Self-Paced Context Evaluation for Contextual Reinforcement Learning

Theresa Eimer [1] **André Biedenkapp** [2] **Frank Hutter** [2 3] **Marius Lindauer** [1]

A. Instance Sampling

AntGoal We uniformly sampled 100 different goals at a distance of at most 750 in both x- and y-direction for both training and test set respectively.

BallCatching The distance and goal coordinates were sampled uniformly for both training and test set. The distance ranged between $0.125 \cdot \pi$ and $0.5 \cdot \pi$, the x-coordinate between 0.6 and 1.1 and the y-coordinate between 0.75 and 4.0. Each instance set contains 100 instances.

PointMass For PointMass, we sampled two different instance sets. First, we used the context bounds of [-4, 4] for the goal position, [0.5, 8] for the goal width and [0, 4] for friction to uniformly sample instances. The goal was to cover the instance space as well as possible. Our second instance set was sampled using the target distribution of SPDRL, which are normal distributions for each context component with means 2.5, 0.5 and 0 respectively as well as standard deviations of 0.004, 0.00375, and 0.002.

B. Experiment Hardware & Hyperparameters

Hardware All experiments with SPACE and the baseline round robin agent were conducted on a slurm CPU cluster (see Table 1). The upper memory limit for these experiments was 1GB per run. The SPDRL experiments were replicated on a slurm GPU cluster consisting of 6 nodes with eight RTX 2080 Ti each. Here maximum memory was 10GB. Slurm scripts for the experiments on PointMass and Ant are provided in the supplementary material. Gridworld experiments are every small and can therefore be found in a jupyter notebook.

Machine no.	CPU model	cores	RAM
1	Xeon E5-2670	16	188 GB
2	Xeon E5-2680 v3	24	251
3-6	Xeon E5-2690 v2	20	125 GB
7-10	Xeon Gold 5120	28	187

Table 1: CPU cluster used for training

CartPole We used a DQN implementation in the top-10 on the environment leaderboard to ensure fair performance for round robin and SPACE agents (Chauhan, 2019). We did not change any hyperparameters from that implementation and used $\kappa = 1$ and $\eta = 2.5\%$ for all experiments.

Other benchmarks For both experiments we used stable baselines version 2.9.0 (Hill et al., 2018) with TRPO for PointMass and PPO2 for all other benchmarks. The policies are encoded by an MLP in both cases, with two layers of 64 units for PPO. For PointMass, we used the default from the SDPRL paper with 21 layers of 64 units each. The discount factor was 0.95. The PPO2 specfic hyperparameters included no gradient clipping, a GAE hyperparameter λ value of 0.99 and an entropy coefficient of 0. For TRPO we used again used the same hyperparameters as SPDRL with a GAE hyperparameter λ of 0.99, a maximum KL-Divergence of 0.004 and value function step size of around 0.24. Any hyperparameters not mentioned were left at the stable baselines' default values. The random seeds were used to seed the environments with the corresponding seeding method.

C. Additional Comparison to SPDRL

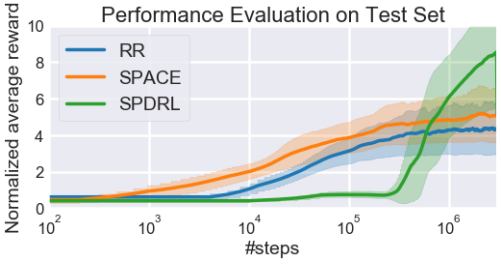

Figure 1: Mean reward per episode on a test set of hard instances with small goals and low friction.

In contrast to SPACE, SPDRL is designed to solve hard instances. To this end, it samples harder and harder instances over time. Therefore, we additionally study how SPACE, round robin (RR) and SPDRL compare on hard instances sampled from the SPDRL target distribution, see Figure 1. Instances in this distribution typically have small goal sizes and low friction, both of which contribute significantly to an increased difficulty.

