Reinforcement Learning of Parameters in Complex Physical Systems

Nemo Fournier

September 4, 2019

Outline

Progress in Robotic Hardware

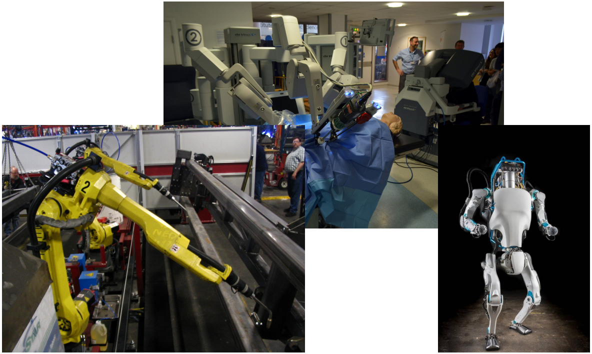

Figure: Up: Da Vinci chirurgical robot. Left: Fanuc welding robot. Right: Boston Dynamics' Atlas robot *(Images from Wikimedia)*

Progress in Machine Learning

Figure: Up: Alphago match. Left: Dota2 AI. Right: Atari AI

Main Principles of Reinforcement Learning

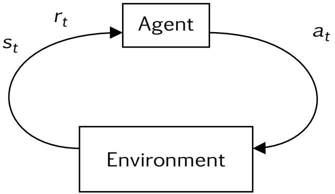

Figure: Reinforcement Learning (RL) feedback loop of the interactions between the agent and the environment.

Some formalization

Markov Decision Process

A MDP $\mathcal{M}$ is a tuple $(\mathcal{S}, \mathcal{A}, r, \gamma, p, p_0)$

- $\mathcal{S}$: set of states
- $\mathcal{A}$: set of actions
- $r : \mathcal{S} \times \mathcal{A} \to \mathbf{R}$: reward
- γ: discount factor
- $p : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$: transition
- $p_0 \in \mathcal{P}(\mathcal{S})$: initial state

Some formalization

Markov Decision Process

A MDP $\mathcal{M}$ is a tuple $(\mathcal{S}, \mathcal{A}, r, \gamma, p, p_0)$

- $\mathcal{S}$: set of states
- $\mathcal{A}$: set of actions
- $r : \mathcal{S} \times \mathcal{A} \to \mathbf{R}$: reward
- γ: discount factor
- $p : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$: transition
- $p_0 \in \mathcal{P}(\mathcal{S})$: initial state

Policy

$\pi : \mathcal{S} \to \mathcal{A}$ or $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$

The RL problem

Trajectory

$\tau = (s_0, a_0, s_1, a_1, s_2, \ldots, s_a)$ is a trajectory over $\mathcal{M}$ using a policy π if $s_0 \sim p_0$, and for $t \geq 0$, $a_t \sim \pi(\cdot \mid s_{t-1})$ and $s_{t+1} \sim p(s_t, a_t)$.
We denote $\mathrm{T}_{\mathcal{M},\pi}$ the distribution of such trajectories.

The RL problem

Trajectory

$\tau = (s_0, a_0, s_1, a_1, s_2, \ldots, s_a)$ is a trajectory over $\mathcal{M}$ using a policy π if $s_0 \sim p_0$, and for $t \geq 0$, $a_t \sim \pi(\cdot \mid s_{t-1})$ and $s_{t+1} \sim p(s_t, a_t)$.
We denote $\mathrm{T}_{\mathcal{M},\pi}$ the distribution of such trajectories.

Performance of a policy

$$J(\pi) = \mathop{\mathbb{E}}_{\tau \sim \mathrm{T}_{\mathcal{M},\pi}} \left[\sum_{t=0}^{H} \gamma^t r(s_t, a_t) \right]$$

The RL problem

Trajectory

$\tau = (s_0, a_0, s_1, a_1, s_2, \ldots, s_a)$ is a trajectory over $\mathcal{M}$ using a policy π if $s_0 \sim p_0$, and for $t \geq 0$, $a_t \sim \pi(\cdot \mid s_{t-1})$ and $s_{t+1} \sim p(s_t, a_t)$.
We denote $\mathrm{T}_{\mathcal{M},\pi}$ the distribution of such trajectories.

