Continuous Deep Q-Learning with Model-based Acceleration

Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, Sergey Levine

ICLR 2016

Presented by Sanghyeon Lee

Motivation

Problem

Sample complexity of model-free algorithms tens to limit their applicability to physical systems

Solution

NAF(normalized advantage functions) as an alternative to policy gradient and actor-critic

Imagination rollout : on-policy samples generated under the learned model

(Model free + Model base ~ like Dyna-Q …)

Contributions:

1. Derive and evaluate an effective Q-function representation in **Continuous** domains.
2. Evaluate several naïve options for Incorporating learned model into model – free Q-learning and they has minimally effective on continuous control tasks
3. Combine locally linear models with local on-policy imagination rollouts to accelerate model free Q-learning, and show that this produces a large improvement in sample complexity

Notation

u_t: Action
x_t: state
$A^{\pi}(x_t, u_t) = Q^{\pi}(x_t, u_t)$- $V^{\pi}(x_t)$: Advantage function

Method

Background

1.1 Model–free Reinforcement Learning

*Q-learning

Greedy policy: $\mu(\mathrm{x_t}) = \mathrm{argmax_u Q(x_t, u_t)}$ & $\pi(\mathrm{u_t}|\mathrm{x_t}) = \delta(\mu(\mathrm{x_t}))$: Deterministic policy

1.2 Model–Based Reinforcement Learning

Model: $\mathrm{p(x_{t+1}|x_t, u_t)}$ Learned model: $\hat{p}$

iLQG Algorithm: Optimizes trajectories by iteratively constructing locally optimal linear feedback controller under a local linearization of the dynamics:

$$\hat{p}(x_{t+1}|x_t, u_t) = N(f_{xt}x_t + f_{ut}u_t, F_t)$$

Method

Background

iLQG: THE ITERATIVE LINEAR QUADRATIC REGULATOR ALGORITHM

Model: $\mathrm{p}(\mathrm{x}_{\mathrm{t}+1}|\mathrm{x}_{\mathrm{t}}, \mathrm{u}_{\mathrm{t}})$ & Learned model: $\hat{p}$

iLQG Algorithm: Optimizes trajectories by iteratively constructing locally optimal linear feedback controller under a local linearization of the dynamics:

$$\hat{p}(x_{t+1}|x_t, u_t) = N(f_{xt}x_t + f_{ut}u_t, F_t)$$

Q and V are locally quadratic

$$\begin{aligned} Q_{\boldsymbol{xu},\boldsymbol{xu}t} &= r_{\boldsymbol{xu},\boldsymbol{xu}t} + \boldsymbol{f}_{\boldsymbol{xu}t}^T V_{\boldsymbol{x},\boldsymbol{x}t+1} \boldsymbol{f}_{\boldsymbol{xu}t} \\ Q_{\boldsymbol{xu}t} &= r_{\boldsymbol{xu}t} + \boldsymbol{f}_{\boldsymbol{xu}t}^T V_{\boldsymbol{x},\boldsymbol{x}t+1} \\ V_{\boldsymbol{x},\boldsymbol{x}t} &= Q_{\boldsymbol{x},\boldsymbol{x}t} - Q_{\boldsymbol{u},\boldsymbol{x}t}^T Q_{\boldsymbol{u},\boldsymbol{u}t}^{-1} Q_{\boldsymbol{u},\boldsymbol{x}t} \\ V_{\boldsymbol{x}t} &= Q_{\boldsymbol{x}t} - Q_{\boldsymbol{u},\boldsymbol{x}t}^T Q_{\boldsymbol{u},\boldsymbol{u}t}^{-1} Q_{\boldsymbol{u}t} \\ Q_{\boldsymbol{x},\boldsymbol{x}T} &= V_{\boldsymbol{x},\boldsymbol{x}T} = r_{\boldsymbol{x},\boldsymbol{x}T} \end{aligned}$$

$$\pi_t^{iLQG}(\boldsymbol{u}_t|\boldsymbol{x}_t) = \mathcal{N}(\hat{\boldsymbol{u}}_t + \boldsymbol{k}_t + \boldsymbol{K}_t(\boldsymbol{x}_t - \hat{\boldsymbol{x}}_t), -cQ_{\boldsymbol{u},\boldsymbol{u}t}^{-1})$$

Method

Continuous Q-Learning with Normalized Advantage Functions

$$Q(x,u|\theta^Q) = A(x,u|\theta^A) + V(x|\theta^V)$$
$$A(x,u|\theta^A) = -\frac{1}{2}(x-\mu(x|\theta^\mu))^T P(x|\theta^P)(u-\mu(x|\theta^\mu)) \text{ s.t}$$
$$P(x|\theta^P) = L(x|\theta^P)L(x|\theta^P)^T$$