As in the original paper, SPDRL was allowed to sample as many instances as needed from the distribution, whereas SPACE and RR still only got access to a finite set of 100 instances. In this setting, agents trained either via SPACE

or RR exhibit a similar learning behaviour as on the space covering instance set. For the first $\sim$ 200 000 steps both agents outperform the agent trained via SPDRL; RR anyway focuses on the whole target distribution from the beginnig and SPACE is more free in the way it can select instances with fast training progress. During this time, SPDRL trains the agents on some easy instances, while gradually adapting the instance distribution to focus on ever more difficult tasks. Note that the level of difficulty is not determined solely by the agent being trained via SPDRL, as done in SPACE, but is determined by an expert beforehand.

Once the agent trained via SPDRL is capable of homing in on the difficult instances it outperforms the other agents, as it can exploit its domain knowledge to sample ever more similarly difficult instances, while SPACE and RR are stuck with the limited number of example instances and still try to cover the entire instance space. To achieve this feat, SPDRL requires substantial expert knowledge about which instances to focus on. In essence, the agent trained via SPDRL in the end is only capable of solving a few hard instances with very little variation and will fail to perform well on instances that are not narrowly aligned with the assumed instance distribution.

To be able to know which instances SPDRL should focus on, additional time and effort have to be spent to identify how to quantify *difficulty* for SPDRL. This effort is not reflected in Figure 1 and would move the curve of SPDRL even further to the right.

D. Does the Training Set Size Matter?

To answer this question, we used SPACE to train agents with varying instance set sizes. Figure 2 shows the test performance for differently sized instance sets. Intuitively, one might think that performance should improve with more instances as they cover the instance space better. Indeed, the results for training sets with only 25 and 50 instances are visibly worse than for larger sets. On the remaining instance sets, the agent show very similar performance, however. Note that the performance seems to increase from an instance set size of 100 to 200, but slightly drops again afterwards. There are multiple factors potentially contributing to this effect.

The first is that the agent cannot incorporate any more information from the additional instances, maybe due to limited network capacity or due to the fact that smaller instance sets already cover the space adequately. Furthermore, as we only extend the instance set by one instance at a time, there are more learning steps between curriculum iterations the larger the instance set is, thereby slowing the process down. Especially an agent trained on 1 600 instances will suffer from this.

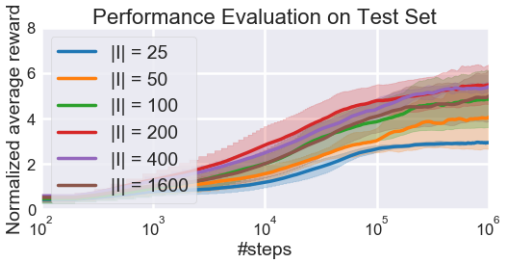

Figure 2: Mean reward per episode on test set for different sized instance sets.

Lastly, SPACE improves upon the RR baseline by ordering training instances and thus smoothing the progression through the instance space. Larger instance sets offer an inherently smoother representation of the instance distribution, therefore diminishing the effect of SPACE. In real-world application settings, we will rarely have access to such large numbers of instances and therefore, it is unlikely that such diminishing performance effects can be observed. This shows that the strength of our method comes to full effect when learning on a sparse representation of our instance space.

E. Comparison of SPACE Curricula

To give some insight into which curricula SPACE found on our benchmark environments, we compare how they behave across random seeds and how they compare to cSPACE curricula. We use Kendall's tau to determine how similar the order in which the instances are added to the training set is.

On PointMass, SPACE finds a curriculum that stay very consistent across all random seeds, showing a correlation of at least 98.9% each to the mean curriculum. The same is true for the cSPACE variation, where the correlation is above 93.8% per seed. Interestingly, these curricula are uncorrelated with a correlation of -0.04. In both we cannot make out a human readable progression in a single context feature (see Figure 3), their curricula do not correspond to any manual instance ordering. As both perform well nonetheless, we can see that learning can be improved by multiple different curricula on this environment.

