Faster Reinforcement Learning via Transfer

John Schulman

2018.09.06
# Overview

## Policy Gradients

- Success stories
- Limitations

## Meta Reinforcement Learning

## Gym Retro
Terminology

Reinforcement Learning (RL): maximize cumulative reward using trial and error.

Deep RL: using neural nets to represent functions in RL algorithm

Meta-Learning: learn how to learn. Master a task that itself involves learning
RL Terminology

Observation, Action, Reward

Trajectory: a sequence of observation, action, reward

Return: sum of rewards received by agent
RL Terminology (II)

Policy: function that chooses action, based on recent observations
usually stochastic

Policy Gradient Method: a class of RL algorithms that optimize a policy
Policy Gradients

Pseudocode

**Initialize policy**

**Loop:**

- Collect trajectories
- Estimate which actions were good and which were bad
- Increase probability of good actions via gradient update

**Why does it work?** Gradient ascent on expected return of policy
Policy Gradients—History

Old idea (Williams ‘92)

Lots of recent variants with deep learning

E.g., Proximal Policy Optimization (PPO)
Policy Gradients—Success Stories

**AlphaGo**

Lee Sedol version used policy gradients (REINFORCE) as part of the pipeline that also involved search

depmind.com
Policy Gradients—Success Stories

Dota2

- LSTM policy trained using PPO
- Beat world champions at 1v1 (2017)
- Beat top players at modified 5v5 (2018)
- Approaching top level at normal 5v5 (2018)
Policy Gradients—Success Stories

Robotic manipulation

LSTM policy trained using PPO

(this is actually meta-RL)

OpenAI, 2018
RL Requires a Lot of Training

<table>
<thead>
<tr>
<th></th>
<th>Chess</th>
<th>Shogi</th>
<th>Go</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mini-batches</strong></td>
<td>700k</td>
<td>700k</td>
<td>700k</td>
</tr>
<tr>
<td><strong>Training Time</strong></td>
<td>9h</td>
<td>12h</td>
<td>34h</td>
</tr>
<tr>
<td><strong>Training Games</strong></td>
<td>44 million</td>
<td>24 million</td>
<td>21 million</td>
</tr>
<tr>
<td><strong>Thinking Time</strong></td>
<td>800 sims</td>
<td>800 sims</td>
<td>800 sims</td>
</tr>
<tr>
<td></td>
<td>40 ms</td>
<td>80 ms</td>
<td>200 ms</td>
</tr>
</tbody>
</table>

Table S3: Selected statistics of *AlphaZero* training in Chess, Shogi and Go.

Alpha Zero Paper (Silver et al. 2017)
RL Requires a Lot of Training

<table>
<thead>
<tr>
<th></th>
<th>OPENAI 1v1 BOT</th>
<th>OPENAI FIVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUs</td>
<td>60,000 CPU cores on Azure</td>
<td>128,000 preemptible CPU cores on GCP</td>
</tr>
<tr>
<td>GPUs</td>
<td>256 K80 GPUs on Azure</td>
<td>256 P100 GPUs on GCP</td>
</tr>
<tr>
<td>Experience collected</td>
<td>~300 years per day</td>
<td>~180 years per day (~900 years per day counting each hero separately)</td>
</tr>
</tbody>
</table>

OpenAI Five (https://blog.openai.com/openai-five/)
Prior Knowledge

from life history

and evolutionary history
Meta Reinforcement Learning: Problem Formulation

Single task RL: Maximize reward in a single task, given K episodes of experience

- Train from scratch
- Most recent deep RL work follows this framework, e.g. using Atari games
Meta RL: Maze Navigation

“Classic RL” task: learn to navigate from start to goal as fast as possible in a single maze

“Meta-RL” task: learn to navigate from start to goal as fast as possible $K$ times in a random maze

Agent doesn’t know what maze it’s in—needs to explore

After first episode, it should know location of goal and go straight there

Through outer training loop, agent needs to learn to explore and learn to remember
Meta RL is a Special Case of Normal RL

Define new RL task in terms of old task

First timestep: agent is placed in random task / world

New observation = (old observation, reward, “done” signal)

New-task episode = K old-task episodes

https://arxiv.org/abs/1611.02779 (Duan et al. ’16)
Meta RL is a Special Case of Normal RL

RL$^2$: “Fast Reinforcement Learning via Slow Reinforcement Learning”

Agent uses recurrent network (LSTM / GRU) as policy—internally implements learning algorithm.

