Continuous Control in Bearing-only Localization

Cedrick Argueta
cedrick@cs.stanford.edu
Department of Computer Science, Stanford University, Stanford, CA 94305

Motivation

Localization is a controls problem that deals with minimizing uncertainty over position. It has applications in robotics, autonomous vehicles, aerospace, and more.

Overview

In this problem, a drone moves in a 2D box, looking for a moving radio transmitter that it can detect with an noisy antenna.

We implement the **TD3** algorithm, a slight modification of the commonly studied **DDPG** algorithm, and use it to control a simulated drone in a bearing-only localization task. We discuss the tradeoffs of continuous control vs. discrete control through **DQN** and a greedy solver, especially with respect to particle filters and partial observability.

Ultimately, all learning methods outperform the greedy solution, but continuous control does not outperform discrete control.

Simulation Environment

Training is performed in an open-source drone simulation environment, **PyFEBOL**. Check it out on GitHub, I wrote it!

The drone takes a noisy measurement of the target with the antenna. The particle filter is updated according to this measurement, and the controller uses this filter to perform inference. The drone must move to minimize uncertainty of the target. The target moves with constant velocity, and the drone moves with constant speed.

References

- L. Dressel and M. J. Kochenderfer. Hunting drones with other drones: Tracking a moving radio target. ICRA 2019
- L. Dressel and M. J. Kochenderfer. Pseudo-bearing measurements for improved localization of radio sources with multirotor uavs. ICRA 2018
- V. Mnih, et al. Playing atari with deep reinforcement learning. CoRR, abs/1312.5602, 2013.
- T. Lillicrap, et al. Continuous control with deep reinforcement learning. ICLR 2016
- S. Fujimoto, et al. Addressing function approximation error in actor-critic methods. CoRR, abs/1802.09477, 2018.

Algorithms and Model

TD3 and **DDPG** are both actor-critic algorithms for reinforcement learning, meaning that they directly optimize a policy w.r.t. expected return and maintain a value function to help. The left equation shows the solution to a **partially observable MDP**, while the right equation shows the basic policy gradient used in **DDPG** and **TD3**.

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(\omega_t))\right] \qquad \nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

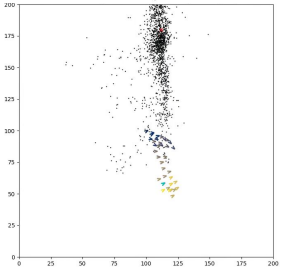
TD3 uses two critics for double Q learning, introduces noise to the actor policy for policy smoothing, and delays policy updates for reduced variance in policies generated.
Each **neural network** in **DQN**, **DDPG**, and **TD3** has approximately the same structure: seven state variables, hidden layers of sizes 512 and 384, then output dependent on the function approximated.

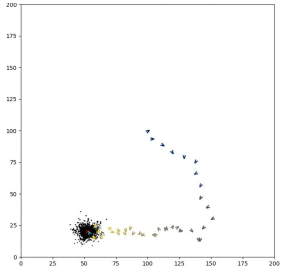
Actions
In the discrete setting, the controller selects one of 24 directions (radial about the drone) to travel in. In the continuous setting, the controller selects the degree indicating the heading.

Observations
The drone takes **imperfect observations** of the environment with a radio antenna, and integrates this information into a **particle filter**. Each state is: the drone's (x, y, h) position and heading, and the particle filter's mean (x, y) position and velocity (dx, dy). Note: the filter is necessarily **stochastic!**
The particle filter can be seen as a form of **feature engineering** or **model-based RL**: rather than directly use the bearing observation (30° east) we create a model of the environment and use that for inference.

Rewards
The goal of the controller is to localize the target *while* respecting a keep-out distance around the target. The target is localized when **entropy** in the particle filter is low.

$$c(s_t) = H(b_t) + \lambda \mathbb{E}_{b_t} \mathbb{1}(\|x_t - \theta_t\| < d) \qquad H(b_t) = -\sum_{i=1}^{M} \tilde{b}_t[i] \log \tilde{b}_t[i]$$





Results

Localization quality is represented by tracking error: if we have low tracking error, our mean hypothesis for the target position is good.
Collision rate shows the tradeoff of better observations vs. collisions, where a higher rate indicates riskier behavior.
Average cost combines these two metrics: it is the sum of nats of entropy in the particle filter and probability of collision according to the particle filter.

	Tracking Error	Collision Rate	Average Cost
Greedy	30.5m	1.8%	-4.18
DQN	**5.4m**	62%	**-2.37**
DDPG	20.1m	**0%**	-3.73
TD3	13.48m	**0%**	-3.75

Discussion

The **greedy** method acts as expected: since it takes the lowest cost action at every step, it never plans ahead to incur some penalty to get close to the target and then receive a much higher reward.

DQN performs fabulously, learning that the long-term reward from incurring the collision penalty is greatly outweighed by reduction in entropy. This also validates our choice of observations, showing that learning the optimal policy is possible.

TD3 and **DDPG** improve on the greedy solver's tracking error, but fail to learn that collisions lead to a more optimal policy. Why? It's likely due to a minimal hyperparameter search, as continuous control often takes longer during training (meaning less trials) and is more sensitive to hyperparameter choice.

Future Work

- Control from the raw particle filter might yield better results, as particle filters are often very non-gaussian and mean/std doesn't accurately represent belief.
- Training time is disappointing, is there a way to leverage the particle filter as a model to do real model-based RL instead of model-free RL?
- The simulations conform to specifications for an actual drone contained in SISL, so maybe these policies can actually be flown!