Automatic Curriculum Graph Generation for Reinforcement Learning Agents

Maxwell Svetlik
University of Texas at Austin
Austin, USA

Matteo Leonetti
University of Leeds
Leeds, UK

Jivko Sinapov
University of Texas at Austin
Austin, USA

Rishi Shah
University of Texas at Austin
Austin, USA

Nick Walker
University of Texas at Austin
Austin, USA

Peter Stone
University of Texas at Austin
Austin, USA

Abstract

In recent years, research has shown that transfer learning methods can be leveraged to construct curricula that sequence a series of simpler tasks such that performance on a final target task is improved. A major limitation of existing approaches is that such curricula are handcrafted by humans that are typically domain experts. To address this limitation, we introduce a method to generate a curriculum based on task descriptors and a novel metric of transfer potential. Our method automatically generates a curriculum as a directed acyclic graph (as opposed to a linear sequence as done in existing work). Experiments in both discrete and continuous domains show that our method produces curricula that improve the agent's learning performance when compared to the baseline condition of learning on the target task from scratch.

1 Introduction

Increasingly, reinforcement learning (RL) agents are being tasked with challenging problems that may be unfeasible to learn directly. To address such situations, transfer learning (TL) methods reuse knowledge learned in a *source* task so that an agent can learn a difficult *target* task faster or converge to a better policy (Taylor and Stone 2009; Lazaric 2012). Most recently, TL methods have been leveraged for the problem of *curriculum learning* in which the goal is to construct a *curriculum* where an agent learns a sequence of tasks (Narvekar et al. 2016; Peng et al. 2016).

The major limitation of current approaches to curriculum learning is that the curriculum is typically hand-crafted or designed by a human, often an expert in the domain. In addition, most TL methods do not generalize to simultaneous transfer from multiple source tasks to a given target task. Thus, each hand-crafted curriculum is represented as a linear sequence of tasks, which does not afford parallelization of tasks when learning.

To address these issues, we introduce a framework for automatic construction of a curriculum. Our method allows an agent to transfer from multiple source tasks into a single target task and, as a result, the generated curriculum is represented as a Directed Acyclic Graph. The construction of the curriculum graph is based on a novel measure of task relatedness, that utilizes task descriptors to estimate the benefit of transferring knowledge from one task to another. We evaluate our method on discrete and continuous RL domains and the results show that employing the generated curricula leads to a substantial speed-up in learning performance.

2 Background

2.1 Markov Decision Process

A Markov Decision Process (MDP) is a 5-tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of possible actions, $\mathcal{P}$ is the transition function, $\mathcal{R}$ is the reward function, and γ is the discount factor. We represent an instance of MDP t as a *task*. The goal of the agent is to maximize the expected value of the *return* $G_t = \sum_{t \geq 0} \gamma^t R_{t+1}$. An agent can learn the optimal policy π^* by learning the optimal action-value function $Q^*(s, a)$ which gives the expected return for taking action a in state s and acting optimally thereafter.

In our experiments, the action-value function was learned using either the Q-learning algorithm (Watkins and Dayan 1992) or SARSA (Singh and Sutton 1996).

2.2 Potential-based Reward Shaping

The method proposed in this paper uses reward shaping (Ng, Harada, and Russell 1999) as a means to transfer knowledge from one or more source tasks to a target task. In reward shaping, the reward signal is modified by adding additional reward, driving the agent towards the desired behavior. The reward function becomes $R'(s, a, s') = R(s, a, s') + F(s, a, s')$, where F is the shaping function. If the reward is sparse, the agent may have no feedback for many actions, making learning more difficult. The shaping function can fill these gaps, providing advice on the value of the actions. We use potential-based look-ahead advice (Wiewiora, Cottrell, and Elkan 2003), where the reward function is augmented by the potential-based advice function:

$$F(s, a, s', a') = \gamma \phi(s', a') - \phi(s, a) \tag{1}$$

If ϕ is a potential function over states and actions, its impact on the value function is nondisruptive, allowing the true value function to be recovered (Wiewiora, Cottrell, and Elkan 2003).

Copyright © 2017, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

3 Related Work

In supervised machine learning, transfer learning has been applied to effectively reuse knowledge acquired in one domain without having to completely start from scratch in a new problem (Pan and Yang 2010). More recently, transfer learning methodologies have been developed for RL settings (Taylor and Stone 2009; Lazaric 2012).

