Self-Paced Context Evaluation for Contextual Reinforcement Learning

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

Abstract

Reinforcement learning (RL) has made a lot of advances for solving a single problem in a given environment; but learning policies that generalize to unseen variations of a problem remains challenging. To improve sample efficiency for learning on such *instances* of a problem domain, we present *Self-Paced Context Evaluation (*SPACE*)*. Based on self-paced learning, SPACE automatically generates instance curricula online with little computational overhead. To this end, SPACE leverages information contained in state values during training to accelerate and improve training performance as well as generalization capabilities to new instances from the same problem domain. Nevertheless, SPACE is independent of the problem domain at hand and can be applied on top of any RL agent with state-value function approximation. We demonstrate SPACE's ability to speed up learning of different value-based RL agents on two environments, showing better generalization capabilities and up to $10\times$ faster learning compared to naive approaches such as round robin or SPDRL, as the closest state-of-the-art approach.

1. Introduction

Although Reinforcement Learning (RL) has performed impressively in settings like continuous control (Lillicrap et al., 2016), robotics (OpenAI et al., 2019) and game playing (Silver et al., 2016; Vinyals et al., 2019), their applicability is often very limited. RL training on a given task takes a lot of training samples, but the skills acquired do not necessarily transfer to similar tasks as they do for humans. An agent that is able to generalize across variations of a task, however, can be applied more flexibly and has a lower chance of succeeding when presented with unseen inputs. Therefore improving generalization means improving sample efficiency and robustness to unknown situations. We view these as important qualities for real-world RL applications.

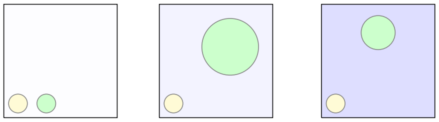

Figure 1: Example instances of the contextual PointMass environment. The agent's yellow starting point, the green goal and floor friction (indicated by shading) are part of the context and vary between instances.

Curriculum learning (Bengio et al., 2009) aims to bridge the gap between agent and human transfer capabilities by training an agent the same way a human would learn: transferring experience from easy to hard variations of the same task. It has been shown that generating such *instances* with increasing difficulty to form a training curriculum can improve training as well generalization performance (Dendorfer et al., 2020; Matiisen et al., 2017; Zhang et al., 2020). As information about instance difficulty is often not readily available, many approaches utilize the agent's progress markers, such as evaluation performance, confidence in its policy or its value function to minimize the need for domain knowledge (Wang et al., 2019; Klink et al., 2020). Because the progression is dictated by the agent's learning progress, this is called Self-Paced Learning (Kumar et al., 2010).

Instances in a curriculum can vary from the core task in different aspects, such as varying goals or movement speeds (see Fig. 1). While only selecting different goals states as instances is common for curriculum learning methods (Dendorfer et al., 2020; Zhang et al., 2020), changing transition dynamics are important considerations regarding the robustness of a policy. A dynamic change in robotics could for example be caused by a broken joint that the agent now has to adapt to. To allow these changes in the transition dynamics, in addition to goal changes in the instances, we consider *contextual* RL instead.

Our contributions are as follows:

1. We propose SPACE, a new self-paced learning algorithm, to automatically generate instance curricula in a

[1] Information Processing Institute (tnt), Leibniz University Hannover, Germany [2] Department of Computer Science, University of Freiburg, Germany [3] Bosch Center for Artificial Intelligence, Renningen, Germany. Correspondence to: Theresa Eimer <eimer@tnt.uni-hannover.de>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

general contextual RL setting, without any knowledge about instance difficulty being required and with access to only a limited set of instances (see Section 4).

2. We show the convergence behavior of SPACE to be at least as good as round robin (see Section 4.2).

3. We demonstrate that SPACE is capable of outperforming a round robin baseline (Speck et al., 2020) as well as similar self-paced methods (see Section 5).

2. Related Work

There are different approaches to increase generalization capability in RL. Their goals and scopes differ substantially, however. MAML (Finn et al., 2017) and related meta-RL methods pre-train an agent such that specializing on one of the training tasks is then very efficient. These take different approaches of aggregating and propagating the gradients in training and are complementary approaches to SPACE.

Domain randomization (DR; Tobin et al., 2017) on the other hand varies the task space. In essence, DR creates new instances of tasks in order to force the agent to adapt to alterations in its observations and policy. Other examples such as POET (Wang et al., 2019) and ADR (OpenAI et al., 2019) sample instances at random but order them by leveraging knowledge about the environment. Without prior knowledge of the target distribution, however, making appropriate changes is hard, resulting in either too little variation to facilitate generalization or deviating so much that the problem becomes too hard to learn. Other approaches utilize human expert knowledge to facilitate generalization performance, such as human-in-the-loop RL (Thomaz & Breazeal, 2006) or imitation learning (Hussein et al., 2017).

Curriculum learning (Bengio et al., 2009) uses expert knowledge to generate an ordering of training instances in such a way that knowledge can be transferred from hard to easy instances. There are different approaches for automatically generating such instance curricula, including learning how to generate training instances (Dendorfer et al., 2020; Such et al., 2020) similar to a teacher (Matiisen et al., 2017; Turchetta et al., 2020) or leveraging self-play as a form of curriculum generation (Sukhbaatar et al., 2018; da Silva et al., 2019). In most of these cases, some knowledge of the instance space is required in order to either define a measure of instance difficulty or how to generate new instances. While instance generation requires only little prior knowledge, a separate agent will need to learn to generate instances of appropriate difficulty, which increases the training overhead significantly.

Value Disagreement based Sampling (VDS; Zhang et al., 2020) on the other hand builds curricula for goal-directed RL. VDS uses the disagreement between different agents trained on the same instances to measure which training instance should be trained on next. Like its building block HER (Andrychowicz et al., 2017), VDS is only compatible with goal-directed off-policy RL.

