Target-based Surrogates for Efficient Sequential Decision-making

Sharan Vaswani (Simon Fraser University)

Based on work with Olivier Bachem, Simone Totaro, Robert Müller, Shivam Garg, Matthieu Geist, Marlos Machado, Pablo Samuel Castro, Jonathan Wilder Lavington, Reza Babanezhad, Mark Schmidt, Nicolas Le Roux

Microsoft Research, Montreal

Problem Formulation

- Optimize functions with a composition structure $h(\theta) := \ell(f(\theta))$.
- Canonical example: Supervised learning where $\ell(z)$ is the loss function (squared loss, cross-entropy) and $z = f(\theta)$ is the model (linear model, neural network).

Problem Formulation

- Optimize functions with a composition structure $h(\theta) := \ell(f(\theta))$.
- Canonical example: Supervised learning where $\ell(z)$ is the loss function (squared loss, cross-entropy) and $z = f(\theta)$ is the model (linear model, neural network).
- Linear Regression: $h(\theta) = \frac{1}{2} \|X\theta - y\|_2^2 = \ell(f(\theta))$ where $\ell(z) = \frac{1}{2} \|z - y\|_2^2$ and $z = f(\theta) = X\theta$.

Problem Formulation

- Optimize functions with a composition structure $h(\theta) := \ell(f(\theta))$.
- Canonical example: Supervised learning where $\ell(z)$ is the loss function (squared loss, cross-entropy) and $z = f(\theta)$ is the model (linear model, neural network).
- Linear Regression: $h(\theta) = \frac{1}{2}\|X\theta - y\|_2^2 = \ell(f(\theta))$ where $\ell(z) = \frac{1}{2}\|z - y\|_2^2$ and $z = f(\theta) = X\theta$.
- $\min_\theta \ell(f(\theta))$ is equivalent to $\min_{z \in \mathcal{Z}_\theta} \ell(z)$ where $\mathcal{Z}_\theta := \{z | \exists \theta \text{ s.t } z = f(\theta)\}$.

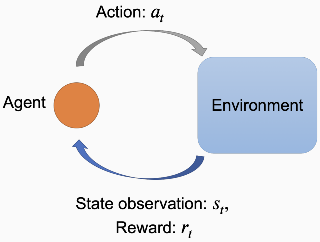
Problem Formulation

- Optimize functions with a composition structure $h(\theta) := \ell(f(\theta))$.
- Canonical example: Supervised learning where $\ell(z)$ is the loss function (squared loss, cross-entropy) and $z = f(\theta)$ is the model (linear model, neural network).
- Linear Regression: $h(\theta) = \frac{1}{2} \|X\theta - y\|_2^2 = \ell(f(\theta))$ where $\ell(z) = \frac{1}{2} \|z - y\|_2^2$ and $z = f(\theta) = X\theta$.
- $\min_\theta \ell(f(\theta))$ is equivalent to $\min_{z \in \mathcal{Z}_\theta} \ell(z)$ where $\mathcal{Z}_\theta := \{z | \exists \theta \text{ s.t } z = f(\theta)\}$.

Focus on problems in sequential decision-making where computing $\{\ell(z), \nabla_z \ell(z)\}$ can be much more computationally expensive compared to $\{f(\theta), \nabla_\theta f(\theta)\}$

Motivating example: Policy Optimization in Reinforcement Learning

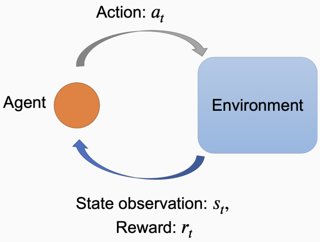
Objective: Given a Markov decision process (MDP) with state space $\mathcal{S}$ and action space $\mathcal{A}$, learn a policy $\pi : \mathcal{S} \to \Delta_A$ that maximizes the expected cumulative discounted reward $J(\pi) := \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots]$.



- Policy π is parameterized by a model f with parameters θ such that $\pi = f(\theta)$.

Motivating example: Policy Optimization in Reinforcement Learning

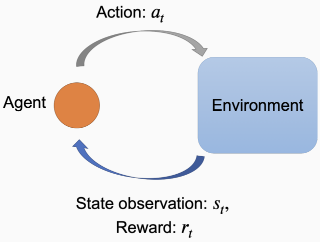
Objective: Given a Markov decision process (MDP) with state space $\mathcal{S}$ and action space $\mathcal{A}$, learn a policy $\pi : \mathcal{S} \to \Delta_A$ that maximizes the expected cumulative discounted reward $J(\pi) := \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots]$.



- Policy π is parameterized by a model f with parameters θ such that $\pi = f(\theta)$.
- Computing $\{J(\pi), \nabla_\pi J(\pi)\}$ requires interacting with the environment and is computationally expensive.

Motivating example: Policy Optimization in Reinforcement Learning

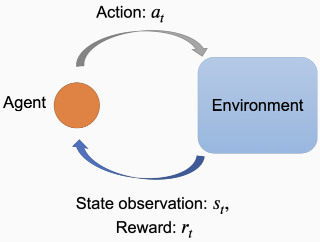
Objective: Given a Markov decision process (MDP) with state space $\mathcal{S}$ and action space $\mathcal{A}$, learn a policy $\pi : \mathcal{S} \to \Delta_A$ that maximizes the expected cumulative discounted reward $J(\pi) := \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots]$.



- Policy π is parameterized by a model f with parameters θ such that $\pi = f(\theta)$.
- Computing $\{J(\pi), \nabla_\pi J(\pi)\}$ requires interacting with the environment and is computationally expensive.
- **Generic Algorithm**: Starting from an initial policy π_0, at every iteration t,
  - Use the current policy π_t to interact with the environment and gather data (Slow step)

Motivating example: Policy Optimization in Reinforcement Learning

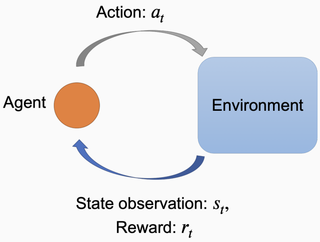
Objective: Given a Markov decision process (MDP) with state space $\mathcal{S}$ and action space $\mathcal{A}$, learn a policy $\pi : \mathcal{S} \to \Delta_A$ that maximizes the expected cumulative discounted reward $J(\pi) := \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots]$.



- Policy π is parameterized by a model f with parameters θ such that $\pi = f(\theta)$.
- Computing $\{J(\pi), \nabla_\pi J(\pi)\}$ requires interacting with the environment and is computationally expensive.
- **Generic Algorithm**: Starting from an initial policy π_0, at every iteration t,
  - Use the current policy π_t to interact with the environment and gather data (Slow step)
  - Use the gathered data to update the policy by solving: $\max_{\pi \in \Pi_\theta} J(\pi) = \max_\theta J(f(\theta))$ where Π_θ is the set of policies that can be expressed by the model.

Motivating example: Policy Optimization in Reinforcement Learning

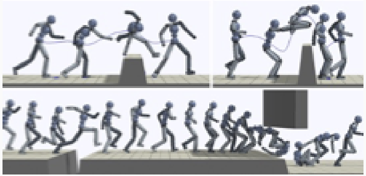
Objective: Given a Markov decision process (MDP) with state space $\mathcal{S}$ and action space $\mathcal{A}$, learn a policy $\pi : \mathcal{S} \to \Delta_A$ that maximizes the expected cumulative discounted reward $J(\pi) := \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots]$.



- Policy π is parameterized by a model f with parameters θ such that $\pi = f(\theta)$.
- Computing $\{J(\pi), \nabla_\pi J(\pi)\}$ requires interacting with the environment and is computationally expensive.
- **Generic Algorithm**: Starting from an initial policy π_0, at every iteration t,
  - Use the current policy π_t to interact with the environment and gather data (Slow step)
  - Use the gathered data to update the policy by solving: $\max_{\pi \in \Pi_\theta} J(\pi) = \max_\theta J(f(\theta))$ where Π_θ is the set of policies that can be expressed by the model.
- If $z = \pi$, $\mathcal{Z}_\theta = \Pi_\theta$, $\ell = -J$, equivalent to solving $\min_\theta h(\theta) = \ell(f(\theta))$.

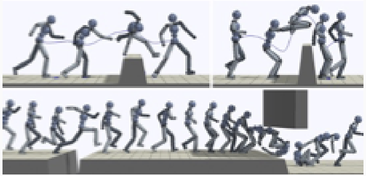
Motivating example: Online Imitation Learning

Objective: Learn a policy that tries to "imitate" the expert. E.g: Learning to control a robot from human supervision.



Motivating example: Online Imitation Learning

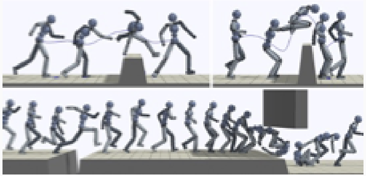
Objective: Learn a policy that tries to "imitate" the expert. E.g: Learning to control a robot from human supervision.



- Policy π is parameterized by a model f with parameters θ such that $\pi = f(\theta)$.

Motivating example: Online Imitation Learning

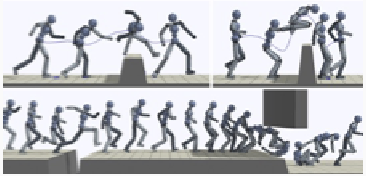
Objective: Learn a policy that tries to "imitate" the expert. E.g: Learning to control a robot from human supervision.



- Policy π is parameterized by a model f with parameters θ such that $\pi = f(\theta)$.
- **Generic Algorithm**:: Starting from an initial policy π_0, at every iteration t,
  - Use the current policy π_t to interact with the environment and receive feedback from the expert (Slow step).

Motivating example: Online Imitation Learning

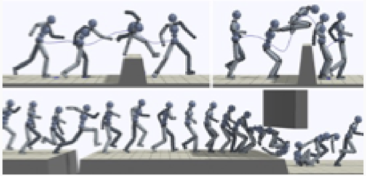
Objective: Learn a policy that tries to "imitate" the expert. E.g: Learning to control a robot from human supervision.