Where multiple source tasks are available, a key question is how to pick a good source task for a given target. Lazaric *et al.* (2008) proposed a framework that allows an agent to select samples from multiple source tasks based on their similarity to samples in the target task. Models (Nguyen, Silander, and Leong 2012) and options (Perkins and Precup 1999) have also been transferred, by identifying which source task is most similar to the target task. Along the same research line, methods have been proposed to estimate the similarity between MDPs known to the agent, and the one that it is trying to learn (Ferns et al. 2012; Ferns, Panangaden, and Precup 2011; Ammar et al. 2014). In cases where samples from the target task are unavailable, *task descriptors* (e.g., attribute-value pairs) have been used to identify the relevant source task(s) (Sinapov et al. 2015; Isele, Rostami, and Eaton 2016).

While these methods have shown promise at selecting appropriate source tasks for a given target, they stopped short of automatically designing a full curriculum. More recently, Narvekar et al. (2016) and Peng et al. (2016) proposed methods for generating tasks that can be useful candidates for a curriculum, learned in a sequence. In these works, the curricula were hand-crafted by either expert or naive users; the goal of our approach, on the other hand, is to automatically construct such curricula.

We use potential based reward shaping (Ng, Harada, and Russell 1999; Wiewiora, Cottrell, and Elkan 2003) as a means of transferring knowledge from source tasks to a target. In reward shaping, there remain two open questions: where the potential function should come from, and how it should be engineered. We propose to answer those questions in the context of curriculum learning. We use the framework of Konidaris and Barto (2006) for learning shaping functions as a basis to engineer a potential function. In their framework, an agent associates local perceptions to the expected value of a state. The agent learns a possibly non-Markovian function from the perceptions (which define the *agent*-space), to a prediction of the reward. These functions are used to initialize (or equivalently as a shaping signal for) the value function, which is defined, problem by problem, over the smallest set of variables necessary to make the representation Markovian (the *problem*-space). We use this notion of local perception to transfer value functions learned from one task to another even in cases where the global state space may differ between tasks.

4 Problem Formulation and Notation

Let $\mathcal{T}$ be a set of tasks generated from the same domain $\mathcal{D}$, where $\mathcal{F}_t$ is the vector of *degrees of freedom* for task t. Each element of $\mathcal{F}_t$ is a *feature*, representing a property of the domain (for instance, the number of ghosts in Ms. Pac-Man, one of the domains we use in our experiments), or the presence of an object (such as pills in Ms. Pac-Man).

Given a finite set of tasks $\mathcal{T}$, and a final task $t_f \in \mathcal{T}$, our aim is to find a suitable curriculum for t_f which uses only tasks from $\mathcal{T}$. We begin formalizing this problem by giving the definition of curriculum in the context of this work.

Definition (Curriculum)**.** *Let $C = \langle V, E \rangle$ be a directed graph, where $V \subseteq \mathcal{T}$ is the set of vertices and $E \subseteq V \times V$ is the set of edges. Furthermore, let $deg^+(v)$ be the out-degree of a vertex $v \in V$. Given a set of tasks $\mathcal{T}$, a task $t_f \in \mathcal{T}$, and a graph C, C is a* curriculum *over $\mathcal{T}$ for t_f iff C is acyclic, weakly connected, and $deg^+(v) > 0$ for each $v \in V$ s.t. $v \neq t_f$, and $deg^+(t_f) = 0$.*

If there is an edge in a curriculum C between two tasks from t_s to t_t, then t_s is a source task for the target task t_t.

4.1 Learning Through a Curriculum

A curriculum $C = \langle V, E \rangle$ imposes a partial order in which the tasks may be executed. It also specifies which tasks are *source* tasks for which *target*. Given a task $t \in V$, the set of source tasks for t is $X_t = \{s \in V | \langle s, t \rangle \in E\}$. We transfer knowledge from sources to target tasks through reward shaping, which allows the agent to merge information coming from multiple sources, and to use a different representation for the value function in each task.

Reward shaping setting As mentioned in Section 3, we use the framework by Konidaris and Barto (2006), in which the agent learns portable non-Markovian predictions from perceptions to the expected cumulative reward. With no constraint on the tasks from which the agent learns, it is impossible to know which perceptions will help in future tasks, and the associated prediction may take the general form of a non-Markovian function. Furthermore, in general, the value function of an MDP may not be applicable in a different MDP. In our setting, however, the intermediate tasks in the curriculum are specifically created to have some features in common with the final task. By defining the value function over a set of features $\mathcal{F}_t$, the value function can be portable to a task whose degrees of freedom are a superset of $\mathcal{F}_t$.