One approach to order the training instances is explicitly using an agent's performance as an ordering criterion instead, called Self-Paced Learning. This can be done using the agent's value function as a substitute for actual episode evaluations. SPDRL (Klink et al., 2020) uses this idea to generate new instances uniquely suited to the agent's current training progress in order to progress towards specific hard instances. While this eliminates the need for a teacher, researchers instead need to know a priori which instances are considered the hard target instances and where the agent should start training in relation to them.

3. Contextual Reinforcement Learning

Before we describe SPACE, we discuss how we can extend the typical RL formulation to allow for the notion of instances. RL problems are generally modeled as Markov Decision Processes (MDPs), i.e., a 4-tuple $\mathcal{M} := (S, A, T, R)$ consisting of a state space S, a set of actions A, a transition function $T : S \times A \rightarrow S$ and a reward function $R : S \times A \rightarrow \mathbb{R}$. This abstraction however, only allows to model a specific instantiation of a problem and does not allow to deviate from a single *fixed* instance.

An instance $i \in \mathcal{I}$ in a set of instances $\mathcal{I}$ could, e.g., determine a different goal position in a maze problem or different gravity conditions (i.e., moon instead of earth) for a navigation task. Information about the instance at hand is called its context c_i. This context can either directly encode information about the instance, e.g., the true goal coordinates, or the kind of robot that should be controlled.

In order to make use of context in our problem description, we consider contextual MDPs (cMDP; Hallak et al., 2015; Modi et al., 2018; Biedenkapp et al., 2020). A contextual MDP $\mathcal{M}_\mathcal{I}$ is a collection of MDPs $\mathcal{M}_\mathcal{I} := \{\mathcal{M}_i\}_{i\in\mathcal{I}}$ with $\mathcal{M}_i := (S, A, T_i, R_i)$. As the underlying problem stays the same, we assume the possible state and action spaces are consistent across all instances; however, the transition and reward functions are unique to each instance.[1]

An optimal policy π^* for such a cMDP optimizes the expected return over all instances $\mathcal{I}$ with discount factor γ:

$$\pi^* \in \arg\max_{\pi \in \Pi} \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \sum_{t}^{T} \gamma^t R_i(s_t, \pi(s_t)) \qquad (1)$$

As the reward depends on the given instance i, an agent solving a cMDP can leverage the context c_i along with

[1] In goal-directed RL, instances can also only vary the reward function and keep dynamics constant.

the current state $s_t \in S$ in order to differentiate between instances.

4. Self-Paced Context Evaluation

In order to generate a curriculum without any prior knowledge of the target domain, our *Self-Paced Context Evaluation (*SPACE*)* takes advantage of the information contained in an agent's state value predictions. By modelling $V^\pi(s_t, c_i)$, the agent learns to predict the expected reward from state s_t on instance i when following the current policy π. Therefore, we propose $V^\pi(s_0, c_i)$ as an estimate of the total expected reward given a starting state s_0.[2]

Definition 1. *The* performance improvement capacity (PIC) *of an instance is the difference in value estimation between point* t *and* $t-1$*, that is:*

$$d_t(i) = V^\pi_t(s_0, c_i) - V^\pi_{t-1}(s_0, c_i). \tag{2}$$

The intuition is, if the instance evaluation changes by a large amount, the agent has learned a lot about this instance in the last iteration and can potentially learn even more on it. Instances that are too easy or too hard will yield relatively small or no improvements. SPACE prefers instances on which it expects to make most learning progress. As most state-of-the-art RL algorithms use a value-based critic, each instance's PIC is easily computed during training.

Algorithm 1 summarizes the idea of SPACE. After some initialization in Lines 1-3, SPACE performs an update step for the current policy π and the value function V^π based on roll-outs on the current instance set $\mathcal{I}_{curr}$. In principle, any RL algorithm with a value-function estimate can be used, such as Q-learning or policy search based on an actor-critic. In Lines 6-7, SPACE updates the average instance evaluation and the difference to the last iteration; note that this only considers the current set of instances $\mathcal{I}_{curr}$. In Lines 8-9, SPACE first checks whether the value function V^π_t changed ΔV^π_t by a factor $\eta < 1.0$ compared to the value function before the update. If the update led to an insignificant change of the value function, SPACE assumes that the learning sufficiently converged and we can add κ new instances to $\mathcal{I}_{curr}$. Starting in Line 10, SPACE determines which instances in $\mathcal{I}$ should be included in $\mathcal{I}_{curr}$. For each instance, SPACE first computes how much the value function changed, $d_t(i)$. The instances with the highest PIC regarding V^π are chosen as $\mathcal{I}_{curr}$ (Lines 12-13), assuming that it is easy to make progress on these instances right now. Note, we evaluate the influence of the η and κ hyperparameters on the learning behaviour of SPACE in our experiments.

Algorithm 1: SPACE curriculum generation

```
Data: policy $\pi$, value function $V$, Instance set $\mathcal{I}$,
      threshold $\eta$, step size $\kappa$, #iterations $T$
1  $S, t := 0$
2  $V_0 := 0$
3  $\mathcal{I}_{curr} := \{i\}$ with $i$ randomly sampled from $\mathcal{I}$
4  for $t = 1 ... T$ do
5      $\pi, V^\pi_t := update(\pi, V^\pi_{t-1}, \mathcal{I}_{curr})$
6      $V^\pi_t := \frac{1}{|\mathcal{I}_{curr}|} \sum_{i \in \mathcal{I}_{curr}} |V^\pi_t(s_0, c_i)|$
7      if $V^\pi_t \in [(1-\eta)V^\pi_{t-1}, (1+\eta)V^\pi_{t-1}]$ then
           // Increase set size
8          $S := S + \kappa$
       // Choose next instance set
9      forall $i \in \mathcal{I}$ do
10         $d_t(i) := V^\pi_t(s_0, c_i) - V^\pi_{t-1}(s_0, c_i)$
11     $\mathcal{I}_{curr} := S$ instances with highest $d_t(i)$
12     $t := t + 1$
```

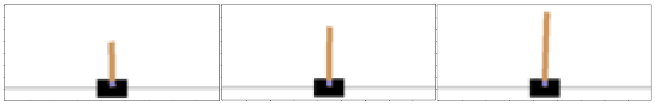

Figure 2: CartPole instances: short (s), medium (m) and long (l) balancing pole.

