Environment Adversarial Reinforcement Learning

John R. Cooper

NASA Langley Research Center

AIAA SciTech, January 2024

Introduction

- Contribution: a method for improving performance of reinforcement learning agents
- Gradually make agent's task more difficult over the course of training – update environment parameters as an adversary to the agent
- Inspired by human learning – start with simplified version of new skill, build up to more difficult versions as proficiency is developed
- Difficulty defined relative to agent – agent's predicted return from environment

Related Work

- Curriculum Learning
  - Update parameters from easy solvability towards desired environment
  - Curriculum design an open problem
- Model-Agnostic Meta-Learning
  - Increase agent generalizability by randomizing task during training
  - Agent then easily fine-tuned for specific tasks
- Deep Reinforcement Learning Self Play
  - An agent can achieve superhuman performance in 2-player games when trained against itself
  - How can this fact be leveraged for real world applications?

Environment Adversarial Reinforcement Learning (EARL)

- Break up reinforcement learning (RL) training into multiple iterations
- After each iteration, collect performance data by running trained agent with randomized environment parameters
- Train performance prediction network – predicted return as a function of environment parameters
- Update environment parameters in direction of negative gradient to decrease predicted performance
- Continue RL training

Loss function for training performance predictor: $L(\phi) = \frac{1}{n_b}\sum_{j=1}^{n_b}\left(R_j - \mathcal{P}_\phi(\theta_j)\right)^2$ (1)

Environment parameter update law: $\mu \leftarrow \mu - \alpha_e \nabla_\theta \mathcal{P}_\phi(\mu)$ (2)

Algorithm

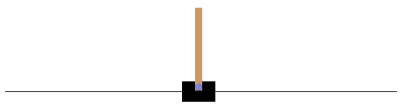
Algorithm 1 EARL algorithm

Require: initial environment parameters μ, environment sampling weight w, RL agent $\mathcal{A}$, number of RL time-steps n_{RL}, predictor learning rate α_p, environment parameter step-size α_e, predictor network $\mathcal{P}$, number of episodes for each predictor training step n_p, number of epochs for each predictor training step n_e, batch size for predictor training n_b, number of EARL iterations N

for $i = 1, \ldots, N$ **do**
- Train $\mathcal{A}$ on the current environment for n_{RL} time-steps.
- Initialize replay buffer $\mathcal{R}$
- **for** $j = 1, \ldots, n_p$ **do**
  - $\sigma \leftarrow \mu / w$
  - $\theta \sim \mathcal{N}(\mu, \sigma^2)$
  - Run $\mathcal{A}$ on the environment with parameters θ. Record return R and parameters θ in $\mathcal{R}$.
- **for** $k = 1, \ldots, n_e$ **do**
  - Randomly sample n_b entries of $\mathcal{R}$
  - Update the parameters of $\mathcal{P}_\phi(\theta)$ using an Adam optimizer with learning rate α_p to minimize the loss function $L(\phi)$ from (1)
- Update μ_θ using an Adam optimizer with learning rate α_e to minimize the loss function $\mathcal{P}(\mu_\theta)$ according to (2)

Experimental Setup

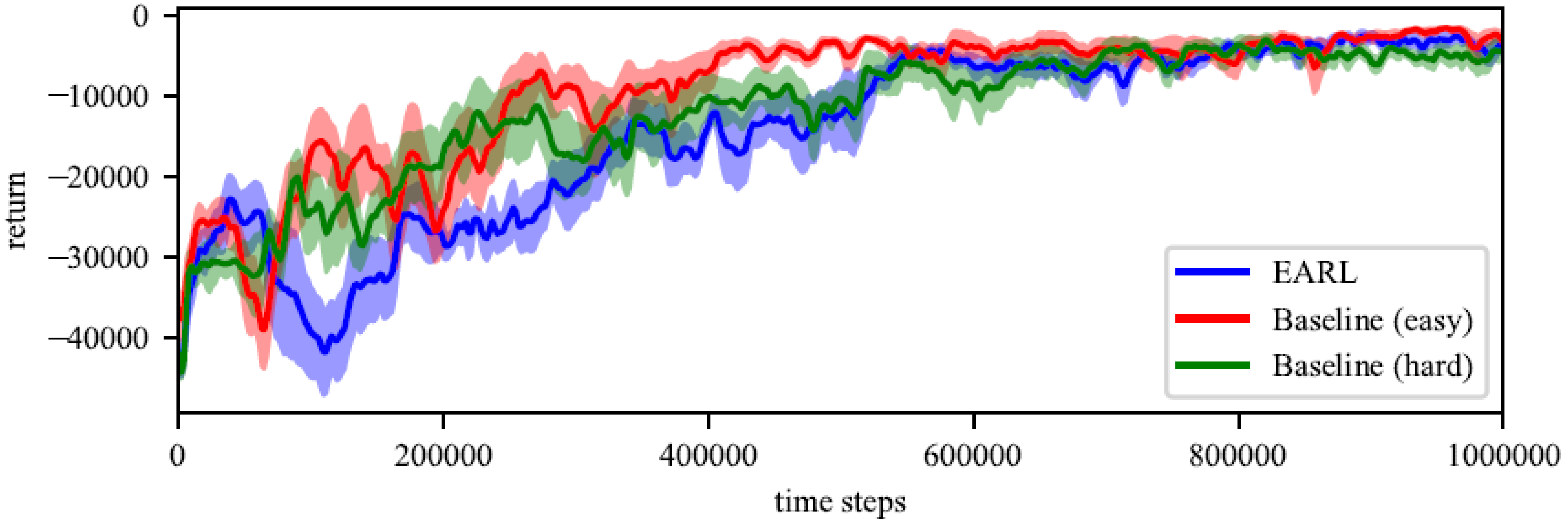
- Tested on continuous CartPole environment from OpenAI gym
  - Environment parameters:
    - Cart mass, initial value = 1
    - Pole mass, initial value = 0.1
    - Pole length, initial value = 0.5
    - Gravitational acceleration, initial value = 9.8