- Policy π is parameterized by a model f with parameters θ such that $\pi = f(\theta)$.
- **Generic Algorithm**:: Starting from an initial policy π_0, at every iteration t,
  - Use the current policy π_t to interact with the environment and receive feedback from the expert (Slow step).
  - Update the policy to minimize the discrepancy D to the expert policy π_e: $\min_{\pi \in \Pi_\theta} \ell_t(\pi) := \mathbb{E}_{s \sim d^{\pi_t}} \left[D(\pi(\cdot|s) || \pi_e(\cdot|s)) \right]$.

Motivating example: Online Imitation Learning

Objective: Learn a policy that tries to "imitate" the expert. E.g: Learning to control a robot from human supervision.



- Policy π is parameterized by a model f with parameters θ such that $\pi = f(\theta)$.
- **Generic Algorithm**:: Starting from an initial policy π_0, at every iteration t,
  - Use the current policy π_t to interact with the environment and receive feedback from the expert (Slow step).
  - Update the policy to minimize the discrepancy D to the expert policy π_e: $\min_{\pi \in \Pi_\theta} \ell_t(\pi) := \mathbb{E}_{s \sim d^{\pi_t}} \left[D(\pi(\cdot|s) || \pi_e(\cdot|s)) \right]$.
- If $z = \pi$, $\mathcal{Z}_\theta = \Pi_\theta$, equivalent to minimizing a sequence of functions of the form $h_t(\theta) := \ell_t(f(\theta))$.

Outline

- **Target-based Surrogate Optimization**
- Functional Mirror Ascent for Policy Gradient (FMA-PG)
- Conclusions and Future Work

Naive Idea: Parametric Gradient Descent

- Ignore the composition structure and optimize $h(\theta) = \ell(f(\theta))$ directly w.r.t θ.
- Parametric Gradient Descent: $\theta_{t+1} = \theta_t - \eta \nabla h(\theta_t) \quad ; \quad z_{t+1} = f(\theta_{t+1})$.
- Example: In RL, this approach is broadly referred to as policy gradient.

Naive Idea: Parametric Gradient Descent

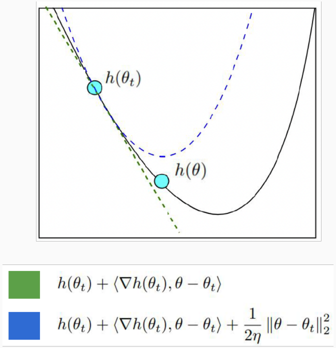
- Ignore the composition structure and optimize $h(\theta) = \ell(f(\theta))$ directly w.r.t θ.
- Parametric Gradient Descent: $\theta_{t+1} = \theta_t - \eta \nabla h(\theta_t) \quad ; \quad z_{t+1} = f(\theta_{t+1})$.
- Example: In RL, this approach is broadly referred to as policy gradient.

× Each policy update requires computing $\nabla_\theta h(\theta) = \nabla_z \ell(z) \nabla_\theta f(\theta)$. Since computing $\nabla_z \ell(z)$ involves interacting with the environment, it is computationally expensive.

Naive Idea: Parametric Gradient Descent

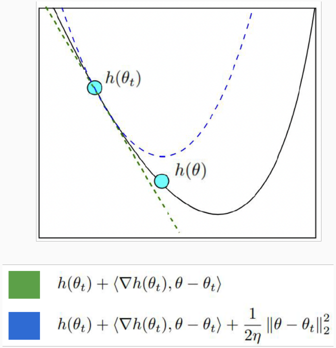
- Ignore the composition structure and optimize $h(\theta) = \ell(f(\theta))$ directly w.r.t θ.
- Parametric Gradient Descent: $\theta_{t+1} = \theta_t - \eta \nabla h(\theta_t) \quad ; \quad z_{t+1} = f(\theta_{t+1})$.
- Example: In RL, this approach is broadly referred to as policy gradient.
- × Each policy update requires computing $\nabla_\theta h(\theta) = \nabla_z \ell(z) \nabla_\theta f(\theta)$. Since computing $\nabla_z \ell(z)$ involves interacting with the environment, it is computationally expensive.
- Idea: Form a surrogate function that exploits the composition structure and enables "reusing" the gathered data i.e can update the policy multiple times without computing $\nabla_z \ell(z)$.
- Example: In RL, methods such as TRPO/PPO construct such surrogate functions and update the policy to maximize the surrogates.

- At iteration t, gradient descent on a smooth (possibly non-convex) function $h(\theta)$ is equivalent to minimizing a local quadratic surrogate function around θ_t:
  $g_t(\theta) := h(\theta_t) + \langle \nabla h(\theta_t), \theta - \theta_t \rangle + \frac{1}{2\eta} \|\theta - \theta_t\|_2^2$.



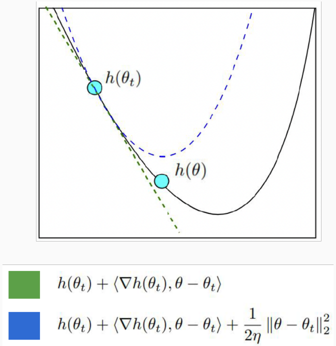
Digression: Parametric GD as Surrogate Minimization

- At iteration t, gradient descent on a smooth (possibly non-convex) function $h(\theta)$ is equivalent to minimizing a local quadratic surrogate function around θ_t:
  $g_t(\theta) := h(\theta_t) + \langle \nabla h(\theta_t), \theta - \theta_t \rangle + \frac{1}{2\eta} \|\theta - \theta_t\|_2^2$.
- For "small" enough η, $\forall \theta$, $g_t(\theta) \geq h(\theta)$ i.e. the surrogate is a global upper-bound on $h(\theta)$.



Digression: Parametric GD as Surrogate Minimization

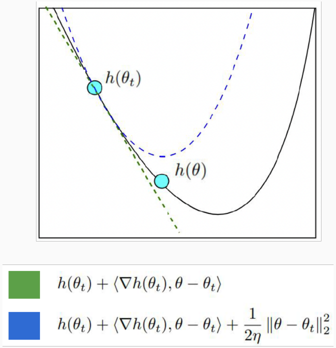
- At iteration t, gradient descent on a smooth (possibly non-convex) function $h(\theta)$ is equivalent to minimizing a local quadratic surrogate function around θ_t:
  $g_t(\theta) := h(\theta_t) + \langle \nabla h(\theta_t), \theta - \theta_t \rangle + \frac{1}{2\eta} \|\theta - \theta_t\|_2^2$.
- For "small" enough η, $\forall \theta$, $g_t(\theta) \geq h(\theta)$ i.e. the surrogate is a global upper-bound on $h(\theta)$.



- Update: $\theta_{t+1} = \arg\min_\theta g_t(\theta) \implies \theta_{t+1} = \theta_t - \eta \nabla h(\theta_t)$.

Digression: Parametric GD as Surrogate Minimization

- At iteration t, gradient descent on a smooth (possibly non-convex) function $h(\theta)$ is equivalent to minimizing a local quadratic surrogate function around θ_t:
  $g_t(\theta) := h(\theta_t) + \langle \nabla h(\theta_t), \theta - \theta_t \rangle + \frac{1}{2\eta} \|\theta - \theta_t\|_2^2$.
- For "small" enough η, $\forall \theta$, $g_t(\theta) \geq h(\theta)$ i.e. the surrogate is a global upper-bound on $h(\theta)$.



- Update: $\theta_{t+1} = \arg\min_\theta g_t(\theta) \implies \theta_{t+1} = \theta_t - \eta \nabla h(\theta_t)$.
- Algorithm: Iteratively form the surrogate $g_t(\theta)$ around θ_t and minimize it to update θ. Exactly gradient descent!

Target-based Surrogate Minimization

- At iteration t, gradient descent on a smooth (possibly non-convex) function $\ell(z)$ is equivalent to minimizing a local quadratic surrogate function around z_t:

$$g_t(z) := \ell(z_t) + \langle \nabla_z \ell(z_t), z - z_t \rangle + \frac{1}{2\eta} \|z - z_t\|_2^2$$

.

Target-based Surrogate Minimization

- At iteration t, gradient descent on a smooth (possibly non-convex) function $\ell(z)$ is equivalent to minimizing a local quadratic surrogate function around z_t:

$$g_t(z) := \ell(z_t) + \langle \nabla_z \ell(z_t), z - z_t \rangle + \frac{1}{2\eta} \|z - z_t\|_2^2$$

- For "small" enough η, $\forall z, g_t(z) \geq \ell(z)$ i.e. the surrogate is a global upper-bound on $\ell(z)$.

Target-based Surrogate Minimization

- At iteration t, gradient descent on a smooth (possibly non-convex) function $\ell(z)$ is equivalent to minimizing a local quadratic surrogate function around z_t:

$$g_t(z) := \ell(z_t) + \langle \nabla_z \ell(z_t), z - z_t \rangle + \frac{1}{2\eta} \|z - z_t\|_2^2$$

- For "small" enough η, $\forall z, g_t(z) \geq \ell(z)$ i.e. the surrogate is a global upper-bound on $\ell(z)$.
- Update: $z_{t+1} = \arg\min_{z \in \mathcal{Z}_\theta} g_t(z)$. Since $z = f(\theta)$ for all $z \in \mathcal{Z}_\theta$,

$$\theta_{t+1} = \arg\min g_t(\theta) := \ell(f(\theta_t)) + \langle \nabla_z \ell(f(\theta_t)), f(\theta) - f(\theta_t) \rangle + \frac{1}{2\eta} \|f(\theta) - f(\theta_t)\|_2^2$$

.

Target-based Surrogate Minimization

- At iteration t, gradient descent on a smooth (possibly non-convex) function $\ell(z)$ is equivalent to minimizing a local quadratic surrogate function around z_t:

$$g_t(z) := \ell(z_t) + \langle \nabla_z \ell(z_t), z - z_t \rangle + \frac{1}{2\eta} \|z - z_t\|_2^2$$

- For "small" enough η, $\forall z, g_t(z) \geq \ell(z)$ i.e. the surrogate is a global upper-bound on $\ell(z)$.
- Update: $z_{t+1} = \arg\min_{z \in \mathcal{Z}_\theta} g_t(z)$. Since $z = f(\theta)$ for all $z \in \mathcal{Z}_\theta$,

$$\theta_{t+1} = \arg\min g_t(\theta) := \ell(f(\theta_t)) + \langle \nabla_z \ell(f(\theta_t)), f(\theta) - f(\theta_t) \rangle + \frac{1}{2\eta} \|f(\theta) - f(\theta_t)\|_2^2$$

- Algorithm: Iteratively form the surrogate $g_t(\theta)$ around z_t, minimize it w.r.t θ and update $z_{t+1} = f(\theta_{t+1})$.