4.1. Exemplary Application of SPACE

As a motivating example, we consider the CartPole environment (Brockman et al., 2016) with three different pole lengths, see Figure 2. We use a small DQN (hyperparameters given in Appendix B) for this example with the pole length being given as an additional state feature. Although CartPole is generally considered as easy to solve, using poles of different length causes the DQN using a round robin curriculum to be unable to improve over time (see Figure 3). SPACE on the other hand is able to generate curricula that allow the DQN to learn how the cart has to be moved for the different poles and thus improve considerably to a mean performance of around 150 per episode compared to round robin's 25.

In Figure 4 we can see the main difference between the two methods. While the round robin agent trains on all three different variations one episode each, SPACE only chooses to train on the cart with the long pole twice before episode 40. Instead, the focus is on a single instance at a time, using either the short or medium pole and changing not every episode but trains on an instance for at least three consecutive episodes. This shows that the value function can provide guidance as to instance similarity, as we would expect that the short and medium sticks behave in a similar way, as well as difficulty, the long pole being the hardest to

[2] For simplicity's sake, we assume that an environment has a single starting state s_0 and we do not integrate over all possible starting states.

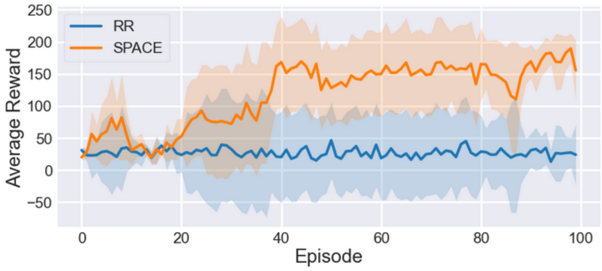

Figure 3: Performance (± std.) comparison of SPACE and default instance ordering on CartPole over 5 seeds each.

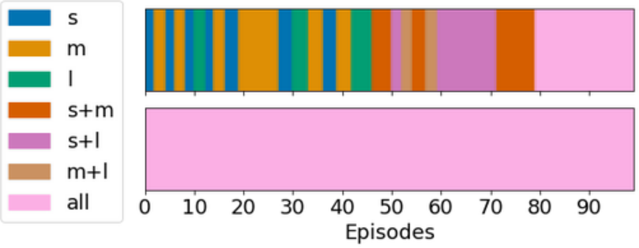

Figure 4: One exemplary run of SPACE (top) and round robin (bottom) curricula on CartPole.

control of the three. While the changes in the value function may not provide a completely stable curriculum, training on one instance for a flexible amount of episodes instead of one episode already has a big impact on overall performance. Furthermore, comparing the curriculum to the performance curve, focusing on only one instance at a time already leads to the agent performing considerably better on all of them. This validates the idea that there are underlying dynamics common to all three pole lengths which are important to learn and then refine according to the instance dynamics.

4.2. Convergence of SPACE

To discuss under which conditions SPACE will converge, we consider two cases.

Theorem 1. *Given a set of instances $\mathcal{I}$ that are sufficiently distinguishable by their context c_i as well as an instance of* SPACE *with $\eta > 0$, $\kappa \geq 1$ and an agent with value function V_t. If the value function estimation V_t^π converges in the limit to some value function V on each instance ($\forall i \in \mathcal{I}.\forall s \in S : \lim_{t\to\infty} V_t^\pi(s, c_i) = V(s, c_i)$) and globally ($\lim_{t\to\infty} V_t^\pi(s) = V(s)$),* SPACE *will eventually include all instances in the curriculum.*

Proof. Since for all $i \in \mathcal{I}$:

$$\lim_{t\to\infty} V_t^\pi(s_0, c_i) = V(s_0, c_i) \tag{3}$$

and therefore it follows that:

$$\lim_{t\to\infty} \Delta V_{t-1}^\pi = ||V_t^\pi(s_0, c_i)| - |V_t^\pi(s_0, c_i)|| \to 0 \tag{4}$$

Thus SPACE is guaranteed to include at least one other instance i' in the new curriculum $\mathcal{I}_{curr}$ at some point t. Now we assume that we are given any $\mathcal{I}' \subseteq \mathcal{I}$ with size $n < |\mathcal{I}|$. As $\forall i \in \mathcal{I}.\forall s \in S : \lim_{t\to\infty} V_t^\pi(s, c_i) = V(s, c_i)$ and Equation 3, convergence of V_t^π on the subset $\mathcal{I}'$ follows:

$$\forall i \in \mathcal{I}' : \lim_{t\to\infty} V^\pi(s_0, c_i) = V(s_0, c_i) \tag{5}$$

Therefore, as in the single instance case:

$$\lim_{t\to\infty} \Delta V_t^\pi \to 0 \tag{6}$$

and a new instance is added.

As the curriculum is guaranteed to be extended for any instance set of size $n = 1$ and $n \leq |\mathcal{I}|$, SPACE will eventually construct a curriculum using the whole instance set.

□

Corollary 1. *If* SPACE *covers all instances at some time point, it will be only slower than round robin by a constant factor in the worst case.*

Proof. Assume $\kappa = 1$ and that the learning agent requires $\mathcal{O}(K)$ steps to converge on a single instance.

