DCUR: Data Curriculum for Teaching via Samples with Reinforcement Learning

Anonymous Author(s)

Affiliation
Address
email

Abstract: Deep reinforcement learning (RL) has shown great empirical successes, but typically suffers from brittleness and sample inefficiency. A potential remedy is to use an already-trained policy as a source of supervision. In this work, we refer to these policies as teachers and study how their expertise can be transferred to other policies, referred to as students, by focusing on data usage. We propose a framework, Data Curriculum for Reinforcement learning (DCUR), which first trains teachers using online deep RL, and stores its logged environment interaction history. Then, students learn by running either offline RL or by using teacher data in combination with a small amount of self-generated data. DCUR’s central idea involves defining a class of data curricula which, as a function of training time, limits the student to sampling from a fixed subset of the full teacher data. We test teachers and students using state-of-the-art deep RL algorithms across a variety of data curricula. Results suggest that the choice of data curriculum significantly impacts student learning, and that it is beneficial to limit the data during early training stages while gradually letting the data availability grow over time. We identify when the student can learn offline from teacher data and match teacher performance without relying on specialized offline RL algorithms. Furthermore, we show that collecting a small fraction of online data can stabilize training in challenging environments and provides complementary benefits with the data curriculum. Supplementary material is available at https://sites.google.com/view/teach-curr/home.

Keywords: Curriculum Learning, Reinforcement Learning, Offline Data

1 Introduction

Humans often learn best when guided through a curricula. When providing expert demonstrations to human students, the demonstrations should fall within a particular range of difficulties. If they are too easy the student learns nothing, but if they are too difficult the student may have trouble learning [1, 2]. With this intuition, we consider the analogous context in Reinforcement Learning (RL) [3], and study how a teacher policy can best “teach” a student policy, where both are trained with reinforcement learning. In this work, we investigate when a teacher provides the student with a dataset of samples (i.e., data tuples) \( D^{(T)} \). For the student, rather than propose a new algorithm to learn from data, we use existing algorithms but focus on data usage. In particular, given a student employing a standard off-the-shelf RL algorithm, how can it better sample from \( D^{(T)} \)?

We propose a framework, DCUR (pronounced dee-curr): Data Curriculum for Reinforcement learning, where we study how to filter a teacher’s static dataset to accelerate a student’s learning progress, measured in terms of environment episodic reward. The framework is compatible with a variety of student learning settings, including pure offline reinforcement learning [4, 5, 6] and scenarios where the student can engage in a small amount of self-generated data, which we refer to as apprenticeship learning. Either case reduces the need for the student to engage in extensive and potentially risky exploratory behavior, and thus may hold tremendous promise for enabling robots to learn from existing, massive datasets. In experiments, we test the DCUR framework by training teachers in a standard fashion with online environment interaction. We store its logged history of

Submitted to the 5th Conference on Robot Learning (CoRL 2021). Do not distribute.
experienced environment interactions as a large dataset $D^{(T)}$ to be used by the student for learning. Results over a range of standard continuous control tasks [7, 8] suggest that students running off-the-shelf, off-policy RL algorithms can make use of curricula to more efficiently learn from static teacher data, even without the use of specialized offline RL algorithms. See Figure 1 for an overview of DCUR. In summary, the contributions of this paper include:

- We introduce the DCUR framework which considers data improvements, rather than algorithmic improvements, for accelerating learning from large, teacher-provided datasets.
- We show that the best curricula, combined with potentially longer training, can enable students training offline to match the teacher’s top performance.
- We show that students can use some online data (2.5-5.0% of the offline data size) in combination with data curricula to aid learning in more complex environments.

Supplementary material is available at [https://sites.google.com/view/teach-curr/home](https://sites.google.com/view/teach-curr/home).

