Meta and Multitask RL in Complex Worlds

By: Ronak Malde, Michael Elabd

CS 330

Abstract

Meta-reinforcement learning allows an agent to quickly learn a new task in a simulated environment by learning from other tasks from the same distribution. Current benchmarks for Meta-RL algorithms, however, such as Cheetah speed in Mujoco are too simple. Thus, we create a new Mujoco Meta-RL environment with complex action and state space with varying tasks. This forces our meta-learning or multi-task learning algorithm to adapt their learnings to problems with much higher variation. In our experimentation, we considered state-of-the-art algorithms for meta-learning, multi-task learning, and fine-tuning to compare how they perform in complex environments with high variation. We find that on simple tasks all learning algorithms achieve very similar performance. However, when tested more religiously, we see that the performance gaps widens between different types of learning methods.

Problem and Motivation

Current Meta-reinforcment learning benchmarks for Meta-RL algorithms are from a narrow distribution and do not reflect the complexities of real world tasks. Recent contributions to Meta-RL algorithms all achieve similar performance on the accepted benchmark tests, meaning that current metrics might be insufficient in testing the true ability of the algorithm. In this project, we propose a new Meta-RL environment generator, in which an agent has to learn tasks centered around pushing boxes to a target in a randomly generated map. These environments have higher flexibility for varying task distributions than current benchmarks, and more resemble real-world tasks. Finally, we plan to explore modifications to state of the art model architectures to be better suited the wider distributions and complexity of these environments.

Related Works

Meta-learning Algorithms

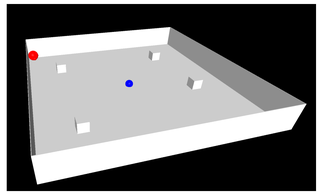
MAML: optimize the model parameters for each task (in addition to general optimization) such that just a few gradient steps on that new task will have a large impact on the end behavior.
RL^2: models an RL problem as multitask by creating an RNN encoder that takes in previous MDP representations of new tasks as inputs.
MQL: In addition to vanilla Q learning, it changes the multi-task objective to maximize reward across training tasks, rather optimize training task.
AdMRL: increase performance on tasks that are have higher variance in distribution or are worst-case tasks.

Meta-learning Benchmark Environments

Meta-World: creates new worlds for robotics tasks from wider distribution to enable better meta-learning model evaluation.

Environment

We propose a new environment to evaluate Meta-RL algorihthms that allows for a wider task distribution and more closely resembles real world tasks. The premise of the environment is that an agent pushes a certain amount and type of box to a certain location in a randomly generated map.



We used the Mujoco WorldGen library to construct our template environment, and then initialized a randomly generated environment for each task. We can model each generated task as an MDP with State Space, Action Space, Transition Function, and Reward Function formulated by 4-tuple (S, A, T_a, R_a).

State Space S - The world generator creates a Mujoco environment with a floor, four walls, an agent, boxes, and a target location. The world can be varied by certain parameters: the size of the floor, the number and type of boxes, and the starting locations of the agent, boxes, and target. Each task is randomly constructed from a range of any of these parameters. For our experimentation, we chose to vary only the starting locations of the agents and boxes, as we found that varying the other parameters required far longer training time to achieve comparable results.

Action Space A - The agent can move in the x-and-y plane and rotate their body's along the vertical z-axis, through Mujoco actuators.

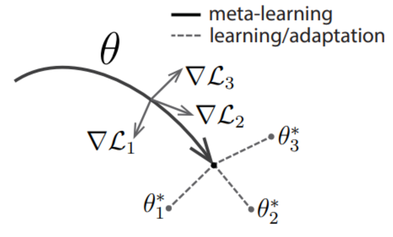
Transition Function T_a - The transitions are described by the Mujoco physics engine. All cubes have very low friction and low mass, so they are easy to push by the agent.

Reward Function R_a - The reward function is parameterized by a negative reward scaled to the distance of all desired objects to the marker. Once the agent pushes an object close to the marker, a high, one-time reward is given, and then that object is removed from future negative reward calculations.

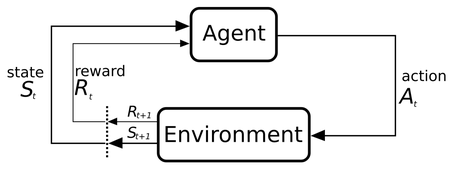
Terminal Conditions - The task is completed when either all desired objects are pushed to the middle, or the maximum episode length is reached.

Algorithms

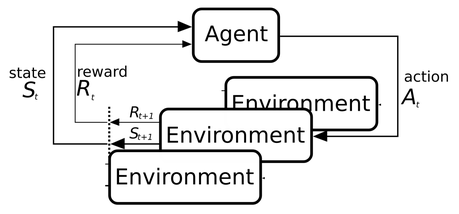
MAML+PPO



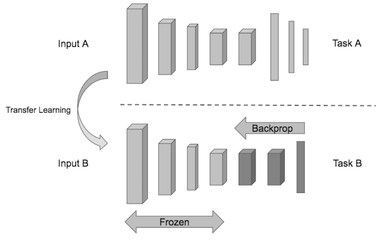
PPO



Multi-Task+PPO



Fine tune+PPO



Results

Test Performance on simple task:

Method	mean Reward	std. Reward	num success	Percent Success
Baseline PPO	-8.32	0.0002	0	0%
Multi-Task PPO	70.7	0.0008	1700	85%
Fine-Tune PPO	**99.88**	**0.0002**	**1998**	**99.9%**
MAMLPPO	16.468	38.23	420	21%
PPO (trained on Env) (Oracle)	99.9	0.882	1998	99.9%

Test Performance on complex task:

Method	mean Reward	std. Reward	num success	Percent Success
Baseline PPO	-8.32	0.0001	0	0%
Multi-Task PPO	-8.28	0.0006	0	0%
Fine-Tune PPO	-8.30	0.004	0	0%
MAMLPPO	**4.21**	**33.0**	**242**	**12.1%**
PPO (trained on Env) (Oracle)	99.9	0.893	1998	99.9%

Analysis

We find that while all models perform well (not tested to convergence) on simple tasks, such as moving the agent from randomized locations to the target, MAMLPPO is the only model able to genuinely learn (again not tested to convergence due to compute constraints) for more complex tasks. Complex tasks, such as having the agent move boxes to the target are not easily learned by either our fine-tuning or multi-task learners.
This shows that on simple tasks, performance differences between those models might be less tangible and that more analysis needs to be done on pushing these models to their limits and seeing how they performance. It is also relevant to compare MAMLPPO to other meta-reinforcement-learning algorithms to see how it compares to the other state of the art algorithms. It is also important to note that while MAMLPPO performed the best on the complex task, it had far higher variance in training than the other algorithms.

Future Work

We plan on modifying existing meta-learning architectures, namely MAMLPPO, in order to improve performance on the complex task distributions. One possible improvement is to leverage domain knowledge and give the model a context encoding for the given tasks, in order to reduce the variance found in the experimentation. Furthermore, we would like to experiment with other meta-learning algorithms such as RL^2 and MQL. We are also considering making the environment more complex by adding variation to the number of boxes present in the world at one time.