Target-based Surrogate Minimization

- At iteration t, gradient descent on a smooth (possibly non-convex) function $\ell(z)$ is equivalent to minimizing a local quadratic surrogate function around z_t:

$$g_t(z) := \ell(z_t) + \langle \nabla_z \ell(z_t), z - z_t \rangle + \frac{1}{2\eta} \|z - z_t\|_2^2$$

- For "small" enough η, $\forall z, g_t(z) \geq \ell(z)$ i.e. the surrogate is a global upper-bound on $\ell(z)$.
- Update: $z_{t+1} = \arg\min_{z \in \mathcal{Z}_\theta} g_t(z)$. Since $z = f(\theta)$ for all $z \in \mathcal{Z}_\theta$,

$$\theta_{t+1} = \arg\min g_t(\theta) := \ell(f(\theta_t)) + \langle \nabla_z \ell(f(\theta_t)), f(\theta) - f(\theta_t) \rangle + \frac{1}{2\eta} \|f(\theta) - f(\theta_t)\|_2^2$$

- Algorithm: Iteratively form the surrogate $g_t(\theta)$ around z_t, minimize it w.r.t θ and update $z_{t+1} = f(\theta_{t+1})$.
- Equivalent to projected gradient descent in the target space: $z_{t+1} = \text{Proj}_{\mathcal{Z}_\theta}[z_t - \eta \nabla \ell(z_t)]$

Target-based Surrogate Minimization

Recall that $g_t(\theta) := \ell(f(\theta_t)) + \langle \nabla_z \ell(f(\theta_t)), f(\theta) - f(\theta_t) \rangle + \frac{1}{2\eta} \|f(\theta) - f(\theta_t)\|_2^2$

- Since $g_t(\theta)$ is a potentially non-convex function, we cannot optimize it exactly.

Target-based Surrogate Minimization

Recall that $g_t(\theta) := \ell(f(\theta_t)) + \langle \nabla_z \ell(f(\theta_t)), f(\theta) - f(\theta_t) \rangle + \frac{1}{2\eta} \|f(\theta) - f(\theta_t)\|_2^2$

- Since $g_t(\theta)$ is a potentially non-convex function, we cannot optimize it exactly.
- Idea: Optimize $g_t(\theta)$ using m_t steps of gradient descent w.r.t θ.

Algorithm 1 Surrogate minimization

Input: θ_0 (initialization), T (number of iterations), m_t (number of inner-loops), η (step-size for the target space), α (step-size for the parametric space)

for $t = 0$ to $T - 1$ **do**
  Access the gradient oracle to construct $g_t(\theta)$
  Initialize inner-loop: $\omega_0 = \theta_t$
  for $k \leftarrow 0$ to m_{t-1} **do**
    $\omega_{k+1} = \omega_k - \alpha \nabla_\omega g_t(\omega_k)$
  end for
  $\theta_{t+1} = \omega_m \quad ; \quad z_{t+1} = f(\theta_{t+1})$
end for
Return θ_T

Target-based Surrogate Minimization

Recall that $g_t(\theta) := \ell(f(\theta_t)) + \langle \nabla_z \ell(f(\theta_t)), f(\theta) - f(\theta_t) \rangle + \frac{1}{2\eta} \|f(\theta) - f(\theta_t)\|_2^2$

- Since $g_t(\theta)$ is a potentially non-convex function, we cannot optimize it exactly.
- Idea: Optimize $g_t(\theta)$ using m_t steps of gradient descent w.r.t θ.
- Forming $g_t(\theta)$ requires access to $\nabla_z \ell(z)$, but optimizing $g_t(\theta)$ does not. Inner-loop can update θ without accessing the expensive gradient oracle.

Algorithm 1 Surrogate minimization

Input: θ_0 (initialization), T (number of iterations), m_t (number of inner-loops), η (step-size for the target space), α (step-size for the parametric space)

for $t = 0$ to $T - 1$ **do**
  Access the gradient oracle to construct $g_t(\theta)$
  Initialize inner-loop: $\omega_0 = \theta_t$
  for $k \leftarrow 0$ to m_{t-1} **do**
    $\omega_{k+1} = \omega_k - \alpha \nabla_\omega g_t(\omega_k)$
  end for
  $\theta_{t+1} = \omega_m \quad ; \quad z_{t+1} = f(\theta_{t+1})$
end for
Return θ_T

Target-based Surrogate Minimization

Recall that $g_t(\theta) := \ell(f(\theta_t)) + \langle \nabla_z \ell(f(\theta_t)), f(\theta) - f(\theta_t) \rangle + \frac{1}{2\eta} \|f(\theta) - f(\theta_t)\|_2^2$

- Since $g_t(\theta)$ is a potentially non-convex function, we cannot optimize it exactly.
- Idea: Optimize $g_t(\theta)$ using m_t steps of gradient descent w.r.t θ.
- Forming $g_t(\theta)$ requires access to $\nabla_z \ell(z)$, but optimizing $g_t(\theta)$ does not. Inner-loop can update θ without accessing the expensive gradient oracle.
- Example: In RL, the inner-loop updates to the policy parameters are referred to as off-policy updates.

Algorithm 1 Surrogate minimization

Input: θ_0 (initialization), T (number of iterations), m_t (number of inner-loops), η (step-size for the target space), α (step-size for the parametric space)

for $t = 0$ to $T - 1$ **do**
  Access the gradient oracle to construct $g_t(\theta)$
  Initialize inner-loop: $\omega_0 = \theta_t$
  for $k \leftarrow 0$ to m_{t-1} **do**
    $\omega_{k+1} = \omega_k - \alpha \nabla_\omega g_t(\omega_k)$
  end for
  $\theta_{t+1} = \omega_m \quad ; \quad z_{t+1} = f(\theta_{t+1})$
end for
Return θ_T

Target-based Surrogate Minimization

Recall that $g_t(\theta) := \ell(f(\theta_t)) + \langle \nabla_z \ell(f(\theta_t)), f(\theta) - f(\theta_t) \rangle + \frac{1}{2\eta} \|f(\theta) - f(\theta_t)\|_2^2$

- Since $g_t(\theta)$ is a potentially non-convex function, we cannot optimize it exactly.
- Idea: Optimize $g_t(\theta)$ using m_t steps of gradient descent w.r.t θ.
- Forming $g_t(\theta)$ requires access to $\nabla_z \ell(z)$, but optimizing $g_t(\theta)$ does not. Inner-loop can update θ without accessing the expensive gradient oracle.
- Example: In RL, the inner-loop updates to the policy parameters are referred to as off-policy updates.
- Using $m = 1$ in the Alg 1. recovers parametric GD.

Algorithm 1 Surrogate minimization

Input: θ_0 (initialization), T (number of iterations), m_t (number of inner-loops), η (step-size for the target space), α (step-size for the parametric space)

```
for t = 0 to T − 1 do
    Access the gradient oracle to construct g_t(θ)
    Initialize inner-loop: ω_0 = θ_t
    for k ← 0 to m_{t−1} do
        ω_{k+1} = ω_k − α∇_ω g_t(ω_k)
    end for
    θ_{t+1} = ω_m    ;    z_{t+1} = f(θ_{t+1})
end for
Return θ_T
```

Target-based Surrogate Minimization – Example

Recall that $g_t(\theta) := \ell(f(\theta_t)) + \langle \nabla_z \ell(f(\theta_t)), f(\theta) - f(\theta_t) \rangle + \frac{1}{2\eta} \left\| f(\theta) - f(\theta_t) \right\|_2^2$

- To gain some intuition, consider linear regression: $\ell(z) = \frac{1}{2} \left\| z - y \right\|_2^2 \;;\; z = f(\theta) = X\theta$.

Target-based Surrogate Minimization – Example

Recall that $g_t(\theta) := \ell(f(\theta_t)) + \langle \nabla_z \ell(f(\theta_t)), f(\theta) - f(\theta_t) \rangle + \frac{1}{2\eta} \|f(\theta) - f(\theta_t)\|_2^2$

- To gain some intuition, consider linear regression: $\ell(z) = \frac{1}{2} \|z - y\|_2^2$; $z = f(\theta) = X\theta$.
- $g_t(\theta) = \frac{1}{2} \|X\theta_t - y\|_2^2 + \left\langle [X\theta_t - y], X(\theta - \theta_t) \right\rangle + \frac{1}{2\eta} \|X(\theta - \theta_t)\|_2^2$

Target-based Surrogate Minimization – Example

Recall that $g_t(\theta) := \ell(f(\theta_t)) + \langle \nabla_z \ell(f(\theta_t)), f(\theta) - f(\theta_t) \rangle + \frac{1}{2\eta} \|f(\theta) - f(\theta_t)\|_2^2$

- To gain some intuition, consider linear regression: $\ell(z) = \frac{1}{2} \|z - y\|_2^2$; $z = f(\theta) = X\theta$.
- $g_t(\theta) = \frac{1}{2} \|X\theta_t - y\|_2^2 + \big\langle [X\theta_t - y], X(\theta - \theta_t) \big\rangle + \frac{1}{2\eta} \|X(\theta - \theta_t)\|_2^2$
- Alg. 1 with $m = 1$ is equivalent to parametric GD: $\theta_{t+1} = \theta_t - \eta X^\mathsf{T}[X\theta_t - y]$

Target-based Surrogate Minimization – Example

Recall that $g_t(\theta) := \ell(f(\theta_t)) + \langle \nabla_z \ell(f(\theta_t)), f(\theta) - f(\theta_t) \rangle + \frac{1}{2\eta} \|f(\theta) - f(\theta_t)\|_2^2$

- To gain some intuition, consider linear regression: $\ell(z) = \frac{1}{2} \|z - y\|_2^2$; $z = f(\theta) = X\theta$.
- $g_t(\theta) = \frac{1}{2} \|X\theta_t - y\|_2^2 + \big\langle [X\theta_t - y], X(\theta - \theta_t) \big\rangle + \frac{1}{2\eta} \|X(\theta - \theta_t)\|_2^2$
- Alg. 1 with $m = 1$ is equivalent to parametric GD: $\theta_{t+1} = \theta_t - \eta X^\top [X\theta_t - y]$
- Alg. 1 with $m = \infty$ and $\eta = 1$ is equivalent to Newton's method: $\theta_{t+1} = \theta_t - \eta (X^\top X)^{-1} [X^\top (X\theta_t - y)]$.