2 Related Work

**Curriculum Learning.** The use of curriculum learning in machine learning dates to at least Elman et al. [9], who showed the benefit of initially training on “easier” samples while gradually increasing the difficulty. Subsequent work by Bengio et al. [10] confirmed these results by accelerating classification and language modeling by arranging samples in order of difficulty. Other work in curriculum learning has included training teacher agents to provide samples or loss functions to a student [11, 12], and larger-scale studies to investigate curricula for image classification [13]. In RL, curriculum learning has shown promise in multi-task [14, 15] and self-play [16] contexts, for selecting one of several teachers to provide samples [17], and for generating a curriculum of start [18] or goal [19, 20] states. In this work, we design a curriculum of samples (i.e., data tuples) for (mostly) offline RL, and we do not focus on the multi-task setting nor do we require self-play or goal generation. When students execute online steps, this can be interpreted as a form of apprenticeship learning [21] where the student can “practice” in addition to using offline data.

**Offline Reinforcement Learning.** Offline RL [6], also referred to as Batch RL [4, 5], has seen an explosion of recent interest. Offline RL is the special case of reinforcement learning [3] without exploration, so the agent must learn from a static dataset. It differs from imitation learning [22] in that data is annotated with rewards, which can be utilized by reinforcement learning algorithms to learn better policies than the underlying data generating policy. Many widely utilized Deep RL algorithms, such as DQN [23, 24] for discrete control and DDPG [25], SAC [26], and TD3 [27] for continuous control are off-policy algorithms and, in principle, capable of learning offline. In practice, however, researchers have found that such off-policy algorithms are highly susceptible to bootstrapping errors and thus may diverge quickly and perform poorly [28, 29, 30, 31, 32]. One remedy is to incorporate conservatism in reinforcement learning such as by regularizing the value functions in model-free RL settings [28, 31, 33, 34, 35, 36, 37]. Other studies have found promising results in model-based contexts [38, 39, 40]. The goals of this work are orthogonal to work that at-

![Figure 1: Visualization of DCUR. First, a teacher policy $\pi_{\theta_T}$ is trained using standard Deep RL with online environment interactions, and fills a replay buffer $D^{(T)}$ with all the experienced data tuples. After training $\pi_{\theta_T}$, we choose a fixed data curriculum $C(t)$ (see Figure 2) for training a student policy $\pi_{\theta_S}$ from $D^{(T)}$. In general, we study when students train offline, but since this can be challenging, we optionally allow students to gather some online data to use for learning, in addition to $D^{(T)}$. For each time $t$ (i.e., minibatch gradient update), the fixed curriculum strategy $C(t)$ restricts the samples the student can draw from $D^{(T)}$, resulting in a set of eligible data tuples (shaded in light gray) that the student can use for gradient updates.](image-url)
Reinforcement Learning with Demonstrations. Combining reinforcement learning with demonstrations is a highly effective technique for training students, particularly for hard exploration environments. In off-policy RL, demonstrations can be inserted into a replay buffer [41] and sampled for learning along with self-generated samples from a student. Examples of such algorithms in discrete control settings include DQfD [42], Ape-X DQfD [43], and R2D3 [44]. Other work has utilized demonstrations in continuous control by adding transitions to a replay buffer [45, 46, 47], specifying an auxiliary loss [48] or estimating value functions for model-based RL [49]. These works enable additional exploration from the student, allowing for self-generated samples, whereas we aim to understand how well students can learn with minimal exploration. Furthermore, these works often use a very low demo ratio, or the fraction of expert (teacher) demonstrations in a given minibatch. For example, R2D3 [44] reported that a demo ratio as small as 1/256 was ideal. That these algorithms perform best when utilizing so little teacher data motivates the need to understand how to use teacher data without requiring frequent student environment interaction.

3 Problem Statement and Preliminaries

We utilize the Markov Decision Process (MDP) framework for reinforcement learning (RL) [3]. An MDP is specified as a tuple $\langle S, A, P, R, \gamma \rangle$ where at each time step $t$, the agent is at state $s_t \in S$ and executes action $a_t \in A$. The environment dynamics map the state-action pair into a successor state $s_{t+1} \sim P(\cdot|s_t, a_t)$, and the agent receives a scalar reward signal $r_t = R(s_t, a_t)$. The objective is to find a policy $\pi : S \rightarrow A$ that maximizes the expected discounted return $E[\sum_{t=0}^{\infty} \gamma^t r_t]$ with discount $\gamma \in (0, 1]$. In Deep RL, the policy $\pi_\theta$ is parameterized by a deep neural network with parameters $\theta$.