Performance of a policy

$$J(\pi) = \mathop{\mathbb{E}}_{\tau \sim \mathrm{T}_{\mathcal{M},\pi}} \left[\sum_{t=0}^{H} \gamma^t r(s_t, a_t) \right]$$

$$\pi^* = \arg\max_{\pi} J(\pi)$$

Solving the RL problem: Policy Gradient Method

Idea: parametrize a policy π_θ and perform gradient ascent:

$$\theta_{t+1} \leftarrow \theta_t + \alpha \nabla J(\theta_t)$$

Solving the RL problem: Policy Gradient Method

Idea: parametrize a policy π_θ and perform gradient ascent:

$$\theta_{t+1} \leftarrow \theta_t + \alpha \nabla J(\theta_t)$$

- REINFORCE
- Actor-Critic
- DPG
- TRPO
- PPO

A Robotics Problem is a RL problem

- $\mathcal{S}$
- $\mathcal{A}$
- $p : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$

A Robotics Problem is a RL problem

- $\mathcal{S}$
- $\mathcal{A}$
- $p : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$
- $r : \mathcal{S} \times \mathcal{A} \to \mathbf{R}$

Issues of RL when applied to robotics

- Sampling efficiency
- Random exploration
- Real-time rollouts

Simulation

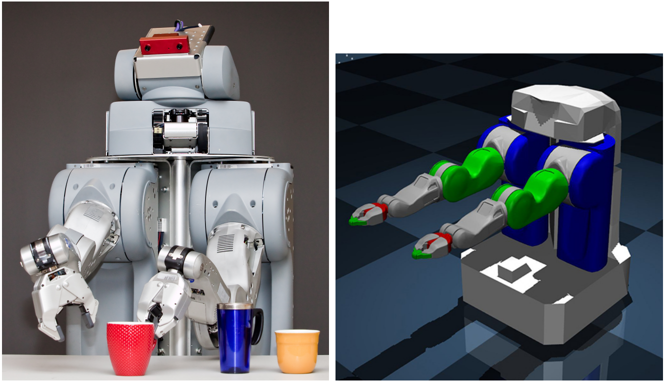

Figure: A real PR2 robot and its simulated equivalent.

Strategies to Cross the Reality Gap

- Several learning phases
- Assess live discrepancies
- Dynamics randomization

Dynamics Randomisation

Peng et al.[1] introduced parametrization of the environment using a vector ϕ.

[1] X.B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. *Sim-to-real transfer of robotic control with dynamics randomization.* 2018

Dynamics Randomisation

Peng et al.[1] introduced parametrization of the environment using a vector ϕ.

$$J_\phi(\pi) = \mathop{\mathbb{E}}_{\tau \sim \mathrm{T}_{\mathcal{M},\pi,\phi}} \sum_{t=0}^{H} \gamma^t r(s_t, a_t)$$

[1] X.B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. *Sim-to-real transfer of robotic control with dynamics randomization.* 2018

Dynamics Randomisation

Peng et al.[1] introduced parametrization of the environment using a vector ϕ.

$$J_\phi(\pi) = \mathop{\mathbb{E}}_{\tau \sim \mathrm{T}_{\mathcal{M},\pi,\phi}} \sum_{t=0}^{H} \gamma^t r(s_t, a_t)$$

$$\pi^* = \arg\max_{\pi} \mathop{\mathbb{E}}_{\phi \sim \rho} \left[J_\phi(\pi) \right]$$

[1] X.B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. *Sim-to-real transfer of robotic control with dynamics randomization.* 2018

Universal policy

Yu et al.[2] introduced the parametrized policy $\pi_\phi = \pi(\cdot \mid \phi)$.

$$\pi^* = \arg\max_{\pi} \mathop{\mathbb{E}}_{\phi \sim \rho} \left[J_\phi(\pi_\phi) \right]$$

[2] W. Yu, J. Tan, C. K. Liu, and G. Turk. *Preparing for the unknown: Learning a universal policy with online system identification.* 2017

Universal policy

Yu et al.[2] introduced the parametrized policy $\pi_\phi = \pi(\cdot \mid \phi)$.

$$\pi^* = \arg\max_{\pi} \mathop{\mathbb{E}}_{\phi \sim \rho} \left[J_\phi(\pi_\phi) \right]$$

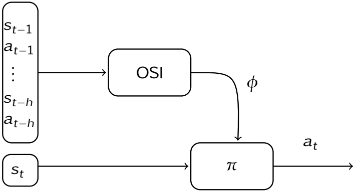

Figure: Universal Policy with Online System Identification

[2] W. Yu, J. Tan, C. K. Liu, and G. Turk. *Preparing for the unknown: Learning a universal policy with online system identification.* 2017

