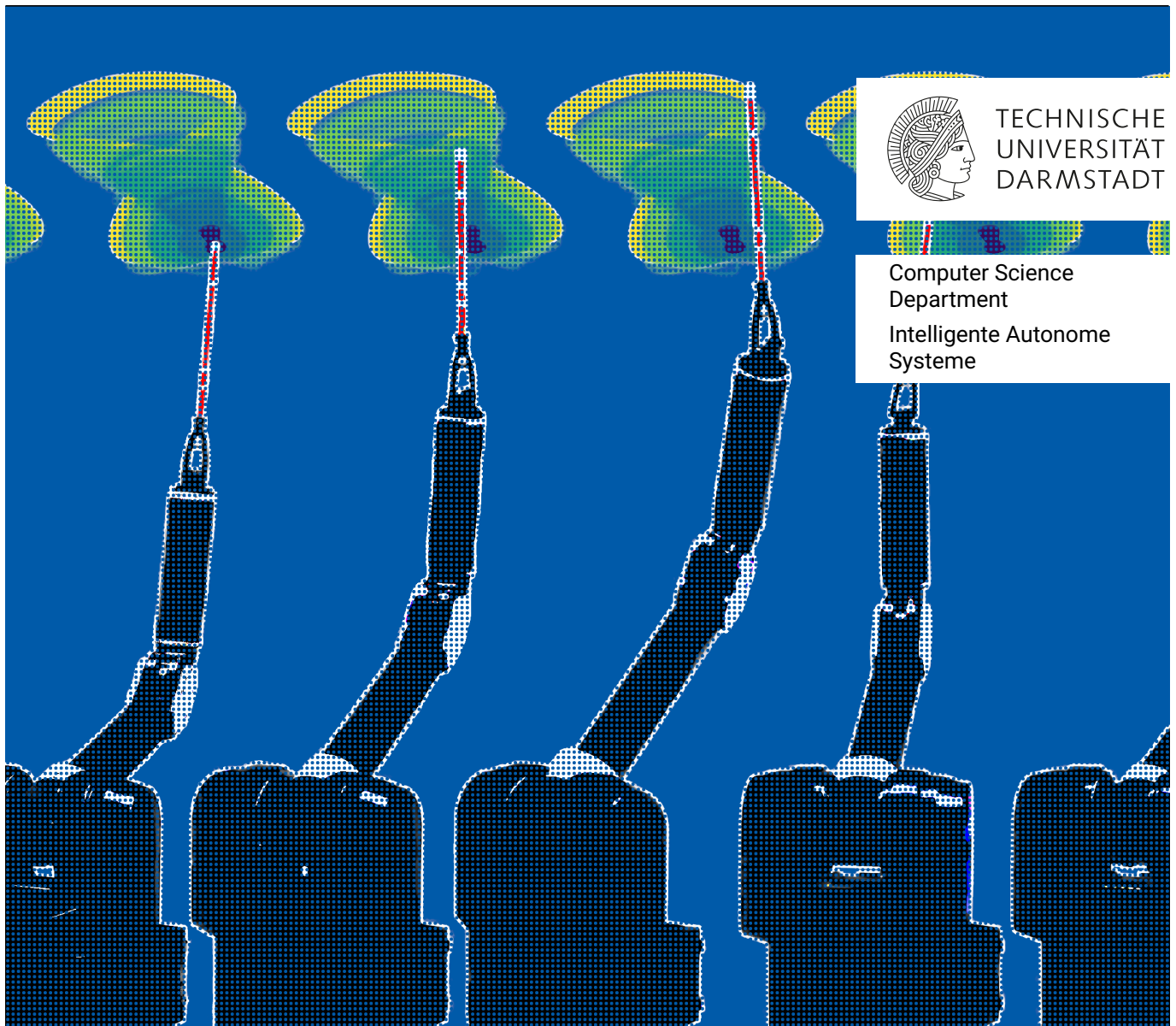


Reinforcement Learning Curricula as Interpolations between Task Distributions

Zur Erlangung des akademischen Grades Doktor-Ingenieur (Dr.-Ing.)
Vorgelegte Dissertation von Pascal Klink aus Weinheim
Tag der Einreichung: 22.09.2023, Tag der Prüfung: 07.11.2023

1. Gutachten: Prof. Jan Peters, Ph. D.
2. Gutachten: Prof. Peter Stone, Ph. D.
3. Gutachten: Asst. Prof. Dr. Joni Pajarinen
Darmstadt – D17



Reinforcement Learning Curricula as Interpolations between Task Distributions

Submitted doctoral thesis by Pascal Klink

Date of submission: 22.09.2023

Date of thesis defense: 07.11.2023

Darmstadt – D17

Bitte zitieren Sie dieses Dokument als:

URN: urn:nbn:de:tuda-tuprints-247829

URL: <http://tuprints.ulb.tu-darmstadt.de/24782>

Dieses Dokument wird bereitgestellt von tuprints,

E-Publishing-Service der TU Darmstadt

<http://tuprints.ulb.tu-darmstadt.de>

tuprints@ulb.tu-darmstadt.de

Die Veröffentlichung steht unter folgender Creative Commons Lizenz:

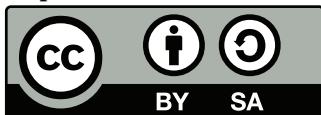
Namensnennung – Weitergabe unter gleichen Bedingungen 4.0 International

<https://creativecommons.org/licenses/by-sa/4.0/>

This work is licensed under a Creative Commons License:

Attribution–ShareAlike 4.0 International

<https://creativecommons.org/licenses/by-sa/4.0/>



Erklärungen laut Promotionsordnung

§ 8 Abs. 1 lit. c PromO

Ich versichere hiermit, dass die elektronische Version meiner Dissertation mit der schriftlichen Version übereinstimmt.

§ 8 Abs. 1 lit. d PromO

Ich versichere hiermit, dass zu einem vorherigen Zeitpunkt noch keine Promotion versucht wurde. In diesem Fall sind nähere Angaben über Zeitpunkt, Hochschule, Dissertationsthema und Ergebnis dieses Versuchs mitzuteilen.


§ 9 Abs. 1 PromO

Ich versichere hiermit, dass die vorliegende Dissertation selbstständig und nur unter Verwendung der angegebenen Quellen verfasst wurde.

§ 9 Abs. 2 PromO

Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Darmstadt, 22.09.2023



P. Klink

Abstract

In the last decade, the increased availability of powerful computing machinery has led to an increasingly widespread application of machine learning methods. Machine learning has been particularly successful when large models, typically neural networks with an ever-increasing number of parameters, can leverage vast data to make predictions.

While *reinforcement learning (RL)* has been no exception from this development, a distinguishing feature of RL is its well-known exploration-exploitation trade-off, whose optimal solution – while possible to model as a partially observable Markov decision process – evades computation in all but the simplest problems. Consequently, it seems unsurprising that notable demonstrations of reinforcement learning, such as an RL-based Go agent (AlphaGo) by Deepmind beating the professional Go player Lee Sedol, relied both on the availability of massive computing capabilities and specific forms of regularization that facilitate learning. In the case of AlphaGo, this regularization came in the form of self-play, enabling learning by interacting with gradually more proficient opponents.

In this thesis, we develop techniques that, similarly to the concept of self-play of AlphaGo, improve the learning performance of RL agents by training on sequences of increasingly complex tasks. These task sequences are typically called *curricula* and are known to side-step problems such as slow learning or convergence to poor behavior that may occur when directly learning in complicated tasks. The algorithms we develop in this thesis create curricula by minimizing distances or divergences between probability *distributions* of learning tasks, generating *interpolations* between an initial distribution of easy learning tasks and a target task distribution. Apart from improving the learning performance of RL agents in experiments, developing methods that realize curricula as interpolations between task distributions results in a nuanced picture of key aspects of successful reinforcement learning curricula.

In Chapter 1, we start this thesis by introducing required reinforcement learning notation and then motivating curriculum reinforcement learning from the perspective of continuation methods for non-linear optimization. Similar to curricula for reinforcement

learning agents, continuation methods have been used in non-linear optimization to solve challenging optimization problems. This similarity provides an intuition about the effect of the curricula we aim to generate and their limits.

In Chapter 2, we transfer the concept of self-paced learning, initially proposed in the supervised learning community, to the problem of RL, showing that an automated curriculum generation for RL agents can be motivated by a regularized RL objective. This regularized RL objective implies generating a curriculum as a sequence of task distributions that trade off the expected agent performance against similarity to a specified distribution of target tasks. This view on curriculum RL contrasts existing approaches, as it motivates curricula via a regularized RL objective instead of generating them from a set of assumptions about an optimal curriculum. In experiments, we show that an approximate implementation of the aforementioned curriculum – that restricts the interpolating task distribution to a Gaussian – results in improved learning performance compared to regular reinforcement learning, matching or surpassing the performance of existing curriculum-based methods.

Subsequently, Chapter 3 builds up on the intuition of curricula as sequences of interpolating task distributions established in Chapter 2. Motivated by using more flexible task distribution representations, we show how parametric assumptions play a crucial role in the empirical success of the previous approach and subsequently uncover key ingredients that enable the generation of meaningful curricula without assuming a parametric model of the task distributions. One major ingredient is an explicit notion of task similarity via a distance function of two Markov Decision Processes. We turn towards optimal transport theory, allowing for flexible particle-based representations of the task distributions while properly considering the newly introduced metric structure of the task space. Combined with other improvements to our first method, such as a more aggressive restriction of the curriculum to tasks that are not too hard for the agent, the resulting approach delivers consistently high learning performance in multiple experiments.

In the final Chapter 4, we apply the refined method of Chapter 3 to a trajectory-tracking task, in which we task an RL agent to follow a three-dimensional reference trajectory with the tip of an inverted pendulum mounted on a Barrett Whole Arm Manipulator. The access to only positional information results in a partially observable system that, paired with its inherent instability, underactuation, and non-trivial kinematic structure, presents a challenge for modern reinforcement learning algorithms, which we tackle via curricula. The technically infinite-dimensional task space of target trajectories allows us to probe the developed curriculum learning method for flaws that have not surfaced in the rather low-dimensional experiments of the previous chapters. Through an improved

optimization scheme that better respects the non-Euclidean structure of target trajectories, we reliably generate curricula of trajectories to be tracked, resulting in faster and more robust learning compared to an RL baseline that does not exploit this form of structured learning. The learned policy matches the performance of an optimal control baseline on the real system, demonstrating the potential of curriculum RL to learn state estimation and control for non-linear tracking tasks jointly.

In summary, this thesis introduces a perspective on reinforcement learning curricula as interpolations between task distributions. The methods developed under this perspective enjoy a precise formulation as optimization problems and deliver empirical benefits throughout experiments. Building upon this precise formulation may allow future work to advance the formal understanding of reinforcement learning curricula and, with that, enable the solution of challenging decision-making and control problems with reinforcement learning.

Kurzfassung

In den letzten zehn Jahren hat die zunehmende Verfügbarkeit leistungsstarker Computer zu einer immer breiteren Anwendung von Methoden des maschinellen Lernens geführt. Das maschinelle Lernen ist besonders erfolgreich, wenn große Modelle, in der Regel neuronale Netze mit einer immer größer werdenden Anzahl von Parametern, große Datenmengen nutzen können, um Vorhersagen zu treffen.

Während das Verstärkungslernen (Reinforcement Learning, RL) keine Ausnahme von dieser Entwicklung darstellt, ist der bekannte “Exploration-Exploitation Trade-Off” ein entscheidendes Merkmal von RL, dessen optimale Lösung - obwohl sie als teilweise beobachtbarer Markov-Entscheidungsprozess modelliert werden kann - sich der Berechnung bei allen außer den einfachsten Problemen entzieht. Daher scheint es nicht überraschend, dass bemerkenswerte Demonstrationen des Reinforcement Learnings, wie z.B. der RL-basierte Go-Agent AlphaGo von Deepmind, der den professionellen Go-Spieler Lee Sedol besiegte, sowohl auf die Verfügbarkeit massiver Rechenkapazitäten als auch auf spezifische Formen der Regularisierung, die das Lernen erleichtern, angewiesen war. Im Fall von AlphaGo kam diese Regularisierung in Form von Selbstspiel, welches das Lernen durch Interaktion mit allmählich besser werdenden Gegnern ermöglicht.

In dieser Arbeit entwickeln wir Techniken, die, ähnlich wie das Konzept des Selbstspiels von AlphaGo, die Lernleistung von RL-Agenten durch das Training auf Sequenzen von zunehmend komplexen Aufgaben verbessern. Diese Aufgabensequenzen werden typischerweise als Curricula bezeichnet und sind dafür bekannt, Probleme wie langsames Lernen oder Konvergenz zu schlechtem Verhalten zu umgehen, die beim direkten Lernen in komplizierten Aufgaben auftreten können. Die Algorithmen, die wir in dieser Arbeit entwickeln, erstellen Curricula, indem sie Abstände oder Divergenzen zwischen Wahrscheinlichkeitsverteilungen von Lernaufgaben minimieren. Dabei erzeugen sie Interpolationen zwischen einer Anfangsverteilung von einfachen Lernaufgaben und einer Zielaufgabenverteilung. Neben der Verbesserung der Lernleistung von RL-Agenten in Experimenten führt die Entwicklung von Methoden, die Curricula als Interpolationen zwischen Aufgabenverteilungen realisieren, zu einem differenzierteren Bild von Schlüsselaspekten erfolgreicher Reinforcement Learning Curricula.

In Kapitel 1 beginnen wir diese Arbeit mit einer Einführung in die erforderliche Notation des Reinforcement Learnings und motivieren dann das Reinforcement Learning mit Curricula aus der Perspektive von Continuation Methods in der nichtlinearen Optimierung. Ähnlich wie bei Curricula für das Verstärkungslernen werden Continuation Methods in der nichtlinearen Optimierung verwendet, um schwierige Optimierungsprobleme zu lösen. Diese Ähnlichkeit vermittelt eine Intuition über den Effekt und die Grenzen der Curricula, die wir erstellen wollen.

In Kapitel 2 übertragen wir das Konzept des Self-Paced Learnings, das ursprünglich im Bereich des Supervised Learnings vorgeschlagen wurde, auf das Problem des RL, indem wir zeigen, dass eine automatisierte Curriculumerstellung für RL-Agenten durch ein regularisiertes RL Problem motiviert werden kann. Dieses regularisierte RL Problem impliziert die Generierung eines Curriculums als eine Sequenz von Aufgabenverteilungen, die die erwartete Agentenleistung gegen die Ähnlichkeit mit einer spezifizierten Verteilung von Zielaufgaben abwägen. Diese Sichtweise auf das Curriculum Reinforcement Learning steht im Gegensatz zu bestehenden Ansätzen, da sie Curricula über ein regularisiertes RL Problem motiviert, anstatt diese aus einer Reihe von Annahmen über einen optimalen Lehrplan zu generieren. In Experimenten zeigen wir, dass eine approximative Implementierung des oben erwähnten Curriculums - die die interpolierende Aufgabenverteilung auf eine Gauß-Verteilung beschränkt - zu einer verbesserten Lernleistung im Vergleich zu regulärem Reinforcement Learning führt und die Leistung bestehender curriculum-basierter Methoden erreicht oder übertrifft.

Anschließend baut Kapitel 3 auf der in Kapitel 2 entwickelten Intuition von Lehrplänen als Sequenzen interpolierender Aufgabenverteilungen auf. Motiviert durch die Verwendung flexiblerer Repräsentationen von Aufgabenverteilungen zeigen wir, wie parametrische Annahmen eine entscheidende Rolle für den empirischen Erfolg des vorherigen Ansatzes spielen, und decken anschließend Schlüsselbestandteile für die Erstellung sinnvoller Curricula ohne Annahmen über parametrische Modelle der Aufgabenverteilungen auf. Ein wichtiger Bestandteil ist ein expliziter Begriff der Aufgabenähnlichkeit über eine Distanzfunktion zweier Markov-Entscheidungsprozesse. Wir wenden uns der Theorie des optimalen Transports zu, die flexible partikelbasierte Darstellungen der Aufgabenverteilungen ermöglicht und gleichzeitig die neu eingeführte metrische Struktur des Aufgabenraums angemessen berücksichtigt. In Kombination mit anderen Verbesserungen unserer ersten Methode, wie einer aggressiveren Beschränkung des Lehrplans auf Aufgaben, die für den Agenten nicht zu schwer sind, liefert der resultierende Ansatz in mehreren Experimenten eine konstant hohe Lernleistung.

Im abschließenden Kapitel 4 wenden wir die verfeinerte Methode aus Kapitel 3 auf eine Bahnverfolgungsaufgabe an, bei der wir einen RL-Agenten damit beauftragen, einer dreidimensionalen Referenztrajektorie mit der Spitze eines, auf einem Barrett-Wholearm-Manipulator montierten, sphärischen Pendels zu folgen. Die partielle Beobachtbarkeit des Systems durch das Fehlen von Geschwindigkeitsinformationen, gepaart mit einem inhärent instabilen, unteraktuierten System und einer nicht-trivialen Kinematik, stellt eine Herausforderung für moderne Reinforcement-Learning-Algorithmen dar, die wir mit Hilfe von Curricula angehen. Der technisch unendlich-dimensionale Aufgabenraum der Zieltrajektorien erlaubt es uns, die entwickelte Curriculum-Lernmethode auf Schwächen zu untersuchen, die in den eher niedrig-dimensionalen Experimenten der vorherigen Kapitel nicht auftraten. Durch ein verbessertes Optimierungsschema, das die nicht-euklidische Struktur der Zieltrajektorien besser berücksichtigt, generieren wir zuverlässig Curricula über zu verfolgende Trajektorien. Das Training auf diesen Trajektorien führt zu schnellerem und robusterem Lernen als direktes Training auf den Zieltrajektorien. Das erlernte Verhalten erreicht auf dem realen System die Leistung eines Reglers, der mit Hilfe von Ansätzen der Optimalen Steuerung entworfen wurde, was das Potenzial von Curriculum-RL für das gemeinsame Erlernen von Zustandsschätzung und Regelung für nichtlineare Bahnverfolgungsaufgaben demonstriert.

Zusammenfassend führt diese Arbeit eine Perspektive auf Reinforcement Learning Curricula als Interpolationen zwischen Aufgabenverteilungen ein. Die unter dieser Sichtweise entwickelten Methoden genießen eine präzise Formulierung als Optimierungsprobleme und liefern in Experimenten empirische Vorteile. Aufbauend auf dieser präzisen Formulierung können zukünftige Arbeiten das formale Verständnis von Reinforcement Learning Curricula vorantreiben und damit die Lösung von anspruchsvollen Entscheidungs- und Kontrollproblemen mit Reinforcement Learning ermöglichen.

Acknowledgement

Looking back at the past years, I want to thank those people who were a part of my professional and private life throughout this time and have, knowingly or not, contributed to the appearance of this thesis.

Without the continued support of my supervisors *Joni Pajarinen* and *Jan Peters* and their tremendous experience and valuable advice, I would not have been able to explore my ideas and, more importantly, carry on with them even in the face of challenges and doubts. Only through the thorough work of my thesis committee this document becomes more than a collection of research results. Therefore, I additionally thank *Peter Stone* for evaluating my thesis and *Max Mühlhäuser*, *Oskar von Stryk*, and *Kristian Kersting* for their work in the committee.

The past years would have been even more intimidating and challenging without knowing a secure place of retreat. Consequently, I am forever indebted to my parents, *Anette* and *Thomas Klink*, my whole family, my love, *Marlene Hecht*, and my friends for guiding and accompanying me to and during this stage of my life.

Apart from my supervisors, my colleagues were essential in creating an inspiring and friendly environment in which to enjoy my time. I want to thank *Hany Abdulsamad* and *Boris Belousov* for their excellent supervision throughout my Master's Thesis, which was an essential part of my decision to continue my path in academia, and their continued support throughout my years at IAS. Together with *Fabio Muratore*, *João Carvalho*, *Joseph Watson*, *Michael Lutter*, *Samuele Tosatto*, *Svenja Stark*, and the rest of the IAS lab, they made my days in the office worthwhile, regardless of the success or failure of experiments and submissions.

Having learned invaluable lessons during my internship outside of the IAS lab, I want to thank the team at Amazon Robotics in Berlin, led by *Alexander Melkozerov* and *Can Erdogan*, for the great time I enjoyed there. My mentor, *Kiru Park*, and the whole team made this time unforgettable.

Apart from my collaborators, who all except for *Carlo D'Eramo*, *Kai Ploeger*, *Peter Nickl*, and *Tuan Dam* have been named above, I want to thank my students for their insightful work and discussions. I want to particularly thank *Haoyi Yang* and *Florian Wolf*, with whom I worked together on those papers that resulted in Chapters 3 and 4 of this thesis.