The RL framework we study involves two agents: a student $S$ and a teacher $T$, following respective policies $\pi_{\theta_S}$ and $\pi_{\theta_T}$ with parameters $\theta_S$ and $\theta_T$. We assume the teacher is trained via standard online RL, and produces a dataset $D^{(T)} = \{(s_i, a_i, r_i, s_{i+1})\}_{i=1}^N$ of $N$ data tuples (also referred to as “samples”), where each contains a state $s_i$, action $a_i$, scalar reward $r_i$, and successor state $s_{i+1}$. In general, we use the $i$ subscript notation in $(s_i, a_i, r_i, s_{i+1})$ to specify a time indexing of the tuples across the full data, and use $(s, a, r, s')$ when precise time indexing is not needed. Data tuples from teacher data $D^{(T)}$ are provided to the student $S$, which we assume runs an off-policy RL algorithm, so that it can in principle learn from just the fixed data. We consider the problem of designing a curriculum to decide, for each time step $t$ of the student’s learning progress,1 which data tuples from $D^{(T)}$ should be “available” to the student when it performs minibatch gradient updates.

4 Method

4.1 Teachers and Data Generation

The DCUR framework is agnostic to the precise algorithm to train students and teachers. Unless stated otherwise, we generate teacher data $D^{(T)}$ using TD3 [27], a state-of-the-art off-policy Deep RL algorithm for continuous control. TD3 is an actor-critic algorithm where the actor $\pi_{\theta_S}$ is a policy, and the critic is a value function which consists of two Q-networks, $Q_\phi$, and $Q_{\phi_2}$, with target networks $Q_{\phi_1, \text{targ}}, Q_{\phi_2, \text{targ}}$. During gradient updates, TD3 mitigates overestimation of Q-values by taking the minimum of the two Q-networks to compute targets $y$ for the Bellman update:

$$y = r + \gamma \min_{i \in \{1, 2\}} Q_{\phi_i, \text{targ}}(s', a'(s'))$$ (1)

with discount factor $\gamma$, and where $a'$ is the action considered at the successor state $s'$:

$$a'(s') = \text{clip}(\pi_{\theta_S}(s') + \epsilon, a_{\text{low}}, a_{\text{high}}), \quad \epsilon \sim \text{clip}(\mathcal{N}(0, \sigma), -\beta, \beta)$$ (2)

which in practice involves adding zero-mean clipped Gaussian noise $\epsilon$ to $\pi_{\theta_S}(s')$ for some $\beta$, then clipping (again) component-wise to an environment-dependent action range $[a_{\text{low}}, a_{\text{high}}]$. For more

---

1In this work, we consider RL contexts where it is standard to have a 1:1 ratio of environment steps and gradient (i.e., training) updates, modulo any initial online data collection to partially fill in a replay buffer before training begins. We thus treat a “time step” $t$ as referring to environment steps and gradient updates interchangeably. If students learn offline, then a “time step” is interpreted as a gradient update only.
details on TD3, we refer the reader to Fujimoto et al. [27]. To generate \( D^{(T)} \), we use the \textit{logged environment interaction history} of TD3 from online training, resulting in a set of ordered tuples:

\[
D^{(T)} = \{(s_i, a_i, r_i, s_{i+1})\}_{i=0}^{N=1M},
\]

in a \textit{replay buffer}, where following standard MuJoCo training [27], the number of environment steps and the buffer capacity are both 1M, so no data tuples are overwritten. To our knowledge, the only prior work that has tested TD3 in offline RL with logged data is from Agarwal et al. [50], who report that TD3 outperformed Batch Constrained Q-learning [28] when learning from logged data generated from DDPG agents. We perform a deeper investigation of training on logged data by showing the utility of curricula (Section 4.2) and self-generated data (Section 4.3).