New Issues of those two Methods (DR and UP-OSI)

- Sampling efficiency

New Issues of those two Methods (DR and UP-OSI)

- Sampling efficiency
- Curse of dimensionality

New Issues of those two Methods (DR and UP-OSI)

- Sampling efficiency
- Curse of dimensionality
- Choice of relevant parameters

Dimensionality Reduction

We want a mapping

$$\Psi : \Phi \to \widetilde{\Phi}$$

Dimensionality Reduction

We want a mapping

$$\Psi : \Phi \rightarrow \widetilde{\Phi}$$

$$\pi^*, \Psi^* = \underset{\pi, \Psi}{\arg\max} \underset{\phi \sim \rho}{\mathbb{E}} \left[J_\phi \left(\pi_{\Psi(\phi)} \right) \right]$$

Autoencoders

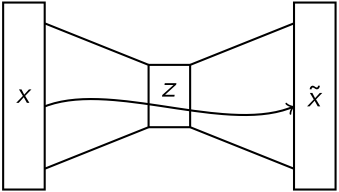

Figure: Standard autoencoder representation

Our architecture

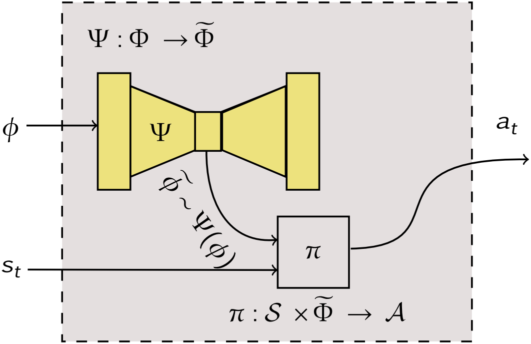

Figure: The new architecture we proposed.

Analysing the embedding

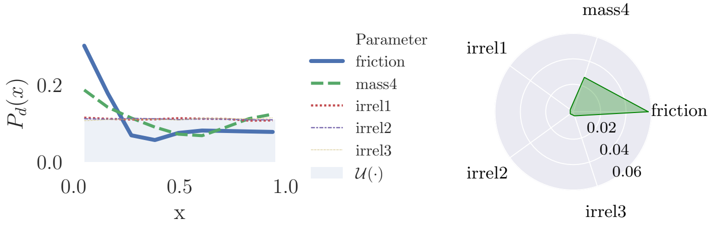

Figure: Toy problem on the Hopper environment.

Training the embedded OSI

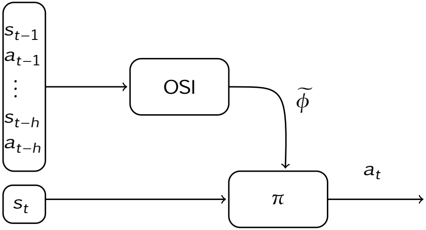

Figure: Embedded Universal Policy with Embedded Online System Identification

Results

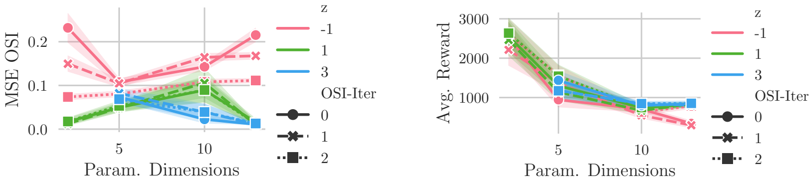

Figure: Effect of the embedding in terms of (Left) OSI prediction error and (Right)

Transferability

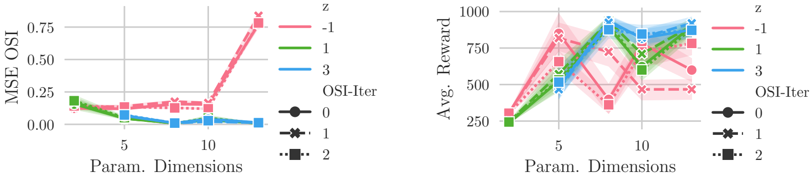

Figure: Effect of the embedding for transfer in terms of (Left) OSI prediction error and (Right) Average reward on the task.

Conclusion

- Promising direction and results
- Better evaluation needs to be done