Generalizing Curricula for Reinforcement Learning

Sanmit Narvekar [1] **Peter Stone** [1]

Abstract

Curriculum learning for reinforcement learning (RL) is an active area of research that seeks to speed up training of RL agents on a target task by first training them through a series of progressively more challenging source tasks. Each task in this sequence builds upon skills learned in previous tasks to gradually develop the repertoire needed to solve the final task. Over the past few years, many automated methods to develop curricula have been developed. However, they all have one key limitation: the curriculum must be regenerated from scratch for each new agent or task encountered. In many cases, this generation process can be very expensive. However, there is structure that can be exploited between tasks and agents, such that knowledge gained developing a curriculum for one task can be reused to speed up creating a curriculum for a new task. In this paper, we present a method to generalize a curriculum learned for one set of tasks to a novel set of unseen tasks.

1. Introduction

Curricula are used throughout human education and development, serving to structure the learning process in everything from cognitive skills such as math and reading to motor skills and sports. These curricula are organized so that each new concept introduced builds upon previously learned skills, facilitating the rapid acquisition of complex skills and behaviors.

Motivated by this observation, recently there have been efforts to apply curricula to improve training speed for reinforcement learning (RL) agents. Over the years, many techniques have been designed to both manually and automatically design curricula for RL agents. However, each of these methods must be run entirely from scratch to design a curriculum for each new agent or each new task. In contrast, curricula designed for humans are used to teach many different students, and can easily be repurposed to teach people how to solve similar tasks.

This paper thus considers the problem of *curriculum generalization*: how can knowledge gained about designing a curriculum for one task be generalized to speed up learning of a curriculum for a similar, but novel unseen task? In other words, how can we transfer or adapt a curriculum learned for one task to a new target task?

Our work builds on a representation of a curriculum as a policy (specifically, a *curriculum policy*) (Narvekar & Stone, 2019), which maps from the state of knowledge of an RL agent to the task it should learn next. Our primary contribution is to show that by combining curriculum policies with universal value functions, where the task is encoded as the goal, we can learn a curriculum policy that can generalize to produce curricula for new unseen tasks. This combination allows us to essentially perform “zero-shot”” curriculum learning, where a curriculum is generated for a novel target task based on experience generating curricula for similar tasks.

2. Background

We begin by introducing some basic notation and background on reinforcement learning, curriculum learning, and universal value functions, which will form the basis for our approach.

2.1. Reinforcement Learning

We model an agent’s interaction with its environment (i.e. a *task*) as an episodic Markov Decision Process (MDP) (Sutton & Barto, 2018). An episodic MDP M is a 4-tuple $(\mathcal{S}, \mathcal{A}, p, r)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $p(s'|s,a)$ is a transition function that gives the probability of transitioning to state s' after taking action a in state s, and $r(s,a,s')$ is a reward function that gives the immediate reward for taking action a in state s and transitioning to state s'. In addition, we use Δs_0 to denote the initial state distribution, and $\mathcal{S}_f$ to denote the set of terminal states.

[1]Department of Computer Science, University of Texas at Austin, Austin, TX, USA. Correspondence to: Sanmit Narvekar <sanmit@cs.utexas.edu>.

Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, PMLR 108, 2020. Copyright 2020 by the author(s).

At each time step t, the agent observes its state and chooses an action according to its *policy* $\pi_\theta(a|s)$, which we assume is parameterized by θ. The goal of the agent is to learn an *optimal policy* π^*, which maximizes the expected *return* (cumulative sum of rewards) until the episode ends. One algorithm for appoximating π^* is DQN (Mnih et al., 2015), which uses a function approximator (specifically, a deep neural network) to learn an action-value function that estimates the expected return for taking an action in a state and following the current policy after. The optimal policy is derived by acting greedily with respect to the values.

2.2. Curriculum Learning in RL

Many RL problems of interest are challenging due to issues such as sparse reward signals, poor state representation, or the presence of adversaries. One way learning can be accelerated in such settings is by first training on a simpler source task, and transferring the knowledge acquired to improve learning on a subsequent target task. Transfer learning (Taylor & Stone, 2009) is an area of research that focuses on how this knowledge can be transferred. For example, one method is value function transfer (Taylor et al., 2007), which initializes the value function in the target task using the value function from the source task. This process in effect creates an exploration bias in the target task. In deep learning, this process is also referred to as finetuning.