### 4.2 Data Curriculum for Reinforcement Learning (DCUR) Framework

We propose to accelerate student learning with a curriculum \( C(t) \) which specifies the range of eligible data tuples in the static teacher data \( D^{(T)} \) which can be sampled for the minibatch gradient update at time \( t \). We define two classes of curricula: \textit{additive} and \textit{scale}. An additive curriculum \( C_{\text{add}} \) uses two parameters, \( p \geq 0 \) and \( f \geq 0 \), specifying the \textit{previous} and \textit{future} data tuples in \( D^{(T)} \) relative to \( t \). For ease of notation, we omit the \( p \) parameter if the intent is to always allow data tuples from index 0, which represents the teacher’s earliest environment interaction. We thus denote the additive curriculum on \( D^{(T)} \) using one of two conventions:

\[
C_{\text{add}}(t; p, f) = \{(s_i, a_i, r_i, s_{i+1})\}_{i=\max(0, t-p)}^{t=\min(1M, t+f)}
\]

where the valid time indices are centered at the current student training time cursor \( t \), and are always limited by the buffer data size of \( |D^{(T)}| = 1M \) studied in this work. The second class of curricula \( C_{\text{scale}} \), is parameterized by a single \textit{scale} parameter \( c > 0 \) and defined as:

\[
C_{\text{scale}}(t; c) = \{(s_i, a_i, r_i, s_{i+1})\}_{i=0}^{t=\min(1M, t-c)}
\]

which enables index 0 up to index \( t \cdot c \). See Figure 2 for a visualization. If the student trains for 1M gradient updates following \( C_{\text{add}}(t; f = 1M) \), it has the full buffer accessible for sampling data tuples at all times. Using \( C_{\text{scale}}(t; c = 1.00) \) means the student can only sample from indices 0 up to \( t \) at time \( t \), so the available offline data grows over time. In addition, \( C_{\text{add}}(t; f = 0) \) and \( C_{\text{scale}}(t; c = 1.00) \) define the same data curriculum, and we default to using the latter notation.

In prior work, Fujimoto et al. [28] tested two special cases of these curricula, named \textit{final buffer} and \textit{concurrent}. The final buffer setting enables the entire data at all times for learning and was later studied in Agarwal et al. [50]. This is equivalent to \( C_{\text{add}}(t; f = 1M) \). The concurrent setting is represented as \( C_{\text{add}}(t; c = 1.00) \). We remark that that limiting the size of the buffer is a known option to stabilize online RL [51], but we aim to study this in an largely offline context in more complex environments, and where the eligible data buffer can grow over time.

![Figure 2](image-url)
4.3 Apprenticeship Learning: Using a Small Amount of Online Data

While we primarily study the student learning purely offline, we additionally explore learning with small amounts of on-policy student data, to see how much this stabilizes training and to also check that such effects are complementary with a data curriculum. This means the student forms a smaller buffer $\mathcal{D}^{(S)}$ of self-generated online data with exploration noise, and where $|\mathcal{D}^{(S)}| \ll |\mathcal{D}^{(T)}|$. We call this setting $X\%$ Online if the student, by the end of its training, has collected self-generated data tuples that amount to $X\%$ of the full teacher data (1M in this work). The student still performs 1M gradient updates, but takes one online step at equally spaced time intervals (i.e., one step every 100/X updates). For example, with “5% Online,” the student takes online environment steps 1 out of every 20 gradient updates, and $\mathcal{D}^{(S)}$ contains 50K data tuples at the end of training. This can equivalently be viewed as the student performing 50K consecutive online environment steps, but with 20 gradient updates between consecutive steps. The data tuples from $\mathcal{D}^{(S)}$ are never discarded. When the student samples a minibatch at time $t$, it first applies the pre-selected curriculum (see Section 4.2) to get the appropriate subset of $\mathcal{D}^{(T)}$, then combines the resulting eligible data tuples with all of $\mathcal{D}^{(S)}$, then uniformly samples from that.