$L(x|\theta^P)$:Lower-triangular matrix & output of Neural networks diagonal terms exponentiated

-Then Q function is always given by deterministic policy $\mu(x|\theta^\mu)$

-This NAF algorithm is considerably simpler than DDPG

➔ Avoids the needs for Actor and Critic

- Locally-Invariant Exploration for Normalized Advantage Function (Adaptive exploration strategy)

$$\pi(u|x) = \frac{\exp(Q(x,u|\theta^Q))}{\int \exp(Q(x,u|\theta^Q)du} = N(\mu(x|\theta^\mu), cP(x|\theta^P)^{-1})$$

Algorithm 1 Continuous Q-Learning with NAF

Randomly initialize normalized Q network $Q(\boldsymbol{x}, \boldsymbol{u}|\theta^Q)$.
Initialize target network Q' with weight $\theta^{Q'} \leftarrow \theta^Q$.
Initialize replay buffer $R \leftarrow \emptyset$.
for episode=1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state $\boldsymbol{x}_1 \sim p(\boldsymbol{x}_1)$
  for t=1, T **do**
    Select action $\boldsymbol{u}_t = \mu(\boldsymbol{x}_t|\theta^\mu) + \mathcal{N}_t$
    Execute $\boldsymbol{u}_t$ and observe r_t and $\boldsymbol{x}_{t+1}$
    Store transition $(\boldsymbol{x}_t, \boldsymbol{u}_t, r_t, \boldsymbol{x}_{t+1})$ in R
    for iteration=1, I **do**
      Sample a random minibatch of m transitions from R
      Set $y_i = r_i + \gamma V'(\boldsymbol{x}_{i+1}|\theta^{Q'})$
      Update θ^Q by minimizing the loss: $L = \frac{1}{N}\sum_i(y_i - Q(\boldsymbol{x}_i, \boldsymbol{u}_i|\theta^Q))^2$
      Update the target network: $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$
    end for
  end for
end for

Advantage function

Method

Accelerating Learning with Imagination Rollouts

Improve data efficiency under some additional assumptions by Exploiting learned model

1) Model Guided Exploration

-Incorporating a learned model into an off-policy algorithm

-Authors utilize the iLQG to generate good trajectories under the model and mix this trajectories by appending them to the replay buffer

- But even use trajectories under the true model, this improvement are small

  : Off policy iLQG exploration is too different from the learned policy

Method

Accelerating Learning with Imagination Rollouts

2)Imagination Rollouts

- In the previous section, 1), incorporating off-policy exploration from iLQG(good, narrow distribution) dose not result in significant improvement for Q-learning

  ➔ Q-learning inherently requires noisy on-policy
- But in the real world, this can be undesirable

  1) Large amounts of on-policy experiences are required

  2) Policy must be allowed to make “its own mistakes”
- Imagination rollouts:

  <u>Adding trajectories base on both Q-learning behavior and iLQG policy</u>

  But this model can suffer from bias➔ Fitting time-varying linear dynamics

Method

Accelerating Learning with Imagination Rollouts

3) Fitting the Dynamics Model

- Author aim to only to obtain a good local model around the latest set of samples

  A1) Initial state to be either deterministic or low-variance Gaussian

  A2) State and action to be continuous

- Gaussian initial state with separate time-varying linear models

$$\hat{p}(x_{t+1}|x_t, u_t) = N(F_t[x_t; u_t] + f_t, N_t)$$

- Every n episodes, we can refit the parameter F_t, f_t , N_t from Gaussian distribution from samples x_t^i, u_t^i, x_{t+1}^i ; i indicates the sample index

- Although this method needs additional assumption, we can gains impressive sample efficiency

Method

Accelerating Learning with Imagination Rollouts

Algorithm 2 Imagination Rollouts with Fitted Dynamics and Optional iLQG Exploration

