Parameter Space Noise for Exploration
Matthias Plappert

APRIL 10, 2018
Agenda

- OpenAI
- Robotics at OpenAI
- A Brief Introduction to Reinforcement Learning
- Parameter Space Noise for Exploration
About Me

- Matthias Plappert
- since 2017: Research at OpenAI Robotics
- 2011 - 2017: Computer Science at Karlsruhe Institute of Technology (B.Sc. and M.Sc.)
- Before that and in between: iOS software development
- I like hiking, camping, running, and climbing
OpenAI's Mission

“OpenAI is a non-profit AI research company, discovering and enacting the path to safe artificial general intelligence.”
OpenAI Charter

- Broadly Distributed Benefits
- Long-Term Safety
- Technical Leadership
- Cooperative Orientation

Full text available on our blog: https://blog.openai.com/openai-charter/
Robotics Release – Environments

- Realistic robot environments
- Integrated with OpenAI Gym
- Goal-based formulation
- Sparse rewards
Robotics Release – Hindsight Experience Replay

- Learning from failed attempts
- Can be combined with any off-policy RL algorithm
- Allows us to solve sparse tasks
Multi-Goal Reinforcement Learning: Challenging Robotics Environments and Request for Research

Martineau Huguet, Michele Andrychowicz, Abhishek Varma, Rob Malcolm, Bertram Piesse, Greg Zissner, Daniel Schuster, Ido Tenev, Martino Chirico, Peter Wolter, Milan Maimon, and Wojciech Zaremba
Crysalis
Correspondence to: Martino Chirico, machi@csail.mit.edu

Abstract
The purpose of this technical report is to describe the Environment with Multiple Goals (E-MG) and to motivate the need for research in this area. The report introduces a suite of environmental contexts that are integrated with the OpenAI Gym environment. The report includes a detailed analysis of the E-MG environments, with a focus on the task definitions, level of complexity, and the level of difficulty. The report also includes a description of the research questions and the potential impact of this work. The report concludes with a discussion of the future research directions and the potential impact of this work.
Robotics Release – Results

Hand Manipulate Block—Rotate XYZ

Median Test Success Rate

Epoch

- DDPG+HER with sparse rewards
- DDPG+HER with dense rewards
- DDPG with sparse rewards
- DDPG with dense rewards
Robotics Release – Results
Robotics Release – Further Reading

- Blog post
  https://blog.openai.com/ingredients-for-robotics-research/

- "Hindsight Experience Replay", Andrychowicz et al., 2017

- "Multi-Goal Reinforcement Learning: Challenging Robotics Environments and Request for Research", Plappert et al., 2018
Robotics – Robots that Learn

- "One-Shot Imitation Learning", Duan et al., 2017
- "Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World", Tobin et al., 2017
Robotics – Generalizing from Simulation

- "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization", Peng et al., 2017
- "Asymmetric Actor Critic for Image-Based Robot Learning", Pinto et al., 2017
Parameter Space Noise for Exploration
A Brief Introduction to Reinforcement Learning
Reinforcement Learning (1)

Agent

Environment

Action $a_t$

State $s_{t+1}$ and Reward $r_t := r(s_t, a_t)$
Formalize as **Markov decision process** $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \rho, r)$ with

- Set of states $\mathcal{S}$
- Set of actions $\mathcal{A}$
- Reward function $r: \mathcal{S} \times \mathcal{A} \to \mathbb{R}$
- Initial state distribution $s_0 \sim \rho(\cdot)$
- State transition distribution: $s_{t+1} \sim \mathcal{P}(\cdot | s_t, a_t)$

Agent uses a **policy** to select actions:

$$a_t \sim \pi(\cdot | s_t)$$
Reinforcement Learning (3)

- Let \( \tau \) denote a trajectory with \( s_0 \sim \rho(\cdot), a_t \sim \pi(\cdot \mid s_t), s_{t+1} \sim \mathcal{P}(\cdot \mid s_t, a_t) \)

- The discounted return is then defined as:

\[
R(\tau) := \sum_{t} \gamma^t r(s_t, a_t), \text{with } \gamma \in [0, 1)
\]

- We wish to find a policy \( \pi^* \) that maximizes the expected discounted return:

\[
\pi^* := \arg\max_{\pi} \mathbb{E}_\tau[R(\tau)]
\]

- Also notice that we have to discover states with high rewards since we assume no a-priori information about our environment → requires exploration
Policy Gradients (1)

- Let’s assume a **parameterized policy** $\pi_\theta$, where $\theta$ is some parameter vector.

