Montgomery, Levine
Guided Policy Search as Approximate Mirror Descent

1 Abstract

In this paper, the authors describe and validate a GPS method that approximates mirror descent (MDGPS). One benefit of MDGPS is *increased simplicity*, e.g. a simpler formulation with fewer hyperparameters. MDGPS also provides improvement and convergence guarantees in convex and linear domains, and is identical to mirror descent in these cases. Meanwhile, in nonlinear domains the error can be bounded.

- **GPS:** an iterative process for optimizing complex nonlinear policies, e.g. deep neural nets, without having to directly compute policy gradients in high-D parameter space. It instead uses supervised learning to mimic a teacher algorithm, such as a trajectory optimizer or a trajectory-centric RL method.
  - GPS methods always provide asymptotic local convergence guarantees.
  - The learned policy is gradually constrained to match the teacher (constrained optimization).
  - Issue: unclear how much the policy improves within a small number of iterations.
  - Issue: need to provide the entire state space as input (and in the case of airplane insertion or another task involving complex objects in reality, do we actually have all of that data?).

2 Introduction

- The KL divergence is used to bound policy cost at each iteration, and provides guidance on how to adjust step size.
- We can represent GPS as approximate mirror descent *if* we allow the policy parameterization to impose constraints (supervised learning equates to projection onto constraint manifold).
- If not convex or linear, we have to project the policy space onto the constraint manifold (only approximately, up to a bound that depends on the step size).
- GPS will always try to complete task in a certain way (even if there are theoretically a lot of options).
- Generic GPS method:
  - For some number of iterations, do
    - **C-step:** improve each $p_i(u_t|x_t)$ based on surrogate cost $\tilde{l}_i(x_t, u_t)$, return samples D_i
    - **S-step:** train $\pi_\theta(u_t|x_t)$ with supervised learning on the dataset $D = \bigcup_i D_i$
    - Modify $\tilde{l}_i(x_t, u_t)$ to enforce agreement between $\pi_\theta(u_t|x_t)$ and each $p_i(u_t|x_t)$
      - (ensures convergence)

Notation

- $\pi_\theta(u_t|x_t)$: a parameterized policy over actions u_t conditioned on the state x_t
- $p_i(u_t|x_t)$: a local policy
- $l(x_t, u_t)$: a cost function (the "true" cost)

- $\tilde{l}_i(x_t, u_t)$: a surrogate cost function for local policy i (takes in state and action)
- $p(x_{t+1}|x_t, u_t)$: stochastic dynamics (probability of state at next time step, given state and action at current time step)
- $J(\theta) = \sum_{t=1}^{T} \mathbb{E}_{\pi_\theta(x_t,u_t)}[l(x_t, u_t)]$: expected cost under the policy's trajectory distribution
- Π_Θ: set of all possible policies π_θ for a given parameterization (i.e. set of trajectory distributions that are possible under the chosen parameterization)
- ϵ: step size (used in local policy optimization)

Terminology

- **KL divergence** D_{KL}: a measure of how much one probability distribution diverges from a second, expected probability distribution ("measure of surprise" like entropy/information gain).
- **Surrogate cost function:** chosen to induce correct behavior (optimize *this* instead of true reward function corresponding to declarative task description).
- **LQR optimization:** takes cost function and dynamics and solves for optimal control; requires low-dimensional state space; involves forward and backward pass.
  - *Cost* is quadratic, *control* is linear.
  - Task-based cost function must be well-shaped (encode priors).
  - Requires sampling to fit dynamics (run policy in simulation a bunch of times and sample).

HGPS Notes

- During GPS, we want to stay within covariance while demonstrating.
- In human GPS, we'll want to replace LQR with human.
- Human can help provide *control* ($\mathbb{R}^6$) and *noise* (scalar).

3 GPS Algorithms

- Aim of GPS: optimize $\pi_\theta(u_t|x_t)$ (equivalently: minimize $J(\theta)$, or the expected cost under policy's trajectory distribution).
- Instead of directly optimizing $J(\theta)$, optimization is split into two phases:
  - **C-step ("control phase"):** finds multiple simple local policies $p_i(u_t|x_t)$ that can solve the task from different initial states; uses local policy optimizer for each of these policies
  - **S-step ("supervised phase"):** optimizes the global policy $\pi_\theta(u_t|x_t)$ to match all of these local policies using standard supervised learning; uses samples from each local policy
- Main advantage of GPS: sample efficiency (policies can be trained with less experience than that required by direct deep RL methods)

4 Mirror Descent GPS (MDGPS)

- Uses constrained LQR optimization to optimize each of the local policies.
- Instead of constraining a local policy against its previous form, constrain it directly against the global policy (and set the surrogate cost to be the true cost).

MDGPS Algorithm

- Initialize random policy
- For GPS iteration k
  - For initial condition i from 0 to N
    - Run current controller M times to get samples for dynamics model (for policy i)
    - Fit linear dynamics $p_i(x_{t+1}|x_t, u_t)$
    - Fit linearized global policy $\bar{\pi}_{\theta i}(u_t|x_t)$
      - the bar means "linearized";
        linearize around specific condition to get good policy for local initial conditions
    - C-step: run LQR to optimize (*task cost*) + (*weighting term*) $\cdot$ (*KL divergence between current policy and global policy*) [Lagrangian of constraint optimization problem], get local policy p_i
    - S-step: use supervised learning to get a global policy π_θ by minimizing the KL divergence between the global policy π_θ and the local policy p_i
    - Shrink ϵ as we continue, because global policy gets better
      - initially ϵ is large and the global policy sucks

Alternatively

- For some number of iterations, do
  - Generate samples
  - Fit linear-Gaussian dynamics using samples
  - Fit linearized global policy using samples
  - C-step (optimize p_i's contingent on KL divergence)
  - S-step (optimize global policy π_0 using supervised learning)
  - Adjust step size

Notes

- ϵ: encodes how much we're willing to deviate from current global policy (small $\epsilon \implies$ small step).
- Usually five or so initial policies: $M \approx$ 3-5, $N \approx 10$.
- After we have the local LQR policies, we can fire off the S-step (supervised training of global policy):
  - Minimize sum of KL divergences with local policies
  - Global policy in distribution space tries to match all local policies
- Use KL divergence between local policies and the global policy. Provides constraint between whole trajectory of each local policy and whole trajectory of global policy.

References

[1] William Montgomery, Sergey Levine. Guided Policy Search as Approximate Mirror Descent. NIPS 2016.