Target-based Surrogate Minimization – Example

Recall that $g_t(\theta) := \ell(f(\theta_t)) + \langle \nabla_z \ell(f(\theta_t)), f(\theta) - f(\theta_t) \rangle + \frac{1}{2\eta} \|f(\theta) - f(\theta_t)\|_2^2$

- To gain some intuition, consider linear regression: $\ell(z) = \frac{1}{2} \|z - y\|_2^2$; $z = f(\theta) = X\theta$.
- $g_t(\theta) = \frac{1}{2} \|X\theta_t - y\|_2^2 + \left\langle [X\theta_t - y], X(\theta - \theta_t) \right\rangle + \frac{1}{2\eta} \|X(\theta - \theta_t)\|_2^2$
- Alg. 1 with $m = 1$ is equivalent to parametric GD: $\theta_{t+1} = \theta_t - \eta X^\intercal [X\theta_t - y]$
- Alg. 1 with $m = \infty$ and $\eta = 1$ is equivalent to Newton's method: $\theta_{t+1} = \theta_t - \eta (X^\intercal X)^{-1} [X^\intercal (X\theta_t - y)]$.
- Hence, Alg. 1 with $m \in [1, \infty)$ interpolates between a first-order and second-order method without explicitly forming the Hessian.

Theoretical Guarantees

Recall that $g_t(\theta) := \ell(f(\theta_t)) + \langle \nabla_z \ell(f(\theta_t)), f(\theta) - f(\theta_t) \rangle + \frac{1}{2\eta} \|f(\theta) - f(\theta_t)\|_2^2$

- Property (i): For an appropriate η (set according to the smoothness of $\ell(z)$), $\forall \theta$, $g_t(\theta) \geq \ell(f(\theta)) = h(\theta)$ and $g_t(\theta_t) = \ell(f(\theta_t)) = h(\theta_t)$.

Theoretical Guarantees

Recall that $g_t(\theta) := \ell(f(\theta_t)) + \langle \nabla_z \ell(f(\theta_t)), f(\theta) - f(\theta_t) \rangle + \frac{1}{2\eta} \left\| f(\theta) - f(\theta_t) \right\|_2^2$

- Property (i): For an appropriate η (set according to the smoothness of $\ell(z)$), $\forall \theta$, $g_t(\theta) \geq \ell(f(\theta)) = h(\theta)$ and $g_t(\theta_t) = \ell(f(\theta_t)) = h(\theta_t)$.
- Property (ii): With an appropriate step-size α (set according to the smoothness of $g_t(\theta)$) and any value of m, GD on $g_t(\theta)$ ensures descent implying that $g_t(\theta_{t+1}) \leq g_t(\theta_t)$.

Theoretical Guarantees

Recall that $g_t(\theta) := \ell(f(\theta_t)) + \langle \nabla_z \ell(f(\theta_t)), f(\theta) - f(\theta_t) \rangle + \frac{1}{2\eta} \|f(\theta) - f(\theta_t)\|_2^2$

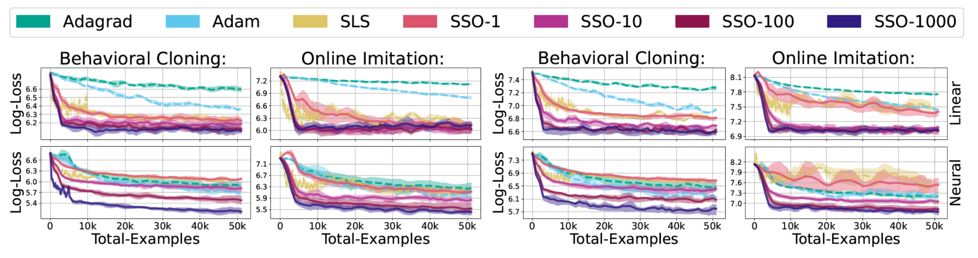
- Property (i): For an appropriate η (set according to the smoothness of $\ell(z)$), $\forall \theta$, $g_t(\theta) \geq \ell(f(\theta)) = h(\theta)$ and $g_t(\theta_t) = \ell(f(\theta_t)) = h(\theta_t)$.
- Property (ii): With an appropriate step-size α (set according to the smoothness of $g_t(\theta)$) and any value of m, GD on $g_t(\theta)$ ensures descent implying that $g_t(\theta_{t+1}) \leq g_t(\theta_t)$.
- **Lemma**: $\ell(z_{t+1}) \overset{Def}{=} h(\theta_{t+1}) \overset{(i)}{\leq} g_t(\theta_{t+1}) \overset{(ii)}{\leq} g_t(\theta_t) \overset{Def}{=} h(\theta_t) \overset{Def}{=} \ell(z_t)$ meaning that Alg. 1 results in a decrease in $\ell(z)$ for any model f.

Theoretical Guarantees

Recall that $g_t(\theta) := \ell(f(\theta_t)) + \langle \nabla_z \ell(f(\theta_t)), f(\theta) - f(\theta_t) \rangle + \frac{1}{2\eta} \|f(\theta) - f(\theta_t)\|_2^2$

- Property (i): For an appropriate η (set according to the smoothness of $\ell(z)$), $\forall \theta$, $g_t(\theta) \geq \ell(f(\theta)) = h(\theta)$ and $g_t(\theta_t) = \ell(f(\theta_t)) = h(\theta_t)$.
- Property (ii): With an appropriate step-size α (set according to the smoothness of $g_t(\theta)$) and any value of m, GD on $g_t(\theta)$ ensures descent implying that $g_t(\theta_{t+1}) \leq g_t(\theta_t)$.
- **Lemma**: $\ell(z_{t+1}) \overset{Def}{=} h(\theta_{t+1}) \overset{(i)}{\leq} g_t(\theta_{t+1}) \overset{(ii)}{\leq} g_t(\theta_t) \overset{Def}{=} h(\theta_t) \overset{Def}{=} \ell(z_t)$ meaning that Alg. 1 results in a decrease in $\ell(z)$ for any model f.
- Other theoretical results:
  - Prove that Alg. 1 converges to a stationary point of $h(\theta)$ at an $O(1/T)$ rate.
  - For convex ℓ, prove convergence rates even when we only have access to a stochastic, unbiased gradient of $\ell(z)$.

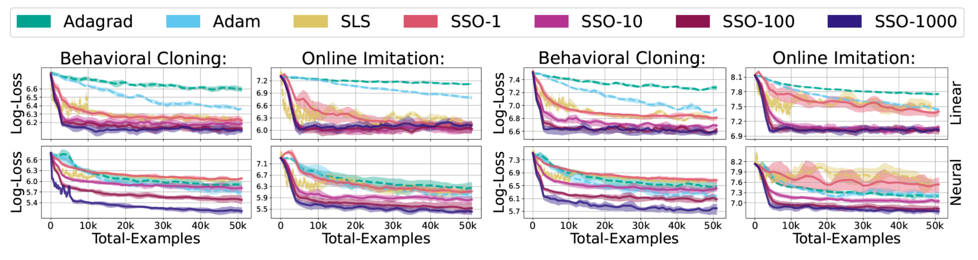
Experiments – Imitation Learning



(a) Hopper-v2 Environment (b) Walker2d-v2 Environment

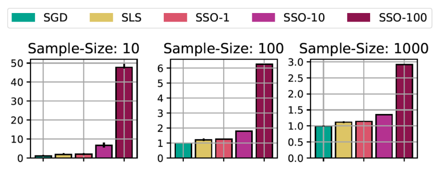
- SSO results in strong empirical performance without the need to tune step-sizes.
- Using larger values of m (more off-policy updates) results in better performance.

Experiments – Imitation Learning



(a) Hopper-v2 Environment (b) Walker2d-v2 Environment

- SSO results in strong empirical performance without the need to tune step-sizes.
- Using larger values of m (more off-policy updates) results in better performance.



- Using a larger number of trajectories results in more computational cost for evaluating $\nabla \ell(z)$, amortizing the cost of multiple off-policy updates.

Conclusion

✓ Constructed a sequence of surrogate functions that exploit the composition structure, and result in efficient updates for imitation learning.

Conclusion

- ✓ Constructed a sequence of surrogate functions that exploit the composition structure, and result in efficient updates for imitation learning.
- ✓ Proved that the resulting surrogate minimization algorithm will result in convergence to a stationary point.

Conclusion

- ✓ Constructed a sequence of surrogate functions that exploit the composition structure, and result in efficient updates for imitation learning.
- ✓ Proved that the resulting surrogate minimization algorithm will result in convergence to a stationary point.
- ✓ Strong empirical performance with better robustness towards hyper-parameters.

Conclusion

- ✓ Constructed a sequence of surrogate functions that exploit the composition structure, and result in efficient updates for imitation learning.
- ✓ Proved that the resulting surrogate minimization algorithm will result in convergence to a stationary point.
- ✓ Strong empirical performance with better robustness towards hyper-parameters.
- ✓ Black-box structure for stochastic optimization: Can use any stochastic optimization algorithm to form surrogates which can then be optimized using any deterministic optimization algorithm.

Outline

- Target-based Surrogate Optimization
- **Functional Mirror Ascent for Policy Gradient (FMA-PG)**
- Conclusions and Future Work

Motivation

- Policy gradient (PG) methods based on REINFORCE:
    - Each policy update requires recomputing the policy gradient.
    - ✓ Theoretical guarantees [Agarwal et al., 2020] with function approximation.
    - × Each update requires computationally expensive interactions with the environment.

Motivation

- Policy gradient (PG) methods based on REINFORCE:
  - Each policy update requires recomputing the policy gradient.
  - ✓ Theoretical guarantees [Agarwal et al., 2020] with function approximation.
  - × Each update requires computationally expensive interactions with the environment.
- Methods such as TRPO, PPO and MPO:
  - Rely on constructing *surrogate functions* and update the policy to maximize these surrogates.
  - ✓ Support *off-policy updates* – can update the policy without requiring additional environment interactions. Have good empirical performance, and widely used.
  - × Only have theoretical guarantees in the tabular setting, and can fail to converge in simple scenarios [Hsu et al., 2020].