- Rewrite our optimization objective as:

  $$ J(\theta) = \mathbb{E}_\tau[R(\tau)], $$

  where $\mathbb{E}_\tau$ has a dependency on $\theta$ through $a_t \sim \pi_\theta(\cdot \mid s_t)$ but, importantly, $R(\tau)$ has no dependence on $\theta$ (it simply computes the return of any given trajectory).

- Simple idea: Let’s **compute the gradient** and do gradient ascent on this objective!
Policy Gradients (2)

- We can expand the expectation as follows:

\[
\nabla_{\theta} J(\theta) = \nabla_{\theta} \mathbb{E}_{\tau}[R(\tau)] \\
= \int R(\tau) \nabla_{\theta} p(\tau) d\tau \\
\]

- Furthermore, we can use the log derivative trick:

\[
\nabla_{\theta} \log p(\tau) = \frac{1}{p(\tau)} \nabla_{\theta} p(\tau) \\
\n\nabla_{\theta} p(\tau) = \nabla_{\theta} \log p(\tau) p(\tau) \\
\]

- Plugging this back in, we obtain:

\[
\nabla_{\theta} J(\theta) = \int R(\tau) \nabla_{\theta} \log p(\tau) p(\tau) \\
= \mathbb{E}_{\tau}[R(\tau) \nabla_{\theta} \log p(\tau)] \\
\]
Policy Gradients (3)

- So we just have to worry about computing $\nabla_\theta \log p(\tau)$!

- $p(\tau) = \rho(s_0) \prod_t P(s_{t+1} | s_t, a_t) \pi(a_t | s_t)$

- Taking the log:
  $\log p(\tau) = \log \rho(s_0) + \sum_t \log P(s_{t+1} | s_t, a_t) + \log \pi_\theta(a_t | s_t)$

- Only $\pi$ has a dependence on $\theta$, thus we obtain:
  $$\nabla_\theta \log p(\tau) = \sum_t \nabla_\theta \log \pi_\theta(a_t | s_t)$$

- $\nabla_\theta J(\theta) = \mathbb{E}_\tau [R(\tau) \nabla_\theta \log p(\tau)]$
  $$= \mathbb{E}_\tau [R(\tau) \sum_t \nabla_\theta \log \pi_\theta(a_t | s_t)]$$
Policy Gradients (4)

- The expectation can be estimated using Monte Carlo sampling, which corresponds to rolling out the policy multiple times to collect $N$ trajectories: $\tau^{(1)}, \tau^{(2)}, \ldots, \tau^{(N)}$

- The estimate for the policy gradient is thus:

$$\hat{g} = \frac{1}{N} \sum_n \left[ R(\tau^{(n)}) \sum_t \nabla_{\theta} \log \pi_\theta (a_t^{(n)} | s_t^{(n)}) \right]$$
Q-learning (1)

- Policy gradients work but ...
  - gradient estimates have large variance
  - throw away all data after every policy update

- Introduce the state-action value function $Q^\pi(s, a)$

- Informally, $Q^\pi(s, a)$ gives us the expected discounted return if we take action $a$ in state $s$ and follow $\pi$ afterwards

- Define $\mathcal{T}^\pi$ as the Bellman operator:
  $$(\mathcal{T}^\pi Q)(s, a) = r(s, a) + \gamma \mathbb{E}_\pi [Q(s', a')]$$