Contents

1. Introduction	1
1.1. A Homotopy Perspective on Curriculum Reinforcement Learning	2
1.1.1. Reinforcement Learning as Optimization	3
1.1.2. Local Optima, Regularization, and Reward Shaping	5
1.1.3. Homotopic-Continuation Methods	8
1.1.4. Homotopies via Changing Task Distributions	9
1.2. Thesis Outline and Contributions	10
1.3. Common Notation	12
2. Self-Paced Reinforcement Learning	17
2.1. Introduction	17
2.2. Related Work	19
2.3. Preliminaries: Self-Paced Learning	21
2.4. A Probabilistic Interpretation of Self-Paced Learning	22
2.5. Self-Paced Learning for Reinforcement Learning	25
2.6. Application to Episodic Reinforcement Learning	27
2.6.1. Algorithmic Implementation	28
2.6.2. Experiments	30
2.7. Application to Step-Based Reinforcement Learning	35
2.7.1. Algorithmic Implementation	35
2.7.2. Experiments	36
2.8. Improved α -Schedule	42
2.9. An Inference Perspective on Self-Paced Reinforcement Learning	44
2.9.1. RL as Inference	44
2.9.2. Connection to Self-Paced Reinforcement Learning	46
2.9.3. Self-Paced Learning as Tempering	46
2.10. Conclusion	48
3. On the Benefit of Optimal Transport for Curriculum Reinforcement Learning	51
3.1. Introduction	51

3.2. Related Work	53
3.3. Divergence-Minimizing Curriculum Reinforcement Learning	55
3.4. Curriculum Reinforcement Learning as Constrained Optimal Transport . .	56
3.4.1. Limitations of the KL Divergence	56
3.4.2. Challenges of Expected Performance Constraints	58
3.5. Approximate Algorithms for Discrete- and Continuous Context Spaces . . .	60
3.5.1. Approximate Wasserstein Barycenters	60
3.5.2. Approximate GRADIENT	61
3.5.3. Approximate CURROT	61
3.6. Experiments	64
3.6.1. E-Maze Environment	64
3.6.2. Unlock-Pickup Environment	67
3.6.3. Point-Mass Environment	69
3.6.4. Sparse Goal-Reaching Environment	70
3.6.5. Teach My Agent	72
3.7. Conclusion	74
4. Tracking Control for a Spherical Pendulum via Curriculum Reinforcement Learning	75
4.1. Introduction	75
4.2. Related Work	77
4.3. Reinforcement Learning System	78
4.3.1. Simulation Environment and Policy Representation	78
4.3.2. Facilitating Sim2Real Transfer	80
4.3.3. Trajectory Representations	81
4.3.4. Curriculum Reinforcement Learning	83
4.4. Improved Curriculum Generation	84
4.4.1. Affine Metrics	85
4.4.2. Sampling-Based Optimization	85
4.4.3. Tracking Metrics Other than Reward	86
4.4.4. GPU Implementation	87
4.5. Experiments	87
4.5.1. Quantitative Results	89
4.5.2. Qualitative Analysis of Generated Curricula	90
4.5.3. Alternative Trajectory Representation	92
4.5.4. Real Robot Results	93
4.6. Conclusion	96

5. Conclusion and Future Work	99
6. Contribution Statements	103
6.1. Contributions to Chapter 2	103
6.2. Contributions to Chapter 3	103
6.3. Contributions to Chapter 4	104
A. Appendix to Chapter 1	105
B. Appendix to Chapter 2	107
B.1. Proof of Theorem 1	107
B.2. Self-Paced Episodic Reinforcement Learning Derivations	109
B.3. Regularized Policy Updates	111
B.4. Experimental Details	112
B.4.1. Episodic Setting	113
B.4.2. Step-Based Setting	118
C. Appendix to Chapter 3	127
C.1. Computational Cost of Optimal Transport	127
C.2. CURROT Search for Feasible Contexts	128
C.3. Experimental Details	128
C.3.1. Algorithm Hyperparameters	129
C.3.2. E-Maze Environment	131
C.3.3. Unlock-Pickup Environment	133
C.3.4. Point-Mass Environment	137
C.3.5. Sparse Goal-Reaching Environment	139
C.3.6. Teach My Agent	140
D. Appendix to Chapter 4	141
D.1. High-Dimensional Ablations	141
D.2. Modeling Network Communication Delays	142
D.3. Analytic Solution to the LTI System Equations	143
Bibliography	145
List of Figures	163
List of Algorithms	165

List of Tables	167
Publication List	169
Curriculum Vitae	171

1. Introduction

Reinforcement learning (RL) research aims to build and understand agents that leverage experience to maximize a reward signal using their actions. In the last decade, pairing this framework for decision-making with advances in neural function approximators has pushed the boundaries of problems that can be tackled with RL [134, 183, 8, 173, 42], receiving widespread attention in the scientific community.

For example, the AlphaGo algorithm by Google Deepmind [182, 183] was the first to beat the professional Go player Lee Sedol in a widely covered match. Apart from the combination of tree search with value function approximation, a key ingredient in the algorithm's success was the concept of self-play, i.e., playing against a previous version of the learning agent. The use of self-play ensured that the proficiency of the encountered opponent gradually increased over learning until ultimately reaching and surpassing the competency of the best human players.

In another work, OpenAI paired function approximation with large-scale reinforcement learning to learn dexterous robotic in-hand manipulation of a Rubik's Cube [8]. The learning agent was trained in simulation, and the transfer of learned behavior to the real system was enabled by adaptively randomizing aspects of the environment, e.g., surface friction, such that the agent encountered heavier randomization as it grew more proficient. Another, more recent, robotics application of RL enabled the locomotion of a quadruped with a policy learned entirely in simulation via RL [173]. Leveraging recent progress in GPU-accelerated simulation allowed the researchers to complete the training on a single GPU in minutes. The final agent could progress through complicated terrain, such as stairs and cluttered scenes, without requiring traditional foot-planning methods. Training started with walking in flat terrain and progressed to increasingly challenging scenarios, such as the stairs shown in Figure 1.1.

All these applications sequentially train the agent on versions of a target learning task with increasing difficulty. Training starts with easy versions of the task and exposes the agent to increasingly challenging ones as it gains proficiency. Depending on the learning task, this form of structured learning can either increase the learning speed or be even indispensable for learning. Interestingly, these ideas have not only been investigated as part of increasingly large-scale applications of RL but have been around for decades, albeit

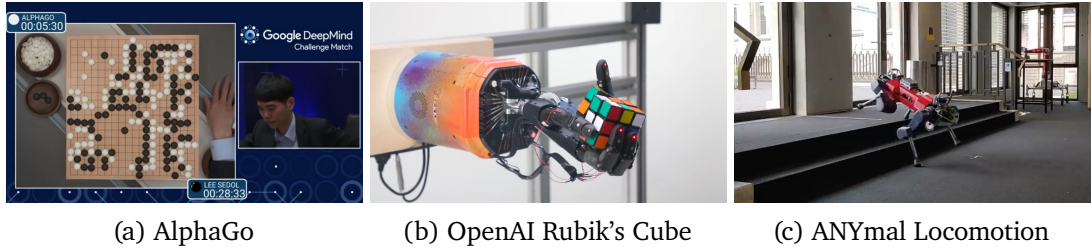


Figure 1.1.: Recent applications of reinforcement learning to solve non-trivial decision making and control tasks using curricula. In Figure 1.1a, we see AlphaGo [182] playing against Lee Sedol. Figure 1.1b shows a robot trained with RL manipulating a Rubik's cube [8]. Finally, Figure 1.1c shows that ANYmal robot climbing stairs using a control-law learned with reinforcement learning in simulation [173].

not giving a particular name [181, 11]. Over time, investigations around these concepts have been carried out under different terms, such as active- [14], transfer- [193], or curriculum learning [20], depending on their particular focus.

This thesis is driven by the question of *how to schedule the complexity of a learning environment to improve the learning performance of RL agents*, or in simpler words, how to build curricula for such agents. We provide answers to this question by developing and benchmarking multiple methods that generate reinforcement learning curricula. All these methods optimize divergences or distances between distributions of learning tasks, or more formally, Markov decision processes, to create interpolations between them. Testing and refining these methods throughout the thesis will result in algorithms that increasingly account for the nuanced aspects of curriculum generation.

Before outlining these developments, we discuss the connection between the proposed methods in this thesis and continuation methods for non-linear optimization in the following section. This connection allows us to introduce reinforcement learning and obtain an intuition on the effect and limits of the curricula we aim to generate.

1.1. A Homotopy Perspective on Curriculum Reinforcement Learning

Aside from the empirically observed benefits of curricula in RL, methods for and investigations of curriculum reinforcement learning are often motivated by concepts from psychology or analogies to human learning, such as shaping [184], intrinsic motivation [15, 14], teacher-student paradigms [162, 90], self-play [182, 186] or the zone of proximal

development [199]. Taking a different perspective, Bengio et al. [20] connect curricula to continuation methods for optimizing non-linear functions and finding roots of systems of non-linear equations. We detail this connection in the following sections, simultaneously introducing required reinforcement learning notation.

1.1.1. Reinforcement Learning as Optimization

To develop a more precise connection between curriculum reinforcement learning (CRL) and continuation methods, we will first introduce the RL objective and policy gradients as one of the working principles of modern on-policy RL algorithms. Reinforcement Learning requires a precise definition of an environment, which is given by a so-called Markov decision process (MDP) \mathcal{M} , a quintuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, p, r, p_0 \rangle$ that defines the state-space \mathcal{S} , action-space \mathcal{A} , dynamics $p: \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}_{\geq 0}$, reward function $r: \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$, and initial state distribution $p_0: \mathcal{S} \mapsto \mathbb{R}_{\geq 0}$ ¹. In this model of an environment, an agent starts in an initial state $\mathbf{s}_0 \sim p_0(\mathbf{s})$ sampled from the initial state distribution. Upon observing the first observation, the agent needs to select an action $\mathbf{a}_0 \in \mathcal{A}$, which then leads to a transition in the environment $\mathbf{s}_1 \sim p(\mathbf{s}|\mathbf{s}_0, \mathbf{a}_0)$ according to the dynamics. As part of this transition, the agent receives the reward $r(\mathbf{s}_0, \mathbf{a}_0)$. This action-observation cycle repeats indefinitely, and the process of action selection is modeled by the so-called policy $\pi: \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}_{\geq 0}$. We will refer to the space of all possible policies as \mathbb{P} .

Using the introduced notation, *reinforcement learning* can be formalized as finding a policy $\pi \in \mathbb{P}$ that maximizes a γ -discounted expected return

$$\max_{\pi \in \mathbb{P}} J(\pi, \mathcal{M}) = \max_{\pi \in \mathbb{P}} \mathbb{E}_{p_0(\mathbf{s}_0), p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t), \pi(\mathbf{a}_t|\mathbf{s}_t)} \left[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) \right]. \quad (1.1)$$

Objective (1.1) is typically optimized knowing neither the analytic form of the dynamics, reward, or initial state distribution. The learning agent relies on experience in the form of transitions $(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, r_t)$ arising from environment interactions, with $\mathbf{s}_{t+1} \sim p(\mathbf{s}|\mathbf{s}_t, \mathbf{a}_t)$ and $r_t = r(\mathbf{s}_t, \mathbf{a}_t)$. The discount factor $\gamma \in [0, 1)$ is one way of dealing with the – in general – unbounded sum of rewards arising from the infinite action-observation cycle, and there is ongoing work on the implications of this choice [107, 143, 4, 179]. Another way of coping with infinity is to look at the average-reward RL setting, which we omit here.

Stating the RL objective (1.1), we ignored the challenge of its optimization in practice. An essential part of this optimization is the value function of a policy π

$$V^\pi(\mathbf{s}, \mathcal{M}) = \mathbb{E}_{\pi(\mathbf{a}|\mathbf{s})} \left[r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{p(\mathbf{s}'|\mathbf{s}, \mathbf{a})} [V^\pi(\mathbf{s}', \mathcal{M})] \right], \quad (1.2)$$

¹The reward function is often also defined as $r: \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$. It is also possible to look at state-dependent action spaces $\mathcal{A}: \mathcal{S} \mapsto \mathcal{P}(\mathcal{A})$, where $\mathcal{P}(\cdot)$ denotes the power set.

which encodes the expected discounted return obtained by the agent following policy π from state s . Expanding the recursive definition of the value function, we see that objective (1.1) maximizes the initial state’s expected value under the policy

$$\max_{\pi \in \mathbb{P}} J(\pi, \mathcal{M}) = \max_{\pi \in \mathbb{P}} \mathbb{E}_{s_0 \sim p_0(s)} [V^\pi(s_0, \mathcal{M})].$$

More importantly, it can be shown that the value function corresponding to the optimal policy π^* , which maximizes the expected discounted return in a given MDP \mathcal{M} , adheres to the following recursive definition

$$V^*(s, \mathcal{M}) = \max_{\mathbf{a} \in \mathcal{A}} r(s, \mathbf{a}) + \gamma \mathbb{E}_{p(s'|s, \mathbf{a})} [V^*(s', \mathcal{M})] = \max_{\mathbf{a} \in \mathcal{A}} Q^*(s, \mathbf{a}, \mathcal{M}),$$

where we have introduced the so-called (optimal) state-action value function $Q^*(s, \mathbf{a}, \mathcal{M})$, also called Q -function. For a given policy π , the Q -function $Q^\pi(s, \mathbf{a}, \mathcal{M})$ encodes, similarly to the value function, the expected discounted return of the agent performing action \mathbf{a} in state s and then following π . Initial work on dynamic programming focused on estimating V^* and Q^* from full process knowledge [17, 75]. Temporal difference (TD) methods [188, 207] have enabled the estimation using online samples only. These approaches have been highly successful for discrete state-action spaces since TD methods are guaranteed to converge to Q^* . Furthermore, Q^* fully encodes the optimal policy π^* . For large or continuous state-action spaces, it is, however, inevitable to resort to some form of function approximation to represent either policy, value-, or Q -function. The necessary introduction of function approximation and the accompanying errors are hard to specify precisely in general settings, and hence, proofs of convergence for approximate dynamic programming or -temporal difference learning rely on assumptions on the MDP and/or function approximator [198, 141, 129].

Opposed to viewing RL as the problem of estimating and approximating the value function, policy-gradient methods [217, 190, 125] allow us to compute the gradient of the RL objective from samples collected by executing the agent’s policy. Starting from a policy π_θ that is parameterized by $\theta \in \Theta$ and differentiable everywhere w.r.t. θ , the policy-gradient theorem by Sutton et al. [190] shows that the gradient of the agent performance w.r.t. the policy parameters can be computed as

$$\nabla_{\theta} J(\pi_{\theta}, \mathcal{M}) = \mathbb{E}_{d^{\pi_{\theta}}(s)} \left[\sum_{\mathbf{a} \in \mathcal{A}} \nabla_{\theta} \pi_{\theta}(s, \mathbf{a}) Q^{\pi_{\theta}}(s, \mathbf{a}, \mathcal{M}) \right], \quad (1.3)$$

with $d^{\pi}(s) = (1 - \gamma) \mathbb{E}_{s_0 \sim p_0(\cdot)} \left[\sum_{t=0}^{\infty} \gamma^t p(s_t | s_0, \pi) \right]$ representing the (discounted) distribution of states visited under the current policy π^2 .

²Note that the original paper only looked at the case of a designated state s_0 but that given the form of the policy gradient, it is easy to just pull the extra expectation over $s_0 \sim p_0(s)$ into d^{π} .

The gradient (1.3) has found widespread application in practice since it does not require computing derivatives of the long-term environment dynamics $d^{\pi\theta}(\mathbf{s})$ w.r.t. θ explicitly. Instead, the Q -function values implicitly capture this interdependence, allowing to approximate the expectation w.r.t. $d^{\pi}(\mathbf{s})$ from samples collected during policy execution. The Q -values that weigh the individual gradients of the policy w.r.t. the parameters θ can, in the simplest form, be approximated from a single rollout $\{(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1}) | t \geq 0\}$, yielding an approximate gradient [6, Lemma 4.10]

$$\frac{\nabla_{\theta} J}{1 - \gamma} \approx \sum_{t=0}^{\infty} \gamma^t \nabla_{\theta} \log(\pi_{\theta}(\mathbf{s}_t, \mathbf{a}_t)) Q^{\pi\theta}(\mathbf{s}_t, \mathbf{a}_t, \mathcal{M}) \approx \sum_{t=0}^{\infty} \gamma^t \left(\nabla_{\theta} \log(\pi_{\theta}(\mathbf{s}_t, \mathbf{a}_t)) \sum_{k=0}^{\infty} \gamma^k r_{t+k} \right). \quad (1.4)$$

In practice, the effect of diminishing returns modeled by γ^t allows practitioners to estimate a finite time-horizon T , after which the contributions of rewards are negligible, and hence, the simulation or experiment can be reset, leading to a practical algorithm.³

In this simplest form, the sample approximation of the outer expectation in the policy gradient (1.3) and the approximation of the Q -values introduce significant variance in the gradients. Williams [217] discussed variance-reduction techniques by subtracting a (possibly state-dependent) baseline. Indeed, variance reduction is an essential part of modern policy-gradient RL algorithms that are, e.g., based on the concept of an advantage function $A^{\pi}(\mathbf{s}, \mathbf{a}, \mathcal{M}) = Q^{\pi}(\mathbf{s}, \mathbf{a}, \mathcal{M}) - V^{\pi}(\mathbf{s}, \mathcal{M})$ [178, 177]. Regardless of the specific instantiation of the policy gradient theorem (1.3), it provides intuition of RL as an ordinary *gradient-based* optimization of the non-linear RL objective $J(\pi_{\theta}, \mathcal{M})$, where the optimization is complicated by having only access to gradients $\nabla_{\theta} J(\pi_{\theta}, \mathcal{M})$ that are corrupted by noise.

1.1.2. Local Optima, Regularization, and Reward Shaping

Reducing RL to an ordinary non-linear optimization problem over policy parameters θ (with typically noisy gradient observations) highlights its susceptibility to challenging loss functions $f(\theta)$ whose local optima or flat regions can lead to sub-optimal optimization outcomes or slow optimization progress.

Figure 1.2a shows an example of a reward function $r(s_0, a)$ for an arbitrary but fixed state $s_0 \in \mathcal{S}$ containing a local optimum. The red line in Figure 1.2a highlights the basin of attraction of the global optimum if we were to optimize $r(s_0, a)$ over the action $a \in \mathcal{A}$ with gradient-based methods. We can construct a minimal MDP \mathcal{M} to formulate the gradient-based optimization of $r(s_0, a)$ w.r.t. a as an RL problem by picturing an MDP in which the

³This truncation is subject to significant discussion and can have significant impact on the quality of the policy [143].

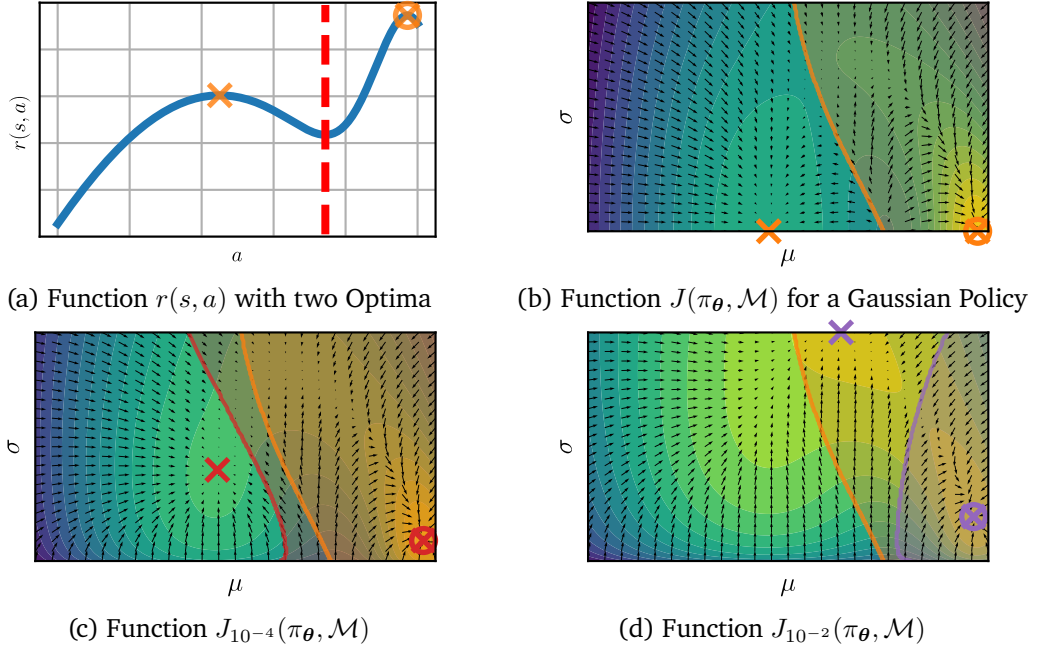


Figure 1.2.: (a) A reward function $r(s, a)$ with a local- (cross) and global (circled cross) optimum. (b) The loss landscape of $J(\pi_\theta, \mathcal{M})$ (Obj. 1.5). We see that the optima coincide with those of $r(s, a)$, since $J(\pi_\theta, \mathcal{M}) = r(s, a)$ for $\sigma=0$. The orange shaded region highlights the basin of attraction of the global optimum. (c+d) Loss landscapes of $J_\lambda(\pi_\theta, \mathcal{M})$ for varying entropy regularization λ . The new optima and the corresponding basins of attraction of the global optima are shown in red and violet. We also visualize the basin of attraction of the unregularized objective for comparison.

agent executes a single step obtaining $r(s_0, a)$, after which the execution terminates⁴. We define the policy π_θ to be a Gaussian $\pi_\theta(a|s_0) = \mathcal{N}(a|\mu, \sigma)$ with parameters $\theta = [\mu \ \sigma]$, as is typical for RL algorithms in continuous spaces [157, 177, 66]. Given this policy choice, the RL objective (1.1) corresponds to the expected function value

$$J(\pi_\theta, \mathcal{M}) = \mathbb{E}_{\mathcal{N}(a|\mu, \sigma)} [r(s_0, a)]. \quad (1.5)$$

Figure 1.2b visualizes the function values and the basin of attraction of the global optimum of $J(\pi_\theta, \mathcal{M})$ over the parameter space θ . Not surprisingly, an optimum $a^* \in \mathcal{A}$ of $r(s_0, a)$ can be translated into an optimum $\theta^* = [a^* \ 0]$ of $J(\pi_\theta, \mathcal{M})$. More interestingly, we can

⁴We show the formal definition of the corresponding MDP in the infinite horizon case in Appendix A.

see that the basin of attraction to the global optimum widens with increasing values of σ , i.e., with increasing initial explorative behavior. Nonetheless, a large set of initial policy parameter values still converge to the local optimum, even for larger values of the parameter σ . A common way of improving the explorative behavior of RL agents is through promoting higher-entropy policies in the loss function [218, 228, 66], i.e., modifying the optimization objective to

$$\max_{\theta} J_{\lambda}(\pi_{\theta}, \mathcal{M}) = \max_{\theta} J(\pi_{\theta}, \mathcal{M}) + \lambda H(\pi_{\theta}), \quad H(p) = - \int_{\mathcal{X}} p(\mathbf{x}) \log(p(\mathbf{x})) d\mathbf{x}.$$

Figure 1.2c shows the benefit of this promotion of explorative behavior via an entropy bonus in our simple example problem, with moderate values of λ increasing the basin of attraction to the global optimum of $J_{\lambda}(\pi_{\theta}, \mathcal{M})$. However, Figure 1.2d also demonstrates a trade-off when using entropy promotion for optimization problems via a fixed value of λ , as both the global optimum of the (now regularized) objective $J_{\lambda}(\pi_{\theta}, \mathcal{M})$ changes with λ and, additionally, a too-large entropy regularization can promote convergence to a sub-optimal high-entropy solution rather than the desired behavior.