Randomly initialize normalized Q network $Q(\boldsymbol{x}, \boldsymbol{u}|\theta^Q)$.
Initialize target network Q' with weight $\theta^{Q'} \leftarrow \theta^Q$.
Initialize replay buffer $R \leftarrow \emptyset$ and fictional buffer $R_f \leftarrow \emptyset$.
Initialize additional buffers $B \leftarrow \emptyset, B_{old} \leftarrow \emptyset$ with size nT.
Initialize fitted dynamics model $\mathcal{M} \leftarrow \emptyset$.
for $episode = 1, M$ **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state $\boldsymbol{x}_1$
  Select $\mu'(\boldsymbol{x}, t)$ from $\{\mu(\boldsymbol{x}|\theta^\mu), \pi_t^{iLQG}(\boldsymbol{u}_t|\boldsymbol{x}_t)\}$ with probabilities $\{p, 1-p\}$
  for $t = 1, T$ **do**
    Select action $\boldsymbol{u}_t = \mu'(\boldsymbol{x}_t, t) + \mathcal{N}_t$
    Execute $\boldsymbol{u}_t$ and observe r_t and $\boldsymbol{x}_{t+1}$
    Store transition $(\boldsymbol{x}_t, \boldsymbol{u}_t, r_t, \boldsymbol{x}_{t+1}, t)$ in R and B
    if $\mod(episode \cdot T + t, m) = 0$ and $\mathcal{M} \neq \emptyset$ **then**
      Sample m $(\boldsymbol{x}_i, \boldsymbol{u}_i, r_i, \boldsymbol{x}_{i+1}, i)$ from B_{old}
      Use $\mathcal{M}$ to simulate l steps from each sample
      Store all fictional transitions in R_f
    end if
    Sample a random minibatch of m transitions $I \cdot l$ times from R_f and I times from R, and update $\theta^Q, \theta^{Q'}$ as in Algorithm 1 per minibatch.
  end for
  if B_f is full **then**
    $\mathcal{M} \leftarrow$ FitLocalLinearDynamics(B_f) (see Section 5.3)
    $\pi^{iLQG} \leftarrow$ iLQG_OneStep$(B_f, \mathcal{M})$ (see appendix)
    $B_{old} \leftarrow B_f, B_f \leftarrow \emptyset$
  end if
end for

Imagination Rollouts

Algorithm 1 Continuous Q-Learning with NAF

Randomly initialize normalized Q network $Q(\boldsymbol{x}, \boldsymbol{u}|\theta^Q)$.
Initialize target network Q' with weight $\theta^{Q'} \leftarrow \theta^Q$.
Initialize replay buffer $R \leftarrow \emptyset$.
for episode=1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state $\boldsymbol{x}_1 \sim p(\boldsymbol{x}_1)$
  for t=1, T **do**
    Select action $\boldsymbol{u}_t = \mu(\boldsymbol{x}_t|\theta^\mu) + \mathcal{N}_t$
    Execute $\boldsymbol{u}_t$ and observe r_t and $\boldsymbol{x}_{t+1}$
    Store transition $(\boldsymbol{x}_t, \boldsymbol{u}_t, r_t, \boldsymbol{x}_{t+1})$ in R
    for iteration=1, I **do**
      Sample a random minibatch of m transitions from R
      Set $y_i = r_i + \gamma V'(\boldsymbol{x}_{i+1}|\theta^{Q'})$
      Update θ^Q by minimizing the loss: $L = \frac{1}{N}\sum_i(y_i - Q(\boldsymbol{x}_i, \boldsymbol{u}_i|\theta^Q))^2$
      Update the target network: $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$
    end for
  end for
end for

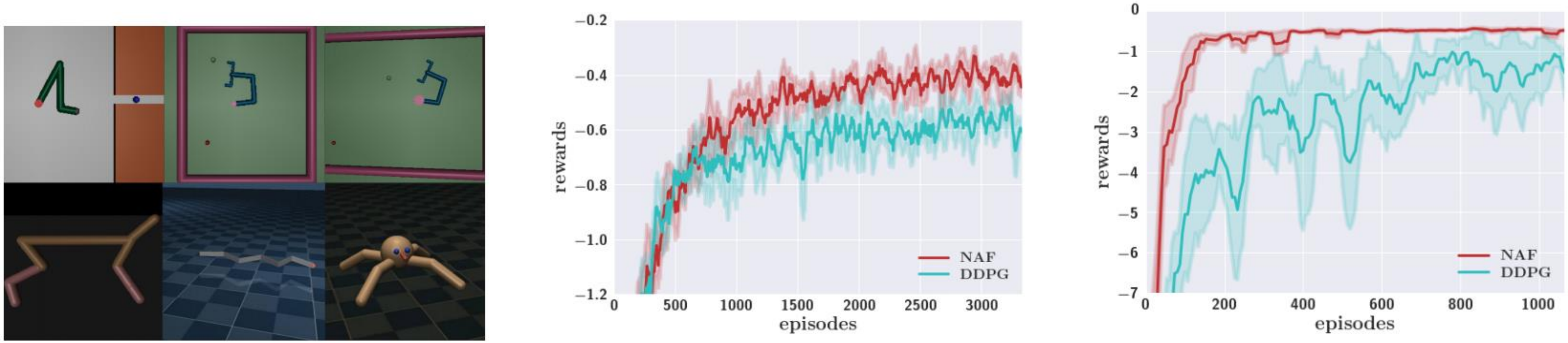
Experiments

Env: Robotic tasks in MuJoCo

Benchmarks paper: Continuous control with deep RL Lillicrap et al.(ICLR2016) -DDPG