- RL algorithm: TD3 from Stable Baselines library with default parameters
- 3 training configurations:
  - EARL
  - Baseline trained on easy environment
  - Baseline trained on hard environment
- Predictor network:
  - Multilayer perceptron
  - 2 hidden layers, 128 units each, ReLU activation
  - Output layer: linear combination of final hidden layer
- 1e6 environment samples for each training run

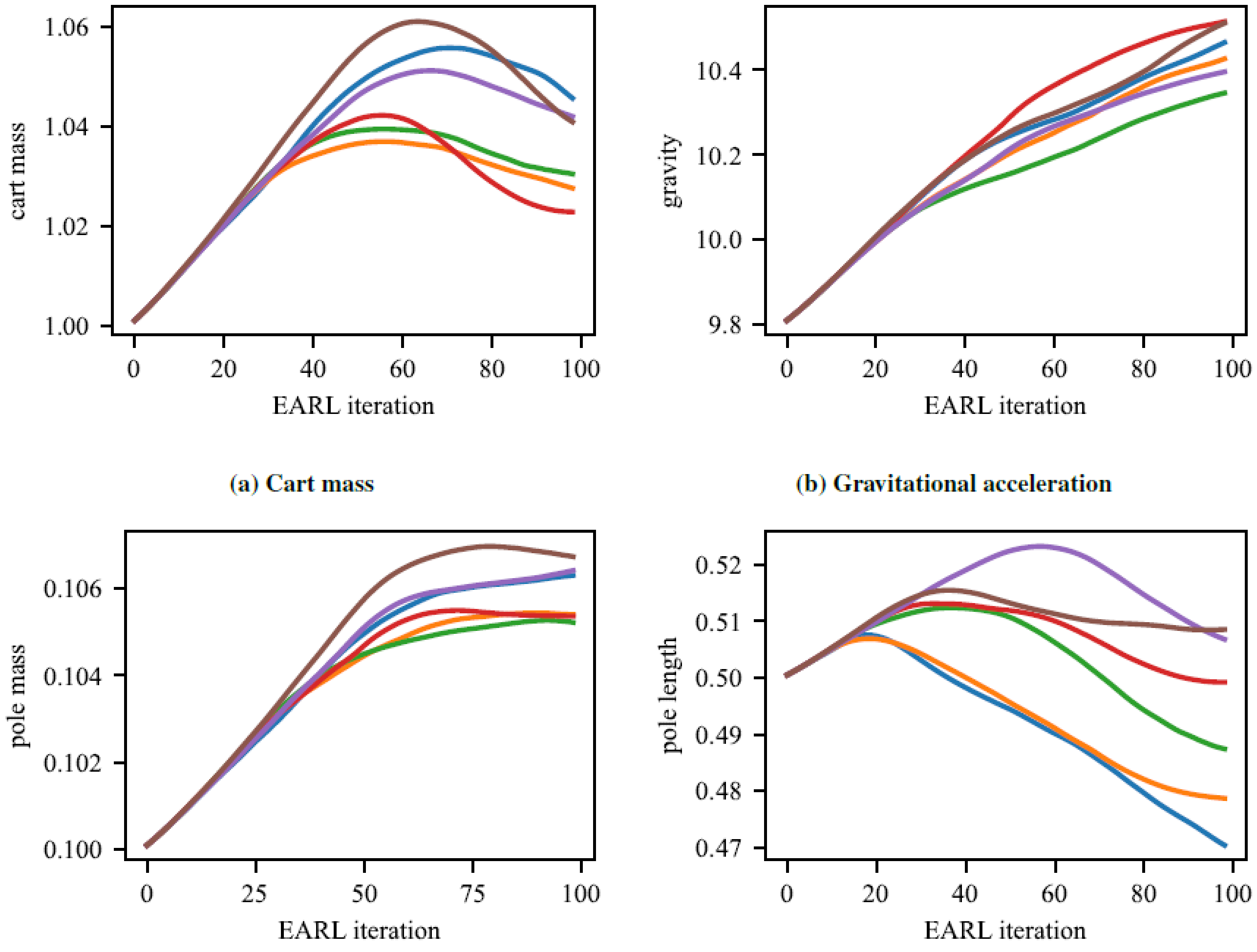
parameter	value
w	5
n_{RL}	1e3
α_p	1e-4
α_e	1e-3
n_p	3e3
n_e	2e3
n_b	512
N	100