This observation is a stereotypical challenge of reward shaping in reinforcement learning, i.e., optimizing a “shaped” reward $\tilde{r}(s, a)$ instead of the nominal one $r(s, a)$ to obtain a more robust convergence to optimal behavior. While the well-known result by Ng, Harada, and Russell [149] shows that there at least exist reward transformations that leave the optimal policy π^* unchanged, the, in practice, more common trial-and-error approach to reward-shaping can easily lead to invalid task specifications [26]. Examples of such invalid task specifications include agents that learn to circle a target instead of reaching it [166] or agents that are hesitant to stack objects to a tower to accumulate intermediate rewards designed to speed up initial exploration [161].

Such perils of reward shaping are one motivation for investigating curricula in reinforcement learning. By not restricting ourselves to training on one reward function $r(s, a)$, we can leverage a sequence of reward functions $(r_i(s, a))_{i \in [1, N]}$ to train the agent. In this sequence, the function $r_1(s, a)$ leads to learning a coarse solution of the target task, which is then iteratively refined by subsequent reward functions $r_i(s, a)$ with $i > 1$. In our example of entropy regularization, this corresponds to annealing the value of λ to zero as the optimization progresses. Looking back at Figure 1.2d, we can see that first optimizing $J_{\lambda}(\pi_{\theta}, \mathcal{M})$ with an entropy regularization of $\lambda=10^{-2}$ and, after convergence, setting $\lambda=0$ would have enabled robust convergence to the global optimum of J since both optima of $J_{10^{-2}}$ are in the basin of attraction of the global optimum of $J_0 = J$. In the next section, we connect this idea of training on sequences of reward functions, or more generally on sequences of MDPs, to continuation methods in non-linear optimization.

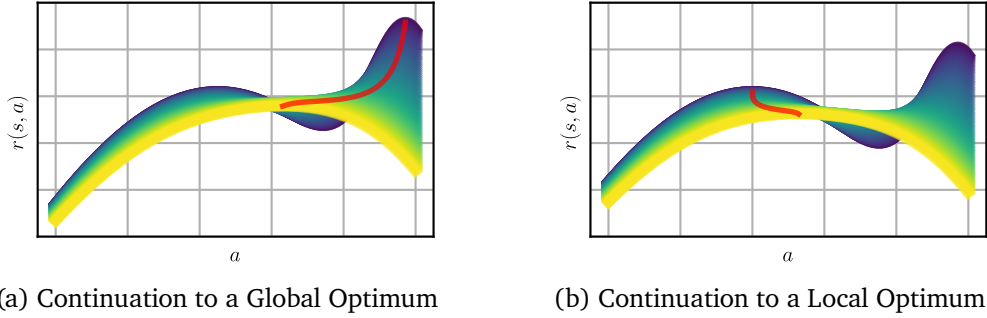


Figure 1.3.: Visualizations of Gaussian continuations on two different functions. (a) shows a Gaussian continuation applied to the function $r(s_0, a)$ from Figure 1.2a. (b) shows a slight variation of $r(s_0, a)$ for which the Gaussian continuation does not lead to the global optima. Bright colors indicate large standard deviation of the Gaussian kernel.

1.1.3. Homotopic-Continuation Methods

Treating the RL objective $J(\pi_\theta, \mathcal{M})$ in Figure 1.2b as a regular function $f([\mu \ \sigma])$, we can re-interpret the function as being a single argument function $f_\sigma(\mu)$ with an external parameter σ . In this case, the function $f_\sigma(\mu)$ is obtained from $r(s_0, a)$ via a convolution with a Gaussian kernel of standard deviation σ

$$\begin{aligned}
 f_\sigma(\mu) &= \mathbb{E}_{\mathcal{N}(a|\mu, \sigma)} [r(s_0, a)] \\
 &= \int_{\mathcal{A}} r(s_0, a) \mathcal{N}(a|\mu, \sigma) da \\
 &= \int_{\mathcal{A}} r(s_0, a) k_\sigma(\mu - a) da = (k_\sigma * r(s_0, \cdot))(\mu), \quad k_\sigma(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}x^2\right).
 \end{aligned}$$

As shown in Figure 1.3a, this family of functions spanned by the parameter σ smoothly deforms the original objective function $r(s_0, a) = f_0(\mu)$ with its two local optima into a function $f_1(\mu)$ with only one global optimum. In mathematical terms, we call such a family of functions that smoothly interpolates between a function f and g via a parameter $\sigma \in [0, 1]$ a homotopy h_σ . Formally, we require that $h_0 = f$, $h_1 = g$, and the map $(x, \sigma) \mapsto h_\sigma(x)$ is continuous.

Such homotopies are exploited in continuation- and homotopy methods [211, 214, 9] to solve general systems of non-linear equations. The central idea of continuation methods via homotopies is to solve an “easy” system of non-linear equations and to follow the solution of this easy problem while gradually changing it using the homotopy. This approach traces paths of solutions to the individual problems along the interpolation spanned by

the parameter σ . For our example problem, the corresponding solution path for varying values of σ is highlighted in red in Figure 1.3a.

Recasting optimization as constraint satisfaction of, e.g., Karush-Kuhn-Tucker conditions [212], homotopy methods have also been applied to general non-linear programming [213, 5, 135] and control problems [59, 78, 37, 172, 139]. The homotopy f_σ used in the preceding example is a form of continuation via Gaussian smoothing [135], which has, e.g., been successfully applied in the context of image alignment [136].

While the behavior of solution paths, as shown in Figure 1.3a, is theoretically well-understood [35, 211, 212, 140] and conditions for the existence of paths from a solution at $\sigma=0$ to a solution at $\sigma=1$ exist [35], it is, in general, not possible to predict which particular solution will be obtained at $\sigma=1$ by the continuation method. Consequently, these methods cannot guarantee convergence to the global optimum but only a stationary point in non-linear optimization. Indeed, a slight change of our example function $r(s_0, a)$ in Figure 1.3b reminds us of this lack of guarantee.

1.1.4. Homotopies via Changing Task Distributions

The minimal RL problem (1.5) allowed us to connect the idea of training on reward function sequences to continuation methods in non-linear optimization, which smoothly deform a challenging objective function via homotopies to avoid convergence to one of its local optima. In this thesis, we build upon this intuition but allow to schedule other aspects of a Markov decision process than the reward function, for which we require a contextual version of the Markov decision process [67]

$$\mathcal{M}(\mathbf{c}) = \langle \mathcal{S}, \mathcal{A}, p_{\mathbf{c}}, r_{\mathbf{c}}, p_{0,\mathbf{c}} \rangle. \quad (1.6)$$

In this contextual MDP, the transition dynamics $p_{\mathbf{c}}$, reward function $r_{\mathbf{c}}$, and initial state distribution $p_{0,\mathbf{c}}$ depend on the parameter $\mathbf{c} \in \mathcal{C}$, which we will refer to as context. This formulation allows for more options to regularize the RL problem, e.g., initializing the agent close to desirable states via $p_{0,\mathbf{c}}$, or providing guidance to such states via the dynamics $p_{\mathbf{c}}$. Based on these contextual MDPs, we can define an “extended” RL objective

$$\max_{\pi \in \mathbb{P}} \mathbb{E}_{\mu(\mathbf{c})} [J(\pi, \mathcal{M}(\mathbf{c}))] \quad (1.7)$$

where $J(\pi, \mathcal{M}(\mathbf{c}))$ is the “default” RL objective (1.1) and $\mu(\mathbf{c})$ is a distribution of learning tasks. This “extended” RL objective is just a reformulation of the original RL objective to highlight the role of the parameter \mathbf{c} . With these definitions in place, curricula for RL can be defined as interpolations in the space of probability distributions that ideally guide the learning agent to well-performing solutions on a target distribution $\mu(\mathbf{c})$.

This formulation is general enough to accommodate the examples presented in Section 1. For AlphaGo, the idea of self-play leads to a change in the MDP dynamics with each policy change. The robotic examples of in-hand manipulation and quadruped locomotion can also be readily modeled by distributions over dynamics parameters, such as friction or gravity in [8] or the floor surface in [173].

To conclude this discussion of homotopies and curricula, we wish to state the limitations of the works presented in this thesis from the perspective of continuation methods. We present no guarantees that the RL problems on a sequence of smoothly changing task distributions $p(\mathbf{c})$ have a connected solution path, i.e., a connected path as in Figure 1.3a, from the initial- to the target distribution. Furthermore, our way of updating the task distribution $p(\mathbf{c})$ during agent training is another break with continuation methods, which trace the solution path by, e.g., numerically simulating an ordinary differential equation. We do not follow such an approach as the approximate nature of modern RL, with its heavy use of function approximation, renders the idea of tracking solution paths with even moderate precision challenging to guarantee or achieve in practice. Finally, we already discussed that continuation methods cannot guarantee convergence to good or global optima, and the same holds for the methods developed in this thesis. We will show an example of such a failed interpolation or continuation resulting from the misspecification of the encountered task structure in Chapter 3.

1.2. Thesis Outline and Contributions

This thesis proposes and benchmarks interpolations $p(\mathbf{c})$ between task distributions such that training on these interpolations improves the learning behavior of RL agents, similar in spirit to continuation methods for solving non-linear systems of equations. The thesis consists of three chapters that present methods for generating task distributions $p(\mathbf{c})$ for reinforcement learning agents and benchmark them in different reinforcement learning tasks. Although the chapters build up on each other, they can be read independently.

Self-Paced Reinforcement Learning

In Chapter 2, we propose an interpolation between an initial- and target task distribution based on the Kullback-Leibler divergence. We connect this interpolation to the self-paced learning framework initially developed by Kumar, Packer, and Koller [105] for supervised learning problems. The intuition of self-paced learning to learn on easy tasks first, where the easiness of a task is measured w.r.t. the obtained agent reward, has already been connected to a majorization minimization scheme on a parameterized implicit objective [131], resulting in a connection between our approximate curriculum RL methods and

homotopy methods. We propose approximate methods for generating the interpolation in an episodic- and step-based RL setting, showing improved performance on different learning tasks. The chapter was published in the Journal of Machine Learning Research [93], summarizing two previous conference papers presented at the Conference on Robot Learning [97] and at Advances in Neural Information Processing Systems [98].

On the Benefit of Optimal Transport for Curriculum Reinforcement Learning

Chapter 3 results from an effort to obtain approximate task distribution interpolations with fewer restrictions on the parametric form of the interpolating distributions $p(\mathbf{c})$. During this pursuit, we realized that the approximate formulations presented in Chapter 2 heavily rely on those parametric forms to regularize the interpolation $p(\mathbf{c})$. To combine a flexible representation of $p(\mathbf{c})$ with well-behaving interpolations, we turn to optimal transport, allowing us to lift a ground metric defined on the context space \mathcal{C} into the space of probability distributions on \mathcal{C} . This change in methodology highlights the critical role of a metric $d(\mathbf{c}_1, \mathbf{c}_2)$ measuring the similarity between two tasks, $\mathbf{c}_1 \in \mathcal{C}$ and $\mathbf{c}_2 \in \mathcal{C}$, when generating curricula for reinforcement learning agents. The optimal transport formulation has the additional benefit of allowing for conceptually straightforward approximate implementations via particles. In experiments, we highlight the role of the task space metric d and further show that approximate implementations of our approach lead to good empirical performance in different benchmark tasks. The chapter is based on a preprint currently under review by the IEEE Transactions on Pattern Analysis and Machine Intelligence and extends upon a conference paper at the International Conference on Machine Learning [95].

Tracking Control for a Spherical Pendulum via Curriculum Reinforcement Learning

The final Chapter 4 tests the behavior of our optimal transport formulation in higher-dimensional task spaces \mathcal{C} . We focus on the task of learning tracking control of a spherical pendulum that is attached to a four degrees-of-freedom (DoF) Barrett Whole Arm Manipulator (WAM), where we build the curriculum directly over the trajectories that are to be followed by the pendulum. Our efforts in building such curricula highlight the importance of the task space metric $d(\mathbf{c}_1, \mathbf{c}_2)$ and an appropriate sampling scheme for updating the distribution $p(\mathbf{c})$ when building curricula in high-dimensional spaces. Ultimately, an improved version of the method developed in Chapter 3 can reliably learn tracking control that transfers to reality, allowing for successful task completion on the real robot without fine-tuning. This work is currently under review by the IEEE Transactions on Robotics.

1.3. Common Notation

To ease the understanding of the following chapters and prevent the repetitive introduction of shared notation, we wish to repeat the most important concepts and their symbols in this section. Most of this notation has already been covered in the previous sections, and only a few additional concepts need to be introduced.

Reinforcement Learning

We have already introduced the reinforcement learning objective (1.1) as part of the introduction. Sometimes, we will focus on the trajectories τ that are generated by the policy $\pi \in \mathbb{P}$, resulting in the following (equivalent) definition of the RL objective given a Markov decision process (MDP) $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, p, r, p_0 \rangle$

$$\max_{\pi \in \mathbb{P}} J(\pi, \mathcal{M}) = \max_{\pi \in \mathbb{P}} \mathbb{E}_{p(\tau|\pi, \mathcal{M})} \left[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) \right], \quad \gamma \in [0, 1) \quad (1.8)$$
$$\tau = \{(\mathbf{s}_t, \mathbf{a}_t) | t = 0, \dots\}, \quad p(\tau|\pi, \mathcal{M}) = p_0(\mathbf{s}_0) \prod_{t=1}^{\infty} p(\mathbf{s}_t | \mathbf{s}_{t-1}, \mathbf{a}_{t-1}) \pi(\mathbf{a}_{t-1} | \mathbf{s}_{t-1}).$$

We emphasize the importance of the MDP \mathcal{M} in the RL objective by using it as an argument to the objective $J(\pi, \mathcal{M})$. This additional emphasis enables a more straightforward extension to the contextual case, where we encounter RL objectives for multiple MDPs. As we saw already, another equivalent formulation of the RL objective is given via the value function

$$\max_{\pi \in \mathbb{P}} J(\pi, \mathcal{M}) = \max_{\pi \in \mathbb{P}} \mathbb{E}_{p_0(\mathbf{s}_0)} [V^\pi(\mathbf{s}_0, \mathcal{M})] \quad (1.9)$$
$$V^\pi(\mathbf{s}, \mathcal{M}) = \mathbb{E}_{\pi(\mathbf{a}|\mathbf{s})} [r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{p(\mathbf{s}'|\mathbf{s}, \mathbf{a})} [V^\pi(\mathbf{s}', \mathcal{M})]].$$

For experiments, we inevitably need to make use of parameterized policies $\pi(\mathbf{a}|\mathbf{s}, \boldsymbol{\theta})$ with $\boldsymbol{\theta} \in \Theta$. We will alternatively denote such policies as $\pi_\theta(\mathbf{a}|\mathbf{s})$. When using a parameterized policy, we replace the parameterized policy symbol $\pi_\theta(\mathbf{a}|\mathbf{s})$ with its parameters $\boldsymbol{\theta}$ to shorten notation, e.g., writing

$$\max_{\boldsymbol{\theta} \in \Theta} J(\boldsymbol{\theta}, \mathcal{M}) = \max_{\boldsymbol{\theta} \in \Theta} J(\pi_\theta, \mathcal{M}), \quad V^\theta(\mathbf{s}, \mathcal{M}) = V^{\pi_\theta}(\mathbf{s}, \mathcal{M}), \quad p(\tau|\boldsymbol{\theta}, \mathcal{M}) = p(\tau|\pi_\theta, \mathcal{M}).$$

Contextual Reinforcement Learning

The subsequent chapters will build up on the idea of a contextual Markov decision process $\mathcal{M}(\mathbf{c}) = \langle \mathcal{S}, \mathcal{A}, p_{\mathbf{c}}, r_{\mathbf{c}}, p_{0, \mathbf{c}} \rangle$ defined over a set of contextual parameters, or contexts,

$\mathbf{c} \in \mathcal{C}$ [67]. This contextual model of optimal decision-making has been investigated by multiple works from different perspectives [148, 175, 137] and is well-suited for learning in multiple related tasks as is the case in multi-task [219], goal-conditioned [175] or curriculum RL [146]. Instead of the word context, we also use the word task to refer to \mathbf{c} or $\mathcal{M}(\mathbf{c})$. We denote the distribution of tasks that the agent is expected to master as $\mu(\mathbf{c})$. Overloading the function $J(\cdot, \cdot)$ with different meanings depending on the given argument, the contextual RL objective is then given by

$$\max_{\pi \in \mathbb{P}} J(\pi, \mu) = \max_{\pi \in \mathbb{P}} \mathbb{E}_{\mu(\mathbf{c})} [J(\pi, \mathcal{M}(\mathbf{c}))]. \quad (1.10)$$

At this point, we see how emphasizing the MDP in the single-task RL objective $J(\pi, \mathcal{M})$ pays off and allows for a simple extension to the contextual setting. Put in words, the agent is now required to perform well on those tasks that are likely under $\mu(\mathbf{c})$. The structure of the policy π requires additional discussion in the contextual setting. Depending on the particular application, the policy may or may not observe the context, i.e., $\pi(\mathbf{s}|\mathbf{a}, \mathbf{c})$ or $\pi(\mathbf{s}|\mathbf{a})$. The former case is preferable when, e.g., encoding a goal state to be reached via the context \mathbf{c} . If \mathbf{c} models disturbances in the environment dynamics, it may be preferable not to provide information about \mathbf{c} to the policy to enforce robustness to unknown disturbances.

As for the policy, we replace $\mathcal{M}(\mathbf{c})$ simply by \mathbf{c} when used as a function argument. The same procedure applies when dealing with parameterized task distributions $p(\mathbf{s}|\boldsymbol{\nu}) = p_{\boldsymbol{\nu}}(\mathbf{s})$ with $\boldsymbol{\nu} \in \mathcal{N}$, resulting in the following abbreviated notations

$$J(\boldsymbol{\theta}, \mathbf{c}) = J(\pi_{\boldsymbol{\theta}}, \mathcal{M}(\mathbf{c})), \quad J(\boldsymbol{\theta}, \boldsymbol{\nu}) = J(\pi_{\boldsymbol{\theta}}, p_{\boldsymbol{\nu}}), \quad p(\boldsymbol{\tau}|\boldsymbol{\theta}, \mathbf{c}) = p(\boldsymbol{\tau}|\pi_{\boldsymbol{\theta}}, \mathcal{M}(\mathbf{c})).$$

To finish the introduction of contextual MDPs, we wish to emphasize once more that they are not a new concept but a specifically structured MDP that emphasizes the role of the context \mathbf{c} . To see this, we assume a contextual MDP $\mathcal{M}(\mathbf{c})$ with a given distribution $\mu(\mathbf{c})$. Now, we define an MDP \mathcal{M}_{ext} with the same action space as $\mathcal{M}(\mathbf{c})$ but an extended state space $\mathcal{S}_{\text{ext}} = \mathcal{S} \times \mathcal{C}$ with states $\mathbf{s}_{\text{ext}} = (\mathbf{s}, \mathbf{c})$. We can then define the initial state distribution of \mathcal{M}_{ext} as $p_0(\mathbf{s}_{\text{ext}}) = p_{0, \mathbf{c}}(\mathbf{s})\mu(\mathbf{c})$, the reward as $r(\mathbf{s}_{\text{ext}}, \mathbf{a}) = r_{\mathbf{c}}(\mathbf{s}, \mathbf{a})$, and finally, the dynamics as

$$p(\mathbf{s}'_{\text{ext}}|\mathbf{s}_{\text{ext}}, \mathbf{a}) = \delta_{\mathbf{c}}(\mathbf{c}')p_{\mathbf{c}}(\mathbf{s}'|\mathbf{s}, \mathbf{a}),$$

where $\delta_{\mathbf{c}}(\mathbf{c}')$ is a Dirac delta distribution centered on \mathbf{c} . The resulting single-task RL objective $J(\pi, \mathcal{M}_{\text{ext}})$ matches precisely the contextual objective $J(\pi, \mu)$. We could, hence, alternatively frame the idea of learning via curricula as optimizing $J(\pi, \mathcal{M}_{\text{ext}})$ over differing initial state distributions.

Divergences and Distances between Probability Distributions

The final ingredients in the algorithms presented in this thesis are similarity measures between two task distributions $p_1(\mathbf{c})$ and $p_2(\mathbf{c})$. In Chapter 2, we will heavily rely on the **Kullback-Leibler (KL) divergence**

$$D_{\text{KL}}(p_1(\mathbf{c}) \parallel p_2(\mathbf{c})) = \int_{\mathcal{C}} p_1(\mathbf{c}) \log \left(\frac{p_1(\mathbf{c})}{p_2(\mathbf{c})} \right) d\mathbf{c}, \quad (1.11)$$

a central measure of information in the domain of information theory. As discussed by Kullback and Leibler [104], the KL Divergence can be seen as a measure of how much information samples from $p_1(\mathbf{c})$ contain for discriminating it from $p_2(\mathbf{c})$. Consequently, if $D_{\text{KL}}(p_1(\mathbf{c}) \parallel p_2(\mathbf{c}))=0$, sampling from $p_1(\mathbf{c})$ will not yield any discriminative information w.r.t. $p_2(\mathbf{c})$. We want to note that the KL divergence is not a metric, as it is not symmetric and does not fulfill the triangle inequality. Furthermore, if $p_1(\mathbf{c})$ and $p_2(\mathbf{c})$ are not absolutely continuous, i.e., there exist contexts $\mathbf{c} \in \mathcal{C}$ for which $p_2(\mathbf{c})=0$ while $p_1(\mathbf{c})>0$, we can perfectly discriminate $p_1(\mathbf{c})$ and $p_2(\mathbf{c})$ from one of those contexts, which is why Kullback and Leibler [104] required absolute continuity of p_2 w.r.t. p_1 .