- Fast RL: learned “algorithm” performed by LSTM
- Slow RL: algorithm used to train LSTM in outer loop

https://arxiv.org/abs/1611.02779 (Duan et al. ’16)
Meta RL: Maze Navigation

https://arxiv.org/abs/1611.02779 (Duan et al. ‘16)
Meta RL: Robotic Manipulation

Real world: robotic hand

Simulation: randomized physics, robot dimensions, visual appearance

Meta RL: Robotic Manipulation

Simulation training: every episode for LSTM policy occurs in a different randomly sampled world

Policy is trained to quickly master a new world through several seconds of interaction

Trained policy is then applied to the real world

Identifies real-world physics parameters after first few seconds of interaction
Meta RL: Robotic Manipulation

https://blog.openai.com/learning-dexterity/
Meta RL: Limitations of Approaches Described Previously

Infinite data: can randomly sample tasks; assume distribution covers the one you care about

- Doesn’t emphasize generalization—performance given finite training set

All “learning” is performed through RNN state updates: might be a poor inductive bias for what learning algorithm should look like

- All meta-learning results so far use short horizons, 1000s of timesteps max

- RL algorithms (policy gradients, Q-learning) find better solutions after longer training periods
Meta RL: changes to problem formulation

Performance on a task is defined as average return after $K$ episodes of learning

Given a finite set of training tasks and a test set of tasks

Train on training tasks as much as desired

Goal: maximize performance on test set
Gym Retro

Prior RL research: achieve human-level performance at games

New challenge: solve a previously unseen game as fast as a human, given prior experience with similar games
Gym Retro
Gym Retro

Over 1000 games integrated
  annotated with reward signal & episode completion

Use various game systems via emulators

Open source: https://github.com/openai/retro/
Challenge: naive definition of reward sometimes leads to "farming" infinite loop
Sonic Benchmark

Reward: moving to the right

Avoid bad guys & traps, use gadgets
Sonic Benchmark

47 training tasks, 11 test tasks
Sonic Benchmark

![Graph showing comparisons between different algorithms and the human average.](https://arxiv.org/pdf/1804.03720.pdf)
PPO Joint Training + Fine Tuning

Joint training: PPO on mixture of all training levels

- use standard ConvNet policy
- save weights at the end

Fine tuning at test time: initialize network to saved weights, run PPO
PPO Joint Training + Fine Tuning

Retro Contest

Train on your hardware
- Training Levels → Metalearn → Fast Learner

Test on our hardware
- Test Levels → Fast Learner → Mean Reward
Retro Contest

contest.openai.com

April 5 to June 5, 2018

Hired level designers to create 11 custom levels

Also created 5 low-quality custom levels for leaderboard

Registration numbers:

- 923 teams registered, 229 submitted solutions
- average 20 submissions per team
Retro Contest
# Retro Contest

<table>
<thead>
<tr>
<th>RANK</th>
<th>TEAM</th>
<th>SCORE</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>Dharmeraja</td>
<td>4692</td>
</tr>
<tr>
<td>#2</td>
<td>mistake</td>
<td>4446</td>
</tr>
<tr>
<td>#3</td>
<td>nborg</td>
<td>4430</td>
</tr>
<tr>
<td>#4</td>
<td>whatever</td>
<td>4274</td>
</tr>
<tr>
<td>#5</td>
<td>Students of Plato</td>
<td>4269</td>
</tr>
<tr>
<td></td>
<td>Joint PPO baseline</td>
<td>4070</td>
</tr>
<tr>
<td></td>
<td>Joint Rainbow baseline</td>
<td>3843</td>
</tr>
<tr>
<td></td>
<td>Rainbow baseline</td>
<td>3498</td>
</tr>
</tbody>
</table>
Retro Contest

<table>
<thead>
<tr>
<th>RANK</th>
<th>TEAM</th>
<th>SCORE</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>DharmaRaja</td>
<td>4692</td>
</tr>
<tr>
<td>#2</td>
<td>mistake</td>
<td>4446</td>
</tr>
<tr>
<td>#3</td>
<td>nborg</td>
<td>4430</td>
</tr>
<tr>
<td>#4</td>
<td>whatever</td>
<td>4274</td>
</tr>
<tr>
<td>#5</td>
<td>Students of Plato</td>
<td>4269</td>
</tr>
</tbody>
</table>

Joint PPO
Rainbow
Joint PPO
Future Work

Improve performance on larger retro benchmark

- Exploration
- Unsupervised learning
- Hierarchy

Improve RNN-based meta-learning

- Better dealing with long time horizon: memory & credit assignment
- Better architectures

More contests!
Acknowledgements

Gym Retro dataset, results, and contest was joint work with Chris Hesse, Oleg Klimov, Alex Nichol, Vicki Pfau

Also highlighted work from OpenAI Robotics & Dota teams

RL² was with Rocky Duan & Pieter Abbeel
Thanks