Curriculum learning (Narvekar et al., 2020) considers how exactly to select and order different source tasks, such that performance or learning speed is improved on the target task. In this work, we consider a curriculum learning model (Narvekar & Stone, 2019) that poses curriculum generation as an interaction between 2 MDPs. The first is an MDP for the student agent, which will be the recipient of the curriculum. This agent interacts in the standard way with a given task. The second is a higher level curriculum MDP (CMDP) for the teacher, whose goal is to select tasks for the student to train on. Formally, a CMDP M^C is a tuple $(\mathcal{S}^C, \mathcal{A}^C, p^C, r^C)$ where the set of states $\mathcal{S}^C$ represent the state of the student agent's knowledge or learning progress, and the set of actions $\mathcal{A}^C$ are the possible tasks the student can train on next. Training on a task updates the student's state according to the transition function p^C, and incurs a cost r^C, which is the time it takes to learn that task. The CMDP terminates when the student is able to solve the target task to a desired performance threshold. Thus, a policy π^C over a CMDP is a mapping from the state of knowledge of an agent to the task it should learn next. The optimal curriculum policy minimizes the time needed to learn the target task to a desired performance level.

2.3. Universal Value Functions

In standard reinforcement learning, the value function $v_\pi(s)$ estimates the return of a policy from a given state s. In deep reinforcement learning, the value function is represented by a deep neural network, and exploits the structure in the state space to learn values for observed states and generalize to unseen states. In goal-oriented tasks, where the environment transition dynamics stay the same but the goal state may differ, much of the structure of a value function can also be shared across goals. Thus, the idea behind Universal Value Functions (Schaul et al., 2015) is to create a value function $v_\pi(s, g)$ that generalize over both states s and goals g by creating an embedding over state features and goal features:

$$v_\pi(s, g) = \mathbb{E}^\pi\left[\sum_{t=0}^{\infty} r_g(s_t, a, s_{t+1}) \middle| s_0 = s\right] \tag{1}$$

3. Curriculum Generalization

Our main idea is to learn a universal value function over a curriculum MDP so that we can generalize over CMDP states and goals. Therefore, we need a way of representing CMDP states and goals.

Recall that a CMDP state parametrically represents the agent's knowledge. One way to represent the agent's state of knowledge is by its policy π_θ. In particular, the class of policies the agent can represent is determined by the structure of the function approximator used, and the instantiation of weights θ determines the exact policy in this class. Thus, when access to the internal representation of the agent is available, we can represent the agent's state of knowledge in the CMDP state $\mathcal{S}^C$ using the vector of weights of the student agent's value function or policy θ.

A goal in the universal value framework is represented as a single state: $g \in \mathcal{S}$. In the CMDP setting, we instead represent goals g^C as target tasks that the agent could be trained on, with one goal for each target task. Note that this effectively represents a goal as a subset of CMDP states $g^C \subseteq \mathcal{S}^C$, where the student agent policies θ represented by those states are able to solve that specific target task.

A key question is how to represent tasks. In this work, we restrict our attention to goal-based navigational tasks, which are defined by a starting position and an ending position. This allows us to easily create a parameterized representation of the task by using the concatenated vector of coordinates corresponding to the starting and ending states. However, an important direction for future work is to extend these ideas to non-goal based tasks, such as those described by language or vision commands (Chevalier-Boisvert et al., 2019).

Given a representation for both the CMDP state and goal, we use a two-stream neural network architecture as used by (Schaul et al., 2015) to learn a universal value function over the CMDP. A two-stream architecture assumes the problem can be factorized into two components. In our case, one component is $\phi : \mathcal{S}^C \mapsto \mathbb{R}^n$, which creates an embedding for CMDP states. The second is $\psi : \mathcal{G}^C \mapsto \mathbb{R}^n$, which creates an embedding for CMDP goals. The two streams are combined using an output function $h : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}^m$. In our case, the mappings ϕ and ψ are represented by multi-layer perceptrons, and the output function is the Hadamard product. See Figure 2 for the architecture we use in the experiments.

A policy extracted from this value function is then able to suggest a task to the student based both on what the student knows, and the task it needs to learn. Given enough experience on a set of "training" target tasks, our experiments will show that learning such a universal value function allows the curriculum policy to generalize and zero-shot produce curricula for unseen "test" target tasks.

4. Experimental Results

We evaluated curriculum transfer on navigation tasks in a static gridworld environment. Our goal was to train a CMDP teacher agent to learn a curriculum policy on a subset of tasks, and show that it can zero-shot produce curricula for a student on novel unseen target tasks.

Gridworld Navigation

The gridworld environment considers goal-oriented navigation tasks in a standard 4-room grid world. The environment consists of 4 connected rooms, where each room is 5x5 in size and connected at one cell to each adjacent room. A navigation task is defined by a pair of (x, y) coordinates for the starting position and goal position. There are 100 possible starting and ending positions. We ignore tasks that start and end on the same position, thus there are 9900 possible different target tasks. See Figure 1(a) for an example of a task.