If the agent is not able to transfer any of its gained knowledge between any of the tasks, SPACE will require to train an agent $\mathcal{O}(k)$ steps before growing the curriculum, where $k \leq K$, depending on η. SPACE will thus require $\mathcal{O}(|\mathcal{I}| \cdot k)$ steps to include all $|\mathcal{I}|$ instances in the curriculum. At this point, SPACE behaves as a round robin schedule does, i.e., iterating over each instance while training the agent. Therefore, even if the construction of a meaningful curriculum should have failed, SPACE can recover by falling back to a round robin scheme after $\mathcal{O}(|\mathcal{I}| \cdot k)$ steps. □

Corollary 2. *If the value function estimation converges to the* true *value function V^*,* SPACE *will also converge to the optimal policy.* □

Assume the worst case in which the value function estimate does not converge, but either oscillates or even diverges. This could happen if SPACE jumps between two disjoint instance sets $\mathcal{I}_1$ and $\mathcal{I}_2$ and the progress on $\mathcal{I}_1$ is lost by switching to $\mathcal{I}_2$ and vice versa.[3] Whenever we detect that learning is not progressing further and convergence is not achieved (i.e., $\Delta V_t^\pi \neq 0$ and $\mathcal{I}_{curr} \neq \mathcal{I}$), SPACE could simply increase η. As this hyperparameter controls how strict the convergence criterion is, increasing the value will allow for new instances to be added to the training set even though the original convergence criterion has not been met. The least value to which η should be set to guarantee an

[3] Note: Though theoretically possible, we have never observed this problem in practice.

increase of instances is $\frac{\Delta V_t^\pi + \epsilon}{V_{t-1}^\pi}$ for any $\epsilon > 0$ to eventually train on all instances.

Theorem 2. *If the value function estimate is not guaranteed to converge (e.g., in deep reinforcement learning),* SPACE *can still recover a round robin scheme by increasing the threshold η if needed.*

Proof. If at any point t, $\Delta V_t^\pi \neq 0$ and $\mathcal{I}_{curr} \neq \mathcal{I}$, we apply the method described above and set $\eta = \frac{\Delta V_t^\pi + \epsilon}{V_{t-1}^\pi}$. Then the condition to increase the instance set size is $\Delta V_t^\pi < \eta \cdot V_{t-1}^\pi \rightarrow \Delta V_t^\pi < \frac{\Delta V_t^\pi + \epsilon}{V_{t-1}^\pi} \cdot V_{t-1}^\pi \rightarrow \Delta V_t^\pi < \Delta V_t^\pi + \epsilon$. Thus the instance set size is guaranteed to be increased. As this is true for any point in training, SPACE can still consider all instances at some point t^* and thus perform as well as round robin from t^* onward. □

5. Experiments

In this section we empirically evaluate SPACE on two different environments. The code for all experiments is available at https://github.com/automl/SPaCE. We first describe the experimental setup before comparing SPACE against a round robin (RR) training scheme and SPDRL (Klink et al., 2020) as a state-of-the-art self-paced RL baseline. Finally we evaluate the influence of SPACE's own hyperparameters and limitations.

5.1. Setup

We evaluated SPACE in settings that readily allow for context information to encode different instances, namely the *Ant* locomotion environment (Coumans & Bai, 2020), the gym-maze environment (Chan, 2019) and the *BallCatching* and contextual *PointMass* environments as used by Klink et al. (2020).

The task in Ant is to control a four legged ant robot towards a goal on a flat 2D surface as quick as possible. The context is given by the x- and y-coordinates of the goal. Goals that are close to the starting position are easier to reach and thus we expect them to be easier to learn and their policies to transfer to more difficult instances. Additionally, the context indicates if no or up to one of the four legs of the ant robot is immobilized, similar to (Seo et al., 2020). We uniformly sampled 200 instances which we split in equal sized, disjoint training and test sets (see Appendix A). The context of the maze environment (Chan, 2019), in which the task is to find the goal state, is given as the flattened 5x5 layout of the current instance. 100 training and test instances each were sampled using the given maze generator. The agent's goal in BallCatching is to direct a robot to catch a ball. The ball's distance from the robot as well as it goal position are given as context information. Training and test sets were each 100 instances large and uniformly sampled between our context bounds. In the PointMass environment (see Figure 1), an agent maneuvers a point mass through a goal in a two-dimensional space. The goal position, the width as well as the friction coefficient of the ground are given as context. We sampled 100 instances for training and testing, each for two different distributions. The first distribution is chosen to cover the space of possible instances, whereas the second distribution follows that of Klink et al. (2020) and focuses on an area around a particularly difficult instance (see Appendix A).

To be consistent and fair with respect to prior work, we trained a PPO agent (Schulman et al., 2017) for Ant and a TRPO agent for PointMass (Schulman et al., 2015) and base our curriculum generation on their value-based actors. For easier readability, all plots are smoothed over 10 steps. In order to monitor generalization progress over time, we evaluated the agent on all instances in the training and test set after each complete run through the training set. As the results on training and test sets were very similar, we only report the test performance. In all experiments we evaluated our agents over 10 random seeds. For hardware specifications and hyperparameters, please see Appendix B.

5.2. Baselines

In our experiments, we use three different baselines to compare SPACE's performance to.

Round Robin (RR) To be sure SPACE outperforms instances without an intentional ordering, we compare against round robin as a common default instance ordering. This means that the training instances are ordered in an arbitrary way and we simply iterate over them, playing one episode per instance. As the instance sets we use are generated randomly, this ordering is chosen at random as well.