Learning Curves



- Solid lines: mean return
- Shaded area: +/- 1 standard deviation
- Baseline trained on easy converges first
- EARL converges with baseline trained on hard
  - Baseline achieves higher return early in training
  - EARL converges to higher return

Parameters Trajectories



(a) Cart mass

(b) Gravitational acceleration

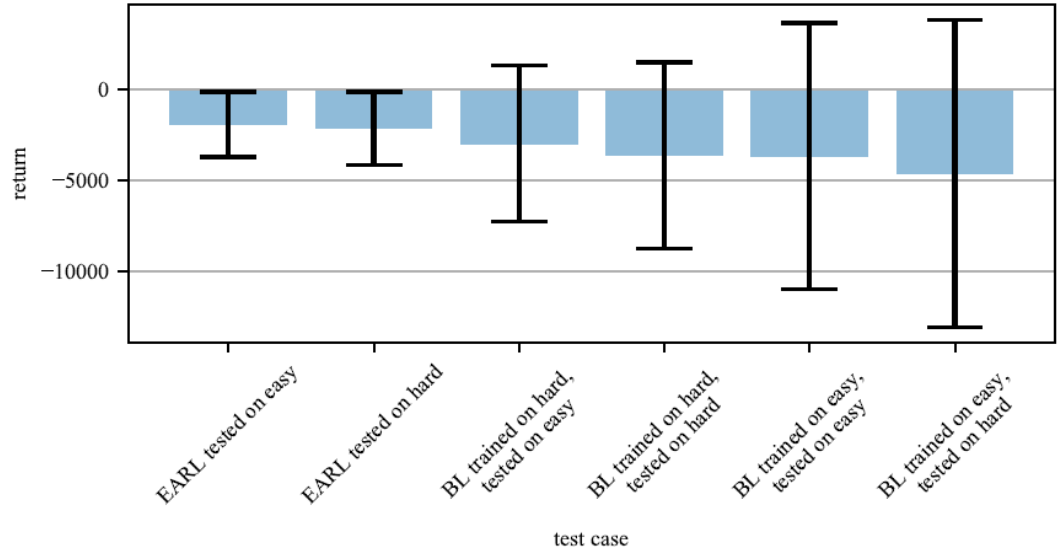
(c) Pole mass

(d) Pole length

Results

- After training, each agent tested for 100 episodes on both easy and hard environments
  - Since each run results in a different hard environment, the baseline agents trained the easy environment are tested on all hard environments from each of the six runs
- EARL outperforms all other test cases

test case	average return	standard deviation
EARL tested on easy	-1938	1793
EARL tested on hard	-2150	2002
BL trained on hard, tested on easy	-3003	4294
BL trained on hard, tested on hard	-3641	5118
BL trained on easy, tested on easy	-3673	7317
BL trained on easy, tested on hard	-4649	8440



Results

- EARL improvements over baseline algorithm

training environment	tested on easy	tested on hard
easy	47%	58%
hard	28%	41%

- P-values for EARL vs. baseline comparison

training environment	tested on easy	tested on hard
easy	2.14e-08	5.78e-15
hard	1.15e-05	4.81e-11

Conclusions

- Experimental results show performance increase compared to standard RL across all variations of training environment when using adversarial training
- Gradient of performance predictor is effective for updating the environment in an adversarial manner
- EARL could be used to learn policies for complicated tasks
- Method presented for increasing difficulty, but decreasing difficulty is an open question
- Future work will test EARL on more environments with other baseline RL algorithms for the inner-loop

Thanks!

John R. Cooper
Autonomous Integrated Systems Research Branch
NASA Langley Research Center
john.r.cooper@nasa.gov