Opposed to this information-theoretic measure of similarity, we use concepts from the field of optimal transport in Chapters 3 and 4. The problem of optimally transporting density between two distributions has been initially investigated by Monge [138]. As of today, generalizations established by Kantorovich [86] have led to so-called **Wasserstein distances** as metrics between probability distributions defined on a metric space $M=(d, \mathcal{C})$ with metric $d : \mathcal{C} \times \mathcal{C} \mapsto \mathbb{R}_{\geq 0}$

$$\mathcal{W}_p(p_1, p_2) = \left(\inf_{\phi \in \Phi(p_1, p_2)} \mathbb{E}_{\phi} [d(\mathbf{c}_1, \mathbf{c}_2)^p] \right)^{1/p}, \quad p \geq 1 \quad (1.12)$$

$$\Phi(p_1, p_2) = \{ \phi : \mathcal{C} \times \mathcal{C} \mapsto \mathbb{R}_{\geq 0} \mid p_i = P_{i\#} \phi, i \in \{1, 2\} \}, \quad (1.13)$$

where $P_{i\#}$ are the push-forwards of the maps $P_1(\mathbf{c}_1, \mathbf{c}_2)=\mathbf{c}_1$ and $P_2(\mathbf{c}_1, \mathbf{c}_2)=\mathbf{c}_2$. We refer to [158, Chapter 2] for an excellent and intuitive introduction to these concepts. The distance between p_1 and p_2 is obtained via the solution to an optimization problem that finds a so-called plan, or coupling, ϕ . This coupling encodes how to equalize p_1 and p_2 considering the cost of moving density between parts of the space \mathcal{C} . This cost is encoded by the metric d . In the following, we will always assume to work with 2-Wasserstein distances, i.e., $p=2$, due to their suitedness for the interpolation between measures [see 158, Chapter 6 and Remark 2.24].

Similar to how (weighted) means can be defined as solutions to optimization problems on a metric space $M=(d, \mathcal{C})$, Wasserstein distances allow us to define what is referred to as **Wasserstein barycenters** [7]

$$\mathcal{B}_2(W, P) = \arg \min_P \sum_{k=1}^K w_k \mathcal{W}_2(p, p_k), \quad (1.14)$$

which represent the (weighted) mean of the distributions $P=\{p_k|k \in [1, K]\}$ with weights $W=\{w_k|k \in [1, K]\}$. The Wasserstein distance and the corresponding barycenters will play a central role in Chapters 3 and 4.

2. Self-Paced Reinforcement Learning

Across machine learning, the use of curricula has shown strong empirical potential to improve learning from data by avoiding local optima of training objectives. For reinforcement learning (RL), curricula are especially interesting, as the underlying optimization has a strong tendency to get stuck in local optima due to the exploration-exploitation trade-off. Recently, a number of approaches for an automatic generation of curricula for RL have been shown to increase performance while requiring less expert knowledge compared to manually designed curricula. However, these approaches are seldomly investigated from a theoretical perspective, preventing a deeper understanding of their mechanics. In this chapter, we present an approach for automated curriculum generation in RL with a clear theoretical underpinning. More precisely, we formalize the well-known self-paced learning paradigm as inducing a distribution over training tasks, which trades off between task complexity and the objective to match a desired task distribution. Experiments show that training on this induced distribution helps to avoid poor local optima across RL algorithms in different tasks with uninformative rewards and challenging exploration requirements.

2.1. Introduction

Research on reinforcement learning (RL) [189] has led to recent successes in long-horizon planning [134, 183] and robot control [101, 110]. A driving factor of these successes has been the combination of RL paradigms with powerful function approximators, commonly referred to as deep RL (DRL). While DRL has considerably pushed the boundary w.r.t. the type and size of tasks that can be tackled, its algorithms suffer from high sample complexity. This can lead to poor performance in scenarios where the demand for samples is not satisfied. Furthermore, crucial challenges such as poor exploratory behavior of RL agents are still far from being solved, resulting in a large body of research that aims to reduce sample complexity by improving this exploratory behavior of RL agents [121, 191, 16, 74, 180].

Another approach to making more efficient use of samples is to leverage similarities between learning environments and tasks in the framework of contextual- or multi-task

RL. In these frameworks, a shared task structure permits simultaneous optimization of a policy for multiple tasks via inter- and extrapolation [106, 175, 77], resulting in tangible speed ups in learning across tasks. Such approaches expose the agent to tasks drawn from a distribution under which the agent should optimize its behavior. Training on such a fixed distribution, however, does not fully leverage the contextual RL setting in case there is a difference in difficulty among tasks. In such a scenario, first training on “easier” tasks and exploiting the generalizing behavior of the agent to gradually progress to “harder” ones promises to make more efficient use of environment interaction. This idea is at the heart of curriculum learning (CL), a term introduced by [20] for supervised learning problems. By now, applications of CL have increasingly expanded to reinforcement learning problems, where the aim is to design task sequences that maximally benefit the learning progress of an RL agent [146].

Recently, an increasing number of algorithms for an automated generation of curricula have been proposed [15, 57, 10, 170]. While empirically demonstrating their beneficial effect on the learning performance of RL agents, the heuristics that guide the generation of the curriculum are, as of now, theoretically not well understood. In contrast, in supervised learning, self-paced learning [105] is an approach to curriculum generation that enjoys wide adaptation in practice [187, 51, 79] and has a firm theoretical interpretation as a majorize-minimize algorithm applied to a regularized objective [131]. In this chapter, we develop an interpretation of self-paced learning as the process of generating a sequence of distributions over samples. We use this interpretation to transfer the concept of self-paced learning to RL problems, where the resulting approach generates a curriculum based on two quantities: the value function of the agent (reflecting the task complexity) and the KL divergence to a target distribution of tasks (reflecting the incorporation of desired tasks).

Contribution: We propose an interpretation of the self-paced learning algorithm from a probabilistic perspective, in which the weighting of training samples corresponds to a sampling distribution (Section 2.4). Based on this interpretation, we apply self-paced learning to the contextual RL setting, obtaining a curriculum over RL tasks that trades-off agent performance and matching a target distribution of tasks (Section 2.5). We connect the approach to the RL-as-inference paradigm [197, 109], recovering well-known regularization techniques in the inference literature (Section 2.9). We experimentally evaluate algorithmic realizations of the curriculum in both episodic- (Section 2.6) and step-based RL settings (Section 2.7). Empirical evidence suggests that the scheme can match and surpass state-of-the-art CL methods for RL in environments of different complexity and with sparse and dense rewards.

2.2. Related Work

Simultaneously evolving the learning task with the learner has been investigated in a variety of fields ranging from behavioral psychology [184] to evolutionary robotics [24] and RL [11, 50, 206]. For supervised learning, this principle was given the name *curriculum learning* by Bengio et al. [20]. The name has by now also been established in the reinforcement learning (RL) community, where a variety of algorithms aiming to generate curricula that maximally benefit the learner have been proposed.

A driving principle behind curriculum reinforcement learning (CRL) is the idea of transferring successful behavior from one task to another, deeply connecting it to the problem of transfer learning [153, 192, 108]. In general, transferring knowledge is—depending on the scenario—a challenging problem on its own, requiring a careful definition of what is to be transferred and what are the assumptions about the tasks between which to transfer. Aside from this problem, Narvekar and Stone [145] showed that learning to create an *optimal* curriculum can be computationally harder than learning the solution for a task from scratch. Both of these factors motivate research on tractable approximations to the problem of transfer and curriculum generation.

To ease the problem of transferring behavior between RL tasks, a shared state-action space between tasks as well as an additional variable encoding the task to be solved are commonly assumed. This variable is usually called a goal [175] or a context [137, 106]. In this chapter, we will adapt the second name, also treating the word “context” and “task” interchangeably, i.e. treating the additional variable and the task that it represents as the same entity.

It has been shown that function approximators can leverage the shared state-action space and the additional task information to generalize important quantities, such as value functions, across tasks [175]. This approach circumvents the complicated problem of transfer in its generality, does however impose assumptions on the set of Markov decision processes (MDPs) as well as the contextual variable that describes them. Results from [137] suggest that one such assumption may be a gradual change in reward and dynamics of the MDP w.r.t. the context, although this requirement would need to be empirically verified. For the remainder of this document, we will disregard this important problem and focus on RL problems with similar characteristics as the ones investigated by [137], as often done for other CRL algorithms. A detailed study of these assumptions and their impact on CRL algorithms is not known to us but is an interesting endeavor. We now continue to highlight some CRL algorithms and refer to the survey by [146] for an extensive overview.

The majority of CRL methods can be divided into three categories w.r.t. the underlying concept. On the one hand, in tasks with binary rewards or success indicators, the idea of keeping the agent’s success rate within a certain range has resulted in algorithms

with drastically improved sample efficiency [56, 57, 10]. On the other hand, many CRL methods [176, 15, 162, 58] are inspired by the idea of ‘curiosity’ or ‘intrinsic motivation’ [152, 23]—terms that refer to the way humans organize autonomous learning even in the absence of a task to be accomplished. The third category includes algorithms that use the value function to guide the curriculum. While similar to methods based on success indicators in sparse reward settings, these methods can allow to incorporate the richer feedback available in dense rewards settings. To the best of our knowledge, only our work and that of Wöhlke, Schmitt, and Hoof [220] fall into this category. The work of Wöhlke, Schmitt, and Hoof [220] defines a curriculum over starting states using the gradient of the value function w.r.t. the starting state. The proposed curriculum prefers starting states with a large gradient norm of the value function, creating similarities to metrics used in intrinsic motivation. In our method, the value function is used as a competence measure to trade-off between easy tasks and tasks that are likely under a target distribution.

Our approach to curriculum generation builds upon the idea of *self-paced learning* (SPL), initially proposed by Kumar, Packer, and Koller [105] for supervised learning tasks and extended by Jiang et al. [81, 80] to allow for user-chosen penalty functions and constraints. SPL generates a curriculum by trading-off between exposing the learner to all available training samples and selecting samples in which the learner performs well. The approach has been employed in a variety of supervised learning problems [187, 51, 79]. Furthermore, Meng, Zhao, and Jiang [131] proposed a theoretical interpretation of SPL, identifying it as a majorize-minimize algorithm applied to a regularized objective function. Despite its well-understood theoretical standing and empirical success in supervised learning tasks, SPL has only been applied in a limited way to RL problems, restricting its use to the prioritization of replay data from an experience buffer in deep Q -networks [168]. Orthogonal to this approach, we will make use of SPL to adaptively select training tasks during agent learning.

Furthermore, we will connect the resulting algorithms to the RL-as-inference perspective during the course of this chapter. Therefore, we wish to briefly point to several works employing this perspective [41, 197, 43, 167, 109]. Taking an inference perspective is beneficial when dealing with inverse problems or problems that require tractable approximations [72, 163]. Viewing RL as an inference problem naturally motivates regularization methods such as maximum- or relative entropy [228, 157, 66] that have proven highly beneficial in practice. Further, this view allows to rigorously reason about the problem of optimal exploration in RL [61]. Finally, it stimulates the development of new, and interpretation of existing, algorithms as different approximations to the intractable integrals that need to be computed in probabilistic inference problems [1, 52], resulting in a highly principled approach to tackling the challenging problem of RL.

2.3. Preliminaries: Self-Paced Learning

Having introduced the necessary notation on (contextual) MDPs and the associated RL objectives in Section 1.3, we are left with defining the concept of self-paced learning (SPL). Self-paced learning (SPL) has been introduced by Kumar, Packer, and Koller [105] for supervised learning settings, in which a function approximator $y = m(\mathbf{x}, \boldsymbol{\theta})$ with parameters $\boldsymbol{\theta} \in \mathbb{R}^{d_\theta}$ is trained w.r.t. a given data set $\mathcal{D} = \{(\mathbf{x}_i, y_i) \mid \mathbf{x}_i \in \mathbb{R}^{d_x}, y_i \in \mathbb{R}, i \in [1, N]\}$. In this setting, SPL generates a curriculum over the data set \mathcal{D} by introducing a vector $\boldsymbol{\nu} = [\nu_1 \ \nu_2 \ \dots \ \nu_N] \in [0, 1]^N$ of weights ν_i for the entries (\mathbf{x}_i, y_i) in the data set. These weights are automatically adjusted during learning via a ‘self-paced regularizer’ $f(\alpha, \boldsymbol{\nu})$ in the SPL objective

$$\boldsymbol{\nu}^*, \boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\nu} \in [0, 1]^N, \boldsymbol{\theta} \in \mathbb{R}^{d_\theta}} r(\boldsymbol{\theta}) + \sum_{i=1}^N (\nu_i l(\mathbf{x}_i, y_i, \boldsymbol{\theta}) + f(\alpha, \nu_i)), \quad \alpha > 0. \quad (2.1)$$

The term $r(\boldsymbol{\theta})$ represents potentially employed regularization of the model and $l(\mathbf{x}_i, y_i, \boldsymbol{\theta})$ represents the error in the model prediction $\tilde{y}_i = m(\mathbf{x}_i, \boldsymbol{\theta})$ for sample (\mathbf{x}_i, y_i) . The motivation for this principle as well as its name are best explained by investigating the solution $\boldsymbol{\nu}^*(\alpha, \boldsymbol{\theta})$ of optimization problem (2.1) when only optimizing it w.r.t. $\boldsymbol{\nu}$ while keeping α and $\boldsymbol{\theta}$ fixed. Introducing the notation $\nu^*(\alpha, l) = \arg \min_{\nu} \nu l + f(\alpha, \nu)$, we can define the optimal $\boldsymbol{\nu}$ for given α and $\boldsymbol{\theta}$ as

$$\boldsymbol{\nu}^*(\alpha, \boldsymbol{\theta}) = [\nu^*(\alpha, l(\mathbf{x}_1, y_1, \boldsymbol{\theta})) \ \nu^*(\alpha, l(\mathbf{x}_2, y_2, \boldsymbol{\theta})) \ \dots \ \nu^*(\alpha, l(\mathbf{x}_N, y_N, \boldsymbol{\theta}))].$$

For the self-paced function $f_{\text{Bin}}(\alpha, \nu_i) = -\alpha \nu_i$ initially proposed by [105], it holds that

$$\nu_{\text{Bin}}^*(\alpha, l) = \begin{cases} 1, & \text{if } l < \alpha \\ 0, & \text{else.} \end{cases} \quad (2.2)$$

We see that the optimal weights $\nu_{\text{Bin}}^*(\alpha, \boldsymbol{\theta})$ focus on examples on which the model under the current parameters $\boldsymbol{\theta}$ performs better than a chosen threshold α . By continuously increasing α and updating $\boldsymbol{\nu}$ and $\boldsymbol{\theta}$ in a block-coordinate manner, SPL creates a curriculum consisting of increasingly ‘‘hard’’ training examples w.r.t. the current model. A highly interesting connection between SPL and well-known regularization terms for machine learning has been established by Meng, Zhao, and Jiang [131]. Based on certain axioms on the self-paced regularizer $f(\alpha, \nu_i)$ (see appendix), Meng, Zhao, and Jiang [131] showed that the SPL scheme of alternatingly optimizing (2.1) w.r.t. $\boldsymbol{\theta}$ and $\boldsymbol{\nu}$ implicitly optimizes

the regularized objective

$$\min_{\boldsymbol{\theta}} r(\boldsymbol{\theta}) + \sum_{i=1}^N F_{\alpha}(l(\mathbf{x}_i, y_i, \boldsymbol{\theta})), \quad F_{\alpha}(l(\mathbf{x}_i, y_i, \boldsymbol{\theta})) = \int_0^{l(\mathbf{x}_i, y_i, \boldsymbol{\theta})} \nu^*(\alpha, \iota) d\iota. \quad (2.3)$$

Using the Leibniz integral rule on F_{α} , we can see that $\nabla_l F_{\alpha}(l) = \nu^*(\alpha, l)$. Put differently, the weight $\nu_i^*(\alpha, \boldsymbol{\theta})$ encodes how much a decrease in the prediction error $l(\mathbf{x}_i, y_i, \boldsymbol{\theta})$ for the training example (\mathbf{x}_i, y_i) decreases the regularized objective (2.3). In combination with the previously mentioned axioms on the self-paced regularizer $f(\alpha, \nu_i)$, this allowed Meng, Zhao, and Jiang [131] to prove the connection between (2.1) and (2.3). Furthermore, they showed that, depending on the chosen self-paced regularizer, the resulting regularizer $F_{\alpha}(l(\mathbf{x}_i, y_i, \boldsymbol{\theta}))$ corresponds exactly to non-convex regularization terms used in machine learning to e.g. guide feature selection [226, 224]. Opposed to feature selection, SPL makes use of these regularizers to attenuate the influence of training examples which the model cannot explain under the current parameters $\boldsymbol{\theta}$. This attenuation of hard training examples on non-proficient models is achieved by reducing their contribution to the gradient w.r.t. $\boldsymbol{\theta}$ via the function F_{α} (see Figure 2.1). This observation naturally explains tendencies of SPL to improve learning e.g. in the presence of extreme noise, as empirically demonstrated by [80].

To summarize, we have seen that SPL formulates a curriculum over a set of training data as an alternating optimization of weights ν for the training data given the current model and the model parameters $\boldsymbol{\theta}$ given the current weights. This alternating optimization performs an implicit regularization of the learning objective, suppressing the gradient contribution of samples that the model cannot explain under the current parameters. Empirically, this has been shown to reduce the likelihood of converging to poor local optima.

2.4. A Probabilistic Interpretation of Self-Paced Learning

In Chapter 1, we have discussed the RL objective (1.8) and highlighted the problem of policy optimization converging to a local optimum or only converging slowly. While ensuring to learn globally optimal policies with optimal sample complexity in its whole generality is an open problem, we discussed in the previous section that for supervised learning, the use of regularizing functions F_{α} can smooth out local optima by transforming the employed loss function. Motivated by this insight, we now apply the aforementioned functions to regularize the contextual RL objective (1.10), obtaining

$$\min_{\boldsymbol{\theta} \in \Theta} \mathbb{E}_{\mu(\mathbf{c})} [F_{\alpha}(-J(\boldsymbol{\theta}, \mathbf{c}))]. \quad (2.4)$$

This objective has two slight differences to the SPL objective (2.3). First, it misses the regularization term $r(\boldsymbol{\theta})$ from (2.3). Second, objective (2.4) is defined as an expectation of the regularized performance $F_\alpha(J(\boldsymbol{\theta}, \mathbf{c}))$ w.r.t. to the context distribution $\mu(\mathbf{c})$ instead of a sum over the regularized performances. This can be seen as a generalization of (2.3), in which we allow to chose $\mu(\mathbf{c})$ differently from a uniform distribution over a discrete set of values. Regardless of these technical differences, one could readily optimize objective (2.4) in a supervised learning scenario e.g. via a form of stochastic gradient descent. As argued in Section 2.3, this results in an SPL optimization scheme (2.1) since the regularizer F_α performs an implicit weighting of the gradients $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, \mathbf{c})$.

In an RL setting, the problem with such a straightforward optimization is that each evaluation of $J(\boldsymbol{\theta}, \mathbf{c})$ and its gradient is typically expensive. If now for given parameters $\boldsymbol{\theta}$ and context \mathbf{c} , the regularizer F_α leads to a negligible influence of $J(\boldsymbol{\theta}, \mathbf{c})$ to the gradient of the objective (see Figure 2.1), evaluating $J(\boldsymbol{\theta}, \mathbf{c})$ wastes the precious resources that the learning agent should carefully utilize. In an RL setting, it is hence crucial to make use of a sampling distribution $p(\mathbf{c})$ that avoids the described wasteful evaluations. At this point, the insight that an SPL weight is equal to the gradient of the regularizing function F_α for the corresponding context, i.e. $\nu^*(\alpha, J(\mathbf{c}, \boldsymbol{\theta})) = \nabla_l F_\alpha(l)|_{l=J(\mathbf{c}, \boldsymbol{\theta})}$, directly yields a method for efficiently evaluating objective (2.4)—that is by sampling a context \mathbf{c} according to its SPL weight $\nu^*(\alpha, J(\mathbf{c}, \boldsymbol{\theta}))$. To make this intuition rigorous, we now introduce a probabilistic view on self-paced learning that views the weights $\boldsymbol{\nu}$ in the SPL objective (2.1) as probabilities of a distribution over samples.