SPDRL SPDRL (Klink et al., 2020) is a state-of-the-art self-paced learning method for contextual RL. This is notable as most curriculum learning methods are explicitly designed for goal-directed RL, which makes them unsuitable in our setting. Counter to SPACE and RR, SPDRL makes use of an instance distribution to continually sample new instances of a specific difficulty level. SPDRL uses this ability to generate new instances to focus on particularly difficult instances, while largely ignoring the remaining instance space. To this end, SPDRL requires additional domain knowledge, besides the context information, to determine which instances SPDRL should focus on. Therefore we provide SPDRL with the distribution of our training and test set to focus the learning on its center.

cSPACE With SPACE we opted for taking an agent's knowledge about the expected reward, i.e. value function, into account to determine the similarity and difficulty of

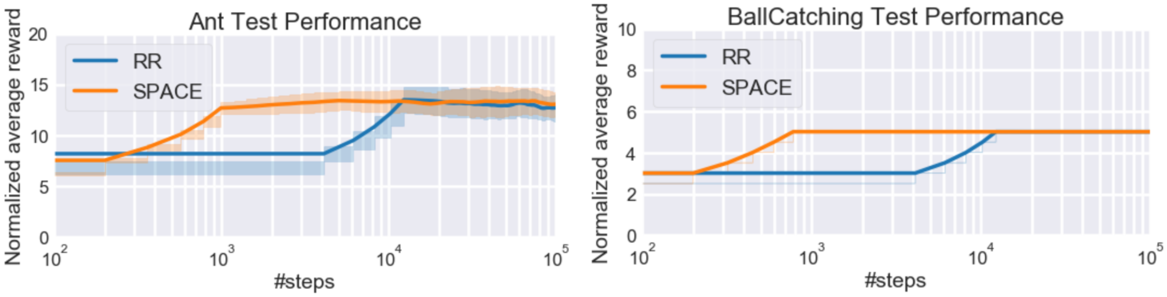

Figure 5: Mean reward (± standard deviation) per episode over 10 runs on Ant (left) and BallCatching (right).

instances. However, as instances can be represented by their context, their similarity could also be quantified directly through their similarity in the context space. This form of similarity quantification is common in fields making use of techniques such as algorithm selection (Rice, 1976) and meta-learning (Brazdil et al., 2008). Such curricula order instances according to their context similarities. A successful application of this approach can be seen in Reverse Curriculum Generation for Reinforcement Learning (Florensa et al., 2017) where robot arm starting positions were ordered into a curriculum according to their similarity. In other words, instead of using an agent's performance evaluations as a basis for the curriculum generation, instances with contexts that are closest to the current curriculum context are added. SPACE's instance ordering criterion can easily be changed to compare context space distance instead of evaluations, yielding a variation we call *context* SPACE (cSPACE). More precisely, we replaced d as our instance selection criterion (see Algorithm 1 Line 10) with the Euclidean distance to the current instance set $\mathcal{I}_{curr}$. In such cases, cSPACE can suffer from the same problems as unsupervised learning. A priori it is not clear how to scale and weight the different context features without having any signal how the features will affect the difficulty of instances and how good the resulting curriculum will be. In contrast to SPACE, we deem this a potential challenge in applying cSPACE. For this reason, we recommend SPACE as the default approach whenever state evaluations are available.

5.3. Does the Instance Order Matter?

We first compare SPACE to a baseline round robin (RR) agent on the Ant and BallCatching environments, to determine if SPACE can find a curriculum that outperforms a random ordering. In Figure 5, both agents reach the same final performance in each environment, but the agent trained via SPACE learns considerably faster. It only requires 10^3 steps to reach a reward of around 11 in Ant whereas RR requires roughly $10\times$ as many steps to train an agent to reach the same reward. The results for BallCatching are similar, with SPACE again being faster to reach the final performance a factor of at least 10. We further compare both methods on the PointMass environment when training on an instance set that was uniformly sampled from the space of possible instances (see Figure 6).

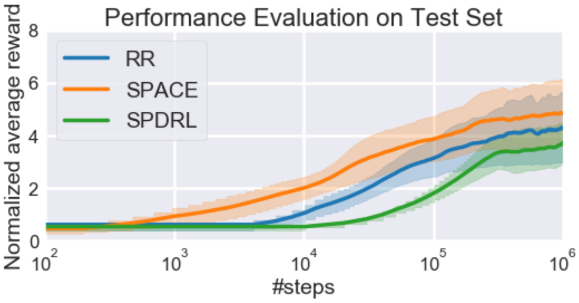

Figure 6: Mean reward per episode on a test set of uniformly sampled instances for PointMass.

Here, the agent trained with SPACE is only roughly twice as fast, but it substantially outperforms round robin in terms of final performance. As the RR baseline does not care about the order in which instances are presented to the agent, we conclude that a more structured learning approach is needed. From SPACE's performance we can conclude that a curriculum, learned in a self-paced fashion can help improve both training performance and generalization. The experiments in the following sections further confirm this finding.

5.4. Comparing SPACE and SPDRL

We further compared SPACE to SPDRL (Klink et al., 2020) on the PointMass environment in order to demonstrate the difference between SPACE and an other self-paced learning method. We used the same implementation and hyperparameters as in (Klink et al., 2020) for SPDRL. The test performance of the agents can be seen in Figure 6.

As SPDRL was developed to train an agent to solve specific hard instances in the PointMass environment, it clearly falls short when it comes to covering the whole instance

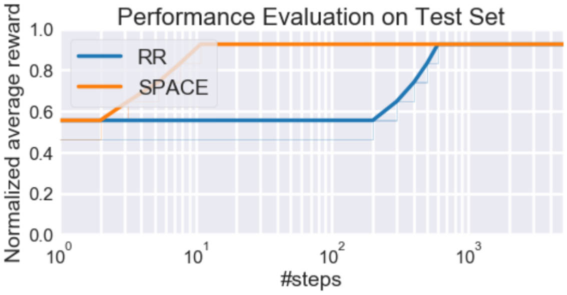

Figure 7: SPACE and RR of a set of mazes over 10 runs (± standard deviation)

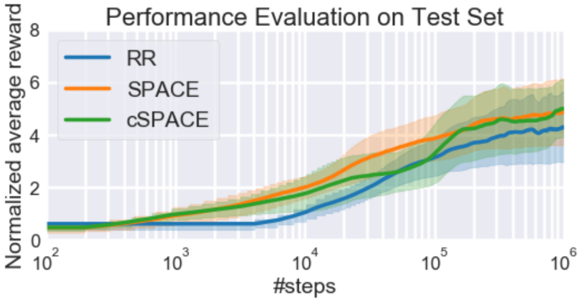

Figure 8: Mean reward on contextual PointMass with additional cSPACE results.

space. The agent trained via SPDRL learns much slower and achieves a worse final performance than an agent trained via RR. Perhaps unsurprisingly, this shows that targeting learning on hard instances does not imply the same agent can achieve good generalization performance on all instances.