This possibility, which differentiates our framework from Konidaris and Barto (2006), is offered by a fundamental difference between curriculum learning and general transfer learning: the system has full control over which tasks the agent will learn. Thus, we do not require the agent to learn additional functions in agent-space, but the value of a task can be reused as a shaping function in other tasks.

In order for a value function to be portable, besides being defined over features shared among the tasks, it must also be agent-centric. For instance, the global position of the agent in Ms. Pac-Man is not portable. However, the position of a ghost with respect to the agent is. If a coordinate change is possible, so that perceptions can be represented with respect to the agent, then the value function is portable.

A portable value function can then be applied to a different task from the one in which it was learned. A value function defined over features $\mathcal{F}_i$ of a source task i is *applicable* to any task j with features $\mathcal{F}_j \supseteq \mathcal{F}_i$. The state space of the two tasks, however, is different in general. The vector representing a state $s_j = [s_i, s_{j \setminus i}]$ in task j, can be considered as partitioned into two components, relative to the features in $\mathcal{F}_i$ and $\mathcal{F}_j \setminus \mathcal{F}_i$ respectively. Let S_j be the state space of task j, and A_i the actions of task i. We define the domain of the application of a value function Q_i over S_j as:

$$D_{Q_i}(j) = \{s_j \in S_j | \exists a \in A_i, \text{ s.t. } Q_i(s_i, a) \text{ is defined}\} \tag{2}$$

The value function over the component of the state s_i may be undefined, for instance, if the same variables have different ranges in the source and the target task.

We define a potential function $\phi_{i,j}$ from task i to task j, according to Equation 1, as:

$$\phi_{i,j}(s, a) = \begin{cases} Q_i(s_i, a) & \text{if } s \in D_{Q_i}(j) \wedge a \in A_i \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

for all $s \in S_j$ and $a \in A_j$.

As introduced at the start of this section, learning through a curriculum requires, in general, transferring from a *set* of source tasks. Interestingly, reward shaping is also well suited to this scenario. We propose to define the potential function ϕ_t used for shaping the reward in a target task t, as a composition of the value functions of the optimal policy (but any policy could be used in principle) of the tasks in X_t. Value functions are potential functions, and the composition must retain this property to avoid modifying the target task. For this paper, we use the sum of the value functions:

$$\phi_t(s, a) = \sum_{i \in X_t} \phi_{i,t}(s, a), \; \forall s \in S_t, \, a \in A_t \tag{4}$$

Curriculum learning procedure By specifying how to construct a potential function for the curriculum case, we have a suitable transfer mechanism. We are now able to specify how an agent should learn through a curriculum C. The agent can learn a task t if it has all the required potential functions for shaping, that is, if it already learned all the tasks in X_t. Algorithm 1 is an algorithm for curriculum learning, under the definitions given in this paper. This simple algorithm is non-deterministic, since the order in which tasks are selected at line 5 is only partially specified. The order between tasks in different paths does not matter, as a consequence of transferring through shaping. Once a task is learned, the advice provided by shaping is canceled out in the value function (Wiewiora, Cottrell, and Elkan 2003), and the history of the previous tasks erased as a consequence. Another interesting consequence of this non-determinism is that tasks for which no ordering is specified may be learned in parallel, if the system allows it (for instance, with simulated agents, or in multi-agent systems).

Algorithm 1 Curriculum learning

```
1: Given: curriculum $C = \langle V, E \rangle$
2: $L = \{\}$ set of learned tasks
3: $Q = \{\}$ set of value functions learned.
4: while $V \setminus L \neq \emptyset$ do
5:     pick a task $t$ in $V \setminus L$ such that $X_t \subseteq L$
6:     $Q_t$ = learn(t,$\{Q_i | t_i \in X_t\}$)
7:     $L = L \cup \{t\}$, $Q = Q \cup \{Q_t\}$
8: Return $Q_{t_f}$
```

4.2 Evaluating curricula

We evaluate a curriculum with a metric based on the *time to threshold* (Taylor and Stone 2009) of the final task, while taking into account the amount of experience gathered through the curriculum, in order to guarantee *strong* transfer (Taylor and Stone 2009). The user should specify a metric for experience, e, which can be, for instance, computation time or number of actions, depending on what is more important to the user for a particular domain.