Motivation

- Policy gradient (PG) methods based on REINFORCE:
  - Each policy update requires recomputing the policy gradient.
  - ✓ Theoretical guarantees [Agarwal et al., 2020] with function approximation.
  - × Each update requires computationally expensive interactions with the environment.
- Methods such as TRPO, PPO and MPO:
  - Rely on constructing *surrogate functions* and update the policy to maximize these surrogates.
  - ✓ Support *off-policy updates* – can update the policy without requiring additional environment interactions. Have good empirical performance, and widely used.
  - × Only have theoretical guarantees in the tabular setting, and can fail to converge in simple scenarios [Hsu et al., 2020].

No systematic way to design theoretically principled surrogate functions, or a unified framework to analyze their properties.

Functional representation vs Policy parameterization

- **Functional representation**: Specifies a policy's sufficient statistics and is implicit.

  Examples:
  - *Direct functional representation*: Conditional distribution over actions $p^\pi(\cdot|s)$ for each s.
  - *Softmax functional representation*: Logits $z^\pi(s, a)$ such that $p^\pi(a|s) = \frac{\exp(z^\pi(s,a))}{\sum_{a'} \exp(z^\pi(s,a'))}$.

Functional representation vs Policy parameterization

- **Functional representation**: Specifies a policy's sufficient statistics and is implicit. *Examples*:
  - *Direct functional representation*: Conditional distribution over actions $p^{\pi}(\cdot|s)$ for each s.
  - *Softmax functional representation*: Logits $z^{\pi}(s, a)$ such that $p^{\pi}(a|s) = \frac{\exp(z^{\pi}(s,a))}{\sum_{a'} \exp(z^{\pi}(s,a'))}$.
- **Policy parameterization**: Practical realization of the sufficient statistics. Determines Π (the set of feasible policies). *Examples*:
  - *Tabular parameterization* for the direct functional representation: $p^{\pi}(a|s) = \theta(s, a)$.
  - *Linear parameterization* for the softmax functional representation: $z^{\pi}(s, a) = \langle \theta, \mathrm{X}(s, a) \rangle$, where $\mathrm{X}(s, a)$ are the state-action features and $\theta \in \mathbb{R}^d$ are the parameters of a linear model.

Functional representation vs Policy parameterization

- **Functional representation**: Specifies a policy's sufficient statistics and is implicit.
  Examples:
  - *Direct functional representation*: Conditional distribution over actions $p^\pi(\cdot|s)$ for each s.
  - *Softmax functional representation*: Logits $z^\pi(s,a)$ such that $p^\pi(a|s) = \frac{\exp(z^\pi(s,a))}{\sum_{a'} \exp(z^\pi(s,a'))}$.
- **Policy parameterization**: Practical realization of the sufficient statistics. Determines Π (the set of feasible policies). *Examples*:
  - *Tabular parameterization* for the direct functional representation: $p^\pi(a|s) = \theta(s,a)$.
  - *Linear parameterization* for the softmax functional representation: $z^\pi(s,a) = \langle \theta, \mathrm{X}(s,a) \rangle$, where $\mathrm{X}(s,a)$ are the state-action features and $\theta \in \mathbb{R}^d$ are the parameters of a linear model.
- The functional representation of a policy is independent of its parameterization.

Functional representation vs Policy parameterization

- **Functional representation**: Specifies a policy's sufficient statistics and is implicit. *Examples*:
  - *Direct functional representation*: Conditional distribution over actions $p^\pi(\cdot|s)$ for each s.
  - *Softmax functional representation*: Logits $z^\pi(s,a)$ such that $p^\pi(a|s) = \frac{\exp(z^\pi(s,a))}{\sum_{a'} \exp(z^\pi(s,a'))}$.
- **Policy parameterization**: Practical realization of the sufficient statistics. Determines Π (the set of feasible policies). *Examples*:
  - *Tabular parameterization* for the direct functional representation: $p^\pi(a|s) = \theta(s,a)$.
  - *Linear parameterization* for the softmax functional representation: $z^\pi(s,a) = \langle \theta, \mathrm{X}(s,a) \rangle$, where $\mathrm{X}(s,a)$ are the state-action features and $\theta \in \mathbb{R}^d$ are the parameters of a linear model.
- The functional representation of a policy is independent of its parameterization.
- **Standard PG approach**: Use a model (with parameters θ) to parameterize (the functional representation of) π and directly maximize $J(\pi(\theta))$ w.r.t. θ.

Functional Mirror Ascent

- Target-based surrogate optimization: Iteratively optimize J w.r.t π and project onto Π_θ.

Functional Mirror Ascent

- Target-based surrogate optimization: Iteratively optimize J w.r.t π and project onto Π_θ.
- Overload π to be a general functional representation, with $\pi(\theta)$ as its parametric realization.

Functional Mirror Ascent

- Target-based surrogate optimization: Iteratively optimize J w.r.t π and project onto Π_θ.
- Overload π to be a general functional representation, with $\pi(\theta)$ as its parametric realization.
- For a strictly convex, differentiable function Φ (*mirror map*), $D_\Phi(\pi, \pi')$ is the *Bregman divergence* between policies π and π'. $D_\Phi(\pi, \pi') := \Phi(\pi) - \Phi(\pi') - \langle \nabla\Phi(\pi'), \pi - \pi' \rangle$.
- E.g. If $\Phi(\pi) = \frac{1}{2}\|\pi\|_2^2$, $D_\Phi(\pi, \pi') = \frac{1}{2}\|\pi - \pi'\|_2^2$.

Functional Mirror Ascent

- Target-based surrogate optimization: Iteratively optimize J w.r.t π and project onto Π_θ.
- Overload π to be a general functional representation, with $\pi(\theta)$ as its parametric realization.
- For a strictly convex, differentiable function Φ (*mirror map*), $D_\Phi(\pi, \pi')$ is the *Bregman divergence* between policies π and π'. $D_\Phi(\pi, \pi') := \Phi(\pi) - \Phi(\pi') - \langle \nabla\Phi(\pi'), \pi - \pi' \rangle$.
- E.g. If $\Phi(\pi) = \frac{1}{2}\|\pi\|_2^2$, $D_\Phi(\pi, \pi') = \frac{1}{2}\|\pi - \pi'\|_2^2$.

In each iteration $t \in [T]$ of *functional mirror ascent* (FMA), with *step-size* η,

$$\pi_{t+1} = \arg\max_{\pi \in \Pi_\theta} \left[\langle \pi, \nabla_\pi J(\pi_t) \rangle - \frac{1}{\eta} D_\Phi(\pi, \pi_t) \right]$$

Functional Mirror Ascent

- Target-based surrogate optimization: Iteratively optimize J w.r.t π and project onto Π_θ.
- Overload π to be a general functional representation, with $\pi(\theta)$ as its parametric realization.
- For a strictly convex, differentiable function Φ (*mirror map*), $D_\Phi(\pi, \pi')$ is the *Bregman divergence* between policies π and π'. $D_\Phi(\pi, \pi') := \Phi(\pi) - \Phi(\pi') - \langle \nabla\Phi(\pi'), \pi - \pi' \rangle$.
- E.g. If $\Phi(\pi) = \frac{1}{2} \|\pi\|_2^2$, $D_\Phi(\pi, \pi') = \frac{1}{2} \|\pi - \pi'\|_2^2$.

In each iteration $t \in [T]$ of *functional mirror ascent* (FMA), with *step-size* η,

$$\pi_{t+1} = \arg\max_{\pi \in \Pi_\theta} \left[\langle \pi, \nabla_\pi J(\pi_t) \rangle - \frac{1}{\eta} D_\Phi(\pi, \pi_t) \right]$$

Since $\pi \in \Pi_\theta$,

$$\pi_{t+1} = \pi(\theta_{t+1}) \quad ; \quad \theta_{t+1} = \arg\max_{\theta \in \mathbb{R}^d} \underbrace{\left[J(\pi(\theta_t)) + \langle \pi(\theta) - \pi(\theta_t), \nabla_\pi J(\pi(\theta_t)) \rangle - \frac{1}{\eta} D_\Phi(\pi(\theta), \pi(\theta_t)) \right]}_{\text{Surrogate function } g_t(\theta)}$$

FMA-PG Framework – Algorithm

Algorithm 1: Generic policy optimization

Input: π (functional representation), θ_0 (initial policy parameterization), T (PG iterations), m (inner-loops), η (step-size for functional update), α (step-size for parametric update)

for $t \leftarrow 0$ **to** $T-1$ **do**
  Compute $\nabla_\pi J(\pi_t)$ and form the surrogate $g_t(\theta)$.
  Initialize inner-loop: $\omega_0 = \theta_t$
  for $k \leftarrow 0$ **to** m **do**
    $\omega_{k+1} = \omega_k + \alpha \nabla_\omega g_t(\omega_k)$ /* Off-policy actor updates */
  $\theta_{t+1} = \omega_m$
  $\pi_{t+1} = \pi(\theta_{t+1})$
Return θ_T

Algorithm 2: Generic policy optimization

Input: π (functional representation), θ_0 (initial policy parameterization), T (PG iterations), m (inner-loops), η (step-size for functional update), α (step-size for parametric update)

for $t \leftarrow 0$ **to** $T-1$ **do**

 Compute $\nabla_\pi J(\pi_t)$ and form the surrogate $g_t(\theta)$.

 Initialize inner-loop: $\omega_0 = \theta_t$

 for $k \leftarrow 0$ **to** m **do**

  $\omega_{k+1} = \omega_k + \alpha \nabla_\omega g_t(\omega_k)$ /* Off-policy actor updates */

 $\theta_{t+1} = \omega_m$

 $\pi_{t+1} = \pi(\theta_{t+1})$

Return θ_T

- If we can guarantee that (i) $g_t(\theta) \leq J(\pi(\theta))$ and (ii) $g_t(\theta_{t+1}) \geq g_t(\theta_t)$, then the above algorithm results in monotonic policy improvement regardless of the policy parameterization.

Instantiating FMA-PG – Direct functional representation

- Policy is represented by distributions $p^{\pi}(\cdot|s)$ over actions for each state $s \in \mathcal{S}$.