5.5. How Well does SPACE Handle Complex Contexts?

While our benchmarks above are common meta-RL problem settings with different complexities, their contexts are given in a rather simple form, i.e., a short context vector directly describes the goal and environment dynamics. The agent can therefore make a direct connection between the changes between instances and the different context descriptions. This may not always be the case with context possibly being given within the observation, e.g., as part of an image.

In order to confirm that such a context description still enables SPACE to select the appropriate next instance, we use a set of 100 5x5 mazes (Chan, 2019) for our agent to generalize over. The observation is the agent's current position while the context is given by the flattened maze layout.

This context is much more complex than the previously used ones by having a structure that has been flattened and its components do not directly correlate to an increase in difficulty. Furthermore, many of the components of the context may not change from instance to instance even though the layout, and therefore the required policy, will.

Figure 7 shows that while the context information for this task is much more complex than previously seen, SPACE still outperforms the round robin agent in a similar way than it does to for Ant and BallCatching. The round robin agent needs several hundred episodes to solve all mazes while SPACE is able to generalize from just 10 episodes. While the context complexity increases in this case, the value function is still able to differentiate between them enough to allow a distinction between different instances. Therefore we expect that the representation of the context is not a major concern for the performance of SPACE.

5.6. Can SPACE Be Applied Without a Value Function?

The PointMass environment has three different context features for which we can easily use the context space distance to construct a curriculum. SPACE and cSPACE perform similarly on PointMass (Figure 8) in terms of learning speed and overall performance, both reaching the same performance at the same speed, with SPACE learning faster on average between 10^4 and 10^5. This makes both SPACE variations a better choice than round robin.

Both cSPACE and SPACE are also consistent in the curricula they find. We measured this by comparing the frequency with which each instance was used in the training set compared to the weighted frequency which gives higher rank to instances chosen at earlier iterations (1 for the instance chosen first down to $\frac{1}{|\mathcal{I}|}$ for the instance chosen last). For cSPACE, both the frequency and weighted frequency stayed the same while for SPACE only the four least used instances differed in order between the two.

We also compared the mean instance distance between curriculum iterations to see which method allows for smoother transitions between tasks. Smooth transitions correlate to a handcrafted curriculum where instances are close together in the context space, making the curriculum easier to learn from a human perspective. SPACE moves the instance set around 4.7% each curriculum iteration while cSPACE moves by around 5.6%. The maximum induced change is 10.1% for SPACE and 13.3% for cSPACE approach.

As we can see from these comparisons, using the information contained in an agent's value function to construct a curriculum is very similar to using the context space distance. It needs to be said, however, that in PointMass the context reflects the environment dynamics in a very direct way, being made up of the x- and y-positions of the goal to reach and the friction coefficient. Therefore we would expect cSPACE to perform very well on such environments. The fact that the default SPACE setting performs similarly indicates that the value function contains the information necessary to

Table 1: Mean reward ± standard deviation for different hyperparameter values on PointMass after 10^6 steps.

	η			
κ	5%	10%	20%	40%
1	5.1 ± 0.7	4.8 ± 1.2	4.7 ± 1.2	5.2 ± 0.7
4	5.5 ± 0.5	5.2 ± 0.7	4.3 ± 1.2	4.4 ± 1.0
16	4.6 ± 1.1	3.7 ± 1.1	5.1 ± 1.2	4.8 ± 1.0
32	4.5 ± 1.1	4.6 ± 1.1	4.7 ± 1.3	5.0 ± 1.2

order instances into a curriculum of similar quality. As not all environments may have such simple changes between instances, we expect that cSPACE has limitations on those kinds of environments while we can expect the value-based SPACE variation to continue constructing high quality curricula even in that case.

5.7. How Robust is SPACE wrt its Hyperparameters?

SPACE comes with two hyperparameters, the performance threshold for curriculum interactions η and the instance increment κ. These hyperparameters interact with each other to make SPACE comparatively stable across different hyperparameter values (as seen in Figure 1).

By varying η for a given value of κ, we alter the degree of stability the agent's value estimates have to reach between training episodes. Depending on the problem at hand, the value estimates may never be perfectly stable, therefore a very low value for η may prevent the training set from expanding. On the other hand, a very large value will move SPACE closer to round robin. Thus we view η as the more important hyperparameter of the two.

Our study shows very little performance differences for different values of κ and η. In part, this is because PointMass instances are not too difficult in the mean, therefore adding many at once does not heavily disturb learning. Larger performance thresholds η are not an issue for this reason. A value of 5% for η seems quite low, but as the instances are relatively easy, the agent can still converge enough very quickly. Different values for κ show similar results here. We expect this hyperparameter to be more important in very diverse settings with large gaps between instances. We can see the effect if we multiply the size of our training set instead of adding instances (see Figure 9). In this case, there is a visible slowdown, supporting that κ has a big influence on training performance.

From these results, we believe that it is reasonable to recommend keeping $\kappa = 1$ for most applications. It can yield more fine-grained curricula which will be important on diverse instance sets and will likely only impact training on very large instance sets. For η, using a low value such as 5% should ensure that the agent will not be overwhelmed with new instances if it takes more than one curriculum iteration to learn from the current training set.

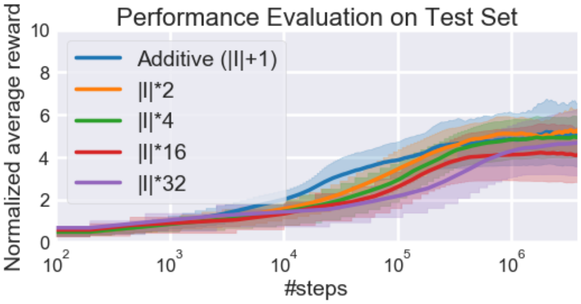

Figure 9: Mean reward per episode on a test set with fast rising instance set size (i.e. varying κ) and fixed $\eta = 10\%$.