More precisely, we define the categorical probability distribution $p(c=i|\boldsymbol{\nu}) = \nu_i$ for $i \in [1, N]$. Note that we restrict ourselves to discrete distributions $p(c=i|\boldsymbol{\nu})$ in this section to both ease the exposition and more easily establish connections to the SPL objective introduced in Section 2.3, although the results can be generalized to continuous distributions $\mu(\mathbf{c})$. For $p(c=i|\boldsymbol{\nu}) = \nu_i$ to be a valid probability distribution, we only need to introduce the constraint $\sum_{i=1}^N \nu_i = 1$, as $\nu_i \geq 0$ per definition of SPL. Hence, we rewrite the SPL objective (2.1) as

$$\boldsymbol{\nu}^*, \boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\nu} \in \Delta(N), \boldsymbol{\theta} \in \mathbb{R}^{d_\theta}} r(\boldsymbol{\theta}) + \mathbb{E}_{p(\mathbf{c}|\boldsymbol{\nu})} [l(\mathbf{x}_c, y_c, \boldsymbol{\theta})] + \sum_{i=1}^N f(\alpha, p(c=i|\boldsymbol{\nu})), \quad \alpha > 0, \quad (2.5)$$

where $\Delta(N) = \{\boldsymbol{\nu} \in \mathbb{R}_{\geq 0}^N, \sum_{i=1}^N \nu_i = 1\}$ is the N -dimensional probability simplex. Apart from changes in notation, the only difference to the SPL objective (2.1) is the constraint that forces the variables ν_i to sum to 1. Interestingly, this constraint does not just normalize the SPL weights obtained by optimizing objective (2.1) since the previously independent SPL weights $\nu^*(\alpha, l(\mathbf{x}_i, y_i, \boldsymbol{\theta}))$ are now coupled via the introduced normalization constraint. The seminal regularizer $f_{\text{Bin}}(\alpha, \nu_i) = -\alpha \nu_i$ explored by Kumar, Packer, and Koller [105]

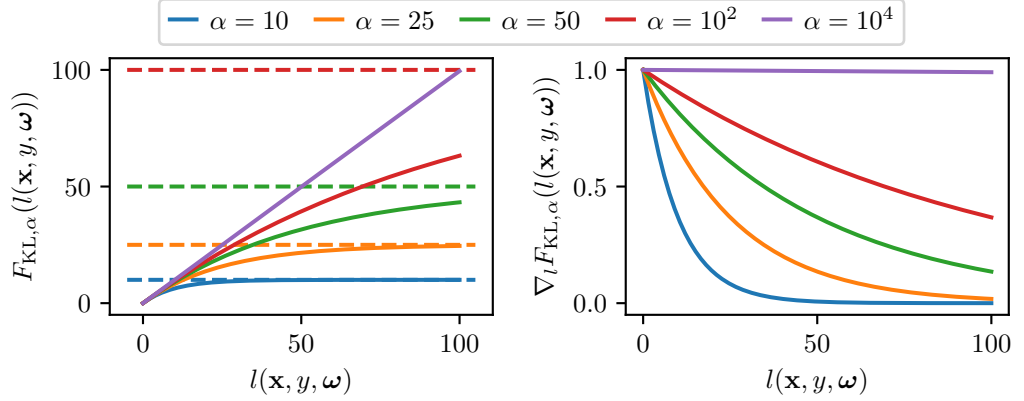


Figure 2.1.: A visualization of the effect of $F_{\text{KL},\alpha}$ (see Equation 2.7) for different values of α and a single data-point (\mathbf{x}, y) . The left plot shows the transformation of the model error $l(\mathbf{x}, y, \boldsymbol{\omega})$ by $F_{\text{KL},\alpha}$. The right plot shows the gradient of $F_{\text{KL},\alpha}$ w.r.t. $l(\mathbf{x}, y, \boldsymbol{\omega})$, i.e. the corresponding weight $\nu_{\text{KL}}^*(\alpha, \boldsymbol{\theta})$.

serves as an example of this behavior. With the additional constraint, the optimal solution ν_{Bin}^* to (2.5) simply puts all weight on the sample with the minimum loss instead of sampling uniformly among samples with a loss smaller than α . Although there seems to be no general connection between objectives (2.1) and (2.5) that holds for arbitrary self-paced regularizers, we can show that for the self-paced regularizer

$$f_{\text{KL},i}(\alpha, \nu_i) = \alpha \nu_i (\log(\nu_i) - \log(\mu(c=i))) - \alpha \nu_i, \quad (2.6)$$

the value of $\nu_{\text{KL},i}^*(\alpha, \boldsymbol{\theta})$ obtained by optimizing the default SPL objective (2.1) and its probabilistic counterpart (2.5) w.r.t. ν is identical up to a normalization constant. The user-chosen distribution $\mu(c)$ in the self-paced regularizer (2.6) represents the likelihood of (\mathbf{x}_c, y_c) occurring and has the same interpretation as in the regularized contextual RL objective (2.4). The corresponding function $F_{\text{KL},\alpha,i}$ is given by

$$F_{\text{KL},\alpha,i}(l(\mathbf{x}_i, y_i, \boldsymbol{\theta})) = \int_0^{l(\mathbf{x}_i, y_i, \boldsymbol{\theta})} \nu_{\text{KL},i}^*(\alpha, \nu) d\nu = \mu(c=i) \alpha \left(1 - \exp\left(-\frac{1}{\alpha} l(\mathbf{x}_i, y_i, \boldsymbol{\theta})\right) \right) \quad (2.7)$$

and is visualized in Figure 2.1. Note the additional subscript i in both $f_{\text{KL},i}$ and $F_{\text{KL},\alpha,i}$. This extra subscript arises due to the appearance of the likelihood term $\mu(c=i)$ in both formulas, resulting in an individual regularizer for each sample (\mathbf{x}_i, y_i) . As can be seen,

$F_{\text{KL},\alpha,i}(l)$ exhibits a squashing effect to limit the attained loss l to a maximum value of α . The closer the non-regularized loss l attains this maximum value of α , the more it is treated as a constant value by $F_{\text{KL},\alpha,i}(l)$. For l increasingly smaller than α , a change in the non-regularized loss l leads to an increasingly linear change in the regularized loss $F_{\text{KL},\alpha,i}(l)$. More interestingly, using $f_{\text{KL},i}(\alpha, \nu_i)$ in objective (2.5) results in a KL-Divergence penalty to $\mu(c)$. Theorem 1 summarizes these findings. The proof can be found in the appendix.

Theorem 1. *Alternatingly solving*

$$\min_{\boldsymbol{\theta}, \boldsymbol{\nu}} \mathbb{E}_{p(c|\boldsymbol{\nu})} [l(\mathbf{x}_c, y_c, \boldsymbol{\theta})] + \alpha D_{\text{KL}}(p(c|\boldsymbol{\nu}) \parallel \mu(c))$$

w.r.t. $\boldsymbol{\theta}$ and $\boldsymbol{\nu}$ is a majorize-minimize scheme applied to the regularized objective

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{\mu(c)} \left[\alpha \left(1 - \exp \left(-\frac{1}{\alpha} l(\mathbf{x}_c, y_c, \boldsymbol{\theta}) \right) \right) \right].$$

In the following section, we make use of the insights summarized in Theorem 1 to motivate a curriculum as an effective evaluation of the regularized RL objective (2.4) under the particular choice $F_\alpha = F_{\text{KL},\alpha,i}$.

2.5. Self-Paced Learning for Reinforcement Learning

Obtaining an efficient way of optimizing the regularized contextual RL objective (2.4) with $F_\alpha = F_{\text{KL},\alpha,i}$ is as easy as exploiting Theorem 1 to define the alternative objective

$$\max_{\boldsymbol{\theta} \in \Theta, \boldsymbol{\nu} \in \mathcal{N}} \mathbb{E}_{p(\mathbf{c}|\boldsymbol{\nu})} [J(\boldsymbol{\theta}, \mathbf{c})] - \alpha D_{\text{KL}}(p(\mathbf{c}|\boldsymbol{\nu}) \parallel \mu(\mathbf{c})).$$

As discussed in the previous section, this formulation introduces a way of computing the desired sampling distribution that efficiently evaluates objective (2.4) given the current agent parameters $\boldsymbol{\theta}$ by optimizing the above optimization problem w.r.t. $\boldsymbol{\nu}$. As discussed in Section 2.3, $p(\mathbf{c}|\boldsymbol{\nu})$ will assign probability mass to a context \mathbf{c} based on its contribution to the gradient of objective (2.4). Before we look at the application to RL problems, we will introduce a regularization that is an important ingredient to achieve practicality. More precisely, we introduce a KL divergence constraint between subsequent context distributions $p(\mathbf{c}|\boldsymbol{\nu})$ and $p(\mathbf{c}|\boldsymbol{\nu}')$, yielding

$$\begin{aligned} & \max_{\boldsymbol{\theta}, \boldsymbol{\nu}} \mathbb{E}_{p(\mathbf{c}|\boldsymbol{\nu})} [J(\boldsymbol{\theta}, \mathbf{c})] - \alpha D_{\text{KL}}(p(\mathbf{c}|\boldsymbol{\nu}) \parallel \mu(\mathbf{c})) \\ & \text{s.t. } D_{\text{KL}}(p(\mathbf{c}|\boldsymbol{\nu}) \parallel p(\mathbf{c}|\boldsymbol{\nu}')) \leq \epsilon, \end{aligned} \tag{2.8}$$

with ν' being the parameters of the previously computed context distribution. In a practical algorithm, this secondary regularization is important because the expected performance $J(\theta, \mathbf{c})$ is approximated by a learned value function, which may not predict accurate values for contexts not likely under $p(\mathbf{c}|\nu')$. The KL divergence constraint helps to avoid exploiting these false estimates too greedily.

Furthermore, it forces the distribution over contextual variables, and hence tasks, to gradually change. Assuming e.g. a Gaussian $p(\mathbf{c}|\nu)$, such a gradual change in distribution implies a gradual change in the sampled contexts \mathbf{c} . This gradual change fosters the extrapolation of learned behavior when the policy $\pi_\theta(\mathbf{a}|\mathbf{s}, \mathbf{c})$ is e.g. represented by (deep) neural networks, which typically exhibit a fair amount of continuity w.r.t. their inputs. Naturally, successful extrapolation of behavior also assumes that a small distance $\|\mathbf{c} - \mathbf{c}'\|$ implies a certain similarity between the tasks $\mathcal{M}(\mathbf{c})$ and $\mathcal{M}(\mathbf{c}')$. Note that the imprecision of this formulation is not by accident but is rather an acknowledgment that the question of similarity between MDPs is a complicated topic on its own.

From a theoretical perspective on SPL, the constraint changes the form of ν^* making it not only dependent on α and θ , but also on the previous parameter ν' . Although it may be possible to relate this modification to a novel regularizer $F_{\alpha,i}$, we do not pursue this idea here but rather connect objective (2.8) to the RL-as-inference perspective in Section 2.9, where we can show highly interesting similarities to the well-known concept of tempering in inference. To facilitate the intuition of the proposed curriculum and its usage, we, however, first present applications and evaluations in the following sections.

An important design decision for such applications is the schedule for α , i.e. the parameter of the regularizing function F_α . As can be seen in (2.8), α corresponds to the trade-off between reward maximization and progression to $\mu(\mathbf{c})$. In a supervised learning scenario, it is preferable to increase α as slowly as possible to gradually transform the objective from an easy version towards the target one. In an RL setting, each algorithm iteration requires the collection of data from the (real) system. Since the required amount of system interaction should be minimized, we cannot simply choose very small step sizes for α , as this would lead to a slower than necessary progression towards $\mu(\mathbf{c})$. In the implementations in sections 2.6 and 2.7, the parameter α is chosen such that the KL divergence penalty w.r.t. the current context distribution $p(\mathbf{c}|\nu_k)$ is in constant proportion ζ to the expected reward under this current context distribution and current policy parameters θ_k

$$\alpha_k = \mathcal{B}(\nu_k, \theta_k) = \zeta \frac{\mathbb{E}_{p(\mathbf{c}|\nu_k)} [J(\theta_k, \mathbf{c})]}{D_{\text{KL}}(p(\mathbf{c}|\nu_k) \parallel \mu(\mathbf{c}))}. \quad (2.9)$$

For the first K_α iterations, we set α to zero, i.e. only focus on maximizing the reward under $p(\mathbf{c}|\nu)$. In combination with an initial context distribution $p(\mathbf{c}|\nu_0)$ covering large

parts of the context space, this allows to tailor the context distribution to the learner in the first iterations by focusing on tasks in which it performs best under the initial parameters. Note that this schedule is a naive choice, that nonetheless worked sufficiently well in our experiments. In Section 2.8, we revisit this design choice and investigate it more carefully.

2.6. Application to Episodic Reinforcement Learning

In this section, we implement and evaluate our formulation of SPL for RL in a slightly different way than the “full” RL setting, which has been described in Chapter 1 and will be evaluated in the next section. Instead, we frame RL as a black-box optimization problem [36, 70]. This setting is interesting for two reasons: Firstly, it has been and still is a core approach to perform RL on real (robotic) systems [106, 155, 159], where “low-level” policies such as Dynamic- and Probabilistic Movement Primitives [174, 154] or PD-control laws [21] are commonly used to ensure smooth and stable trajectories while keeping the dimensionality of the search space reasonably small. Secondly, the different mechanics of the employed episodic RL algorithm and the resulting different implementation of objective (2.8) serve as another validation of our SPL approach to CRL apart from the deep RL experiments in the next section. Readers not interested in or familiar with the topic of black-box optimization (and episodic RL) can skip this section and continue to the experiments with deep RL algorithms in Section 2.7.

The episodic RL setting arises if we introduce an additional “low-level” policy $\pi_\omega(\mathbf{a}|\mathbf{s})$ with parameters $\omega \in \mathbb{R}^{d_\omega}$ and change the agent policy π to not generate actions given the current state and context, but only generate a parameter ω for the low-level policy given the current context, i.e. $\pi_\theta(\omega|\mathbf{c})$. Defining the expected reward for a parameter ω in context \mathbf{c} as

$$r(\omega, \mathbf{c}) = \mathbb{E}_{p_{0,\mathbf{c}}(\mathbf{s})} [V^\omega(\mathbf{s}, \mathbf{c})], \quad (2.10)$$

where $V^\omega(\mathbf{s}, \mathbf{c})$ is the value function defined in Section 1.3, we see that we can simply interpret $r(\omega, \mathbf{c})$ as a function that, due to its complicated nature, does only allow for noisy observations of its function value without any gradient information. The noise in function observations arises from the fact that a rollout of policy $\pi_\omega(\mathbf{a}|\mathbf{s})$ in a context \mathbf{c} corresponds to approximating the expectations in $r(\omega, \mathbf{c})$ with a single sample.

As a black-box optimizer for the experiments, we choose the contextual relative entropy policy search (C-REPS) algorithm [148, 106, 155], which frames the maximization of (2.10) over a task distribution $\mu(\mathbf{c})$ as a repeated entropy-regularized optimization

$$\max_{q(\omega, \mathbf{c})} \mathbb{E}_{q(\omega, \mathbf{c})} [r(\omega, \mathbf{c})] \quad \text{s.t.} \quad D_{\text{KL}}(q(\omega, \mathbf{c}) \parallel p(\omega, \mathbf{c})) \leq \epsilon \quad \int q(\omega, \mathbf{c}) \, d\omega = \mu(\mathbf{c}) \quad \forall \mathbf{c} \in \mathcal{C},$$

where $p(\boldsymbol{\omega}, \mathbf{c}) = p(\boldsymbol{\omega}|\mathbf{c})\mu(\mathbf{c})$ is the distribution obtained in the previous iteration. Note that the constraint in the above optimization problem implies that only the policy $q(\boldsymbol{\omega}|\mathbf{c})$ is optimized since the constraint requires that $q(\boldsymbol{\omega}, \mathbf{c}) = q(\boldsymbol{\omega}|\mathbf{c})\mu(\mathbf{c})$. This notation is common for this algorithm as it eases the derivations of the solution via the concept of Lagrangian multipliers. Furthermore, this particular form of the c-REPS algorithm allows for a straightforward incorporation of SPL, simply replacing the constraint $\int q(\boldsymbol{\omega}, \mathbf{c}) d\boldsymbol{\omega} = \mu(\mathbf{c})$ by a penalty term on the KL divergence between $q(\mathbf{c}) = \int q(\boldsymbol{\omega}, \mathbf{c}) d\boldsymbol{\omega}$ and $\mu(\mathbf{c})$

$$\begin{aligned} \max_{q(\boldsymbol{\omega}, \mathbf{c})} \mathbb{E}_{q(\boldsymbol{\omega}, \mathbf{c})} [r(\boldsymbol{\omega}, \mathbf{c})] - \alpha D_{\text{KL}}(q(\mathbf{c}) \parallel \mu(\mathbf{c})) \\ \text{s.t. } D_{\text{KL}}(q(\boldsymbol{\omega}, \mathbf{c}) \parallel p(\boldsymbol{\omega}, \mathbf{c})) \leq \epsilon. \end{aligned} \quad (2.11)$$

The above objective does not yet include the parameters $\boldsymbol{\theta}$ or $\boldsymbol{\nu}$ of the policy or the context distribution to be optimized. This is because both c-REPS and also our implementation of SPL for episodic RL solve the above optimization problem analytically to obtain a re-weighting scheme for samples $(\boldsymbol{\omega}_i, \mathbf{c}_i) \sim p(\boldsymbol{\omega}|\mathbf{c}, \boldsymbol{\theta}_k)p(\mathbf{c}|\boldsymbol{\nu}_k)$ based on the observed rewards $r(\boldsymbol{\omega}_i, \mathbf{c}_i)$. The next parameters $\boldsymbol{\theta}_{k+1}$ and $\boldsymbol{\nu}_{k+1}$ are then found by a maximum-likelihood fit to the set of weighted samples. The following section will detail some of the practical considerations necessary for this.

2.6.1. Algorithmic Implementation

Solving (2.11) analytically using the technique of Lagrangian multipliers, we obtain the following form for the variational distributions

$$\begin{aligned} q(\boldsymbol{\omega}, \mathbf{c}) \propto p(\boldsymbol{\omega}, \mathbf{c}|\boldsymbol{\theta}_k, \boldsymbol{\nu}_k) \exp\left(\frac{r(\boldsymbol{\omega}, \mathbf{c}) - V(\mathbf{c})}{\eta_q}\right) &= p(\boldsymbol{\omega}, \mathbf{c}|\boldsymbol{\theta}_k, \boldsymbol{\nu}_k) \exp\left(\frac{A(\boldsymbol{\omega}, \mathbf{c})}{\eta_q}\right), \quad (2.12) \\ q(\mathbf{c}) \propto p(\mathbf{c}|\boldsymbol{\nu}_k) \exp\left(\frac{V(\mathbf{c}) + \alpha(\log(\mu(\mathbf{c})) - \log(p(\mathbf{c}|\boldsymbol{\nu}_k)))}{\alpha + \eta_{\bar{q}}}\right) &= p(\mathbf{c}|\boldsymbol{\nu}_k) \exp\left(\frac{\beta(\mathbf{c})}{\alpha + \eta_{\bar{q}}}\right), \quad (2.13) \end{aligned}$$

with $\eta_q, \eta_{\bar{q}}$ as well as $V(\mathbf{c})$ being Lagrangian multipliers that are found by solving the dual objective

$$\mathcal{G} = (\eta_q + \eta_{\bar{q}})\epsilon + \eta_q \log\left(\mathbb{E}_p\left[\exp\left(\frac{A(\boldsymbol{\omega}, \mathbf{c})}{\eta_q}\right)\right]\right) + (\alpha + \eta_{\bar{q}}) \log\left(\mathbb{E}_p\left[\exp\left(\frac{\beta(\mathbf{c})}{\alpha + \eta_{\bar{q}}}\right)\right]\right). \quad (2.14)$$

The derivation of the dual objective, as well as the solution to objective (2.11), are shown in the appendix. As previously mentioned, in practice the algorithm has only access to a

Algorithm 1 Self-Paced Episodic Reinforcement Learning (SPRL)

Input: Initial context distribution- and policy parameters ν_0 and θ_0 , Target context distribution $\mu(\mathbf{c})$, KL penalty proportion ζ , Offset K_α , Number of iterations K , Rollouts per policy update M , Relative entropy bound ϵ

for $k = 1$ **to** K **do**

Collect Data:

 Sample contexts: $\mathbf{c}_i \sim p(\mathbf{c}|\nu_{k-1})$, $i \in [1, M]$

 Sample parameters: $\omega_i \sim p(\omega|\mathbf{c}_i, \theta_{k-1})$

 Execute $\pi_{\omega_i}(\cdot|\mathbf{s})$ in \mathbf{c}_i and observe reward: $r_i = r(\omega_i, \mathbf{c}_i)$

 Create sample set: $\mathcal{D}_k = \{(\omega_i, \mathbf{c}_i, r_i)|i \in [1, M]\}$

Update Policy and Context Distributions:

 Update schedule: $\alpha_k = 0$, if $k \leq K_\alpha$, else $\mathcal{B}(\nu_{k-1}, \theta_{k-1})$ (2.9)

 Optimize dual function: $[\eta_q^*, \eta_{\bar{q}}^*, V^*] \leftarrow \arg \min \mathcal{G}(\eta_q, \eta_{\bar{q}}, V)$ (2.14)

 Calculate sample weights: $[w_i, \tilde{w}_i] \leftarrow \left[\exp\left(\frac{A(\omega_i, \mathbf{c}_i)}{\eta_q^*}\right), \exp\left(\frac{\beta(\mathbf{c}_i)}{\alpha_k + \eta_{\bar{q}}^*}\right) \right]$ (2.12), (2.13)

 Infer new parameters: $[\theta_k, \nu_k] \leftarrow \{(\omega_i, \tilde{w}_i, \mathbf{c}_i)|i \in [1, M]\}$

end for

set of samples $\mathcal{D} = \{(\omega_i, \mathbf{c}_i, r_i)|i \in [1, M]\}$ and hence the analytic solutions (2.12) and (2.13) are approximated by re-weighting the samples via weights w_i . To compute the optimal weights w_i , the multipliers V^* , η_q^* , and $\eta_{\bar{q}}^*$ need to be obtained by minimizing the dual (2.14), to which two approximations are introduced: First, the expectations w.r.t. $p(\omega, \mathbf{c}|\theta, \nu)$ (abbreviated as p in Equation 2.14) are replaced by a sample-estimate from the collected samples in \mathcal{D} . Second, we introduce a parametric form for the value function $V(\mathbf{c}) = \chi^T \phi(\mathbf{c})$ with a user-chosen feature function $\phi(\mathbf{c})$, such that we can optimize (2.14) w.r.t. χ instead of V .

After finding the minimizers χ^* , η_q^* , and $\eta_{\bar{q}}^*$ of (2.14), the weights w_i are then given by the exponential terms in (2.12) and (2.13). The new policy- and context distribution parameters are fitted via maximum likelihood to the set of weighted samples. In our implementation, we use Gaussian context distributions and policies. To account for the error that originates from the sample-based approximation of the expectations in (2.14), we enforce the KL divergence constraint $D_{\text{KL}}(p(\omega, \mathbf{c}|\theta_k, \nu_k) \parallel q(\omega, \mathbf{c}|\theta_{k+1}, \nu_{k+1})) \leq \epsilon$ when updating the policy and context distribution. Again, details on this maximum likelihood step can be found in the appendix. To compute the schedule for α according to (2.9), we approximate the expected reward under the current policy with the mean of the observed rewards, i.e. $\mathbb{E}_{p(\mathbf{c}|\nu_k)}[J(\theta_k, \mathbf{c})] \approx \frac{1}{M} \sum_{i=1}^M r_i$. The overall procedure is summarized in Algorithm 1.