Instantiating FMA-PG – Direct functional representation

- Policy is represented by distributions $p^\pi(\cdot|s)$ over actions for each state $s \in \mathcal{S}$.
- We choose $D_\Phi(\pi, \pi') = \sum_s d^\pi(s)\, D_\phi(p^\pi(\cdot|s), p^{\pi'}(\cdot|s))$.

Instantiating FMA-PG – Direct functional representation

- Policy is represented by distributions $p^\pi(\cdot|s)$ over actions for each state $s \in \mathcal{S}$.
- We choose $D_\Phi(\pi, \pi') = \sum_s d^\pi(s)\, D_\phi(p^\pi(\cdot|s), p^{\pi'}(\cdot|s))$.

Since $\frac{\partial J(\pi)}{\partial p^\pi(a|s)} = d^\pi(s) Q^\pi(s, a)$, the surrogate function at iteration t is given by,

Instantiating FMA-PG – Direct functional representation

- Policy is represented by distributions $p^\pi(\cdot|s)$ over actions for each state $s \in \mathcal{S}$.
- We choose $D_\Phi(\pi, \pi') = \sum_s d^\pi(s)\, D_\phi(p^\pi(\cdot|s), p^{\pi'}(\cdot|s))$.

Since $\frac{\partial J(\pi)}{\partial p^\pi(a|s)} = d^\pi(s) Q^\pi(s,a)$, the surrogate function at iteration t is given by,

$$g_t(\theta) = \mathbb{E}_{(s,a)\sim\mu^{\pi_t}}\left[\left(Q^{\pi_t}(s,a)\,\frac{p^\pi(a|s,\theta)}{p^\pi(a|s,\theta_t)}\right)\right] - \frac{1}{\eta}\mathbb{E}_{s\sim d^{\pi_t}}\left[D_\phi(p^\pi(\cdot|s,\theta), p^\pi(\cdot|s,\theta_t))\right] + C.$$

Instantiating FMA-PG – Direct functional representation

- Policy is represented by distributions $p^\pi(\cdot|s)$ over actions for each state $s \in \mathcal{S}$.
- We choose $D_\Phi(\pi, \pi') = \sum_s d^\pi(s)\, D_\phi(p^\pi(\cdot|s), p^{\pi'}(\cdot|s))$.

Since $\frac{\partial J(\pi)}{\partial p^\pi(a|s)} = d^\pi(s) Q^\pi(s, a)$, the surrogate function at iteration t is given by,

$$g_t(\theta) = \mathbb{E}_{(s,a)\sim\mu^{\pi_t}}\left[\left(Q^{\pi_t}(s, a)\,\frac{p^\pi(a|s,\theta)}{p^\pi(a|s,\theta_t)}\right)\right] - \frac{1}{\eta}\mathbb{E}_{s\sim d^{\pi_t}}\left[D_\phi(p^\pi(\cdot|s,\theta), p^\pi(\cdot|s,\theta_t))\right] + C.$$

For the negative entropy mirror-map i.e. when $\phi(p^\pi(\cdot|s)) = \sum_a p^\pi(a|s)\log p^\pi(a|s)$,

$$g_t(\theta) = \mathbb{E}_{(s,a)\sim\mu^{\pi_t}}\left[\left(Q^{\pi_t}(s, a)\,\frac{p^\pi(a|s,\theta)}{p^\pi(a|s,\theta_t)}\right)\right] - \frac{1}{\eta}\mathbb{E}_{s\sim d^{\pi_t}}\left[\mathrm{KL}\left(p^\pi(\cdot|s,\theta)||p^\pi(\cdot|s,\theta_t)\right)\right] + C.$$

Instantiating FMA-PG – Direct functional representation

- Policy is represented by distributions $p^\pi(\cdot|s)$ over actions for each state $s \in \mathcal{S}$.
- We choose $D_\Phi(\pi, \pi') = \sum_s d^\pi(s)\, D_\phi(p^\pi(\cdot|s), p^{\pi'}(\cdot|s))$.

Since $\frac{\partial J(\pi)}{\partial p^\pi(a|s)} = d^\pi(s) Q^\pi(s,a)$, the surrogate function at iteration t is given by,

$$g_t(\theta) = \mathbb{E}_{(s,a)\sim\mu^{\pi_t}}\left[\left(Q^{\pi_t}(s,a)\,\frac{p^\pi(a|s,\theta)}{p^\pi(a|s,\theta_t)}\right)\right] - \frac{1}{\eta}\mathbb{E}_{s\sim d^{\pi_t}}\left[D_\phi(p^\pi(\cdot|s,\theta), p^\pi(\cdot|s,\theta_t))\right] + C.$$

For the negative entropy mirror-map i.e. when $\phi(p^\pi(\cdot|s)) = \sum_a p^\pi(a|s) \log p^\pi(a|s)$,

$$g_t(\theta) = \mathbb{E}_{(s,a)\sim\mu^{\pi_t}}\left[\left(Q^{\pi_t}(s,a)\,\frac{p^\pi(a|s,\theta)}{p^\pi(a|s,\theta_t)}\right)\right] - \frac{1}{\eta}\mathbb{E}_{s\sim d^{\pi_t}}\left[\mathrm{KL}\left(p^\pi(\cdot|s,\theta)||p^\pi(\cdot|s,\theta_t)\right)\right] + C.$$

Setting η for the direct functional representation with negative entropy mirror map

For any policy parameterization, $\forall\theta$, $J(\pi(\theta)) \geq g_t(\theta)$ for $\eta \leq \frac{(1-\gamma)^3}{2\gamma|A|}$.

Instantiating FMA-PG – Direct functional representation

- Policy is represented by distributions $p^\pi(\cdot|s)$ over actions for each state $s \in \mathcal{S}$.
- We choose $D_\Phi(\pi, \pi') = \sum_s d^\pi(s)\, D_\phi(p^\pi(\cdot|s), p^{\pi'}(\cdot|s))$.

Since $\frac{\partial J(\pi)}{\partial p^\pi(a|s)} = d^\pi(s) Q^\pi(s,a)$, the surrogate function at iteration t is given by,

$$g_t(\theta) = \mathbb{E}_{(s,a)\sim\mu^{\pi_t}}\left[\left(Q^{\pi_t}(s,a)\,\frac{p^\pi(a|s,\theta)}{p^\pi(a|s,\theta_t)}\right)\right] - \frac{1}{\eta}\mathbb{E}_{s\sim d^{\pi_t}}\left[D_\phi(p^\pi(\cdot|s,\theta), p^\pi(\cdot|s,\theta_t))\right] + C.$$

For the negative entropy mirror-map i.e. when $\phi(p^\pi(\cdot|s)) = \sum_a p^\pi(a|s) \log p^\pi(a|s)$,

$$g_t(\theta) = \mathbb{E}_{(s,a)\sim\mu^{\pi_t}}\left[\left(Q^{\pi_t}(s,a)\,\frac{p^\pi(a|s,\theta)}{p^\pi(a|s,\theta_t)}\right)\right] - \frac{1}{\eta}\mathbb{E}_{s\sim d^{\pi_t}}\left[\mathrm{KL}\left(p^\pi(\cdot|s,\theta)||p^\pi(\cdot|s,\theta_t)\right)\right] + C.$$

Setting η for the direct functional representation with negative entropy mirror map

For any policy parameterization, $\forall\theta$, $J(\pi(\theta)) \geq g_t(\theta)$ for $\eta \leq \frac{(1-\gamma)^3}{2\gamma|A|}$.

× Involves the importance-sampling ratio $\frac{p^\pi(a|s,\theta)}{p^\pi(a|s,\theta_t)}$ that could be potentially large.

× Involves the reverse KL divergence making it *mode seeking* hindering exploration.

Instantiating FMA-PG – Softmax functional representation

- Policy is represented by the logits $z^\pi(s, a)$ such that $p^\pi(a|s) \propto \exp(z^\pi(s, a))$ for each state.

Instantiating FMA-PG – Softmax functional representation

- Policy is represented by the logits $z^\pi(s,a)$ such that $p^\pi(a|s) \propto \exp(z^\pi(s,a))$ for each state.
- We choose $D_\Phi(\pi, \pi') = \sum_s d^\pi(s)\, D_{\phi_z}(z(s,\cdot), z'(s,\cdot))$.

Instantiating FMA-PG – Softmax functional representation

- Policy is represented by the logits $z^\pi(s,a)$ such that $p^\pi(a|s) \propto \exp(z^\pi(s,a))$ for each state.
- We choose $D_\Phi(\pi, \pi') = \sum_s d^\pi(s)\, D_{\phi_z}(z(s,\cdot), z'(s,\cdot))$.

Since $\frac{\partial J(\pi)}{\partial z^\pi(s,a)} = d^\pi(s) A^\pi(s,a) p^\pi(a|s)$, the surrogate function at iteration t is given by,

Instantiating FMA-PG – Softmax functional representation

- Policy is represented by the logits $z^\pi(s, a)$ such that $p^\pi(a|s) \propto \exp(z^\pi(s, a))$ for each state.
- We choose $D_\Phi(\pi, \pi') = \sum_s d^\pi(s)\, D_{\phi_z}(z(s, \cdot), z'(s, \cdot))$.

Since $\frac{\partial J(\pi)}{\partial z^\pi(s,a)} = d^\pi(s) A^\pi(s, a) p^\pi(a|s)$, the surrogate function at iteration t is given by,

$$g_t(\theta) = E_{(s,a)\sim\mu^{\pi_t}}\left[A^{\pi_t}(s, a)\, z^\pi(s, a|\theta)\right] - \frac{1}{\eta}\sum_s d^{\pi_t}(s)\, D_{\phi_z}\left(z^\pi(s, \cdot|\theta), z^\pi(s, \cdot|\theta_t)\right) + C.$$

Instantiating FMA-PG – Softmax functional representation

- Policy is represented by the logits $z^\pi(s,a)$ such that $p^\pi(a|s) \propto \exp(z^\pi(s,a))$ for each state.
- We choose $D_\Phi(\pi,\pi') = \sum_s d^\pi(s)\, D_{\phi_z}(z(s,\cdot), z'(s,\cdot))$.