5 Experiments

Environments. We test DCUR using standard MuJoCo environments [8] for Deep RL: Ant-v3, HalfCheetah-v3, Hopper-v3, and Walker2d-v3 from OpenAI [7]. Earlier versions of these environments (-v1 or -v2) have also been benchmarked in work focusing on offline RL [28, 31, 36].

Teachers and Students. We train TD3 teachers using the standard 1M online steps [27, 26, 36], and store all encountered data tuples $(s, a, r, s')$ to make $\mathcal{D}^{(T)}$. Each training epoch consists of 4000 time steps, so there are 250 training epochs for teachers and students. In Section 6.2 we investigate training students for 10X more epochs. We apply standard noise levels for the teacher’s exploration; the noise added to actions is $\mathcal{N}(0, 0.1)$ instead of $\mathcal{N}(0, 0.5)$ as done in some experiments in [50, 28] to increase data diversity. In the Appendix, we have results from SAC [26] teachers and students.

The code we use is built on top of SpinningUp [52] from OpenAI.

Data Curriculum Experiments. We test additive and scale curricula from Section 4.2. For additive curricula, we adjust the “forward” samples allowed: $f \in \{50K, 100K, 200K, 500K, 1M\}$, and we also check if ignoring older samples in $\mathcal{D}^{(T)}$ helps (with $p = 800K$). We report scale curricula with $c \in \{0.50, 0.75, 1.00, 1.10, 1.25\}$, where $c < 1$ tests whether the student needs more gradient updates on data tuples in $\mathcal{D}^{(T)}$ relative to the teacher, and $c > 1$ tests whether it helps to have additional “forward” samples, which is similar to $f > 0$ in additive curricula, but where $f$ increases throughout student training because the maximum eligible index $c \cdot t$ is a function of $t$. Due to computational limitations, for most experiments after Section 6.1, we test two particular curricula (from [28]): all the data ($C_{add}(t; f = 1M)$) or concurrent data ($C_{scale}(t; c = 1.00)$). We test with $C_{add}(t; f = 1M)$ because it serves a baseline of using all data, which can intuitively be viewed as using no curriculum. As Deep RL evaluations are notoriously noisy [53], all experiments are reproducible from code available on the project website.

Evaluation. To evaluate student and teacher performance, we use two metrics, “M1” and “M2”:

- **M1 (Final):** the average reward of the student’s final 100 test-time episodes.
- **M2 (Average):** the average reward across all student test-time episodes.

In all experiments, students and teachers do 10 test episodes after every epoch. We use 5 random seeds for students, with respect to one teacher (per environment), to reduce the source of variability that would result from different teachers. We compute M1 and M2 statistics for each of the 5 runs, then average those 5 to get final numbers for M1 and M2. In the tables, when comparing a relevant set of students, we bold the best M1 and M2 results, and additionally bold other results with overlapping standard errors for a fairer comparison; see the Appendix for the exact formula.

6 Results

We present experimental results of DCUR under various settings, and report M1 and M2 metrics. Due to space constraints, in all following subsections, we defer numerous results to the Appendix.
6.2 Data Curriculum for Training 10X Longer

Here, as in Section 6.1, we run the $C_{add}(t; f = 1M)$ and $C_{scale}(t; c = 1.00)$ curricula again, but train 10X longer to understand how this affects learning. The choice of curriculum will only affect the first 1/10 of training (i.e., the first 250 epochs), since after that, both curricula reduce to the