Given a function $e : 2^{\mathcal{T}} \times \mathcal{T} \rightarrow \mathcal{R}$, we denote with $e(X_t, t)$ the amount of experience to learn a task t with advice from the tasks in X_t (which may be empty). Furthermore, we denote with $e_f : 2^{\mathcal{T}} \times \mathcal{R} \rightarrow \mathcal{R}$ the function which measures the experience accumulated in the final task, with advice from the tasks in X_f, up to reaching the threshold l.

The expected total experience required by a curriculum C to reach threshold l is therefore:

$$E_l(C) = \mathbb{E}\left[\left(\sum_{t \in V s.t. t \neq t_f} e(X_t, t)\right) + e_f(X_f, l)\right]. \tag{5}$$

The expected total experience will be estimated, in practice, by sampling multiple runs of curriculum learning. The curriculum $C_f = \langle\{t_f\}, \emptyset\rangle$ which contains only the final task provides the baseline expected experience $\mathbb{E}[e_f(\emptyset, l)]$. Ideally, we would like a curriculum which minimizes the function E_l. Since the component function e, however, depends on the power set of the tasks, the optimization quickly becomes combinatorially intractable. In the next section we define a heuristic algorithm which is shown to create curricula that improve learning performance when compared to the baseline condition of learning the final target task from scratch.

5 Generating Curriculum Graphs

We define a heuristic to guide our algorithm, which estimates the usefulness of transferring from a task, for a given target task. We name this heuristic *transfer potential*. We use transfer potential to first determine which tasks are relevant enough to the final task, and then how to connect the most relevant tasks.

5.1 Measuring Transfer Potential

We define transfer potential according to two observations. The first one is suggested by the definitions of the transfer mechanism. In particular, Equation 3 shows that a source task s is useful for a given target proportionally to the size of $D_{Q_s}(t)$, that is, to the applicability of its value function to the target. The second observation is derived from the definition of the total experience in Equation 5. Even if the source is strongly related with the target, there has to be a significant advantage in the experience necessary to learn both tasks over the target only.

Algorithm 2 Intra-group transfer

```
1: Given: task group $g$
2: $\mathcal{H} = \{\}$, $sort(g)\ descending$
3: for $t_i \in g$ do
4:     $t_m = argmax_{t_j \in g | i < j}\ \nu(t_j, t_i)$
5:     if $\nu(t_m, t_i) > \epsilon$ then
6:         $\mathcal{H} \leftarrow \mathcal{H} \cup \langle t_m, t_i \rangle$
7: Return $\mathcal{H}$
```

The experience necessary to learn a task depends on many factors: the size of the state space, the stochasticity of the dynamics, the learning algorithm involved, and the quality of the advice. The only way of measuring experience is to execute the task. For this paper, we use a simple proxy for experience, based on one of these factors: the size of the state space. Holding everything else constant, the larger the task, the more difficult it is to learn. This allows us to estimate transfer potential without acting in the environment. Given these observations, we define transfer potential as:

$$\nu(s, t) = \frac{|\mathcal{D}_{Q_s}(t)|}{1 + |\mathcal{S}_t| - |\mathcal{S}_s|} \tag{6}$$

Modeling the cost in this way naturally prioritizes tasks with smaller state space and drives a behavior akin to learning from easy missions (Asada et al. 1996).

5.2 Task Elimination and Grouping

In order to ensure that transfer is facilitated between tasks that ultimately give the agent information about the final target task, we first prune the task pool and eliminate tasks with low potential. The first step of the algorithm is computing the set V of the curriculum, as:

$$V = \{t \in \mathcal{T} | \nu(t, t_f) > \epsilon\}, \tag{7}$$

where ϵ is the minimum amount of potential for a task to be part of the curriculum, with respect to the final task.

Transfer potential is defined between pairs of tasks, while experience in Equation 5 depends on sets of source tasks. While our definition of the potential allows us to carry out only pair-wise comparisons, we also introduce a mechanism to allow more than one source task. We exploit our initial knowledge about the tasks' degrees of freedom, $\mathcal{F}$, to group tasks which share certain features. We construct a binary descriptor $\mathcal{B}_t$ from $\mathcal{F}_t$, in order to have a coarse representation of how tasks are related. $\mathcal{B}_{t,i}$ and $\mathcal{F}_{t,i}$ are then the i-th element of vectors $\mathcal{B}_t$ and $\mathcal{F}_t$ respectively, and we let $B_{t,i} = (\mathcal{F}_{t,i} \neq 0)$. The descriptor $\mathcal{B}$ effectively partitions the task set. We say that the tasks which share the descriptor $\mathcal{B}_k$ belong to group $g_k \in \mathcal{G}$, where $\mathcal{G}$ is the set of groups, that is a partition of V (the curriculum's set of tasks).