1. Normalized Advantage Function

− Compare NAF and DDPG



(a) Example task domains. (b) NAF and DDPG on multi-target reacher. (c) NAF and DDPG on peg insertion.

Figure 1. (a) Task domains: top row from left (manipulation tasks: peg, gripper, mobile gripper), bottom row from left (locomotion tasks: cheetah, swimmer6, ant). (b,c) NAF vs DDPG results on three-joint reacher and peg insertion. On reacher, the DDPG policy continuously fluctuates the tip around the target, while NAF stabilizes well at the target.

Experiments

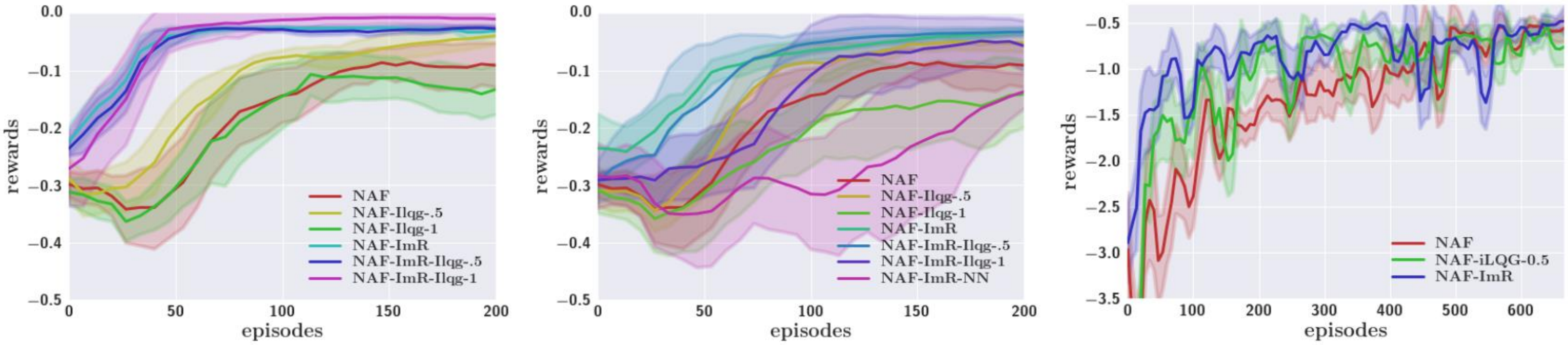
Domains	-	DDPG	episodes	NAF	episodes
Cartpole	-2.1	-0.601	420	-0.604	**190**
Reacher	-2.3	-0.509	1370	**-0.331**	**1260**
Peg	-11	-0.950	690	**-0.438**	**130**
Gripper	-29	1.03	2420	**1.81**	**1920**
GripperM	-90	-20.2	1350	**-12.4**	**730**
Canada2d	-12	-4.64	1040	**-4.21**	900
Cheetah	-0.3	**8.23**	**1590**	7.91	2390
Swimmer6	-325	-174	220	**-172**	**190**
Ant	-4.8	-2.54	2450	-2.58	**1350**
Walker2d	0.3	**2.96**	**850**	1.85	1530

Table 1. Best test rewards of DDPG and NAF policies, and the episodes it requires to reach within 5% of the best value. "-" denotes scores by a random agent.

Experiments

2. Evaluating Best-Case Model-Based Improvement with True Model

3.Guided Imagination Rollouts with Fitted Dynamics



(a) NAF on single-target reacher. (b) NAF on single-target reacher. (c) NAF on single-target gripper.

Figure 2. Results on NAF with iLQG-guided exploration and imagination rollouts (a) using true dynamics (b,c) using fitted dynamics. "ImR" denotes using the imagination rollout with $l = 10$ steps on the reacher and $l = 5$ steps on the gripper. "iLQG-x" indicates mixing x fraction of iLQG episodes. Fitted dynamics uses time-varying linear models with sample size $n = 5$, except "-NN" which fits a neural network to global dynamics.