Since $\frac{\partial J(\pi)}{\partial z^\pi(s,a)} = d^\pi(s) A^\pi(s,a) p^\pi(a|s)$, the surrogate function at iteration t is given by,

$$g_t(\theta) = E_{(s,a)\sim\mu^{\pi_t}}\left[A^{\pi_t}(s,a)\, z^\pi(s,a|\theta)\right] - \frac{1}{\eta}\sum_s d^{\pi_t}(s)\, D_{\phi_z}\left(z^\pi(s,\cdot|\theta), z^\pi(s,\cdot|\theta_t)\right) + C.$$

For the log-sum-exp mirror-map i.e. when $\phi_z(z(s,\cdot)) = \log\left(\sum_a \exp(z^\pi(s,a))\right)$,

$$g_t(\theta) = E_{(s,a)\sim\mu^{\pi_t}}\left[\left(A^{\pi_t}(s,a) + \frac{1}{\eta}\right)\log\frac{p^\pi(a|s,\theta)}{p^\pi(a|s,\theta_t)}\right] + C.$$

Instantiating FMA-PG – Softmax functional representation

- Policy is represented by the logits $z^\pi(s, a)$ such that $p^\pi(a|s) \propto \exp(z^\pi(s, a))$ for each state.
- We choose $D_\Phi(\pi, \pi') = \sum_s d^\pi(s)\, D_{\phi_z}(z(s, \cdot), z'(s, \cdot))$.

Since $\frac{\partial J(\pi)}{\partial z^\pi(s,a)} = d^\pi(s) A^\pi(s, a) p^\pi(a|s)$, the surrogate function at iteration t is given by,

$$g_t(\theta) = E_{(s,a)\sim\mu^{\pi_t}}\left[A^{\pi_t}(s, a)\, z^\pi(s, a|\theta)\right] - \frac{1}{\eta}\sum_s d^{\pi_t}(s)\, D_{\phi_z}\left(z^\pi(s, \cdot|\theta), z^\pi(s, \cdot|\theta_t)\right) + C.$$

For the log-sum-exp mirror-map i.e. when $\phi_z(z(s, \cdot)) = \log\left(\sum_a \exp(z^\pi(s, a))\right)$,

$$g_t(\theta) = E_{(s,a)\sim\mu^{\pi_t}}\left[\left(A^{\pi_t}(s, a) + \frac{1}{\eta}\right)\log\frac{p^\pi(a|s, \theta)}{p^\pi(a|s, \theta_t)}\right] + C.$$

Setting η for the softmax functional representation with log-sum-exp mirror map

For any policy parameterization, $\forall\theta$, $J(\pi(\theta)) \geq g_t(\theta)$ for $\eta \leq 1 - \gamma$.

Instantiating FMA-PG – Softmax functional representation

The surrogate can be rewritten as

$$g_t(\theta) = \mathbb{E}_{s\sim d^{\pi_t}}\left[\mathbb{E}_{a\sim p^{\pi_t}}\left(A^{\pi_t}(s,a)\log\frac{p^\pi(a|s,\theta)}{p^\pi(a|s,\theta_t)}\right) - \frac{1}{\eta}\mathrm{KL}(p^\pi(\cdot|s,\theta_t)||p^\pi(\cdot|s,\theta))\right] + C.$$

Instantiating FMA-PG – Softmax functional representation

The surrogate can be rewritten as

$$g_t(\theta) = \mathbb{E}_{s\sim d^{\pi_t}}\left[\mathbb{E}_{a\sim p^{\pi_t}}\left(A^{\pi_t}(s,a)\log\frac{p^\pi(a|s,\theta)}{p^\pi(a|s,\theta_t)}\right) - \frac{1}{\eta}\mathrm{KL}(p^\pi(\cdot|s,\theta_t)||p^\pi(\cdot|s,\theta))\right] + C.$$

- ✓ Above surrogate depends on the log of the importance sampling ratio.
- ✓ Surrogate involves the forward KL divergence making it *mode covering* encouraging exploration.

Instantiating FMA-PG – Softmax functional representation

The surrogate can be rewritten as

$$g_t(\theta) = \mathbb{E}_{s\sim d^{\pi_t}}\left[\mathbb{E}_{a\sim p^{\pi_t}}\left(A^{\pi_t}(s,a)\log\frac{p^{\pi}(a|s,\theta)}{p^{\pi}(a|s,\theta_t)}\right) - \frac{1}{\eta}\text{KL}(p^{\pi}(\cdot|s,\theta_t)||p^{\pi}(\cdot|s,\theta))\right] + C.$$

- ✓ Above surrogate depends on the log of the importance sampling ratio.
- ✓ Surrogate involves the forward KL divergence making it *mode covering* encouraging exploration.

Compared to TRPO: $\max_{\theta\in\mathbb{R}^d}\mathbb{E}_{(s,a)\sim\mu^{\pi_t}}[A^{\pi_t}(s,a)\,\frac{p^{\pi}(a|s,\theta)}{p^{\pi}(a|s,\theta_t)}]$ s.t. $\mathbb{E}_{s\sim d^{\pi_t}}\,[\text{KL}(p^{\pi_t}(\cdot|s,\theta_t)||p^{\pi}(\cdot|s,\theta))] \leq \delta$,

- $g_t(\theta)$ involves the log of the importance sampling ratio, and enforces proximity between policies using a regularization (with parameter $1/\eta$) rather than a constraint.
- we ensure monotonic policy improvement for any policy parameterization.

FMA-PG – Experimental Evaluation - Continuous control

- FMA-PG with the softmax representation suggests sPPO with the following surrogate:

$$g_t(\theta) = \mathbb{E}_{(s,a)\sim\mu^{\pi_t}}\left[A^{\pi_t}(s,a)\log\left(\text{clip}\left(\frac{p^{\pi}(a|s,\theta)}{p^{\pi}(a|s,\theta_t)}, \frac{1}{1+\epsilon}, 1+\epsilon\right)\right)\right]$$

.

- Compare to PPO on standard Mujoco tasks, with both algorithms using a critic.

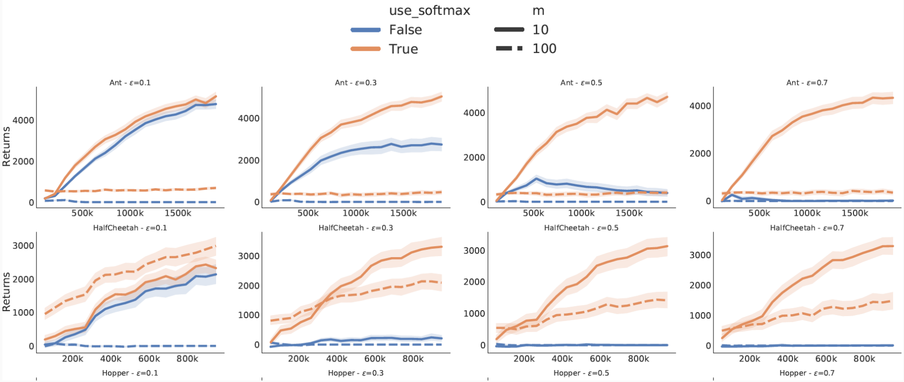
FMA-PG – Experimental Evaluation - Continuous control

- FMA-PG with the softmax representation suggests sPPO with the following surrogate:

$$g_t(\theta) = \mathbb{E}_{(s,a)\sim\mu^{\pi_t}}\left[A^{\pi_t}(s,a)\log\left(\text{clip}\left(\frac{p^\pi(a|s,\theta)}{p^\pi(a|s,\theta_t)}, \frac{1}{1+\epsilon}, 1+\epsilon\right)\right)\right]$$

.

- Compare to PPO on standard Mujoco tasks, with both algorithms using a critic.



Outline

- Target-based Surrogate Optimization
- Functional Mirror Ascent for Policy Gradient (FMA-PG)
- **Conclusion**

Conclusion

✓ Used functional mirror ascent to propose FMA-PG, a systematic way to define surrogate functions for generic policy optimization. Ensures monotonic policy improvement for arbitrary policy parameterization.

Conclusion

- ✓ Used functional mirror ascent to propose FMA-PG, a systematic way to define surrogate functions for generic policy optimization. Ensures monotonic policy improvement for arbitrary policy parameterization.
- ✓ Can use the FMA-PG framework to "lift" existing theoretical guarantees [Mei et al., 2020, Xiao, 2022] for policy optimization algorithms in the tabular setting to use off-policy updates and function approximation.

Conclusion

- ✓ Used functional mirror ascent to propose FMA-PG, a systematic way to define surrogate functions for generic policy optimization. Ensures monotonic policy improvement for arbitrary policy parameterization.
- ✓ Can use the FMA-PG framework to "lift" existing theoretical guarantees [Mei et al., 2020, Xiao, 2022] for policy optimization algorithms in the tabular setting to use off-policy updates and function approximation.
- ✓ Show experimental evidence that on simple tabular MDPs, the algorithms instantiated with FMA-PG are competitive with popular PG algorithms such as TRPO, PPO. The framework suggests an alternative method, `sPPO` that out-performs PPO on the MuJoCo suite.

Conclusion

- ✓ Used functional mirror ascent to propose FMA-PG, a systematic way to define surrogate functions for generic policy optimization. Ensures monotonic policy improvement for arbitrary policy parameterization.
- ✓ Can use the FMA-PG framework to "lift" existing theoretical guarantees [Mei et al., 2020, Xiao, 2022] for policy optimization algorithms in the tabular setting to use off-policy updates and function approximation.
- ✓ Show experimental evidence that on simple tabular MDPs, the algorithms instantiated with FMA-PG are competitive with popular PG algorithms such as TRPO, PPO. The framework suggests an alternative method, `sPPO` that out-performs PPO on the MuJoCo suite.
- ✓ Recent work [Vaswani et al., 2023]: Generalized FMA-PG to design a decision-aware actor-critic framework where the actor and critic are trained cooperatively to optimize a joint objective.

Questions?

Papers: https://arxiv.org/abs/2108.05828, https://arxiv.org/abs/2302.02607

Contact: vaswani.sharan@gmail.com, nicolas.le.roux@gmail.com

References i

Alekh Agarwal, Sham M. Kakade, Jason D. Lee, and Gaurav Mahajan. Optimality and approximation with policy gradient methods in Markov decision processes. In *Conference on Learning Theory (COLT)*, pages 64–66, 2020.