Our student agent has a tabular representation for the state space, and learns using Sarsa(λ) with exploration $\epsilon = 0.1$, learning rate $\alpha = 0.1$, discount factor $\gamma = 1.0$, and eligibility trace decay rate $\lambda = 0.7$. We use value function transfer to transfer information between tasks in the curriculum.

TEACHER (CMDP) AGENT DESCRIPTION

State and Goal Space

The CMDP needs to learn to generalize over both the agent's knowledge and task space. Conceptually, the agent's current policy – its function for selecting actions in each state, which we assume to be known to the teacher – is its state of knowledge. We thus represent the agent's knowledge using the vector of weights associated with the student's Q-function table. Tasks are represented using the pair of (x, y) coordinates associated with the starting and goal states.

Action Space

We create nine different source tasks. Eight of these are static tasks that don't change based on the target task. These tasks initialize the agent in one of the 4 rooms, and terminate when the agent moves into one of the adjacent two rooms. There is one such task for each room and adjacent room pair. In addition, all corridors between rooms are blocked except for the one required to complete the task. The ninth source task is a dynamic source task that changes based on the current target task. This task initializes the agent in the same room as the goal of the target task, and sets the goal tile to be the same as the target task. As with the static sources, all corridors to other rooms are blocked off. These sources, together with the target task, form the action space of the CMDP. Note that these tasks were intentionally designed to give rise to a natural and interpretable curriculum for each target task: use the static sources to navigate to the goal room, and follow with the dynamic source task to complete the path.

Reward Function

When a task is selected, it is trained on until convergence. We consider a task converged when the steps taken to reach the goal averaged over the last 5 episodes is less than the Manhattan distance between the start $\mathbf{s}$ and goal positions $\mathbf{g}$ plus a slack term δ:

$$\text{converged} := \big(\text{steps taken} < ||\mathbf{s} - \mathbf{g}||_1 + \delta\big) \qquad (2)$$

We set δ to 0 when the start and goal positions are in the same room, 5 when they are in adjacent rooms, and 10 when they are diagonally across, to account for navigating around walls. The cost of learning a task is the number of steps needed to learn to convergence. Therefore, the reward given is the negative of the steps taken.

Architecture and Learning Parameters

We use DQN (Mnih et al., 2015) to learn the CMDP. The neural network uses a two-stream architecture, where the features relating to the task/goal space pass through a single hidden layer, while the agent knowledge features pass through three hidden layers. Each hidden layer has 128 units followed by a $\tanh$ activation function. The two streams are subsequently merged via element-wise multiplication, and pass through a final hidden layer to produce action-values. A diagram of this network can be seen in Figure 2. The learning rate is set to 5e-4, replay buffer size to 5000, batch size 64, exploration fraction to 0.05, and has the target network updated every 50 steps. To speed up training, we

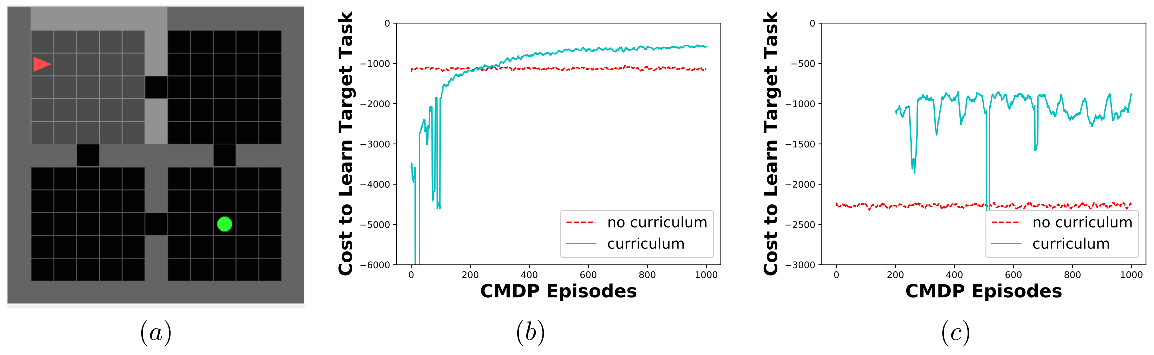

Figure 1. (a) An example of a task in the gridworld environment. The red arrow is the agent, and the green circle is the goal. (b) CMDP learning curves for the interpolation experiments. (c) CMDP learning curves for the extrapolation experiments. In the CMDP learning curves, the x-axis represents a CMDP episode, where each episode is an entire run of a curriculum. The y-axis is the cost of that curriculum in game steps.

also capped the number of actions the CMDP could take to 5, and trained on the target task thereafter. The learning parameters were not extensively optimized.

