Meta-World: A benchmark and evaluation for multi-task and meta RL

Summary

1. Propose a new benchmark with broader distribution of tasks.
2. Show that current methods perform poorly on this new benchmark.
3. Argue that: . prev. benchmark for multi-task (ML) RL has overly disjoint tasks (Atari,
   . prev " " " for Meta RL has too narrow task dist.
4. They found that current methods are unable to learn diverse tasks, much less generalize to entirely new tasks.

Sec 3: the multi-task and meta-RL problem statements

Multi-task RL

1. The goal is to learn a ~~goal-conditioned~~ task-conditioned policy $\pi(a|s, z)$ where z indicates the encoding of the task ID.
2. Aim to maximize expected return across all tasks. $\mathop{E}_{\tau \sim p(\tau)}\left[\mathop{E}_{\pi}\left[\sum_{t=0}^{I} \gamma^t R_t(s_t, a_t)\right]\right]$
3. There is no separate test set of tasks, multi-task RL algorithms are evaluated on their average performance over the training tasks.

Meta RL

Standard notations and definitions

Sec 4 : meta-world

4.1 Parametric & non-parametric variability :

1. The benchmark consists of 50 qualitatively distinct manipulation tasks, where continuous variation of a parameter cannot be used to describe the differences between tasks.
2. However, if there was no variation within each task, they worry that the algorithm will simply memorize the task.
3. They thus introduce parametric variation in obj. and goal positions.
4. Each task requires the robot arm to execute a combination of reaching, pushing and grasping.

4.2 A, S, R :

1. All tasks are performed by a simulated Sawyer robot arm.
2. The action space is 3D end-effector position.
3. For all tasks, the robot either manipulates one obj with a variable goal position, or manipulate 2 objs with a fixed goal pos.
4. The observation is always 9 dimensional, either the 3D cartesian pos of :
   - end-effector, the obj & the goal
   - " " ", 1st obj & 2nd obj

5. The reward fun. is a sum of multiple components : a reaching reward R_{reach}, R_{grasp} and a placing reward R_{place}.
6. They argue that the benefit of this reward function is :
   - well-shaped, so that each task is individually learnable.
   - Different tasks will have the same scales. Otherwise, the policy might be bias towards tasks with high-magnitude rewards.
   - have similar structure across tasks, thereby supporting transfer.

4.3 Evaluation Protocol

1. Their benchmark consists of 5 different settings:

2. Meta-learning 1 (ML1): few-shot adaptation to goal variation within one task.
   - algorithms are evaluated on 3 individual tasks: reaching, pushing and pick & place where the variation is over reaching position or goal object position
   - the goal position is not provided in the observation.

3. Multi-task 10, multi-task 50 (MT10, MT50): learning one multi-task policy that generalizes to 10 and 50 training tasks.
   - the policy is provided with one-hot vector indicating the current task.
   - the positions of objs and goal positions are fixed in all tasks in this evaluation.

4. Meta-learning 10, meta-learning 45 (ML10, ML45): few-shot adaption to new test tasks.
   - with the objective to test generalization to new tasks.
   - hold out 5 tasks and meta-train on either 10 or 45 tasks.
   - obj & goal positions are randomized.
   - intentionally select training tasks with structural similarity to the test tasks.
   - task ID is not provided as input, requiring the meta-RL algo to identify the tasks from experience.

5. They mention that values of reward are not indicative of how successful a policy is.

6. They thus define an interpretable success metric for each task, which is used as the evaluation criterion instead of the reward.

7. Since all of the tasks require involve manipulating one or more objs into a goal configuration, the new success metric is based on the distance between the task-relevant object and its final goal pose, i.e. $\|o - g\|_2 < \varepsilon$.

Sec 5 : experimental results

1. In the multi-task evaluation, they tried :
   - multi-task PPO
   - multi-task TRPO
   - " " " SAC

   } standard model-free algo adapted to solve MT. one-hot task ID is provided as ipt.

   - multi-task multi-head SAC : multi-head SAC with one head per task.
   - task embeddings : parameterizes the learned policies via shared embedding space.
2. In the meta RL evaluation, they tried MAML, RL^2 & PEARL.
3. Figure 7 illustrates perf in ML1 :
   - in reach : all 3 methods have similar perf after 2.5 m steps. pearl asymtotic perf is significantly higher than the other 2.
   - in push : pearl does not learn anything meaningful. RL^2 is much better than MAML.
   - in pick & place : performance of all 3 is pretty low.
4. In MT, they show that multi-head SAC is much better than ~~one~~ ~~loss~~ head SAC.
5. In MT50, the best approach only achieves x 35% success rate.

Questions :

1. Section 4.2 mentions that "the obs space is represented as a 3-tuple of either the 3D Cartesian positions of the end-effector, the obj and the goal, or the 3D Cartesian positions of the end-effector, the 1st obj and the 2nd obj.". Since the semantics of the dimension of the observation space change across tasks, is it exceedingly hard for the policy to perform well on all tasks? Why does this not seem to be an issue?

2. What are the computational resources to run the experiments and how long does each one take in terms of wall-clock time?

3. Figure 6 illustrates the performance of SAC & PPO when trained independently for each task.
   - how many samples does each algorithm take to reach success rate of 1.0 in each task?
   - SAC has low success rate in the bottom 3 tasks. Are these tasks even solvable?

4. In section 4.3, they mention that "they" intentionally select training tasks with structural similarity to the test tasks". What do they mean by structural similarity here and did they have a quantitative measure for it?

5. It seems odd that the multi-head SAC is so much better than SAC with single head. Why is this the case? Do both have the same no. of parameters, or does multi-head SAC have more?

6. In the MT50 experiments, it is surprising that PPO is so much worse than TRPO, which contradicts results from model-free RL. ~~What happen~~ Is there an explanation for this?

7. The results presented in Table 1 and Figure 8 are inconsistent. Consider for example the ML45 case, PEARL is best on meta-test according to table 1 but is ~~worse~~ the worst according to Figure 8. Which result is correct?

8. In conclusion, they mention that "special care needs to be taken to ensure that the task cannot be inferred directly from the image, else ML algo. will memorize the training tasks rather than learning to adapt.". Do they have evidence for this claim?

9. Figure 7 illustrates performance of PEARL, MAML & RL^2 in ML1. Why is Pick Place so ~~can~~ much harder than Reach and Push? Is it because of reward sparsity? Longer task horizon?