- $\mathcal{T}^\pi$ is a contraction and applying it iteratively convergences to the fixed point $Q^\pi(s, a)$, for any given initial $Q$! (Banach fixed-point theorem)
Q-learning (2)

- Assuming we have the optimal $Q^*$, the optimal policy is given by $\pi^*$:
  \[ \pi^*(s) = \arg\max_a Q^*(s, a) \]

- Conversely, $\mathcal{T}^*$ has the same properties as $\mathcal{T}^\pi$ and we can therefore compute $Q^*$ by iteratively applying:
  \[
  (\mathcal{T}^* Q)(s, a) = r(s, a) + \gamma \mathbb{E}_{\pi^*}[Q(s', a')]
  = r(s, a) + \gamma \arg\max_{a'} Q^*(s', a')
  \]

- Still need to explore in order to “see” entire state space

- Tabular case breaks down as both state and action space become large
  \[ \rightarrow \text{approximate } Q, \text{ e.g. using neural networks} \]
Deep Q-Networks

- DQN, Mnih et al., 2013 & 2015

- Parameterize Q-function using a deep neural network and learn:
  \[ Q_\theta(s, a) = r(s, a) + \gamma \max_{a'} Q_\theta(s', a') \]
  i.e. it uses Q-learning

- Policy is defined implicitly: \( \pi(s) := \arg\max_{a'} Q_\theta(s, a') \)

- Works with discrete action spaces

- Typically uses \( \varepsilon \)-greedy exploration:
  - With probability \( \varepsilon \), select a random action
  - Otherwise, select \( \pi(s) \)

- No formal convergence guarantees anymore due to approximate nature
Deep Deterministic Policy Gradient

- DDPG, Lillicrap et al., 2015

- What if we want to use DQN but have a continuous action space?

- DDPG uses actor-critic architecture with two networks:
  \[
  Q_\phi(s, a) = r(s, a) + \gamma Q_\phi(s', \pi_\theta(s')) \\
  \pi_\theta(s) = \arg\max_{\pi_\theta} Q_\phi(s, \pi_\theta(s))
  \]

- Typically uses additive Gaussian exploration:
  \[
  \hat{\pi}(s) := \pi_\theta(s) + \mathcal{N}(\mathbf{0}, \sigma^2 I)
  \]
Further Reading


Selected deep RL papers

- DQN: https://www.nature.com/articles/nature14236
- DDPG: https://arxiv.org/abs/1509.02971
- PPO: https://arxiv.org/abs/1707.06347
Parameter Space Noise for Exploration
Motivation

- Typically, exploration is realized in the action space:
  \[
  \hat{\pi}(s) := \pi_\theta(s) + \mathcal{N}(0, \sigma^2 I)
  \]

- However, this leads to inconsistent exploration since the noise is not conditioned on the state.
Formulation

What if we apply noise to the parameters of the policy instead?

Define \[ \hat{\pi}(s) := \pi_{\theta}(s) \text{ with } \theta := \theta + \mathcal{N}(0, \sigma^2 I) \]
Problems

- Recall that $\hat{\theta} := \theta + \mathcal{N}(0, \sigma^2 I)$

- We use a scalar $\sigma$ to perturb the weights of a deep network (Problem 1)
  - Such a network will likely have many layers
  - Each layer likely has different sensitivities to noise

- We have to pick a suitable scalar $\sigma$ (Problem 2)
  - In action space noise, the effect is intuitively understandable
  - In contrast, what does perturbing the weights of the policy mean?
  - Furthermore, the sensitivity of the policy to perturbations is likely changing as training progresses
Problem 1

- Use a similar re-parameterization as proposed in Salimans et al., 2017

- We use layer normalization (Ba et al., 2016)

\[ h = f \left[ \frac{g}{\sigma} \odot (a - \mu) + b \right] \]

with \( a = Wx \) and \( \mu \) and \( \sigma \) are estimated over \( a \)

- \( h \) ends up with approximately zero mean and unit variance

- Perturbation to \( W \) becomes invariant to sensitivity of that layer

- Hinges on assumption that units within layer are similar
Problem 2

- Reasoning about $\sigma$ in parameter space is hard

- Idea: Think about the effect of a perturbation in action space:

![Diagram showing the effect of noise on parameter space with states and actions.]