\begin{table}[h]
\centering
\begin{tabular}{l|cccc|cccc}
\hline
Data Curriculum & \multicolumn{4}{c|}{Ant-v3} & \multicolumn{4}{c}{HalfCheetah-v3} \\
& M1 & M2 & M1 & M2 & M1 & M2 & M1 & M2 \\
\hline
$C_{add}(t; f = 50K)$ & 219.9 & 382.0 & 8067.3 & 7147.0 & 2986.2 & 1669.5 & 2715.6 & 1712.1 \\
$C_{add}(t; f = 100K)$ & -604.2 & -409.6 & 7416.2 & 7052.6 & 2627.4 & 1980.1 & 2746.7 & 1810.5 \\
$C_{add}(t; f = 200K)$ & 288.4 & -620.4 & 8028.4 & 6521.5 & 2525.3 & 1887.9 & 2691.8 & 1651.9 \\
$C_{add}(t; f = 500K)$ & 283.3 & -801.1 & 7108.7 & 5041.2 & 2498.0 & 1546.2 & 1517.7 & 1183.5 \\
$C_{add}(t; f = 1M)$ & -1552.1 & -1390.5 & 7447.3 & 3846.8 & 2323.0 & 1610.3 & 1741.4 & 1096.4 \\
$C_{add}(t; p = 800K, f = 0)$ & -889.7 & 1497.9 & 7650.4 & 6942.7 & 2132.2 & 1924.4 & 792.0 & 1521.8 \\
\hline
$C_{scale}(t; c = 0.50)$ & 285.0 & 301.4 & 7467.8 & 6261.9 & 2332.3 & 1450.5 & 1866.0 & 799.9 \\
$C_{scale}(t; c = 0.75)$ & 167.0 & 412.2 & 7392.6 & 6689.2 & 3284.7 & 1818.2 & 1864.9 & 1251.5 \\
$C_{scale}(t; c = 1.00)$ & 825.6 & 1103.7 & 8305.3 & 6980.0 & 2984.3 & 2092.6 & 2178.5 & 1305.7 \\
$C_{scale}(t; c = 1.10)$ & 2952.5 & 2212.0 & 8306.0 & 7095.6 & 3185.2 & 2317.1 & 2423.9 & 1698.1 \\
$C_{scale}(t; c = 1.25)$ & -1851.1 & 199.6 & 7843.8 & 7175.4 & 2755.5 & 1891.9 & 2839.4 & 1747.4 \\
\hline
\end{tabular}
\caption{Offline RL Results with DCUR. Student performance as a function of 11 data curricula, with teacher performance (bottom row) as a reference. We report the “M1” and “M2” evaluation metrics (Section 5). We run 1 teacher per environment, then use 5 random seeds for training students from that same teacher data. Hence, all numbers reported below for student data curriculum experiments are averages over 5 independent offline RL runs; standard error values are in the Appendix. For each column, values are bolded for the best M1 and M2 results among all 11 curricula for students, and for other students with overlapping standard errors.}
\end{table}

Figure 3: Offline student performance on HalfCheetah-v3 based on additive curricula (left) and scale curricula (right); these experiments correspond to numerical values in Table 1. In both subplots, the teacher’s performance for reference (at test time, without noise) is shown in the dashed black line. Results from additive curricula suggest a clear pattern that access to more samples initially (i.e., larger $f$) slows learning.

6.1 Effect of Data Curricula on Student Training Offline

We train students offline using 11 data curricula and list results in Table 1. The overall results suggest that a curriculum allowing for the available samples from $D^{(T)}$ to grow over time, and to include a few samples “ahead” relative to the student’s training time $t$ produces stronger results. In particular, $C_{scale}(t; c = 1.10)$ has the best results, as it obtains the top scores or close to it (i.e., with overlapping standard error) on 7 out of the 8 columns in Table 1. The next best curriculum, with 4 out of 8 top scores, is $C_{add}(t; f = 50K)$, and it similarly allows the available range of samples to go slightly past $t$, in this case by a fixed $t + 50K$. In contrast, allowing too much data with $C_{add}(t; f = 1M)$ or $C_{add}(t; f = 500K)$ results in poor performance, with neither of these curricula ranking among the best in any of the environments with respect to either metric. Ignoring older samples from the teacher’s early training history with $C_{add}(t; p = 800K, f = 0)$ also exhibits weak performance. See Figure 3 for a representative set of learning curves for HalfCheetah-v3, showing some curricula that result in the student essentially matching teacher performance, despite known challenges associated with pure offline learning, even with relying on concurrent-style training [28].