5.3 Intra-Group Transfer

In intra-group transfer, we consider transfer between tasks within the same group. Intra-group transfer, shown in Algorithm 2, takes as input a group $g \in \mathcal{G}$, which is then sorted with respect to the transfer potential to the final task.

Algorithm 3 Inter-group transfer

```
1: Given: ordered transfer groups $g$,$g'$
2: $\mathcal{H} = \{\}$
3: for $t_i \in g'$ do
4:     $t_m = argmax_{t_j \in g}\ \nu(t_j, t_i)$
5:     if $\nu(t_m, t_i) > \epsilon$ then
6:         $\mathcal{H} \leftarrow \mathcal{H} \cup \langle t_m, t_i \rangle$
7: return $\mathcal{H}$
```

Algorithm 4 Curriculum Graph Generation

```
 1: Given: task set $\mathcal{T}$, final target task $t_f$
 2: $\mathcal{G}, V' = partition(\mathcal{T})$
 3: $C = \langle V, E \rangle, V = V', E = \{\}$
 4: for $g \in \mathcal{G}$ do
 5:     $E \leftarrow E \cup$ Algorithm 2($g$)
 6: for $t_s \in V'$ s.t. $deg^+(t_s) = 0,\ t_s \neq t_f$ do
 7:     $E \leftarrow E\ \cup \langle t_s, t_f \rangle$
 8: for $g' \in \mathcal{G}$ do
 9:     $\mathcal{G}_s \leftarrow \{g \in \mathcal{G} | \mathcal{B}_{\top g} \subset \mathcal{B}_{\top g'}\}$
10:     for $g \in \mathcal{G}_s$ do
11:         $E \leftarrow E \cup$ Algorithm 3($g$, $g'$)
12: return $C$
```

The algorithm iterates over the group, assigning an edge at line 6 between task j and task i if j has the highest potential among the tasks coming after i in the group, and the potential is above ϵ.

The resulting set of edges is returned.

5.4 Inter-Group Transfer

While any given group contains tasks that are themselves related, there may be tasks in separate groups that would also benefit from transfer. This is the goal of Inter-group transfer.

Two task groups are compared directly for transfer. The group g is considered the source group and the other, g', is considered the target group. The algorithm iterates over g, and adds an edge to a task in g' under the same condition as Intra-group transfer.

5.5 Curriculum Graph Generation

Algorithm 4 combines the three phases into a single algorithm. The *partition()* function in line 2 corresponds to the task grouping stage in section 5.2. Intra-group transfer constructs disjoint graphs on each task group. For all graphs generated, all vertices $v \in V,\ v \neq t_f$, where $deg^+(v) = 0$ are given an edge $\langle v, t_f \rangle$ into the final target task at line 7. Inter-group transfer is then carried out amongst the task groups to further assign edges. Target and source groups are subject to the following imposition: a target group will consider all other groups as source groups if the corresponding binary feature descriptor is a subset of the target group's. We let $\mathcal{B}_{\top g}$ be the set of features that are true in $\mathcal{B}_g$ (line 9).

Theorem 5.1. *The graph C produced by Algorithm 4 is a curriculum.*

Proof. The curriculum C is directed by construction. We need to show that C is weakly connected, acyclic and that

t_f is the only sink. During the execution of line 5, edges for all V' are considered within their groups. For each group of tasks, each $v \in V'$ is either assigned an outgoing edge so that $deg^+(v) \geq 1$ or no edge is assigned, in which case $deg^+(v) = 0$. Connectivity of C follows from line 7, where all sinks are given a directed edge into t_f, leaving the single node t_f with $deg^+(t_f) = 0$. Let reach(t_i,t_j) be the reachability relation between tasks, such that reach(t_i,t_j) $\Leftrightarrow$ $\langle t_i, t_j \rangle \in E \vee \exists t_k$ s.t. reach(t_i,t_k) $\wedge$ $\langle t_k, t_j \rangle \in E$. From line 4 in Algorithm 2 it follows that there can exist an edge between t_i and t_j within a group only if $j > i$. There is a cycle including task t_i iff $\exists t_j$ s.t. reach(t_i,t_j) $\wedge$ reach(t_j,t_i), which would imply $i < j < i$. Hence, each group is internally acyclic. Likewise the assignment of edges between tasks in different groups is acyclic by the same reasoning, where the relationship $\subset$ between group descriptors is used instead of $<$ between task indices, at line 9. □