Results

We consider two types of generalization that may be possible in CMDPs: interpolation and extrapolation. In the interpolation case, we randomly shuffle all the possible tasks, and present them to the CMDP agent one by one. This is the lifelong learning setting, where each task encountered is new. The results are shown in Figure 1(b). As the CMDP learns and the coverage of tasks increase, the curricula produced gradually improve, until they pass the baseline of just training on the target task after having seen 300 tasks. Thus, the results show how many tasks a curriculum must be learned for before the teacher agent is able to zero-shot produce curricula for novel unseen tasks.

In the extrapolation case, we explicitly split the set of target tasks into a train set and a test set. The test set contains all tasks that start in the top left room, and end in the bottom right room, while the train set contains tasks with all other possible start and end goal pairs. The extrapolation case is more challenging, because in the previous interpolation setting, generalization was expected because goals were represented with similar features and the set of training tasks covered pairings from all the different rooms. However, in this case, the curriculum of navigating from the top left room to the bottom right room has not been seen before. We train on tasks in the train set for the first 200 CMDP episodes, and subsequently evaluate on the test set. The results are shown in Figure 1(c), with the curriculum graph offset to reflect time spent training on the train set. The results once again show that curricula learned on a different set of tasks can transfer to produce curricula for new unseen tasks.

5. Related Work

The idea of using curricula to train learning agents can be traced back at least as far as (Elman, 1993). Over the years, curricula have been used to train agents on complex tasks in areas such as robotics (Asada et al., 1996; MacAlpine & Stone, 2018) and games (Wu & Tian, 2017). Most of these methods relied on manual specification of the curriculum, which often requires a lot of human time, domain expertise, and is based on intuition rather than quantitative analysis or metrics.

In recent years, automated approaches to construct a curriculum have been devised. Due to the complexity of the problem, many of these methods rely on heuristics to perform sequencing, such as learning progress (Graves et al., 2017), transfer potential (Svetlik et al., 2017; Da Silva & Costa, 2018), and task similarity (Narvekar et al., 2017). Other approaches make simplifying assumptions on the types of tasks that can be used in the curriculum. For example, some only allow the initial and terminal state distribution to vary between source and target (Florensa et al., 2017; 2018), or only change the reward function (Riedmiller et al., 2018; Sukhbaatar et al., 2018). Finally, another line of work considers the formulation of curricula as an MDP (Narvekar & Stone, 2019) or POMDP (Matiisen et al., 2017), which our work builds upon. This formulation uses learning as opposed to heuristics to guide task sequencing, and does not make any restrictions on the source tasks. However, as far as we know, in all of these previous works, the curriculum is designed for a single agent to learn on a single target task. Generating a curriculum for a new target task or agent requires performing the same generation process from scratch, which can be redundant and expensive. Our work attempts to address this shortcoming.

Lastly, curriculum learning has also been explored in supervised learning (Bengio et al., 2009). For example, (Fan et al.,

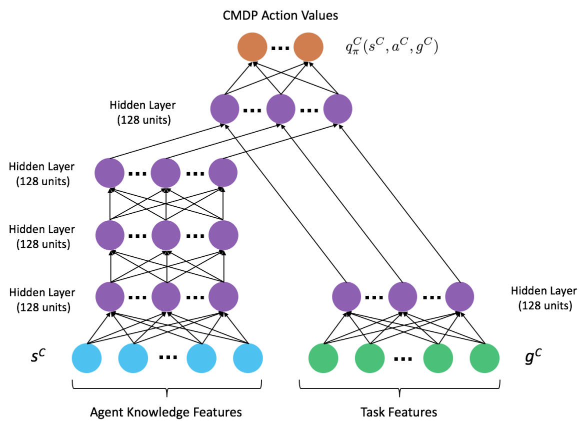

Figure 2. The two-stream network architecture used for the teacher CMDP agent.

2018) use a similar MDP style formulation to order samples of training data. They showed that a teacher MDP can train a student on a new task, where a task in this case is a subset of samples from the same distribution (e.g. one task was the first half of MNIST while another was the second half). However, unlike in reinforcement learning, the distribution of the training and test data does not change during learning. Our work provides a parallel to (Fan et al., 2018) for the reinforcement learning setting.