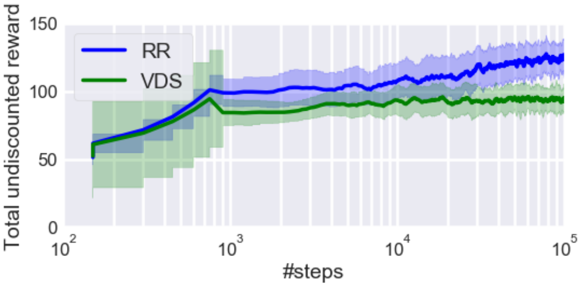

Figure 10: Total undiscounted reward of VDS and its RR baseline on AntGoal.

5.8. Can Goal-Directed RL Achieve Similar Results?

Goal-directed RL and contextual RL are closely related flavours of RL. The main difference between the two is that in goal-directed RL, the context is restricted to only contain information about a desired goal-location. Thereby the context indicates a (final) state in which the agent should reach. Policies learning with goals as context information thus can learn the value of executing an action in a certain state for reaching the desired goal. Crucially however, in this setting transition dynamics are assumed to never change for different contexts.

Contextual RL subsumes goal-directed RL by also allowing the environment dynamics for the same goal to vary. Both environments we evaluate on exemplify this. In contextual PointMass, we specify both a goal and the friction coefficient to describe an instance. Even if the goal is kept the same, however, the friction is a deciding factor in the amount of force that is necessary to reach the goal correctly. So while the force direction is the same for instances with the same goal, the required amount of force depends on the friction and therefore the agent needs both information to learn to generalize across different instances.

On AntGoal, we can see how this looks in practice. VDS (Zhang et al., 2020) is a recent state-of-the-art method for goal-directed curriculum construction based on HER (Andrychowicz et al., 2017). As a result, it can take only the ant's goal into consideration when selecting the next instance and crucially misses that in different instances the ant has different defects in some of its joints. As a result, the method conflates all instances with the same goals and fails to actually learn how to act on any of them.

We used the implementation and baseline of Zhang et al. (2020) to demonstrate that while goal-directed curriculum generation approaches seem similar to SPaCE, our problem setting is out of scope for them (see Figure 10). Their RR baseline has a different learning curve as ours as the algorithm used is different, but it clearly is able to improve over time. As VDS uses goals to describe the necessary behaviour, it cannot do the same. Therefore, curriculum learning methods for contextual RL and goal-directed RL have different scopes and cannot be compared fairly in the contextual RL setting.

6. Limitations

Even though SPACE performed very well on the benchmarks used in this paper, there are several limitations of SPACE to be considered. The first one is that the problem and the instance set both need to support curriculum learning to some degree. For the problem itself this means that the policy to solve it is influenced by context to a large degree, but that there is an underlying structure that can be exploited using a curriculum. The instance set then needs to be large enough to actually give SPACE the opportunity to do so. In settings with too little or very large amounts of instances, SPACE becomes less efficient (see Appendix D).

Furthermore, if the instance set is very homogeneous, similar to the specific instance SPDRL uses on PointMass (see Appendix C), using different instances for training might not make a difference. Conversely, if the instance set is heterogeneous, preliminary experiments showed that SPACE requires a larger amount of instances to speed up the learning. Thus not every problem is suited for curriculum learning.

Lastly, SPACE is constructed to work for discrete instance spaces only, where the instance ordering is essential for learning efficiency. We stress the fact that SPACE is designed for use cases with only a few instance examples. In settings with instance generators or a lot of domain knowledge available, it is likely better to exploit them which SPACE is not designed for.

7. Conclusion

Self-Paced Context Evaluation (SPACE) provides an adaptive curriculum learning method for problem settings constrained to a fixed set of training instances. Thereby we facilitate generalization in practical applications of RL. We demonstrated that the order of instances on which agents learn their behaviour policies indeed is important and can produce a better learning efficiency. In addition, SPACE outperformed a simple round robin baseline as well as more specialized curriculum learning methods requiring access to unlimited instance generators to perform well. Finally we evaluated the influence of SPACE's own hyperparameters and showed that they are robust on the chosen environments.

Future research could address how to derive performance expectations for practical applications of RL with a limited amount of instances with respect to the amount of information available. Furthermore, we might be able to use value estimation to further improve training efficiency for example by clustering instances of similar difficulty and limiting the amount of training on very easy ones to a minimum. Another important factor for contextual RL in general is catastrophic forgetting (see Appendix **??**), which is not yet sufficiently understood, especially in the continuous context spaces we applied SPACE to.

8. Acknowledgements

Theresa Eimer and Marius Lindauer acknowledge funding by the German Research Foundation (DFG) under LI 2801/4-1. All authors acknowledge funding by the Robert Bosch GmbH.

References

Andrychowicz, M., Crow, D., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., and Zaremba, W. Hindsight experience replay. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Proceedings of the 31st International Conference on Advances in Neural Information Processing Systems (NeurIPS'17)*, pp. 5048–5058, 2017.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. In Bottou, L. and Littman, M. (eds.), *Proceedings of the 26th International Conference on Machine Learning (ICML'09)*, pp. 41–48. Omnipress, 2009.

Biedenkapp, A., Bozkurt, H. F., Eimer, T., Hutter, F., and Lindauer, M. Dynamic Algorithm Configuration: Foundation of a New Meta-Algorithmic Framework. In Lang, J., Giacomo, G. D., Dilkina, B., and Milano, M. (eds.), *Proceedings of the Twenty-fourth European Conference* *on Artificial Intelligence (ECAI'20)*, pp. 427–434, June 2020.