6 Experimental Results

We evaluate the proposed method for curriculum generation in two discrete state-space domains, Gridworld and Block Dude, and one continuous domain, Ms. Pac-Man. Some of the parameters such as the choice of learning algorithm, learning rates, and stopping criteria used to end training on source tasks differed across domains as they were implemented independently. [1]

6.1 Gridworld Domain

Gridworld, implemented through BURLAP (MacGlashan 2016), is a 2-D domain, similar to Wumpus World (Russell and Norvig 2003). Each cell in the 2-D grid may be empty or occupied by a *fire*, a *pit*, or *treasure*. The available actions are up, down, left and right. The agent receives -1 reward per action, -250 and -500 reward for going next to and into a fire, -2500 reward for going into a pit, and +200 reward for getting to the treasure (entering a pit or treasure terminates the episode). Each episode also ends after 200 steps.

The final target task was a 10×10 world with 10 pits, 10 fires, and 1 treasure. The degrees of freedom of the domain were the height and width of the grid (represented as integers) and number of fires, pits and treasure. The set of possible source tasks $\mathcal{T}$ used as input to our method was generated by applying the Task Simplification Heuristic introduced in Narvekar et al. (2016). In other words, tasks were generated by reducing the size of the grid, and/or removing certain elements of the game (e.g., a 5×5 grid without any pits). In total, there were 30 candidate source tasks as input to our curriculum generation method, which output a curriculum of 5 of these tasks.

Figure 1 plots the agent's expected reward when playing the final target task as a function of the number of game steps used for learning so far. The figure also shows the learning curve for the baseline condition of learning the target task from scratch as well as when following the two best subsets of the generated curriculum, with a size of 3 (Subset 1) and 4 tasks (Subset 2). Note that the plots corresponding to the curricula conditions were offset horizontally to account for the number of game steps used to learn the source tasks. Each task in a curriculum was learned until the expected reward per episode stopped changing for 10 consecutive games. Using the curriculum or, subsets of it, leads to faster convergence than learning the target task on its own.

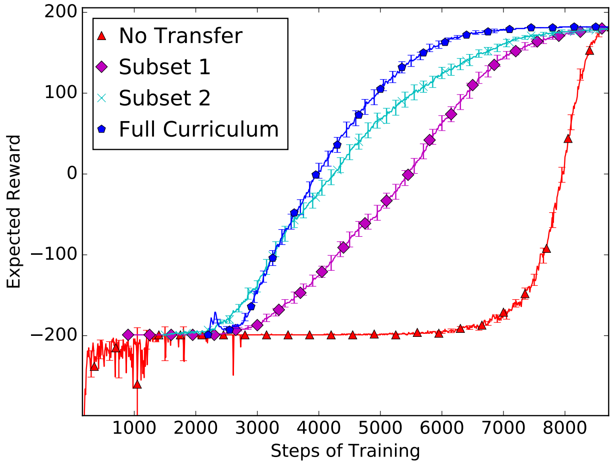


Figure 1: The expected reward while learning a 10x10 Gridworld task. We compare a generated curriculum, and subsets of the generated curriculum, to a baseline Q-Learning agent.

6.2 Block Dude Domain

Block Dude, implemented in BURLAP (MacGlashan 2016), is a 2-D puzzle game in which the agent must reach an exit by stacking blocks to climb over obstacles made out of bricks. The agent can move left and right, pick up and put down blocks, and climb up only if there is a single block or brick in front of it. A -1 reward is given for each action, and a +50 reward is given for reaching the exit. The game ends after the exit is reached, the agent gets trapped (e.g., by building a tower of blocks that it can not climb over), or 200 actions have been taken.

The domain's degrees of freedom were the width and height of the world, the number of blocks present, and the number of brick columns of height 1, 2, 3, and 4. Manipulating these attributes resulted in 10 task variants used as the set of source tasks. Once the curriculum was generated, the agent learned the tasks in the curriculum with the SARSA algorithm using a Radial Basis Function approximator as implemented in BURLAP. Convergence for tasks in a curriculum were detected in the same manner as in Gridworld.