SPACE and cSPACE produce almost equally unrelated curricula on AntGoal (correlation of 0.07), but while the curriculum stays as consistent across seeds for cSPACE, the same cannot be said for SPACE. Here the correlation to the average curriculum ranges from 14.1% to 52.4%. The correlations between the seed curricula fall into the same range, confirming that the SPACE agent trains on a very different curriculum for each seed. CartPole shows a similar behaviour, the curriculum varying quite a bit between

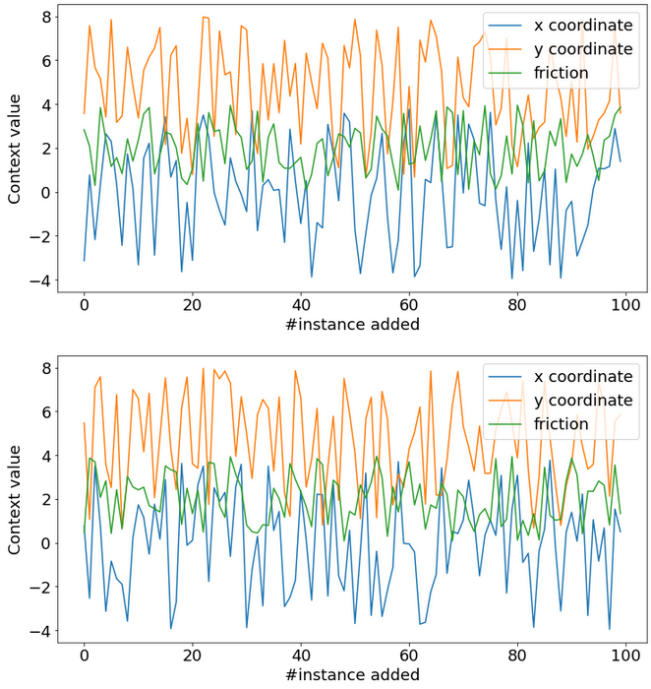

Figure 3: Context feature progression during training for SPACE curriculum (top) and cSPACE curriculum (bottom).

seeds. Therefore we can conclude that SPACE does not find a singular curriculum, but depends on the initialization of environment and model. This is in contrast to cSPACE which stays relatively static due to the context features being constant.

These comparisons suggest that we neither SPACE nor cSPACE finds an optimal curriculum for PointMass, AntGoal or CartPole. It seems, however, that we do not need an optimal curriculum for training at all, as even the 10 very different curricula SPACE finds on AntGoal perform vastly superior to the round robin default. Curriculum Learning should thus focus on reliably and quickly finding good curricula in addition to finding qualitatively better ones.

F. The Influence of Catastrophic Forgetting

When training across multiple instances, forgetting already learnt policies on a subset of instances is a concern (Beaulieu et al., 2020). We analyze how often SPACE and RR agents forget policy components in our PointMass experiments by observing performance development during training. We selected PointMass for this analysis as here policies that are diverse both in how they react to different goal settings and different friction levels are required. That means the policy has to completely change between the extremes of the context which is not required of our other benchmarks where underlying mechanics, e.g. walking for the Ant, stay very similar.

During the training on PointMass, we observed 8 out of 100 instances for which the performance decays after an initial improvement. We would expect the performance to stay at least constant if no forgetting takes place, so the agent likely forgets parts of the policy for these instances in favor of improving on others. The effect is about the same size for round robin agents where we can observe the same for 6 out of 100 instances.

Another reason for attributing this performance decay to forgetting is that on a purely goal-based PointMass variation, the number of instances on which we can observe this effect is slightly smaller (only 4 instances), though not significantly so. All performance decay happens after learning has stagnated on all instances, however. In this easier, purely goal-based setting we could therefore stop training early and would avoid performance decay entirely. This points towards the added complexity of the setting being harder to capture for our agents.

While the effects on both SPACE and RR agents are not very large in our experiments, catastrophic forgetting is therefore certainly important in the field of contextual RL. Future work could on integrate SPACE with existing efforts to reduce this effect like ANML (Beaulieu et al., 2020). A specific aspect of this research that would need to be extended is preventing forgetting in continuous context spaces in addition to the existing successes in discrete ones.

References

Beaulieu, S., Frati, L., Miconi, T., Lehman, J., Stanley, K. O., Clune, J., and Cheney, N. Learning to continually learn. In *ECAI 2020 - 24th European Conference on Artificial Intelligence*, 2020.

Chauhan, K. Cartpole_dqn. https://github.com/kapilnchauhan77/CartPole$_DQN$, 20

Hill, A., Raffin, A., Ernestus, M., Gleave, A., Kanervisto, A., Traore, R., Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., and Wu, Y. Stable baselines. https://github.com/hill-a/stable-baselines, 2018.