GradientDICE: Rethinking Generalized Offline Estimation of Stationary Values

Shangtong Zhang[1], Bo Liu[2], Shimon Whiteson[1]

[1] University of Oxford
[2] Auburn University

Preview

- Off-policy evaluation with density ratio learning
- Use the Perron-Frobenius theorem to reduce the constraints from 3 to 2, reducing the positiveness constraint, making the problem convex in both tabular and linear setting
- A special weighted L_2 norm
- Improvements over DualDICE and GenDICE in tabular, linear and neural network settings

Off-policy evaluation is to estimate the performance of a policy with off-policy data

- The target policy π
- A data set $\{s_i, a_i, r_i, s_i'\}_{i=1,\ldots,N}$
  - $s_i, a_i \sim d_\mu(s, a), r_i = r(s_i, a_i), s_i' \sim p(\cdot \mid s_i, a_i)$
- The performance metric $\rho_\gamma(\pi) \doteq \sum_{s,a} d_\gamma(s, a) r(s, a)$
  - $d_\gamma(s, a) \doteq (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \Pr(S_t = s, A_t = a \mid \pi, p) \quad (\gamma < 1)$
  - $d_\gamma(s, a) \doteq \lim_{t \to \infty} \Pr(S_t = s, A_t = a \mid \pi, p) \quad (\gamma = 1)$

Density ratio learning is promising for off-policy evaluation (Liu et al, 2018)

- Learn $\tau_*(s,a) \doteq \frac{d_\gamma(s,a)}{d_\mu(s,a)}$ with function approximation
- $\rho_\gamma(\pi) = \sum_{s,a} d_\mu(s,a)\tau_*(s,a)r(s,a) \approx \frac{1}{N}\sum_{i=1}^{N} \tau_*(s_i,a_i)r_i$

Density ratio satisfies a Bellman-like equation (Zhang et al, 2020)

- $D\tau_* = (1-\gamma)\mu_0 + \gamma P_\pi^\top D\tau_*$
  - $D \in \mathbb{R}^{N_{sa}\times N_{sa}}, D \doteq \text{diag}(d_\mu)$
  - $\tau_* \in \mathbb{R}^{N_{sa}}$
  - $\mu_0 \in \mathbb{R}^{N_{sa}}, \mu_0(s,a) \doteq \mu_0(s)\pi(a\,|\,s)$
  - $P_\pi \in \mathbb{R}^{N_{sa}\times N_{sa}}, P_\pi((s,a),(s',a')) \doteq p(s'\,|\,s,a)\pi(a'\,|\,s')$

$\gamma < 1$ is easy as it implies a unique solution

- $D\tau = (1 - \gamma)\mu_0 + \gamma P_\pi^\top D\tau$
  - $(I - \gamma P_\pi^\top)^{-1}$ exists

Previous work requires three constraints for $\gamma = 1$

1. $D\tau = P_\pi^\top D\tau$

2. $D\tau \succ 0$

3. $\mathbf{1}^\top D\tau = 1$

GenDICE (Zhang et al, 2020) considers 1 & 3 explicitly

$$L(\tau) \doteq \textbf{divergence}(D\tau, P_\pi^\top D\tau) + (1 - \mathbf{1}^\top D\tau)^2$$

and implements 2 with positive function approximation (e.g. τ^2, e^τ), projected SGD, or stochastic mirror descent

Mousavi et al. (2020) implements 3 with self-normalization over all state-action pairs

Previous work requires three constraints for $\gamma = 1$

1. $D\tau = P_\pi^\top D\tau$

2. $D\tau \succ 0$

3. $1^\top D\tau = 1$

The objective becomes non-convex with positive function approximation or self-normalization, even in tabular or linear setting.

Projected SGD is computationally infeasible.

Stochastic mirror descent significantly reduces the capacity of the (linear) function class.

We actually need only two constraints!

1. $D\tau = P_\pi^\top D\tau$

2. $D\tau \succ 0$

3. $\mathsf{1}^\top D\tau = 1$

Perron-Frobenius theorem: the solution space of 1 is one-dimensional
Either 2 or 3 is enough to guarantee a unique solution

GradientDICE considers a special L_2 norm for the loss

- GenDICE:
  $L(\tau) \doteq \text{divergence}((1-\gamma)\mu_0 + \gamma P_\pi^\top D\tau, D\tau) + (1 - 1^\top D\tau)^2$
  subject to $\quad Dy \succ 0$

- $L(\tau) \doteq ||(1-\gamma)\mu_0 + \gamma P_\pi^\top D\tau - D\tau||_{D^{-1}} + (1 - 1^\top D\tau)^2$

  - GradientTD loss: $||\ldots||_D$

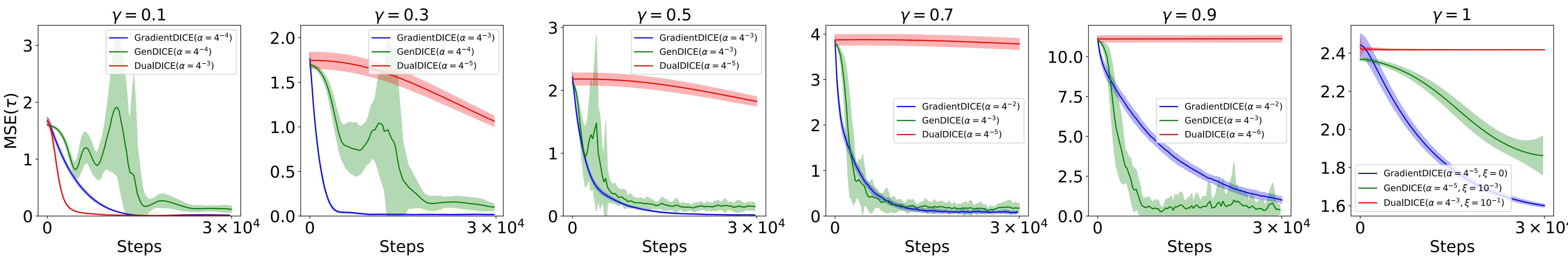
GradientDICE considers a special L_2 norm for the loss