Figure 2 shows the generated curriculum. Following the curriculum allowed the agent to converge to the optimal policy on the final target task in approximately 13,500 game steps (which include learning both source tasks and the final target task). Convergence when learning the final target task from scratch took 17,750 game steps.

6.3 Ms. Pac-Man Domain

In the Ms. Pac-Man domain, the goal of the agent is to traverse a maze, avoid ghosts, and eat edible items such as

[1] All source code used in our evaluation is available at https://github.com/LARG/curriculum_learning_aaai2017

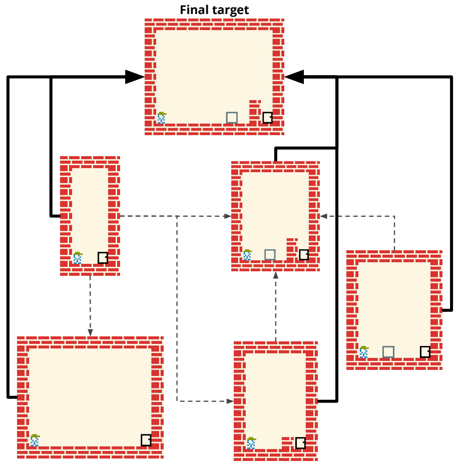
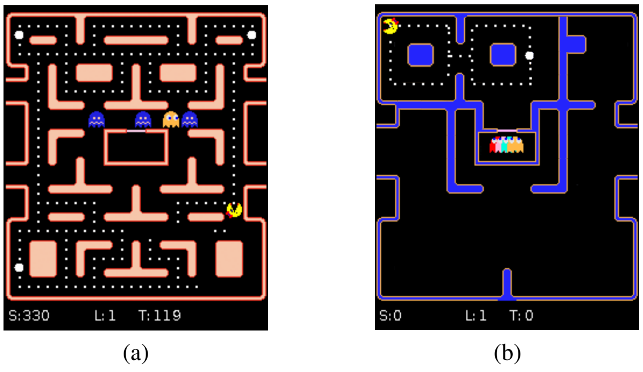


Figure 2: A visualization of the autonomously generated curriculum in the Block Dude domain. Arrows represent inter-group transfer, with the bold arrow showing transfer to the final target task. In this domain, the generated curriculum had 5 groups, each containing a single task.

pills, power-pills, and "scared" ghosts. We used the Ms. Pac-Man implementation described by Taylor *et al.* (2014) where the state space is represented by local features that are egocentric with respect to the agent's position. Learning was performed using the Q-learning algorithm with a linear function approximator. The set of source tasks used as input to our method consisted of 20 tasks which varied in maze type, number of pills, ghosts, and power-pills. Figure 3 (a) shows the final target task while (b) shows an example source task.

The output curriculum consisted of 9 tasks and was compared with a few handcrafted curricula designed by some of the authors, using the same task pool the generated curriculum was derived from. The methods for detecting convergence when learning source tasks used in Gridworld and Block Dude were not able to detect convergence in the non-deterministic Ms. Pac-Man domain. Instead, we employed a heuristic approach in which a source task was learned until the agent obtained at least a quarter of the maximum possible reward for that particular task. Figure 4 shows the learning curves of Q-learning agents when using the automated curriculum, the best handcrafted curriculum and the baseline condition.



(a) (b)

Figure 3: Tasks in the Ms. Pac-Man domain where (a) is the maze of the final target task and (b) is a maze of reduced dimensionality included in the generated curriculum.

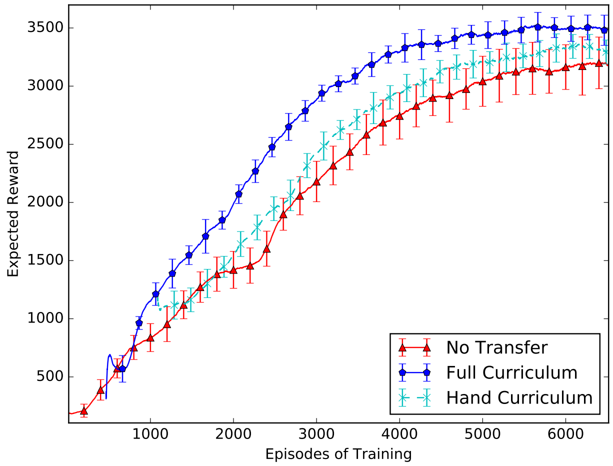


Figure 4: Expected reward (score) of learned policies in the Ms. Pac-Man domain. The policies shown are from a baseline agent, an agent adhering to a generated curriculum, and a curriculum made by a domain expert.