2.6.2. Experiments

We now evaluate the benefit of the SPL paradigm in the episodic RL scenario (SPRL). Besides facilitating learning on a diverse set of tasks, we are also interested in the idea of facilitating the learning of a hard target task via a curriculum. This modulation can be achieved by choosing $\mu(\mathbf{c})$ to be a narrow probability distribution focusing nearly all probability density on the particular target task. To judge the benefit of our SPL adaptation for these endeavors, we compared our implementation to C-REPS, CMA-ES [69], GOALGAN [56] and SAGG-RIAC [15]. With CMA-ES being a non-contextual algorithm, we only use it in experiments with narrow target distributions, where we then train and evaluate only on the mean of the target context distributions. We will start with a simple point-mass problem, where we evaluate the benefit of our algorithm for broad and narrow target distributions. We then turn towards more challenging tasks, such as a modified version of the reaching task implemented in the OpenAI Gym simulation environment [27] and a sparse ball-in-a-cup task. Given that GOALGAN and SAGG-RIAC are algorithm agnostic curriculum generation approaches, we combine them with C-REPS to make the results as comparable as possible.

In all experiments, we use radial basis function (RBF) features to approximate the value function $V(\mathbf{c})$, while the policy $p(\boldsymbol{\omega}|\mathbf{c}, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\omega}|\mathbf{A}_{\boldsymbol{\theta}}\boldsymbol{\phi}(\mathbf{c}), \boldsymbol{\Sigma}_{\boldsymbol{\theta}})$ uses linear features $\boldsymbol{\phi}(\mathbf{c})$. SPRL and C-REPS always use the same number of RBF features for a given environment. SPRL always starts with a wide initial sampling distribution $p(\mathbf{c}|\nu_0)$ that, in combination with setting $\alpha = 0$ for the first K_{α} iterations, allows the algorithm to automatically choose the initial tasks on which learning should take place. After the first K_{α} iterations, we then choose α following the scheme outlined in the previous section. Experimental details that are not mentioned here to keep the section short can be found in the appendix.¹

Point-Mass Environment

In the first environment, the agent needs to steer a point-mass in a two-dimensional space from the starting position $[0 \ 5]$ to the goal position at the origin. The dynamics of the point-mass are described by a simple linear system subject to a small amount of Gaussian noise. Complexity is introduced by a wall at height $y = 2.5$, which can only be traversed through a gate. The x -position and width of the gate together define a task \mathbf{c} . If the point-mass crashes into the wall, the experiment is stopped and the reward is computed based on the current position. The reward function is the exponentiated negative distance to the goal position with additional L2-Regularization on the generated actions. The point-mass is controlled by two linear controllers, whose parameters need to

¹Code is publicly available under <https://github.com/pscl1nk/self-paced-rl>.

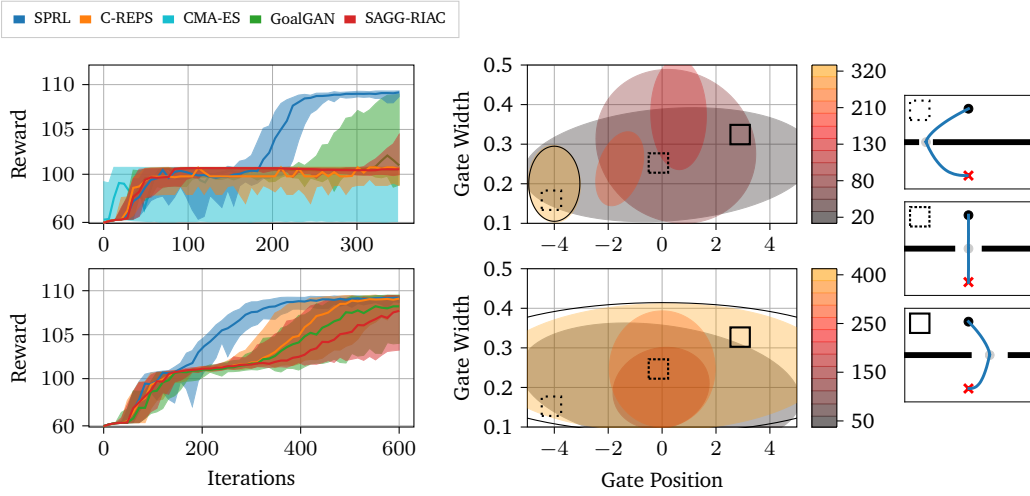


Figure 2.2.: Left: Reward in the “precision” (top row) and “global” setting (bottom row) on the target context distributions in the gate environment. Thick lines show the 50%-quantiles and shaded areas the intervals from 10%- to 90%-quantile for 40 seeds. Middle: Evolution of the sampling distribution $p(c|\nu)$ (colored areas) of one SPRL run together with the target distribution $\mu(c)$ (black line). Right: Task visualizations for different gate positions and widths. Boxes mark the corresponding positions in the context space.

be tuned by the agent. The controllers are switched as soon as the point-mass reaches the height of the gate, which is why the desired y -position of the controllers are fixed to 2.5 (the height of the gate) and 0, while all other parameters are controlled by the policy π , making ω a 14-dimensional vector. We evaluate two setups in this gate environment, which differ in their target context distribution $\mu(c)$: In the first one, the agent needs to be able to steer through a very narrow gate far from the origin (“precision”) and in the second it is required to steer through gates with a variety of positions and widths (“global”). The two target context distributions are shown in Figure 2.2. Figure 2.2 further visualizes the obtained rewards for the investigated algorithms, the evolution of the sampling distribution $p(c|\nu)$ as well as tasks from the environment. In the “global” setting, we can see that SPRL converges significantly faster to the optimum than the other algorithms while in the “precision” setting, SPRL avoids a local optimum to which C-REPS and CMA-ES converge and which, as can be seen in Figure 2.3, does not encode desirable behavior. Furthermore, both curriculum learning algorithms SAGG-RIAC and GOALGAN only slowly escape this local optimum in the “precision” setting. We hypothesize that this slow convergence to the optimum is caused by SAGG-RIAC and GOALGAN not having a notion of

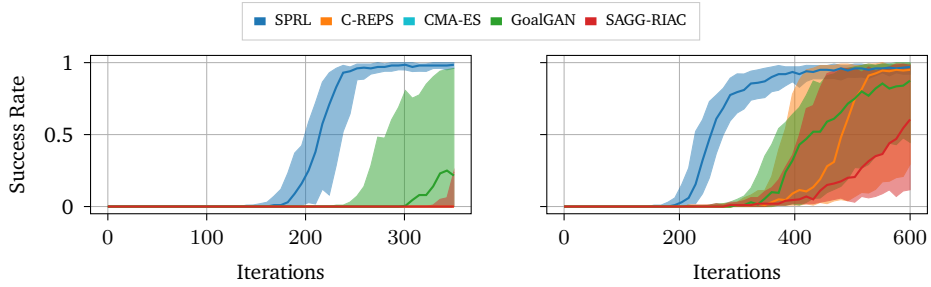


Figure 2.3.: Success rates in the “precision” (left) and “global” setting (right) of the gate environment. Thick lines represent the 50%-quantiles and shaded areas show the intervals from 10%- to 90%-quantile. Quantiles are computed using 40 algorithm executions.

a target distribution. Hence, these algorithms cannot guide the sampling of contexts to sample relevant tasks according to $\mu(\mathbf{c})$. This is especially problematic if $\mu(\mathbf{c})$ covers only a small fraction of the context space with a non-negligible probability density. The visualized sampling distributions in Figure 2.2 indicate that tasks with wide gates positioned at the origin seem to be easier to solve starting from the initially zero-mean Gaussian policy, as in both settings SPRL first focuses on these kinds of tasks and subsequently changes the sampling distributions to match $\mu(\mathbf{c})$. Interestingly, the search distribution of CMA-ES did not always converge in the “precision” setting, as shown in Figure 2.2. This behavior persisted across hyperparameters and population sizes.

Reacher Environment

For the next evaluation, we modify the three-dimensional reacher environment of the OpenAI Gym toolkit [27]. In our version, the goal is to move the end-effector along the surface of a table towards the goal position while avoiding obstacles that are placed on the table. With the obstacles becoming larger, the robot needs to introduce a more pronounced curve movement to reach the goal without collisions. To simplify the visualization of the task distribution, we only allow two of the four obstacles to vary in size. The sizes of those two obstacles make up a task \mathbf{c} in this environment. Just as in the first environment, the robot should not crash into the obstacles, and hence the movement is stopped if one of the four obstacles is touched. The policy π encodes a ProMP [154], from which movements are sampled during training. In this task, ω is a 40-dimensional vector.

Looking at Figure 2.4, we can see that C-REPS and CMA-ES find a worse optimum compared to SPRL. This local optimum does—just as in the previous experiment—not encode optimal behavior, as we can see in Figure 2.5. GOALGAN and SAGG-RIAC tend to find the same

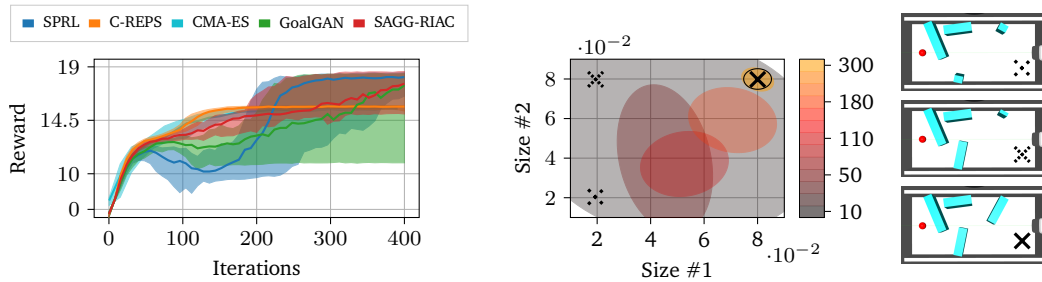


Figure 2.4.: Left: 50%-quantiles (thick lines) and intervals from 10%- to 90%-quantile (shaded areas) of the reward in the reacher environment. Quantiles are computed over 40 algorithm runs. Middle: The sampling distribution $p(c|\nu)$ at different iterations (colored areas) of one SPRL run together with the target distribution (black line). Right: Task visualizations for different contexts with black crosses marking the corresponding positions in context space.

optimum as SPRL, however with slower convergence. This is nonetheless surprising given that—just as for the “precision” setting of the previous experiment—the algorithm deals with a narrow target context distribution. Although the 10%-90% quantile of SAGG-RIAC and GOALGAN contain policies that do not manage to solve the task (i.e. are below the performance of C-REPS), the performance is in stark contrast to the performance in the previously discussed “precision” setting, in which the majority of runs did not solve the task. Nonetheless, the 10%-50% quantile of the performance displayed in Figure 2.4 still indicates the expected effect that SPRL leverages the knowledge of the target distribution to yield faster convergence to the optimal policy in the median case.

Another interesting artifact is the initial decrease in performance of SPRL between iterations 50 – 200. This can be accounted to the fact that in this phase, the intermediate distribution $p(c|\nu)$ only assigns negligible probability density on areas covered by $\mu(c)$ (see Figure 2.4). Hence the agent performance on $\mu(c)$ during this stage is completely dependent on the extrapolation behavior of the agent, which seems to be rather poor in this setting. This once more illustrates the importance of appropriate transfer of behavior between tasks, which is, however, out of the scope of this chapter.

The sampling distributions visualized in Figure 2.4 indicate that SPRL focuses on easier tasks with smaller obstacle sizes first and then moves on to the harder, desired tasks. Figure 2.5 also shows that PPO [177], a step-based reinforcement learning algorithm, is not able to solve the task after the same amount of interaction with the environment, emphasizing the complexity of the learning task.

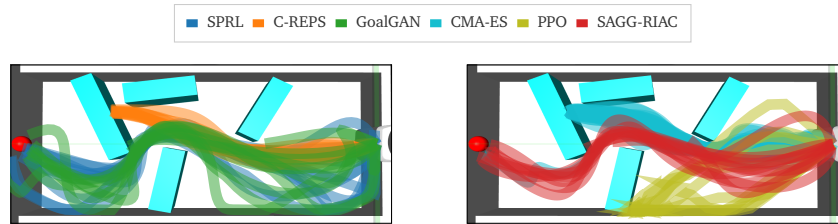


Figure 2.5.: Trajectories generated by final policies learned with different algorithms in the reacher environment. The trajectories should reach the red dot while avoiding the cyan boxes. Please note that the visualization is not completely accurate, as we did not account for the viewpoint of the simulation camera when plotting the trajectories.

Sparse Ball-in-a-Cup

We conclude this experimental evaluation with a ball-in-a-cup task, in which the reward function exhibits a significant amount of sparsity by only returning a reward of 1 minus an L2 regularization term on the policy parameters, if the ball is in the cup after the policy execution, and 0 otherwise. The robotic platform is a Barrett WAM, which we simulate using the MuJoCo physics engine [195]. The policy represents again a ProMP encoding the desired position of the first, third and fifth joint of the robot. Achieving the desired task with a poor initial policy is an unlikely event, leading to mostly uninformative rewards and hence poor learning progress. However, as can be seen in Figure 2.6, giving the learning

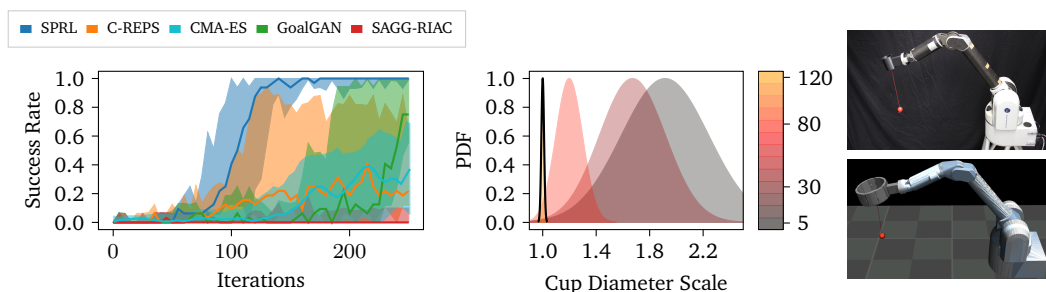


Figure 2.6.: Left: 50%-quantiles (thick lines) and intervals from 10%- to 90%-quantile (shaded areas) of the success rates for the sparse ball-in-a-cup task. Quantiles are computed from the 10 best runs out of 20. Middle: The sampling distribution $p(c|\nu)$ at different iterations (colored areas) of one SPRL run together with the target distribution $\mu(c)$ (black line). Right: Task visualization on the real robot (upper) and in simulation with a scale of 2.5 (lower).

agent control over the diameter of the cup significantly improves the learning progress by first training with larger cups and only progressively increasing the precision of the movement to work with smaller cups. Having access to only 16 samples per iteration, the algorithms did not always learn to achieve the task. However, the final policies learned by SPRL outperform the ones learned by C-REPS, CMA-ES, GOALGAN and SAGG-RIAC. The movements learned in simulation were finally applied to the robot with a small amount of fine-tuning.

2.7. Application to Step-Based Reinforcement Learning

The experiments in the previous section demonstrate that the self-paced learning paradigm can indeed be beneficial in the episodic RL—or black-box optimization—setting, so that as a next step we want to investigate its application when using a stochastic policy of the form $\pi_\theta(\mathbf{a}|\mathbf{s}, \mathbf{c})$. In this setting, we derive an implementation of SPL that is agnostic to the RL algorithm of choice by using the possibility of updating the SPL objective (2.8) in a block-coordinate manner w.r.t. θ and ν . The resulting approximate implementation allows to create learning agents following the SPL paradigm using arbitrary RL algorithms by making use of the value functions that the RL algorithms estimate during policy optimization.

2.7.1. Algorithmic Implementation

Optimizing objective (2.8) w.r.t. the policy parameters θ using an RL algorithm of choice under the current context distribution $p(\mathbf{c}|\nu_k)$ generates a data set \mathcal{D}_k of trajectories

$$\mathcal{D}_k = \{(\mathbf{c}_i, \tau_i) \mid \mathbf{c}_i \sim p(\mathbf{c}|\nu_k), \tau_i \sim p(\tau|\mathbf{c}_i, \theta_k), i \in [1, M]\},$$

where the distribution $p(\tau|\mathbf{c}_i, \theta_k)$ is defined in Section 1.3. One unifying property of many RL algorithms is their reliance on estimating the state-value function $V^\theta(\mathbf{s}, \mathbf{c})$, each in their respective way, as a proxy to optimizing the policy. We make use of this approximated value function $\tilde{V}^\theta(\mathbf{s}, \mathbf{c})$ (note the \sim indicating the approximation) to compute an estimate of the expected performance $J(\theta, \mathbf{c}_i) = \mathbb{E}_{p_0, \mathbf{c}_i(\mathbf{s}_0)} [V^\theta(\mathbf{s}_0, \mathbf{c}_i)] \approx \tilde{V}^\theta(\mathbf{s}_{i,0}, \mathbf{c}_i)$ in context \mathbf{c}_i , where $\mathbf{s}_{i,0}$ is the initial state of trajectory τ_i . This yields an approximate form of objective (2.8) given by

$$\begin{aligned} \max_{\nu_{k+1}} \frac{1}{M} \sum_{i=1}^M \frac{p(\mathbf{c}_i|\nu_{k+1})}{p(\mathbf{c}_i|\nu_k)} \tilde{V}^\theta(\mathbf{s}_{i,0}, \mathbf{c}_i) - \alpha_k D_{\text{KL}}(p(\mathbf{c}|\nu_{k+1}) \parallel \mu(\mathbf{c})) \\ \text{s.t. } D_{\text{KL}}(p(\mathbf{c}|\nu_{k+1}) \parallel p(\mathbf{c}|\nu_k)) \leq \epsilon. \end{aligned} \quad (2.15)$$

Algorithm 2 Self-Paced Deep Reinforcement Learning (SPDL)

Input: Initial context distribution- and policy parameters ν_0 and θ_0 , Target context distribution $\mu(\mathbf{c})$, KL penalty proportion ζ and offset K_α , Number of iterations K , Rollouts per policy update M , Relative entropy bound ϵ

for $k = 1$ **to** K **do**

Agent Improvement:

 Sample contexts: $\mathbf{c}_i \sim p(\mathbf{c}|\nu_k)$, $i \in [1, M]$

 Rollout trajectories: $\tau_i \sim p(\tau|\mathbf{c}_i, \theta_k)$, $i \in [1, M]$

 Obtain θ_{k+1} from RL algorithm of choice using $\mathcal{D}_k = \{(\mathbf{c}_i, \tau_i)|i \in [1, M]\}$

 Estimate $\tilde{V}^{\theta_{k+1}}(s_{i,0}, \mathbf{c}_i)$ (or use estimate of RL agent) for contexts \mathbf{c}_i

Context Distribution Update:

IF $k \leq K_\alpha$: Obtain ν_{k+1} from (2.15) with $\alpha_k = 0$

ELSE: Obtain ν_{k+1} optimizing (2.15), using $\alpha_k = \mathcal{B}(\nu_k, \mathcal{D}_k)$ (2.9)

end for

The first term in objective (2.15) is an approximation to $\mathbb{E}_{p(\mathbf{c}|\nu_{k+1})}[J(\theta, \mathbf{c})]$ via importance-weights. The above objective can be solved using any constrained optimization algorithm. In our implementation, we use the trust-region algorithm implemented in the SciPy library [204]. The two KL divergences in (2.15) can be computed in closed form since $\mu(\mathbf{c})$ and $p(\mathbf{c}|\nu)$ are Gaussians in our implementations. However, for more complicated distributions, the divergences can also be computed using samples from the respective distributions and the corresponding (unnormalized) log-likelihoods. The resulting approach (SPDL) is summarized in Algorithm 2.

2.7.2. Experiments

We evaluate SPDL in three different environments (Figure 2.7) with different deep RL (DRL) algorithms: TRPO [178], PPO [177], and SAC [66]. For all DRL algorithms, we use the implementations from the Stable Baselines library [73].²

The first environment for testing SPDL is again a point-mass environment but with an additional parameter to the context space, as we will detail in the corresponding section. The second environment extends the point-mass experiment by replacing the point-mass with a torque-controlled quadruped ‘ant’, thus increasing the complexity of the underlying control problem and requiring the capacity of deep neural network function approximators used in DRL algorithms. Both environments focus on learning a specific hard target task.

²Code for running the experiments can be found at <https://github.com/psc1k1nk/spdl>.

The final environment is a robotic ball-catching environment. This environment constitutes a shift in curriculum paradigm as well as reward function. Instead of guiding learning towards a specific target task, this third environment requires to learn a ball-catching policy over a wide range of initial states (ball position and velocity). The reward function is sparse compared to the dense ones employed in the first two environments. To judge the performance of SPDL, we compare the obtained results to state-of-the-art CRL algorithms ALP-GMM [162], which is based on the concept of Intrinsic Motivation, and GOALGAN [56], which

relies on the notion of a success indicator to define a curriculum. Further, we also compare to curricula consisting of tasks uniformly sampled from the context space (referred to as ‘Random’ in the plots) and learning without a curriculum (referred to as ‘Default’). Additional details and qualitative evaluations of them can be found in the appendix.

Point-Mass Environment

As previously mentioned, we again focus on a point-mass environment, where now the control policy is a neural network. Furthermore, the contextual variable $\mathbf{c} \in \mathbb{R}^3$ now changes the width and position of the gate as well as the dynamic friction coefficient of the ground on which the point-mass slides. The target context distribution $\mu(\mathbf{c})$ is a narrow Gaussian with a negligible variance that encodes a small gate at a specific position and a dynamic friction coefficient of 0. Figure 2.7 shows two different instances of the environment, one of them being the target task.