6. Conclusion

Most existing work on automated curriculum learning has relied on heuristics, or limited the types of source tasks that can be used in a curriculum, because learning a full curriculum directly from experience can be computationally expensive. In this paper, we showed how curriculum policies can generalize to produce curricula for new unseen tasks. This process effectively amortizes the cost of learning a curriculum and opens the door to using more learning from experience in curriculum design. In human and animal training, as well as more recently in supervised machine learning, curricula have been adapted to train multiple types of learners for different target tasks. This work provides a similar result for the reinforcement learning setting.

Acknowledgements

This work has taken place in the Learning Agents Research Group (LARG) at the Artificial Intelligence Laboratory, The University of Texas at Austin. LARG research is supported in part by grants from the National Science Foundation (CPS-1739964, IIS-1724157, NRI-1925082), the Office of Naval Research (N00014-18-2243), Future of Life Institute (RFP2-000), Army Research Office (W911NF-19-2-0333), DARPA, Lockheed Martin, General Motors, and Bosch. The views and conclusions contained in this document are those of the authors alone. Peter Stone serves as the Executive Director of Sony AI America and receives financial compensation for this work. The terms of this arrangement have been reviewed and approved by the University of Texas at Austin in accordance with its policy on objectivity in research.

References

Asada, M., Noda, S., Tawaratsumida, S., and Hosoda, K. Purposive behavior acquisition for a real robot by vision-based reinforcement learning. *Machine learning*, 23(2-3): 279–303, 1996.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 41–48. ACM, 2009.

Chevalier-Boisvert, M., Bahdanau, D., Lahlou, S., Willems, L., Saharia, C., Nguyen, T. H., and Bengio, Y. BabyAI: First steps towards grounded language learning with a human in the loop. In *International Conference on Learning Representations*, 2019.

Da Silva, F. L. and Costa, A. H. R. Object-oriented curriculum generation for reinforcement learning. In *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2018.

Elman, J. L. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99, 1993.

Fan, Y., Tian, F., Qin, T., Li, X.-Y., and Liu, T.-Y. Learning to teach. In *International Conference on Learning Representations*, 2018.

Florensa, C., Held, D., Wulfmeier, M., Zhang, M., and Abbeel, P. Reverse curriculum generation for reinforcement learning. In *Proceedings of the 1st Annual Conference on Robot Learning*, 2017.

Florensa, C., Held, D., Geng, X., and Abbeel, P. Automatic goal generation for reinforcement learning agents. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1515–1528, Stockholmsmssan, Stockholm Sweden, July 2018.

Graves, A., Bellemare, M. G., Menick, J., Munos, R., and Kavukcuoglu, K. Automated curriculum learning for neural networks. In *International Conference on Machine Learning (ICML)*, 2017.

MacAlpine, P. and Stone, P. Overlapping layered learning. *Artificial Intelligence*, 254:21–43, 2018.

Matiisen, T., Oliver, A., Cohen, T., and Schulman, J. Teacher-student curriculum learning. *IEEE transactions on neural networks and learning systems*, 2017.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015.

Narvekar, S. and Stone, P. Learning curriculum policies for reinforcement learning. In *Proceedings of the 18th International Conference on Autonomous Agents and Multi-Agent Systems*, pp. 25–33. International Foundation for Autonomous Agents and Multiagent Systems, 2019.

Narvekar, S., Sinapov, J., and Stone, P. Autonomous task sequencing for customized curriculum design in reinforcement learning. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, August 2017.

Narvekar, S., Peng, B., Leonetti, M., Sinapov, J., Taylor, M. E., and Stone, P. Curriculum learning for reinforcement learning domains: A framework and survey. *arXiv preprint arXiv:2003.04960*, 2020.

Riedmiller, M., Hafner, R., Lampe, T., Neunert, M., Degrave, J., Van de Wiele, T., Mnih, V., Heess, N., and Springenberg, J. T. Learning by playing-solving sparse reward tasks from scratch. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

Schaul, T., Horgan, D., Gregor, K., and Silver, D. Universal value function approximators. In *International Conference on Machine Learning*, pp. 1312–1320, 2015.

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

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT Press, 2018.

Svetlik, M., Leonetti, M., Sinapov, J., Shah, R., Walker, N., and Stone, P. Automatic curriculum graph generation for reinforcement learning agents. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*, February 2017.

Taylor, M. E. and Stone, P. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(1):1633–1685, 2009.

Taylor, M. E., Stone, P., and Liu, Y. Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research*, 8(1):2125–2167, 2007.

Wu, Y. and Tian, Y. Training agent for first-person shooter game with actor-critic curriculum learning. In *International Conference on Learning Representations (ICLR)*, 2017.