7 Conclusion and Future Work

We introduced a method to arrange a set of source tasks into a curriculum. The proposed method was evaluated on two discrete and one continuous RL domain. The results showed that following this curriculum in full produces the greatest learning benefit over any subset of the curriculum as well as learning the final target task directly. Furthermore, in one of the domains the generated curriculum outperformed a series of handcrafted ones designed by domain experts.

One limitation of our method is that it assumes a set of source tasks exists. In our experiments, the sets of source tasks were generated according to the heuristics proposed by Narvekar et al. (2016) but the process was not automated. In future work, we plan to integrate the proposed curriculum generation method with a methodology for automatically constructing potential source tasks.

8 Acknowledgments

This work has taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (CNS-1330072, CNS-1305287, IIS-1637736, IIS-1651089), ONR (21C184-01), and AFOSR (FA9550-14-1-0087). Peter Stone serves on the Board of Directors of, Cogitai, Inc. 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

Ammar, H. B.; Eaton, E.; Taylor, M. E.; Mocanu, D. C.; Driessens, K.; Weiss, G.; and Tuyls, K. 2014. An automated measure of MDP similarity for transfer in reinforcement learning. In *Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Asada, M.; Noda, S.; Tawaratsumida, S.; and Hosoda, K. 1996. Purposive behavior acquisition for a real robot by vision-based reinforcement learning. In *Recent Advances in Robot Learning*. Springer. 163–187.

Ferns, N.; Castro, P. S.; Precup, D.; and Panangaden, P. 2012. Methods for computing state similarity in Markov decision processes. *arXiv preprint arXiv:1206.6836*.

Ferns, N.; Panangaden, P.; and Precup, D. 2011. Bisimulation metrics for continuous Markov decision processes. *SIAM Journal on Computing* 40(6):1662–1714.

Isele, D.; Rostami, M.; and Eaton, E. 2016. Using task features for zero-shot knowledge transfer in lifelong learning. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*.

Konidaris, G., and Barto, A. 2006. Autonomous shaping: Knowledge transfer in reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, 489–496. ACM.

Lazaric, A.; Restelli, M.; and Bonarini, A. 2008. Transfer of samples in batch reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, 544–551. ACM.

Lazaric, A. 2012. Transfer in reinforcement learning: a framework and a survey. In *Reinforcement Learning*. Springer. 143–173.

MacGlashan, J. 2016. Brown-UMBC reinforcement learning and planning (BURLAP).

Narvekar, S.; Sinapov, J.; Leonetti, M.; and Stone, P. 2016. Source task creation for curriculum learning. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, 566–574. International Foundation for Autonomous Agents and Multiagent Systems.

Ng, A. Y.; Harada, D.; and Russell, S. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, 278–287.

Nguyen, T.; Silander, T.; and Leong, T. Y. 2012. Transferring expectations in model-based reinforcement learning. In *Advances in Neural Information Processing Systems*, 2555–2563.

Pan, S. J., and Yang, Q. 2010. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on* 22(10):1345–1359.

Peng, B.; MacGlashan, J.; Loftin, R.; Littman, M. L.; Roberts, D. L.; and Taylor, M. E. 2016. An empirical study of non-expert curriculum design for machine learners. In *Proceedings of the IJCAI Interactive Machine Learning Workshop*.

Perkins, T. J., and Precup, D. 1999. Using options for knowledge transfer in reinforcement learning. In *Technical Report*. University of Massachusetts.

Russell, S. J., and Norvig, P. 2003. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition.

Sinapov, J.; Narvekar, S.; Leonetti, M.; and Stone, P. 2015. Learning inter-task transferability in the absence of target task samples. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, 725–733. International Foundation for Autonomous Agents and Multiagent Systems.

Singh, S. P., and Sutton, R. S. 1996. Reinforcement learning with replacing eligibility traces. *Machine learning* 22(1-3):123–158.

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

Taylor, M. E.; Carboni, N.; Fachantidis, A.; Vlahavas, I.; and Torrey, L. 2014. Reinforcement learning agents providing advice in complex video games. *Connection Science* 26(1):45–63.

Watkins, C. J., and Dayan, P. 1992. Q-learning. *Machine learning* 8(3-4):279–292.

Wiewiora, E.; Cottrell, G.; and Elkan, C. 2003. Principled methods for advising reinforcement learning agents. In *ICML*, 792–799.