Brazdil, P., Giraud-Carrier, C., Soares, C., and Vilalta, R. *Metalearning: Applications to Data Mining*. Springer Publishing Company, Incorporated, 1 edition, 2008.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym, 2016.

Chan, M. gym-maze. `https://github.com/MattChanTK/gym-maze`, 2019.

Coumans, E. and Bai, Y. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2020.

da Silva, F. L., Costa, A. H. R., and Stone, P. Building self-play curricula online by playing with expert agents in adversarial games. In *8th Brazilian Conference on Intelligent Systems, BRACIS '19*, pp. 479–484, 2019.

Dendorfer, P., Osep, A., and Leal-Taixé, L. Goal-GAN: Multimodal trajectory prediction based on goal position estimation. In *Proceedings of the 15th Asian Conference on Computer Vision (ACCV'20)*, 2020.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML '17)*, volume 70, pp. 1126–1135, 2017.

Florensa, C., Held, D., Wulfmeier, M., Zhang, M., and Abbeel, P. Reverse curriculum generation for reinforcement learning. In *Proceedings of the 1st Conference on Robot Learning (CoRL'17)*, volume 78, pp. 482–495, 2017.

Hallak, A., Castro, D. D., and Mannor, S. Contextual markov decision processes. *arXiv:1502.02259 [stat.ML]*, 2015.

Hussein, A., Gaber, M. M., Elyan, E., and Jayne, C. Imitation learning: A survey of learning methods. *ACM Comput. Surv.*, 50(2):21:1–21:35, 2017.

Klink, P., D'Eramo, C., Peters, J., and Pajarinen, J. Self-paced deep reinforcement learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS'20)*, 2020.

Kumar, M. P., Packer, B., and Koller, D. Self-paced learning for latent variable models. In Lafferty, J., Williams, C., Shawe-Taylor, J., Zemel, R., and Culotta, A. (eds.), *Proceedings of the 24th International Conference on Advances in Neural Information Processing Systems (NeurIPS'10)*, pp. 1189–1197, 2010.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In *Proceedings of the International Conference on Learning Representations (ICLR'16)*, 2016. Published online: `iclr.cc`.

Matiisen, T., Oliver, A., Cohen, T., and Schulman, J. Teacher-student curriculum learning. *CoRR*, abs/1707.00183, 2017.

Modi, A., Jiang, N., Singh, S. P., and Tewari, A. Markov decision processes with continuous side information. In *Algorithmic Learning Theory (ALT'18)*, volume 83, pp. 597–618, 2018.

OpenAI, Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R., Schneider, J., Tezak, N., Tworek, J., Welinder, P., Weng, L., Yuan, Q., Zaremba, W., and Zhang, L. Solving rubik's cube with a robot hand. *arXiv:1910.07113 [cs.LG]*, 2019.

Rice, J. The algorithm selection problem. *Advances in Computers*, 15:65–118, 1976.

Schulman, J., Levine, S., Moritz, P., Jordan, M. I., and Abbeel, P. Trust region policy optimization. In Bach, F. and Blei, D. (eds.), *Proceedings of the 32nd International Conference on Machine Learning (ICML'15)*, volume 37, pp. 1889–1897. Omnipress, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv:1707.06347 [cs.LG]*, 2017.

Seo, Y., Lee, K., Gilaberte, I. C., Kurutach, T., Shin, J., and Abbeel, P. Trajectory-wise multiple choice learning for dynamics generalization in reinforcement learning. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS'20)*, 2020.

Silver, D., Huang, A., Maddison, C., Guez, A., Sifre, L., Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Speck, D., Biedenkapp, A., Hutter, F., Mattmüller, R., and Lindauer, M. Learning heuristic selection with dynamic algorithm configuration. In *Workshop on Bridging the Gap Between AI Planning and Reinforcement Learning (PRL@ICAPS'20)*, October 2020.

Such, F. P., Rawal, A., Lehman, J., Stanley, K. O., and Clune, J. Generative teaching networks: Accelerating neural architecture search by learning to generate synthetic training data. In III, H. D. and Singh, A. (eds.), *Proceedings of the 36th International Conference on Machine Learning (ICML'20)*, volume 98. Proceedings of Machine Learning Research, 2020.

Sukhbaatar, S., Lin, Z., Kostrikov, I., Synnaeve, G., Szlam, A., and Fergus, R. Intrinsic motivation and automatic curricula via asymmetric self-play. In *6th International Conference on Learning Representations (ICLR '18)*, 2018.

Thomaz, A. L. and Breazeal, C. Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. In *Proceedings of the Twenty-first National Conference on Artificial Intelligence (AAAI'06)*, pp. 1000–1006, 2006.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '17)*, pp. 23–30, 2017.

Turchetta, M., Kolobov, A., Shah, S., Krause, A., and Agarwal, A. Safe reinforcement learning via curriculum induction. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, (NeurIPS'20)*, 2020.

Vinyals, O., Babuschkin, I., Czarnecki, W., Mathieu, M., Dudzik, A., Chung, J., Choi, D., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J., Jaderberg, M., Vezhnevets, A., Leblond, R., Pohlen, T., Dalibard, V., Budden, D., Sulsky, Y., Molloy, J., Paine, T., Gülçehre, Ç., Wang, Z., Pfaff, T., Wu, Y., Ring, R., Yogatama, D., Wünsch, D., McKinney, K., Smith, O., Schaul, T., Lillicrap, T., Kavukcuoglu, K., Hassabis, D., Apps, C., and Silver, D. Grandmaster level in starcraft II using multi-agent reinforcement learning. *Nature*, 575(7782): 350–354, 2019.

Wang, R., Lehman, J., Clune, J., and Stanley, K. O. POET: open-ended coevolution of environments and their optimized solutions. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO'19*, 2019.

Zhang, Y., Abbeel, P., and Pinto, L. Automatic curriculum learning through value disagreement. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS'20)*, 2020.