Figure 2.8 shows the results of two different experiments in this environment, one where the curriculum is generated over the full three-dimensional context space and one in which the friction parameter is fixed to its target value of 0 so that the curriculum is generated only in a two-dimensional subspace. As Figure 2.8 and Table 2.1 indicate, SPDL significantly increases the asymptotic reward on the target task compared to other methods. Increasing the dimension of the context space harms the performance of the other CRL

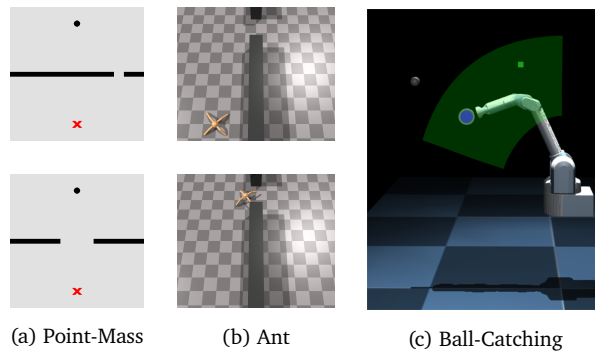


Figure 2.7.: Environments used for experimental evaluation. For the point-mass environment (a), the upper plot shows the target task. The shaded areas in picture (c) visualize the target distribution of ball positions (green) as well as the ball positions for which the initial policy succeeds (blue).

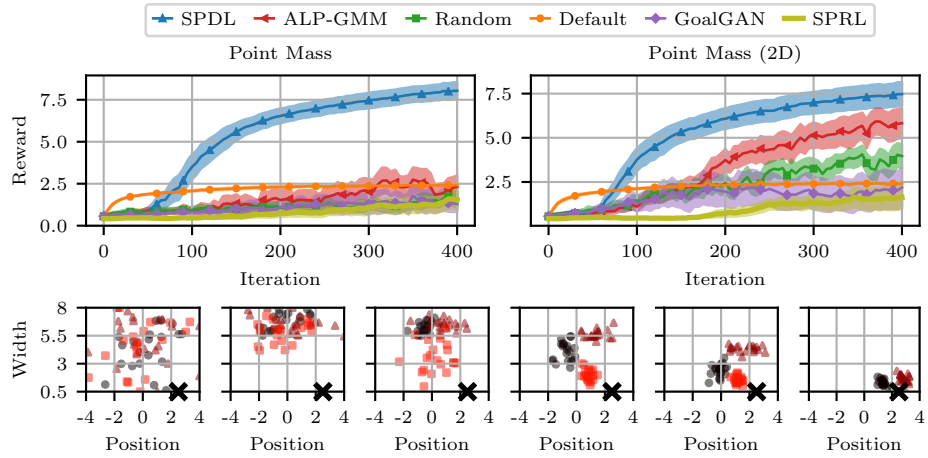


Figure 2.8.: Reward of different curricula in the point-mass (2D and 3D) environment for TRPO. Mean (thick line) and two times standard error (shaded area) is computed from 20 algorithm runs. The lower plots show samples from the context distributions $p(\mathbf{c}|\nu)$ in the point-mass 2D environment at iterations 0, 20, 30, 50, 65 and 120 (from left to right). Different colors and shapes of samples indicate different algorithm runs. The black cross marks the mean of the target distribution $\mu(\mathbf{c})$.

algorithms. For SPDL, there is no statistically significant difference in performance across the two settings. This observation is in line with the hypothesis posed in Section 2.6.2, that SPDL leverages the notion of $\mu(\mathbf{c})$ compared to other CRL algorithms that are not aware of it. As the context dimension increases, the volume of those parts in context space that carry non-negligible probability density according to $\mu(\mathbf{c})$ become smaller and smaller compared to the volume of the whole context space. Hence curricula that always target the whole context space tend to spend less time training on tasks that are relevant under $\mu(\mathbf{c})$. By having a notion of a target distribution, SPDL ultimately samples contexts that are likely according to $\mu(\mathbf{c})$, regardless of the dimension. The context distributions $p(\mathbf{c}|\nu)$ visualized in Figure 2.8 show that the agent focuses on wide gates in a variety of positions in early iterations. Subsequently, the size of the gate is decreased and the position of the gate is shifted to match the target one. This process is carried out at different paces and in different ways, sometimes preferring to first shrink the width of the gate before moving its position while sometimes doing both simultaneously. More interestingly, the behavior of the curriculum is consistent with the one observed in Section 2.6.2. We further see that the episodic version (SPRL), which we applied by defining the episodic RL policy $p_{\theta}(\omega|\mathbf{c})$ to choose the weights ω of a policy network for a given context \mathbf{c} , learns much

slower compared to its step-based counterpart, requiring up to 800 iterations to reach an average reward of 5 (only the first 400 are shown in Figure 2.8). To keep the dimension of the context space moderate, the policy network for SPRL consisted of one layer of 21 tanh-activated hidden units, leading to 168 and 189 parameter dimensions in the two 2D and 3D context space instances. We also evaluated SPDL with this particular policy architecture, still significantly outperforming SPRL with an average reward of around 8 after 800 iterations.

Ant Environment

We replace the point-mass in the previous environment with a four-legged ant similar to the one in the OpenAI Gym simulation environment [27].³ The goal is to reach the other side of a wall by passing through a gate, whose width and position are determined by the contextual variable $\mathbf{c} \in \mathbb{R}^2$ (see Figure 2.7). We only evaluated the CRL algorithms using PPO since the implementations of TRPO and SAC in the Stable-Baselines library do not allow to make use of the parallelization capabilities of the Isaac Gym simulator, leading to prohibitive running times (details in the appendix).

Looking at Figure 2.9, we see that SPDL allows the learning agent to escape the local optimum which results from the agent not finding the gate to pass through. ALP-GMM and a random curriculum do not improve the reward over directly learning on the target task. However, as we show in the appendix, both ALP-GMM and a random curriculum improve the qualitative performance, as they sometimes allow the ant to move through the gate. Nonetheless, this behavior is less efficient than the one learned by GOALGAN and SPDL, causing the action penalties in combination with the discount factor to prevent this better behavior from being reflected in the reward.

Ball-Catching Environment

Due to a sparse reward function and a broad target task distribution, this final environment is drastically different from the previous ones. In this environment, the agent needs to control a Barrett WAM robot to catch a ball thrown towards it. The reward function is sparse, only rewarding the robot when it catches the ball and penalizing excessive movements. In the simulated environment, the ball is considered caught if it is in contact with the end effector. The context $\mathbf{c} \in \mathbb{R}^3$ parameterizes the distance to the robot from which the ball is thrown as well as its target position in a plane that intersects the base of the robot. Figure 2.7 shows the robot as well as the target distribution over the ball

³We use the Nvidia Isaac Gym simulator [151] for this experiment.

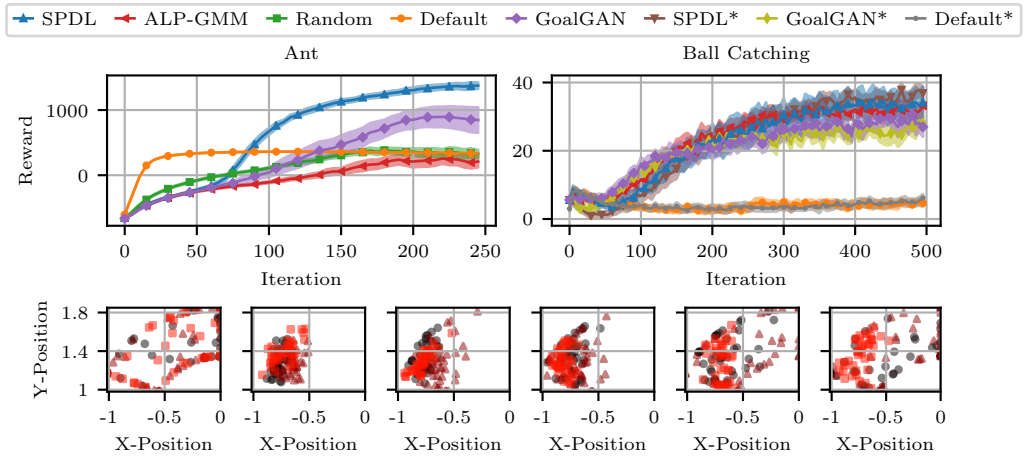


Figure 2.9.: Mean (thick line) and two times standard error (shaded area) of the reward achieved with different curricula in the ant environment for PPO and in the ball-catching environment for SAC (upper plots). The statistics are computed from 20 seeds. For ball-catching, runs of SPDL/GOALGAN with an initialized context distribution and runs of Default learning without policy initialization are indicated by asterisks. The lower plots show ball positions in the ‘catching’ plane sampled from the context distributions $p(c|\nu)$ in the ball-catching environment at iterations 0, 50, 80, 110, 150 and 200 (from left to right). Different sample colors and shapes indicate different algorithm runs. Given that $p(c|\nu)$ is initialized with $\mu(c)$, the samples in iteration 0 visualize the target distribution.

positions in the aforementioned ‘catching’ plane. The context c is not visible to the policy, as it only changes the initial state distribution $p(s_0)$ via the encoded target position and initial distance to the robot. Given that the initial state is already observed by the policy, observing the context is superfluous. To tackle this learning task with a curriculum, we initialize the policy of the RL algorithms to hold the robot’s initial position. This creates a subspace in the context space in which the policy already performs well, i.e. where the target position of the ball coincides with the initial end effector position. This can be leveraged by CRL algorithms.

Since SPDL and GOALGAN support to specify the initial context distribution, we investigate whether this feature can be exploited by choosing the initial context distribution to encode the aforementioned tasks in which the initial policy performs well. When directly learning on the target context distribution without a curriculum, it is not clear whether the policy initialization benefits learning. Hence, we evaluate the performance both with and without a pre-trained policy when not using a curriculum.

	PPO (P3D)	SAC (P3D)	PPO (P2D)	SAC (P2D)	TRPO (BC)	PPO (BC)
ALP-GMM	2.43 ± 0.3	4.68 ± 0.8	5.23 ± 0.4	5.11 ± 0.7	39.8 ± 1.1	46.5 ± 0.7
GOALGAN	0.66 ± 0.1	2.14 ± 0.6	1.63 ± 0.5	1.34 ± 0.4	42.5 ± 1.6	42.6 ± 2.7
GOALGAN *	-	-	-	-	45.8 ± 1.0	45.9 ± 1.0
SPDL	8.45 ± 0.4	6.85 ± 0.8	8.94 ± 0.1	5.67 ± 0.8	47.0 ± 2.0	53.9 ± 0.4
SPDL *	-	-	-	-	43.3 ± 2.0	49.3 ± 1.4
Random	0.67 ± 0.1	2.70 ± 0.7	2.49 ± 0.3	4.99 ± 0.8	-	-
Default	2.40 ± 0.0	2.47 ± 0.0	2.37 ± 0.0	2.40 ± 0.0	21.0 ± 0.3	22.1 ± 0.3
Default *	-	-	-	-	21.2 ± 0.3	23.0 ± 0.7

Table 2.1.: Average final reward and standard error of different curricula and RL algorithms in the two point-mass environments with three (P3D) and two (P2D) context dimensions as well as the ball-catching environment (BC). The data is computed from 20 algorithm runs. Significantly better results according to Welch’s t-test with $p < 1\%$ are highlighted in bold. The asterisks mark runs of SPDL/GOALGAN with an initialized context distribution and runs of default learning without policy initialization.

Figure 2.9 and Table 2.1 show the performance of the investigated curriculum learning approaches. We see that sampling tasks directly from the target distribution does not allow the agent to learn a meaningful policy, regardless of the initial one. Further, all curricula enable learning in this environment and achieve a similar reward. The results also highlight that initialization of the context distribution slightly improves performance for GOALGAN while slightly reducing performance for SPDL. The context distributions $p(\mathbf{c}|\nu)$ visualized in Figure 2.9 indicate that SPDL shrinks the initially wide context distribution in early iterations to recover the subspace of ball target positions, in which the initial policy performs well. From there, the context distribution then gradually matches the target one. As in the point-mass experiment, this progress takes place at a differing pace, as can be seen in the visualizations of $p(\mathbf{c}|\nu)$ in Figure 2.9 for iteration 200: Two of the three distributions fully match the target distribution while the third only covers half of it. The similar performance across curriculum learning methods is indeed interesting. Clearly, the wide target context distribution $\mu(\mathbf{c})$ better matches the implicit assumptions made by both ALP-GMM and GOALGAN that learning should aim to accomplish tasks in the whole context space. However, both ALP-GMM and GOALGAN are built around the idea to sample tasks that promise a maximum amount of learning progress, typically avoiding to sample tasks that the agent can already solve. SPDL achieves the same performance by simply growing the sampling distribution over time, not at all avoiding to sample tasks that the agent has mastered. Hence, a promising direction for further improving the performance of CRL methods is to combine ideas of SPDL and methods such as ALP-GMM and GOALGAN.

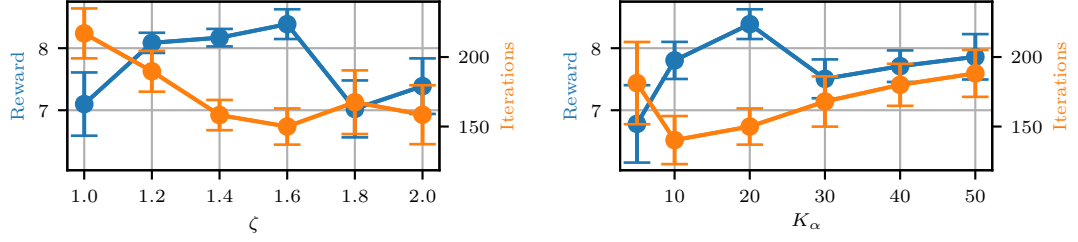


Figure 2.10.: Final performance (blue) of SPDL on the point-mass (3D) environment for different values of K_α and ζ as well as the average number of iterations required to reach a reward larger or equal to 5 on tasks sampled from $\mu(\mathbf{c})$ (orange). The results are computed from 20 environment runs. The error bars indicate the standard error. When varying ζ , K_α was fixed to a value of 20. When varying K_α , ζ was fixed to a value of 1.6.

2.8. Improved α -Schedule

The evaluation in the previous settings showed that choosing the trade-off parameter α_k in each iteration according to (2.9) was sufficient to improve the performance of the learner in the investigated experiments. However, the introduced schedule requires an offset parameter K_α as well as a penalty proportion ζ to be specified. Both these parameters need to be chosen adequately. If K_α is chosen too small, the agent may not have enough time to find a subspace of the context space containing tasks of adequate difficulty. If K_α is chosen too large, the learner wastes iterations focusing on tasks of small difficulty, not making progress towards the tasks likely under $\mu(\mathbf{c})$. The parameter ζ exhibits a similar trade-off behavior. Too small values will lead to an unnecessarily slow progression towards $\mu(\mathbf{c})$ while too large values lead to ignoring the competence of the learner on the tasks under $p(\mathbf{c}|\nu)$, resulting in a poor final agent behavior because of a too speedy progression towards $\mu(\mathbf{c})$. This trade-off is visualized in Figure 2.10 for the point-mass environment. Despite the clear interpretation of the two parameters K_α and ζ , which allows for a fairly straightforward tuning, we additionally explore another approach of choosing α_k in this section. This approach only requires to specify an expected level of performance V_{LB} that the agent should maintain under the chosen context distribution $p(\mathbf{c}|\nu)$. Assuming the decoupled optimization of the policy π_θ and the context distribution $p(\mathbf{c}|\nu)$ investigated in the last section, this can be easily realized by rewriting (2.8) as

$$\begin{aligned}
 & \min_{\nu} D_{\text{KL}}(p(\mathbf{c}|\nu) \parallel \mu(\mathbf{c})) \\
 & \text{s.t. } \mathbb{E}_{p(\mathbf{c}|\nu)} [J(\theta, \mathbf{c})] \geq V_{\text{LB}} \\
 & \quad D_{\text{KL}}(p(\mathbf{c}|\nu) \parallel p(\mathbf{c}|\nu')) \leq \epsilon.
 \end{aligned} \tag{2.16}$$

The main modification is to avoid the explicit trade-off between expected agent performance and KL divergence by minimizing the KL divergence w.r.t. $\mu(\mathbf{c})$ subject to a constraint on the expected agent performance. Investigating the Lagrangian of objective (2.16)

$$L(\boldsymbol{\nu}, \alpha, \eta) = D_{\text{KL}}(p(\mathbf{c}|\boldsymbol{\nu}) \parallel \mu(\mathbf{c})) + \alpha(V_{\text{LB}} - \mathbb{E}_{p(\mathbf{c}|\boldsymbol{\nu})}[J(\boldsymbol{\theta}, \mathbf{c})]) \\ + \eta(D_{\text{KL}}(p(\mathbf{c}|\boldsymbol{\nu}) \parallel p(\mathbf{c}|\boldsymbol{\nu}')) - \epsilon), \quad \alpha, \eta \geq 0$$

we see that the constraint reintroduces a scalar α that trades-off the expected agent performance and KL divergence to $\mu(\mathbf{c})$. The value of this scalar is, however, now automatically chosen to fulfill the imposed constraint on the expected agent performance. For an implementation of (2.16), we again replace $J(\boldsymbol{\theta}, \mathbf{c})$ by an importance-sampled Monte Carlo estimate as done in (2.15). At this point, we have replaced the parameter ζ with V_{LB} . The benefit is a more intuitive choice of this hyperparameter since it is directly related to the expected performance, i.e. the quantity being optimized. Furthermore, we can easily remove the need for the offset parameter K_α by setting $\alpha_k=0$ in (2.15) until we first reach V_{LB} . Consequently, this new schedule only requires one hyperparameter V_{LB} to be specified. From then on, we compute the new context distribution by optimizing (2.16). If during learning, the performance of the agent falls below V_{LB} again, we simply do not change the context distribution until the performance exceeds V_{LB} again. We now compare this new schedule with the schedule for α that is based on K_α and ζ . The corresponding experiment data is shown in Table 2.2. We can see that the final rewards achieved with the two heuristics are not significantly different according to Welch’s t-test. This indicates that the simpler heuristic performs just as well in the investigated environments in terms of final reward. However, the often significantly smaller average number of iterations required to reach a certain average performance on $\mu(\mathbf{c})$ shows that the schedule based on V_{LB} tends to lead to a faster progression towards $\mu(\mathbf{c})$. In a sense, this is not surprising, given that the explicit minimization of the KL divergence w.r.t. $\mu(\mathbf{c})$ under a constraint on the expected performance optimizes the trade-off between agent performance and KL divergence to $\mu(\mathbf{c})$ in each iteration. With the schedule based on K_α and ζ , this trade-off was compressed into the parameter ζ that stayed constant for all iterations. Furthermore, the lower bound V_{LB} on the achieved average reward under $p(\mathbf{c}|\boldsymbol{\nu})$ exhibits certain similarities to the GOALGAN algorithm in which the context distribution resulted from a constraint of encoding only tasks of intermediate difficulty. In a sense, GOALGAN uses an upper and lower bound on the task difficulty. However, it enforces this constraint per context while our formulation enforces the lower bound in expectation.

	SPDL(K_α, ζ)		SPDL(V_{LB})	
	Performance	Iterations to Threshold	Performance	Iterations to Threshold
TRPO (P3D)	8.04 ± 0.25	198 ± 18	7.79 ± 0.28	220 ± 19
PPO (P3D)	8.45 ± 0.42	165 ± 14	8.66 ± 0.07	120 ± 10
SAC (P3D)	6.85 ± 0.77	94 ± 8	7.13 ± 0.71	67 ± 4
TRPO (P2D)	7.47 ± 0.33	201 ± 20	7.67 ± 0.2	198 ± 19
PPO (P2D)	8.94 ± 0.10	132 ± 5	9.01 ± 0.07	119 ± 3
SAC (P2D)	5.67 ± 0.77	134 ± 30	6.56 ± 0.82	59 ± 3
PPO (ANT)	1371 ± 23	131 ± 3	1305 ± 38	131 ± 2
TRPO (BC)	47.0 ± 2.0	379 ± 21	50.0 ± 1.5	320 ± 20
PPO (BC)	53.9 ± 0.4	285 ± 19	51.6 ± 1.7	234 ± 12
SAC (BC)	34.1 ± 2.3	205 ± 12	34.1 ± 1.3	139 ± 8
TRPO (BC*)	43.3 ± 2.0	354 ± 18	46.0 ± 1.5	285 ± 20
PPO (BC*)	49.3 ± 1.4	224 ± 7	51.8 ± 0.5	212 ± 17
SAC (BC*)	36.9 ± 1.0	235 ± 23	37.1 ± 1.2	173 ± 20

Table 2.2.: Comparison between the two SPDL heuristics on the point-mass (P3D and P2D), ant, and ball-catching (BC) environments over 20 seeds. The asterisks mark runs of SPDL with an initialized context distribution. We compare the final average reward \pm standard error (Performance) and the average number of iterations required to reach 80% of the lower of the two rewards (Iterations to Threshold). Statistically significant differences according to Welch’s t-test are highlighted in **bold** for $p < 1\%$ and **brown** for $p < 5\%$.

2.9. An Inference Perspective on Self-Paced Reinforcement Learning

As noted in Section 2.5, the KL divergence regularization w.r.t. ν in (2.8) was done to stabilize the overall learning procedure. In this final section, we show that the resulting learning scheme can be connected to a modified version of the expectation-maximization algorithm, a well-known majorize-minimize algorithm for inference problems. Before we conclude this chapter, we want to briefly point out this connection, especially highlighting a connection between self-paced learning and the concept of tempering [91, 202].