- $L(\tau) \doteq ||(1-\gamma)\mu_0 + \gamma P_\pi^\top D\tau - D\tau||_{D^{-1}} + (1 - 1^\top D\tau)^2$

$$\min_{\tau \in \mathbb{R}^{N_{sa}}} \max_{f \in \mathbb{R}^{N_{sa}}, \eta \in \mathbb{R}} L(\tau, \eta, f) \doteq (1-\gamma)\mathbb{E}_{\mu_0}[f(s,a)]$$
$$+\gamma \mathbb{E}_p[\tau(s,a)f(s',a')]$$
$$-\mathbb{E}_{d_\mu}[\tau(s,a)f(s,a)]$$
$$-\frac{1}{2}\mathbb{E}_{d_\mu}[f(s,a)^2]$$
$$+\lambda\left(\mathbb{E}_{d_\mu}[\eta\tau(s,a) - \eta] - \frac{\eta^2}{2}\right)$$

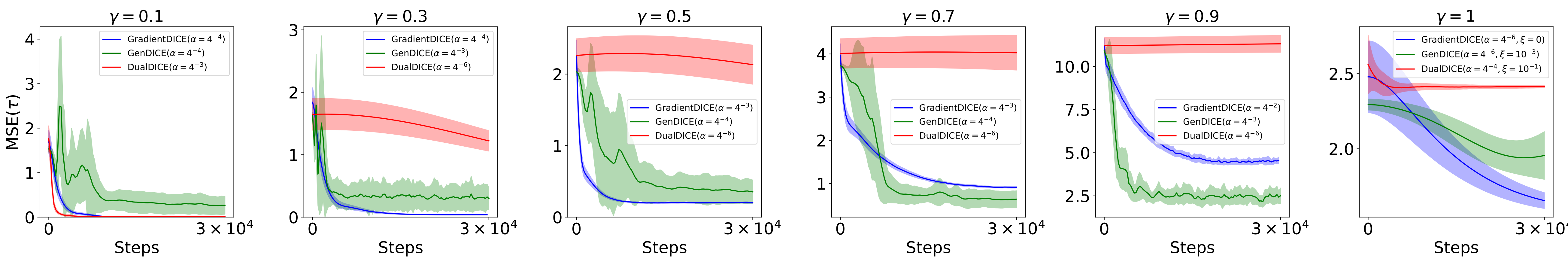
- Convergence in both tabular and linear setting with $\gamma \in [0,1]$

GradientDICE outperforms baselines in Boyan's Chain (Tabular)



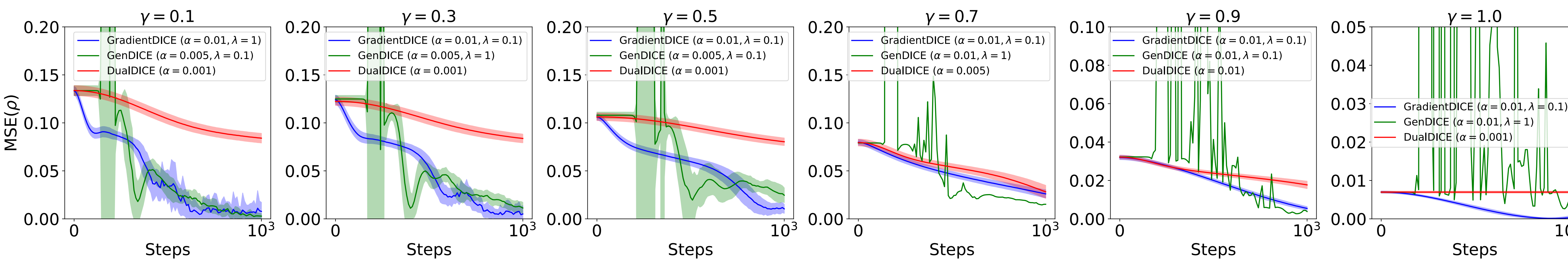
- 30 runs (mean + standard errors)
- Grid Search for hyperparameters, e.g., learning rates from $\{4^{-6}, 4^{-5}, \ldots, 4^{-1}\}$
- Tuned to minimize final prediction error

GradientDICE outperforms baselines in Boyan's Chain (Linear)



- 30 runs (mean + standard errors)
- Grid Search for hyperparameters, e.g., learning rates from $\{4^{-6}, 4^{-5}, \ldots, 4^{-1}\}$
- Tuned to minimize final prediction error

GradientDICE outperforms baselines in Reacher-v2 (Network)



- 30 runs (mean + standard errors)

- Grid Search for hyperparameters, e.g.,
  Learning rates from $\{0.01, 0.005, 0.001\}$
  Penalty from $\{0.1, 1\}$

- Tuned to minimize final prediction error

Thanks

- Code and Dockerfile are available at https://github.com/ShangtongZhang/DeepRL