6.2 Data Curriculum for Training 10X Longer

Here, as in Section 6.1, we run the $C_{add}(t; f = 1M)$ and $C_{scale}(t; c = 1.00)$ curricula again, but train 10X longer to understand how this affects learning. The choice of curriculum will only affect the first 1/10 of training (i.e., the first 250 epochs), since after that, both curricula reduce to the
Figure 4: Hopper-v3 test-time episodic returns comparing the teacher performance (black curve) over its training period of 250 epochs (dashed vertical line), versus students trained offline over 2500 (10X more) epochs. For the two data curricula, we train 5 independent students from the fixed teacher data $D(T)$ with different seeds. Results suggest that the $C_{\text{scale}}(t; c = 1.00)$ curriculum (blue curve) can lead to better average performance over 2500 epochs, and that it matches the teacher’s best performance (dashed horizontal line).

Table 2: Offline RL Results with DCUR, 10X Training. Student performance as a function of the data curriculum, where the students train for 10X longer (2500 epochs) as compared to student results in Table 1. Besides this one change, the setup and table formatting is identical to Table 1. See Section 6.2 for details. The teacher metrics are repeated for reference, and are not considered when bolding numbers in the table.

<table>
<thead>
<tr>
<th>Curriculum</th>
<th>Ant-v3</th>
<th>HalfCheetah-v3</th>
<th>Hopper-v3</th>
<th>Walker2d-v3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M1</td>
<td>M2</td>
<td>M1</td>
<td>M2</td>
</tr>
<tr>
<td>$C_{\text{add}}(t; f = 1M)$</td>
<td>-2770.0</td>
<td>-2088.3</td>
<td>6041.6</td>
<td>6847.9</td>
</tr>
<tr>
<td>$C_{\text{scale}}(t; c = 1.00)$</td>
<td>-2780.6</td>
<td>-1511.2</td>
<td>6496.2</td>
<td>7555.5</td>
</tr>
<tr>
<td>TD3 Teacher</td>
<td>4876.2</td>
<td>3975.8</td>
<td>8573.6</td>
<td>7285.5</td>
</tr>
</tbody>
</table>

student sampling data tuples from the entire teacher buffer $D(T)$. To make results comparable with those in Section 6.1, we use the same teacher buffers. Table 2 has results in a similar manner as Table 1. We find that, somewhat surprisingly, the curricula affects the long-term performance of the student in the remaining 9/10 of training. Across all 8 columns (4 environments and the M1/M2 metrics), using $C_{\text{scale}}(t; c = 1.00)$ outperforms $C_{\text{add}}(t; f = 1M)$ or is statistically similar to it based on standard errors, suggesting that the initial curriculum assists TD3 in finding a stable set of policy and value functions so that it can continue training without significant deterioration.

Figure 4 shows a representative set of learning curves for Hopper-v3, which shows the student with $C_{\text{scale}}(t; c = 1.00)$ matching the teacher’s performance given sufficient training.

Figure 5: Ant-v3 test-time performance for students with various amounts of online data, ranging from 0% (i.e., offline) to 10% online. We show the teacher curve (dashed black line) for reference. We plot results from two curricula, $C_{\text{add}}(t; f = 1M)$ to the left, and $C_{\text{scale}}(t; c = 1.00)$ to the right.

6.3 Apprenticeship Learning: Results with Small Amounts of Online Data

We next study when students can use a small amount of online data to address challenges with pure offline learning. We train students for 250 epochs using 2.0%, 5.0% and 10.0% online data collection, i.e., at the end of 1M training steps, students get 25K, 50K, and 100K self-generated data tuples, respectively, for $D(S)$, which they can sample from in addition to the (filtered) data tuples from $D(T)$. Table 3 has a complete overview of the M1 and M2 results across all 4 environments,
Table 3: DCUR Results with Online Data. Performance of student learning based on one of two data curriculum choices and the addition of a small fraction of online data. At the end of 250 training epochs, the student has collected self-generated, online data that constitutes 2.5%, 5.0%, or 10.0% of the original teacher data size \( \vert D^{(T)} \vert = 1M \). We use the same teacher data as in Table 1 and report the M1 and M2 metrics. We bold values by comparing only the two curricula with the same X% online experiments per column and bolding the maximum only (if non-overlapping standard error) or both (if otherwise).