Distance $d(\pi, \tilde{\pi})$
Problem 2

- Reasoning about $\sigma$ in parameter space is hard

- Idea: Think about the **effect of a perturbation in action space**:

  $d_k := \mathbb{E}_s[d(\pi(\cdot \mid s), \hat{\pi}(\cdot \mid s))]$

  using some distance / divergence measure $d(\cdot, \cdot)$

- Adaptively change $\sigma$:

  $\sigma_{k+1} = \begin{cases} 
  \alpha \sigma_k, & d_k \leq \delta \\
  \frac{1}{\alpha} \sigma_k, & \text{otherwise}
  \end{cases}$
Experiments (1)

- We test for exploration on a simple but scalable toy environment

$ r = 0.001 \quad S_1 \quad S_2 \quad S_3 \quad \cdots \quad S_{N-1} \quad S_N \quad r = 1$

- Experiments on DQN with different exploration methods

![Graphs showing performance of different exploration methods](chart.png)
Experiments (2)

- Evaluation on 20 Atari games
- DQN with different exploration methods
- Exploration behavior of $\epsilon$-greedy (left) vs. parameter space noise (right)
Experiments (3)

Enduro

- Parameter noise
- $\epsilon$-greedy, separate policy head
- $\epsilon$-greedy
Experiments (4)

- Evaluation on 7 MuJoCo continuous control problems
- DDPG with different exploration methods
- Exploration of additive Gaussian noise (left) vs. parameter space noise (right)
Experiments (5)

HalfCheetah-v1

- parameter space noise
- correlated action noise
- uncorrelated action noise
- no exploration

Median Return vs Step

0.0 0.2 0.4 0.6 0.8 1.0

0 1000 2000 3000 4000 5000

1e6
Experiments (6)

- Combine parameter space noise with Hindsight Experience Replay (HER)
- Evaluate on 3 robotics tasks with sparse rewards
Experiments (7)

Pick & Place

Median Success Rate

Epoch

DDPG w/ parameter noise
DDPG w/ action noise
HER w/ parameter noise
HER w/ action noise
Related Work

- Concurrently to our work, DeepMind has proposed “Noisy Networks for Exploration”, Fortunato et al., 2017

- “Deep Exploration via Bootstrapped DQN”, Osband et al., 2016

- “Evolution strategies as a scalable alternative to reinforcement learning”, Salimans et al., 2017

- “State-dependent exploration for policy gradient methods”, Rückstieß et al., 2008

- And a lot of other papers on the general topic of exploration in RL
Conclusion

- Conceptually simple and designed as a drop-in replacement for action space noise (or as an addition)

- Often leads to better performance due to better exploration

- Especially helps when exploration is really important (i.e. sparse rewards)

- Seems to escape local optima (e.g. HalfCheetah)

- Works for off- and on-policy algorithms for discrete and continuous action spaces
Future Directions

- More sophisticated adaptation mechanisms

- Learned perturbations

- Combinations of action and parameter space noise
  - Both simultaneously
  - Switching between action and parameter noise

- Combination with Bayesian Neural Networks
  - Noise proportional to parameter uncertainty?
Collaborators

- Rein Houthooft  
- Prafulla Dhariwal  
- Szymon Sidor  
- Richard Y. Chen  
- Xi Chen  
- Tamim Asfour  
- Pieter Abbeel  
- Marcin Andrychowicz  

OpenAI

UC Berkeley / Embodied Intelligence

KIT
Future Reading


- “Parameter Space Noise for Exploration“, Plappert et al., International Conference on Learning Representations, Vancouver, 2018
Thank you for your attention!

Learn more about OpenAI:

We’re also hiring!
https://openai.com/jobs/