Motion Planning

Pros:
• Is able to explore efficiently without a reward signal.
• Most approaches require a model of the robot.

Cons:
• Learns trajectories, and trajectory following requires a model of the robot.

MP algorithm of choice — RRT [Lavalle, 1998].

We use a variant of RRT which does not require a model (the action is chosen randomly instead of using a heuristic).

Plan

Motion Planning is used to find a single valid trajectory from the environment start to a rewarded state. We designed a new algorithm called Ex inspired from [Hsu et al., 1997].

Algorithm 2: Ex algorithm

In: \( s_0 \in S \) the initial state
step: \( S \times A \rightarrow S \times R \times B \) the step function
Bin: \( S \rightarrow N \) a function that partitions the state-space in bins
Out: \( T \): an exploration tree

1. Initialize the exploration set \( T \) to a single node \( s_0 \)
2. \( c\_\_\_R \leftarrow 0 \)
3. while \( |\text{tree}| < \text{iterations} \) do
4. \( b = \arg \min \{ c(s), s \in T \} \)
5. \( s = \text{RANDOM\_ACTION}(s) \)
6. \( s\_s\_\_R \leftarrow c(s) \)
7. Increment \( c\_\_\_R \)
8. \( a = \text{RANDOM\_ACTION}(s) \)
9. \( s' = \text{STEP}(s, a) \)
10. \( T \leftarrow T \cup \{s', a\} \)
11. end

In this maze a sparse-reward target at coordinates \((3, 1.5)\) is easily found. The color range is the depth in the exploration tree.

Backplay

The trajectory obtained in the “Plan” phase is used as a curriculum for training DDPG. At first, the starting point of the environment is close to the goal, then moved backwards along the curriculum trajectory. This technique is similar to Go-Explore [Ezofet et al., 2019] and Backplay [Resnick et al., 2018].

Algorithm 3: Phase 2 of PBCS

In: \( \tau_0 \ldots \tau_T \) the output of the “Plan” phase
Out: \( \pi_0 \ldots \pi_N \), a chain of policies with activation acts \( A_0 \ldots A_N \)

1. \( T = N \)
2. \( n = 0 \)
3. while \( T > 0 \) do
4. \( \pi_n \leftarrow \text{Backplay}(\tau_n \ldots \tau_T) \)
5. \( A_n = \text{ball of radius } e \) centered around \( \tau_n \)
6. \( n \leftarrow n + 1 \)
7. end

Problem: the gradient descent of Eq. (2) may have local minima even when the reward is close and found through random actions! [Matheron et al., 2019]

To counter this, when training becomes stuck the last good policy and \( K \) are saved and used as stepping stones for skill chaining.

Reinforcement Learning

Pros:
• Does not require a model of the robot.
• Learns a robust controller.

Cons:
• Explores with inefficient random noise when no reward gradient is available.

Algorithm 4: RL algorithm of choice — DDPG [Lillicrap et al., 2015].

Experience is collected in a replay buffer and the equations to the right are used to update the networks.

DDPG maintains two function approximators:
• an actor \( \pi \) parameterized by \( \Psi \)
• a critic \( Q \) parameterized by \( \Theta \) that estimates the state-action value function \( Q^\ast \).

\[
Q(s_t, a_t) = \min_{\pi} \mathbb{E}[r_t + \gamma Q(s_{t+1}, \pi(s_{t+1}))]
\]

Problem: before finding the reward DDPG is only driven by random noise and explores poorly.

Chain Skills

The backplay process is wrapped in a skill chaining framework: when backplay fails, it outputs an intermediate point \( \tau \) and a policy \( \pi_\max \) that can drive the agent from \( \tau \) to \( \tau_\max \). Then a new controller is trained to solve the rest recursively.

Bibliography