<table>
<thead>
<tr>
<th>Curriculum; % Online</th>
<th>Ant-v3</th>
<th>HalfCheetah-v3</th>
<th>Hopper-v3</th>
<th>Walker2d-v3</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C_{add}(t; f = 1M); 2.5% )</td>
<td>3093.1</td>
<td>894.4</td>
<td>8581.7</td>
<td>5275.3</td>
</tr>
<tr>
<td>( C_{scale}(t; c = 1.00); 2.5% )</td>
<td>4044.7</td>
<td>2857.0</td>
<td>8417.9</td>
<td>7212.2</td>
</tr>
<tr>
<td>( C_{add}(t; f = 1M); 5.0% )</td>
<td>3693.9</td>
<td>1556.6</td>
<td>8658.8</td>
<td>5634.6</td>
</tr>
<tr>
<td>( C_{scale}(t; c = 1.00); 5.0% )</td>
<td>3251.1</td>
<td>3068.2</td>
<td>8601.4</td>
<td>7274.0</td>
</tr>
<tr>
<td>( C_{add}(t; f = 1M); 10.0% )</td>
<td>4229.0</td>
<td>2007.6</td>
<td>8864.2</td>
<td>6024.5</td>
</tr>
<tr>
<td>( C_{scale}(t; c = 1.00); 10.0% )</td>
<td>4510.9</td>
<td>3349.0</td>
<td>8608.5</td>
<td>7290.5</td>
</tr>
<tr>
<td>TD3 Teacher</td>
<td>4876.2</td>
<td>3975.8</td>
<td>8573.6</td>
<td>7285.5</td>
</tr>
</tbody>
</table>

Figure 6: Students’ estimated Q-values in HalfCheetah-v3. With more data available at the start of training (i.e., increasing \( f \), shown left to right) this creates initial over-estimation of Q-values. All 5 subplots are derived from students reported in Table 1, and all curves average over the 5 random seeds and show standard errors.

and the Appendix contains further details. The results indicate that even with as little as 2.5% online data, students are able to significantly improve versus offline learning, with the improvement most notable in the complex Ant-v3 environment as shown in Figure 5. Furthermore, \( C_{scale}(t; c = 1.00) \) continues to provide some benefit to learning speed compared to \( C_{add}(t; f = 1M) \) with online data, particularly with respect to M2.

6.4 Investigation of Q-Values

To understand why the data curriculum matters, we investigate the student’s estimated Q-values. With too much data available from \( D^{(T)} \) during early stages of training, this causes overestimation of Q-values, whereas a curriculum that restricts data tuples results in more stable, monotonically increasing Q-values. Figure 6 shows different additive curricula on HalfCheetah-v3 and plots the estimated Q-values. A similar trend holds for the other environments.

7 Conclusion and Future Work

This work introduces the DCUR framework, which studies how to filter a given dataset for existing reinforcement learning algorithms. Results across a variety of curricula and training settings suggest that the choice significantly impacts the learning speed of students running RL. The greatest benefits come from curricula that strictly limit the availability of data tuples at the start, but gradually lets the available data grow throughout training. We caution that the results presented are contingent on the way we generated the datasets, i.e., the logged training history of an RL algorithm. In future work, we will test other datasets and environments, such as those from D4RL [54] or RL Unplugged [55]. We will explore learning from multiple teachers [56]. While we kept relevant experience replay hyperparameters such as the \( \text{replay ratio} \) constant, we will make use of findings from recent research [57] to further understand the interplay between experience replay and data curriculum strategies. Finally, we plan to devise and test more sophisticated data curricula.
References


