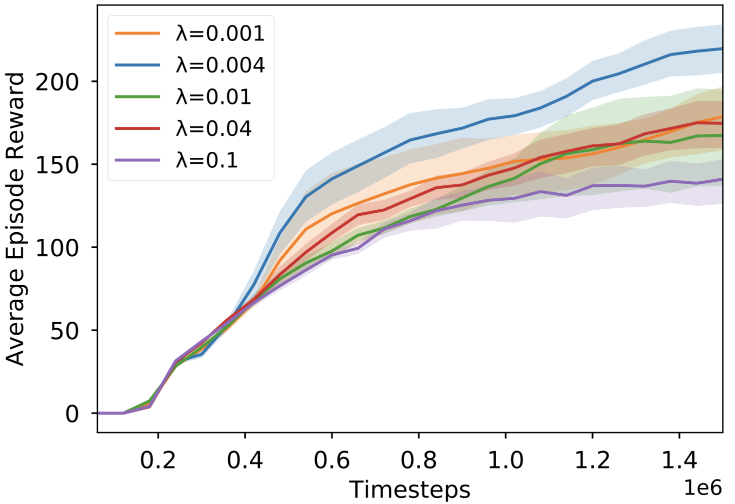
```
Initialize w = (w_V, w_π, w_ρ) and π_b randomly, set ϵ.
for episode i = 1, ..., T do
    for size k = 1, ..., K do
        Add new transition (s, a, r, s') into D by executing behavior policy π_b.
    end for
    for iteration j = 1, ..., N do
        Update w_ρ^j by solving
            min_{w_ρ} Ê_{{s,a,s'}~D} [ (δ(s, a, s') − ρ(s, a))^2 ].
        Decay the stepsize ζ_j in rate O(1/j).
        Compute the stochastic gradients w.r.t. w_V and w_π as ∇̂_{w_V} ℓ̄(V, π) and ∇̂_{w_π} ℓ̄(V, π).
        Update the parameters of primal function by solving the prox-mappings, i.e.,
            update V:  w_V^j = P_{w_V^{j−1}}(ζ_j ∇̂_{w_V} ℓ̄(V, π))
            update π:  w_π^j = P_{w_π^{j−1}}(ζ_j ∇̂_{w_π} ℓ̄(V, π))
    end for
    Update behavior policy π_b = π^N.
end for
```

Experiments

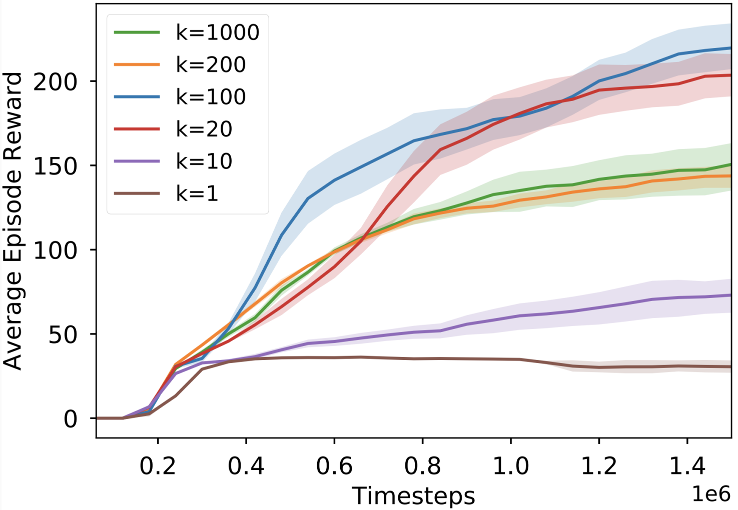
- Use Mujoco on OpenAI as benchmark
- Compare to state-of-the-art baselines:
  - Dual-AC (Dai et al. 18)
  - TRPO (Schulman et al. 15)
  - DDPG (Lillicrap et al. 15)

(from http://www.mujoco.org)

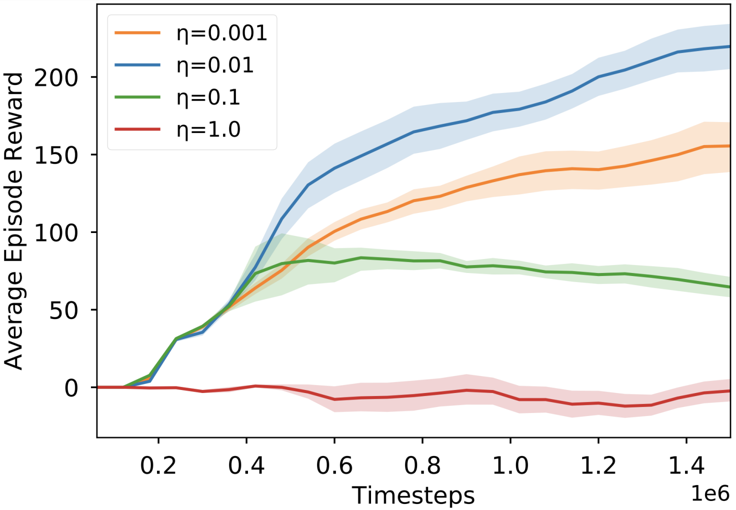
Role of Smoothing Parameter λ



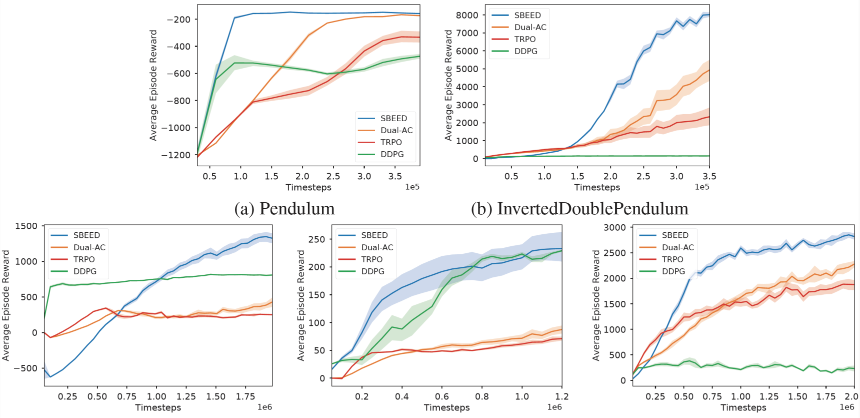
Role of Bootstrapping Distance k



Role of Dual Embedding η



Comparison against Baselines



(a) Pendulum (b) InvertedDoublePendulum

(c) HalfCheetah (d) Swimmer (e) Hopper