Matthieu Geist, Bruno Scherrer, and Olivier Pietquin. A theory of regularized Markov decision processes. In *International Conference on Machine Learning*, pages 2160–2169. PMLR, 2019.

Chloe Ching-Yun Hsu, Celestine Mendler-Dünner, and Moritz Hardt. Revisiting design choices in proximal policy optimization. *arXiv preprint arXiv:2009.10897*, 2020.

Sham Kakade. A natural policy gradient. In *NIPS*, volume 14, pages 1531–1538, 2001.

Jincheng Mei, Chenjun Xiao, Csaba Szepesvari, and Dale Schuurmans. On the global convergence rates of softmax policy gradient methods. In *International Conference on Machine Learning*, pages 6820–6829. PMLR, 2020.

Lior Shani, Yonathan Efroni, and Shie Mannor. Adaptive trust region policy optimization: Global convergence and faster rates for regularized mdps. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5668–5675, 2020.

Manan Tomar, Lior Shani, Yonathan Efroni, and Mohammad Ghavamzadeh. Mirror descent policy optimization. *arXiv preprint arXiv:2005.09814*, 2020.

Sharan Vaswani, Amirreza Kazemi, Reza Babanezhad, and Nicolas Le Roux. Decision-aware actor-critic with function approximation and theoretical guarantees. *arXiv preprint arXiv:2305.15249*, 2023.

Ronald J Williams and Jing Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268, 1991.

Lin Xiao. On the convergence rates of policy gradient methods. *Journal of Machine Learning Research*, 23(282): 1–36, 2022.

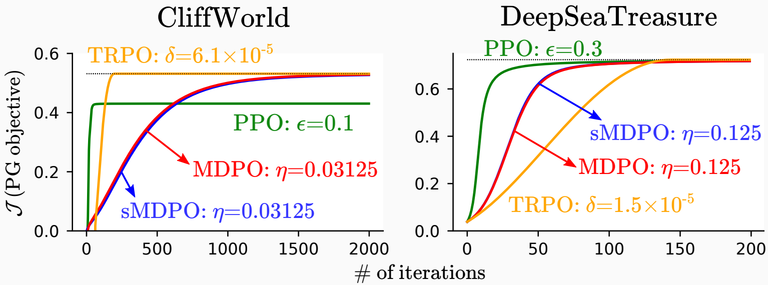
Backup Slides

FMA-PG – Experimental Evaluation - Tabular MDP

- Compare MDPO (direct + negative entropy mirror map), sMDPO (softmax + logsumexp mirror map), PPO, TRPO with access to exact Q^{π}, A^{π} values (no function approximation).
- Use best-tuned values of the functional step-size η for MDPO and sMDPO, clipping value ϵ for PPO and KL constraint value δ for TRPO. Using best-tuned α for each method.

FMA-PG – Experimental Evaluation - Tabular MDP

- Compare MDPO (direct + negative entropy mirror map), sMDPO (softmax + logsumexp mirror map), PPO, TRPO with access to exact Q^{π}, A^{π} values (no function approximation).
- Use best-tuned values of the functional step-size η for MDPO and sMDPO, clipping value ϵ for PPO and KL constraint value δ for TRPO. Using best-tuned α for each method.

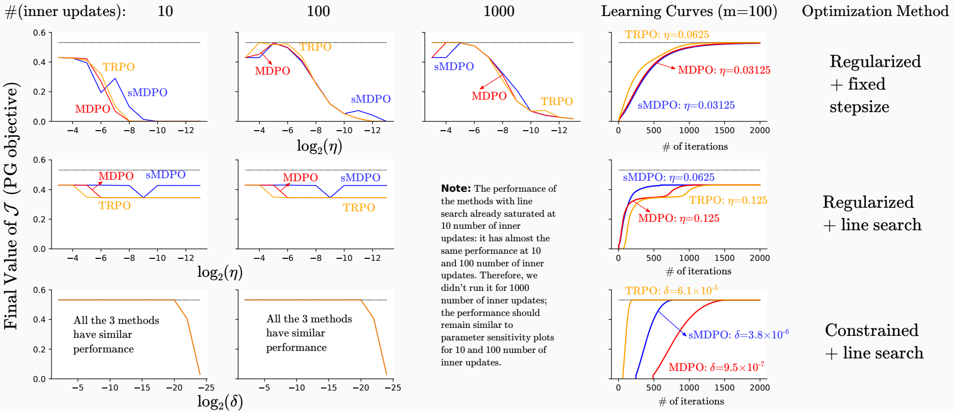


FMA-PG – Experimental Evaluation - Tabular MDP

- Ablation study on MDPO, sMDPO, TRPO to evaluate the effect of m, algorithm hyperparameters (η, δ) and design decisions – line-search in the inner-loop and using a constraint vs regularization.

FMA-PG – Experimental Evaluation - Tabular MDP

- Ablation study on MDPO, sMDPO, TRPO to evaluate the effect of m, algorithm hyperparameters (η, δ) and design decisions – line-search in the inner-loop and using a constraint vs regularization.

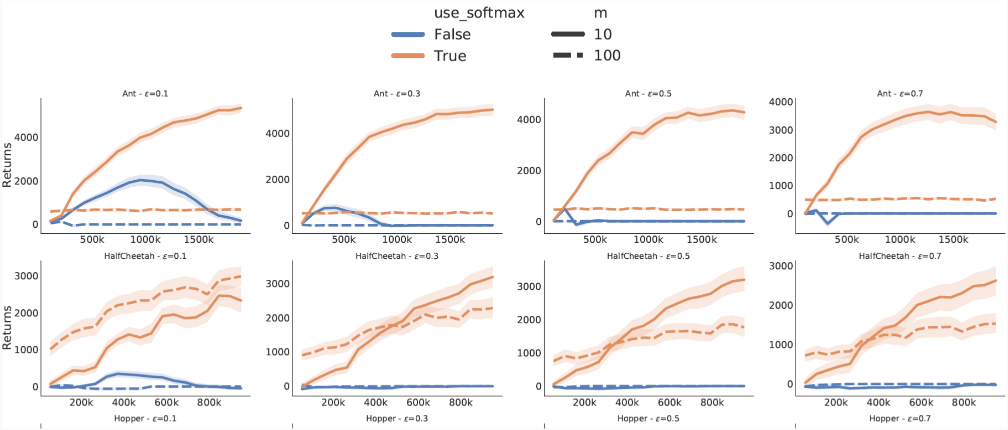


FMA-PG – Experimental Evaluation - Continuous control

- Ablation study on sPPO, PPO disabling both learning rate decay and gradient clipping.

FMA-PG – Experimental Evaluation - Continuous control

- Ablation study on sPPO, PPO disabling both learning rate decay and gradient clipping.



Extra Slides

Instantiating FMA-PG – Direct functional representation

Recall that $g_t(\theta) = \mathbb{E}_{(s,a)\sim\mu^{\pi_t}}\left[\left(Q^{\pi_t}(s,a)\,\frac{p^{\pi}(a|s,\theta)}{p^{\pi}(a|s,\theta_t)}\right)\right] - \frac{1}{\eta}\mathbb{E}_{s\sim d^{\pi_t}}\left[\mathrm{KL}\left(p^{\pi}(\cdot|s,\theta)||p^{\pi}(\cdot|s,\theta_t)\right)\right] + C.$

- With the tabular parameterization,

Instantiating FMA-PG – Direct functional representation

Recall that $g_t(\theta) = \mathbb{E}_{(s,a)\sim\mu^{\pi_t}}\left[\left(Q^{\pi_t}(s,a)\,\frac{p^\pi(a|s,\theta)}{p^\pi(a|s,\theta_t)}\right)\right] - \frac{1}{\eta}\mathbb{E}_{s\sim d^{\pi_t}}\left[\mathrm{KL}\left(p^\pi(\cdot|s,\theta)||p^\pi(\cdot|s,\theta_t)\right)\right] + C.$

- With the tabular parameterization,
    - similar to uniform TRPO [Shani et al., 2020] and Mirror Descent Modified Policy Iteration [Geist et al., 2019].

Instantiating FMA-PG – Direct functional representation

Recall that $g_t(\theta) = \mathbb{E}_{(s,a)\sim\mu^{\pi_t}}\left[\left(Q^{\pi_t}(s,a)\,\frac{p^\pi(a|s,\theta)}{p^\pi(a|s,\theta_t)}\right)\right] - \frac{1}{\eta}\mathbb{E}_{s\sim d^{\pi_t}}\left[\mathrm{KL}\left(p^\pi(\cdot|s,\theta)||p^\pi(\cdot|s,\theta_t)\right)\right] + C$.

- With the tabular parameterization,
  - similar to uniform TRPO [Shani et al., 2020] and Mirror Descent Modified Policy Iteration [Geist et al., 2019].
  - with $m = \infty$ (exact maximization of the surrogate), and,
    (i) squared Euclidean distance mirror map, same as REINFORCE [Williams and Peng, 1991]
    (ii) negative entropy mirror map, same as natural policy gradient [Kakade, 2001].

Instantiating FMA-PG – Direct functional representation

Recall that $g_t(\theta) = \mathbb{E}_{(s,a)\sim\mu^{\pi_t}}\left[\left(Q^{\pi_t}(s,a)\,\frac{p^\pi(a|s,\theta)}{p^\pi(a|s,\theta_t)}\right)\right] - \frac{1}{\eta}\mathbb{E}_{s\sim d^{\pi_t}}\left[\mathrm{KL}\left(p^\pi(\cdot|s,\theta)||p^\pi(\cdot|s,\theta_t)\right)\right] + C.$

- With the tabular parameterization,
  - similar to uniform TRPO [Shani et al., 2020] and Mirror Descent Modified Policy Iteration [Geist et al., 2019].
  - with $m = \infty$ (exact maximization of the surrogate), and,
    (i) squared Euclidean distance mirror map, same as REINFORCE [Williams and Peng, 1991]
    (ii) negative entropy mirror map, same as natural policy gradient [Kakade, 2001].
- For gradient-based maximization of the surrogate, the resulting update is the same as Mirror Descent Policy Optimization [Tomar et al., 2020], but we set the step-sizes that ensure monotonic policy improvement for any policy parameterization and any number of inner-loops.