2.9.1. RL as Inference

To establish the aforementioned connection, we need to introduce a probabilistic interpretation of the contextual RL problem as being an inference task [41, 197, 109]. In this formula-

tion, the goal is to maximize the probability of an optimality event $\mathcal{O} \in \{0, 1\}$ that depends on the sum of rewards along a trajectory of states and actions $\boldsymbol{\tau} = \{(s_t, \mathbf{a}_t) | t = 0, 1, \dots\}$

$$p(\mathcal{O} | \boldsymbol{\tau}, \mathbf{c}) \propto \exp(R(\boldsymbol{\tau}, \mathbf{c})) = \exp\left(\sum_{t=0}^{\infty} r_{\mathbf{c}}(s_t, \mathbf{a}_t)\right). \quad (2.17)$$

Together with the probability for a trajectory $\boldsymbol{\tau}$ given the policy parameters $\boldsymbol{\theta}$

$$p(\boldsymbol{\tau} | \boldsymbol{\theta}, \mathbf{c}) = p_{0,\mathbf{c}}(s_0) \prod_{t \geq 0} \bar{p}_{\mathbf{c}}(s_{t+1} | s_t, \mathbf{a}_t) \pi_{\boldsymbol{\theta}}(s_t | \mathbf{a}_t, \mathbf{c}) \quad (2.18)$$

we can marginalize over the trajectories $\boldsymbol{\tau}$ generated by $p(\boldsymbol{\tau} | \boldsymbol{\theta}, \mathbf{c})$, i.e. generated by the agent. This results in a probabilistic equivalent of the expected performance $J(\boldsymbol{\theta}, \mathbf{c})$ in context \mathbf{c} under policy parameters $\boldsymbol{\theta}$. This probabilistic equivalent is given by the marginal likelihood of event \mathcal{O} in an MDP $\mathcal{M}(\mathbf{c})$

$$J(\boldsymbol{\theta}, \mathbf{c}) = p(\mathcal{O} | \mathbf{c}, \boldsymbol{\theta}) = \int p(\mathcal{O} | \boldsymbol{\tau}, \mathbf{c}) p(\boldsymbol{\tau} | \boldsymbol{\theta}, \mathbf{c}) d\boldsymbol{\tau}. \quad (2.19)$$

The transition probabilities $\bar{p}_{\mathbf{c}}$ in (2.18) are a modified version of the original transition probabilities $p_{\mathbf{c}}$ that introduce a “termination” probability in each step that can occur with a probability of $1 - \gamma$ [see 109]. This introduces the concept of a discounting factor γ into the probabilistic model. Introducing a context distribution $p(\mathbf{c} | \boldsymbol{\nu})$ then yields a probabilistic interpretation of the contextual RL objective

$$J(\boldsymbol{\theta}, p(\mathbf{c} | \boldsymbol{\nu})) = p(\mathcal{O} | \boldsymbol{\theta}, \boldsymbol{\nu}) = \int p(\mathcal{O} | \boldsymbol{\tau}, \mathbf{c}) p(\boldsymbol{\tau} | \boldsymbol{\theta}, \mathbf{c}) p(\mathbf{c} | \boldsymbol{\nu}) d\mathbf{c} d\boldsymbol{\tau}. \quad (2.20)$$

When not making use of a curriculum, we would simply set $p(\mathbf{c} | \boldsymbol{\nu}) = \mu(\mathbf{c})$. The above model is called a latent variable model (LVM), as the trajectories $\boldsymbol{\tau}$, as well as the contexts \mathbf{c} are marginalized out to form the likelihood of the event \mathcal{O} . These marginalizations make the direct optimization w.r.t. $\boldsymbol{\theta}$ and $\boldsymbol{\nu}$ challenging. The so-called expectation-maximization algorithm is commonly applied to split this complicated optimization into two simpler steps: The E- and M-Step

$$\text{E-Step : } q_k(\boldsymbol{\tau}, \mathbf{c}) = \arg \min_{q(\boldsymbol{\tau}, \mathbf{c})} D_{\text{KL}}(q(\boldsymbol{\tau}, \mathbf{c}) \| p(\boldsymbol{\tau}, \mathbf{c} | \mathcal{O}, \boldsymbol{\theta}_k, \boldsymbol{\nu}_k)) \quad (2.21)$$

$$\text{M-Step : } \boldsymbol{\theta}_{k+1}, \boldsymbol{\nu}_{k+1} = \arg \max_{\boldsymbol{\theta}, \boldsymbol{\nu}} \mathbb{E}_{q_k(\boldsymbol{\tau}, \mathbf{c})} [\log(p(\mathcal{O}, \boldsymbol{\tau}, \mathbf{c} | \boldsymbol{\theta}, \boldsymbol{\nu}))]. \quad (2.22)$$

Iterating between these two steps is guaranteed to find a local optimum of the marginal likelihood $p(\mathcal{O} | \boldsymbol{\theta}, \boldsymbol{\nu})$.

2.9.2. Connection to Self-Paced Reinforcement Learning

At this point, two simple reformulations are required to establish the connection between the KL-regularized objective (2.8) and the expectation-maximization algorithm on LVM (2.20). First, we can reformulate the M-Step as an M-projection (i.e. a maximum-likelihood fit of the parametric model $q(\boldsymbol{\tau}, \mathbf{c}|\boldsymbol{\theta}, \boldsymbol{\nu})$ to $q_k(\boldsymbol{\tau}, \mathbf{c})$)

$$\arg \max_{\boldsymbol{\theta}, \boldsymbol{\nu}} \mathbb{E}_{q_k(\boldsymbol{\tau}, \mathbf{c})} [\log(p(\mathcal{O}, \boldsymbol{\tau}, \mathbf{c}|\boldsymbol{\theta}, \boldsymbol{\nu}))] = \arg \min_{\boldsymbol{\theta}, \boldsymbol{\nu}} D_{\text{KL}}(q_k(\boldsymbol{\tau}, \mathbf{c}) \parallel q(\boldsymbol{\tau}, \mathbf{c}|\boldsymbol{\theta}, \boldsymbol{\nu})).$$

Second, the E-Step can, for this particular model, be shown to be equivalent to a KL-regularized RL objective

$$\begin{aligned} & \arg \min_{q(\boldsymbol{\tau}, \mathbf{c})} D_{\text{KL}}(q(\boldsymbol{\tau}, \mathbf{c}) \parallel p(\boldsymbol{\tau}, \mathbf{c}|\mathcal{O}, \boldsymbol{\theta}_k, \boldsymbol{\nu}_k)) \\ &= \arg \max_{q(\boldsymbol{\tau}, \mathbf{c})} \mathbb{E}_{q(\boldsymbol{\tau}, \mathbf{c})} [R(\boldsymbol{\tau}, \mathbf{c})] - D_{\text{KL}}(q(\boldsymbol{\tau}, \mathbf{c}) \parallel p(\boldsymbol{\tau}, \mathbf{c}|\boldsymbol{\theta}, \boldsymbol{\nu})), \end{aligned}$$

in which we penalize a deviation of the policy and context distribution from the current parametric distribution $p(\boldsymbol{\tau}, \mathbf{c}|\boldsymbol{\theta}, \boldsymbol{\nu})$. Adding a term $-\alpha D_{\text{KL}}(q(\mathbf{c}) \parallel \mu(\mathbf{c}))$ and optimizing this modified E-Step only w.r.t. the context distribution $q(\mathbf{c})$ while keeping $q(\boldsymbol{\tau}|\mathbf{c})$ fixed at $p(\boldsymbol{\tau}|\boldsymbol{\theta}_k, \mathbf{c})$, we obtain

$$\arg \max_{q(\mathbf{c})} \mathbb{E}_{q(\mathbf{c})} [\mathbb{E}_{p(\boldsymbol{\tau}|\boldsymbol{\theta}_k, \mathbf{c})} [R(\boldsymbol{\tau}, \mathbf{c})]] - \alpha D_{\text{KL}}(q(\mathbf{c}) \parallel \mu(\mathbf{c})) - D_{\text{KL}}(q(\mathbf{c}) \parallel p(\mathbf{c}|\boldsymbol{\nu}_k)). \quad (2.23)$$

This result resembles (2.8), where however the optimization is carried out w.r.t. $q(\mathbf{c})$ instead of $\boldsymbol{\nu}$ and the KL divergence w.r.t. $p(\mathbf{c}|\boldsymbol{\nu}_k)$ is treated as a penalty term instead of a constraint. Not fitting the parameters $\boldsymbol{\nu}_{k+1}$ directly but in a separate (M-)step is also done by C-REPS and our episodic RL implementation of SPL. Hence, in the light of these results, the step-based implementation can be interpreted as skipping an explicit M-Step and directly optimizing the E-Step w.r.t. to the parametric policy. Such a procedure can be found in popular RL algorithms, as detailed by [1].

2.9.3. Self-Paced Learning as Tempering

The previously derived E-Step has a highly interesting connection to a concept in the inference literature called *tempering* [91, 202]. This connection is revealed by showing that the penalty term $\alpha D_{\text{KL}}(q(\mathbf{c}) \parallel \mu(\mathbf{c}))$ in the modified E-Step (2.23) results in an E-Step

to a modified target distribution. That is

$$\begin{aligned} & \arg \min_{q(\mathbf{c})} D_{\text{KL}}(q(\mathbf{c}) \parallel p(\mathbf{c}|\mathcal{O}, \boldsymbol{\theta}_k, \boldsymbol{\nu}_k)) + \alpha D_{\text{KL}}(q(\mathbf{c}) \parallel \mu(\mathbf{c})) \\ &= \arg \min_{q(\mathbf{c})} D_{\text{KL}}\left(q(\mathbf{c}) \parallel \frac{1}{Z} p(\mathbf{c}|\mathcal{O}, \boldsymbol{\theta}_k, \boldsymbol{\nu}_k)^{\frac{1}{1+\alpha}} \mu(\mathbf{c})^{\frac{\alpha}{1+\alpha}}\right). \end{aligned} \quad (2.24)$$

The modified target distribution in (2.24) is performing an interpolation between $\mu(\mathbf{c})$ and $p(\mathbf{c}|\mathcal{O}, \boldsymbol{\theta}_k, \boldsymbol{\nu}_k)$ based on the parameter α . Looking back at the sampling distribution induced by the probabilistic SPL objective (2.5) for the regularizer $f_{\text{KL},i}$ (see Equation B.1 in the appendix)

$$p(\mathbf{c}|\alpha, \boldsymbol{\theta}) \propto \boldsymbol{\nu}_{\text{KL},\mathbf{c}}^*(\alpha, \boldsymbol{\theta}) = \mu(\mathbf{c}) \exp(J(\boldsymbol{\theta}, \mathbf{c}))^{\frac{1}{\alpha}}, \quad (2.25)$$

we can see that, similarly to the modified E-Step, the distribution encoded by $\boldsymbol{\nu}_{\text{KL},\mathbf{c}}^*$, i.e. the optimizers of (2.5), interpolates between $\mu(\mathbf{c})$ and the distribution $p(\mathbf{c}|\boldsymbol{\theta}) \propto \exp(J(\boldsymbol{\theta}, \mathbf{c}))$. Both of these distributions would be referred to as tempered distributions in the inference literature.

The concept of tempering has been explored in the inference literature as a tool to improve inference methods when sampling from or finding modes of a distribution $\mu(\mathbf{c})$ with many isolated modes of density [91, 127, 200]. The main idea is to not directly apply inference methods to $\mu(\mathbf{c})$ but to make use of a tempered distribution $p_\alpha(\mathbf{c})$ which interpolates between $\mu(\mathbf{c})$ and a user-chosen reference distribution $\rho(\mathbf{c})$ from which samples can be easily drawn by the employed inference method (e.g. a Gaussian distribution). Doing repeated inference for varying values of α allows to explore the isolated modes more efficiently and with that yielding more accurate samples from $\mu(\mathbf{c})$. Intuitively, initially sampling from $\rho(\mathbf{c})$, chosen to be free from isolated modes, and gradually progressing towards $\mu(\mathbf{c})$ while using the previous inferences as initializations avoids getting stuck in isolated modes of $\mu(\mathbf{c})$ that encode comparatively low density. This technique makes the inference algorithm less dependent on a good initialization.

We can easily identify both (2.24) and (2.25) to be particular tempered distributions $p_\alpha(\mathbf{c})$. There, however, seems to be a striking difference to the aforementioned tempering scheme: The target density $\mu(\mathbf{c})$ is typically trivial, not requiring any advanced inference machinery. However, although $\mu(\mathbf{c})$ may be trivial from an inference perspective, the posterior over policy parameters

$$p(\boldsymbol{\theta}|\mathcal{O}) \propto p(\boldsymbol{\theta}) \int p(\mathcal{O}|\mathbf{c}, \boldsymbol{\theta}) \mu(\mathbf{c}) d\mathbf{c}$$

is highly challenging for contexts \mathbf{c} distributed according to $\mu(\mathbf{c})$, potentially containing multiple, highly isolated modes, many of which only encode suboptimal behavior. In these cases, tempering helps to achieve better performance when employed in combination with RL. For low values of α , it is easier to find high-density modes of

$$p_\alpha(\boldsymbol{\theta}|\mathcal{O}) \propto p(\boldsymbol{\theta}) \int p(\mathcal{O}|\mathbf{c}, \boldsymbol{\theta}) p_\alpha(\mathbf{c}) d\mathbf{c}.$$

These modes can then be “tracked” by the RL algorithm while increasing the value of α . The connection between SPL and the concept of tempering yields interesting insights into the problem of choosing both a good schedule for α and also the general design of $p_\alpha(\mathbf{c})$. As introduced in Section 2.3, the particular choice of the self-paced regularizer $f(\alpha, \nu)$, and hence the regularizer $F_\alpha(l)$, is closely related to the particular form of $p_\alpha(\mathbf{c})$. A ubiquitous decision is the choice of the particular regularizer or tempered distribution for a given problem. Gelman and Meng [60] show that the particular choice of p_α has a tremendous effect on the error of Monte Carlo estimates of ratios between normalization constants. Furthermore, they compute the optimal form of p_α for a Gaussian special case that achieves minimum variance of the Monte Carlo estimator. It may be possible to draw inspiration from their techniques to design specialized regularizers for problems of particular structures.

For the application of SPL to RL, another important design decision is the schedule of α . The value of α should be increased as fast as possible while ensuring the stability of the RL agent. We proposed two schedules that accomplished this task sufficiently well. However, there may be a tremendous margin for improvement. In the inference literature, people readily investigated the problem of choosing α , as they face a similar trade-off problem between required computation time of inference methods and the usefulness of their results [124, 63, 119]. Again, it may be possible to draw inspiration from these works to design better schedules for α in RL problems.

2.10. Conclusion

We have presented an interpretation of self-paced learning as inducing a sampling distribution over tasks in a reinforcement learning setting when using the KL divergence w.r.t. a target distribution $\mu(\mathbf{c})$ as a self-paced regularizer. This view renders the induced curriculum as an approximate implementation of a regularized contextual RL objective that samples training tasks based on their contribution to the overall gradient of the objective. Furthermore, we identified our approximate implementations to be a modified version of the expectation-maximization algorithm applied to the common latent variable

model for RL. These findings, in turn, revealed connections to the concept of tempering in the inference literature.

The observations in this chapter motivate further theoretical investigations, such as identifying the particular regularized SPL objective that is related to our approximate implementation (2.8). Furthermore, we only explored the KL divergence as a self-paced regularizer. Although we showed that the probabilistic interpretation of SPL does not hold for arbitrary regularizers, it may be possible to derive the presented results for a wider class of divergences.

From an experimental point of view, we focused on RL tasks with a continuous context space in this chapter. In the future, we want to conduct experiments in discrete context spaces, where we do not need to restrict the distribution to some tractable analytic form since we can exactly represent discrete probability distributions.

Our implementations of the SPL scheme for RL demonstrated remarkable performance across RL algorithms and tasks. The presented algorithms are, however, by far no perfect realizations of the theoretical concept. The proposed ways of choosing α in each iteration are just ad-hoc choices. At this point, insights gained through the inference perspective into our curriculum generation scheme presented in Section 2.9 may be particularly useful. Furthermore, the use of Gaussian context distributions is a major limitation that restricts the flexibility of the context distribution. Specifically in higher-dimensional context spaces, such a restriction could lead to poor performance. Here, it may be possible to use advanced inference methods [115, 216] to sample from the distribution (2.24) without approximations even in continuous spaces.

3. On the Benefit of Optimal Transport for Curriculum Reinforcement Learning

Curriculum reinforcement learning (CRL) allows solving complex tasks by generating a tailored sequence of learning tasks, starting from easy ones and subsequently increasing their difficulty. Although the potential of curricula in RL has been clearly shown in various works, it is less clear how to generate them for a given learning environment, resulting in various methods aiming to automate this task. In this chapter, we focus on framing curricula as interpolations between task distributions, which has previously been shown to be a viable approach to CRL. Identifying key issues of existing methods, we frame the generation of a curriculum as a constrained optimal transport problem between task distributions. Benchmarks show that this way of curriculum generation can improve upon existing CRL methods, yielding high performance in various tasks with different characteristics.

3.1. Introduction

Reinforcement learning (RL) [189] has celebrated great successes as a framework for the autonomous acquisition of desired behavior. With ever-increasing computational power, this framework and the algorithms developed under it have allowed to create learning agents capable of solving non-trivial long-horizon planning [134, 183] and control tasks [8]. However, these successes have highlighted the need for certain forms of regularization, such as leagues in the context of board games [183], gradual diversification of simulated training environments for robotic manipulation [8] and -locomotion [173], or a tailored training pipeline in the context of humanoid control for soccer [117]. These regularizations can help to overcome the shortcomings of modern RL agents such as poor exploratory behavior – an active topic of research [16, 61, 121].

One can view the regularizations mentioned above under the umbrella term of curriculum reinforcement learning [146], which aims to avoid the shortcomings of modern (deep) RL agents by learning on a tailored sequence of tasks. Such task sequences can materialize in

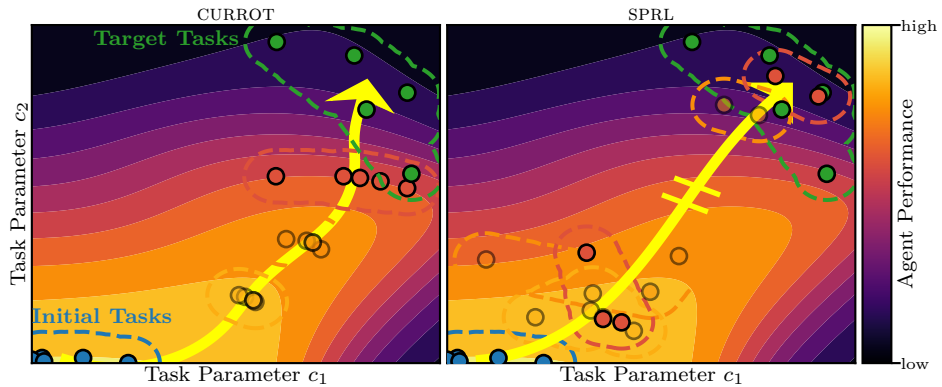


Figure 3.1.: Our approach (CURROT) addresses problems of existing curriculum RL methods, such as SPRL, which create curricula between a distribution of initial tasks (blue) and a distribution of target tasks (green). In this example, the curriculum can change the task via two parameters c_1 and c_2 , leading to more or less challenging learning environments for an agent. Looking at the different stages of the curricula (colored points), we see that existing methods can lead to distributions that encode hard- and easy tasks, but ignore tasks of intermediate difficulty. Our method avoids such a splitting behavior, resulting in interpolations that gradually increase the task difficulty throughout the curriculum. Please see Sections 3.4 and 3.5 for a detailed description.

a variety of ways, and they are motivated by different perspectives in the literature, such as intrinsic motivation or regret minimization, to name some of them [10, 57, 206, 162, 220, 83]. Following the perspective from Chapter 2, we interpret curricula as sequences of task distributions that interpolate between an auxiliary task distribution – with the sole purpose of facilitating learning – and a distribution of target tasks. We refer to these approaches as interpolation-based curricula. While algorithmic realizations of such curricula have been successfully evaluated in the literature [97, 98, 29], some evaluations indicated a relatively poor learning performance of these methods [171]. Furthermore, applications of interpolation-based curricula have been limited to scenarios with somewhat restricted distributions, such as Gaussian- or uniform ones. The observed performance gaps and lack of flexibility w.r.t. distribution parameterization call for a better understanding of the inner workings of these methods to improve their performance and applicability. This chapter investigates the shortcomings of methods that realize curricula as a scheduled interpolation between task distributions based on the KL divergence and an expected performance constraint. We show how both these concepts can fail to produce meaningful curricula in simple examples. The demonstrated failure cases a) illustrate the importance

of explicitly reasoning about the similarity of tasks when building a curriculum and b) show how parametric assumptions on the generated task distributions can masquerade failures of the underlying framework used to generate curricula. To resolve the observed issues, we explicitly specify the similarity of learning tasks via a distance function and use the framework of optimal transport to generate interpolating distributions that, independent of their parameterization, result in gradual task changes. Based on this explicit notion of task similarity, we then propose our approach to curriculum RL (CURROT), which additionally replaces the expected performance constraint with a more strict condition to obtain the behavior visualized in Figure 3.1. Furthermore, we contrast our approach with an alternative method that has been recently proposed by Huang et al. [76] and outline how both approaches use optimal transport to generate curricula but differ in their use of the agent performance to constrain the curriculum while avoiding the demonstrated pitfalls of expected performance constraints.

In experiments, we a) validate the correct behavior of both CURROT and GRADIENT free from approximations and parametric assumptions in a small discrete MDP and b) compare approximate implementations on a variety of tasks featuring discrete- and continuous task spaces, as well as Euclidean- and non-Euclidean measures of distance between learning tasks. In these experiments, both approaches show convincing performance with CURROT consistently matching and surpassing the performance of all other algorithms.

3.2. Related Work

This chapter focuses on generating training curricula for reinforcement learning (RL) agents. Unlike supervised learning, where there is an ongoing discussion about the mechanics and effects of curricula in different learning situations [215, 221], the mechanics seem to be more agreed upon in RL.

Curriculum Reinforcement Learning: In RL, curricula improve the learning performance of an agent by adapting the training environments to its proficiency. This adaptation of task complexity can reduce the sample complexity of RL, e.g., by bypassing poor exploratory behavior of non-proficient agents [113]. Using curricula can avoid the need for extensively engineered reward functions, which come with risks, such as failing to encode the intended behavior [26]. Applications of curricula to RL are widespread, and different terms have been established. Adaptive Domain Randomization [8] uses curricula to gradually diversify the training parameters of a simulator to facilitate sim-to-real transfer. Similarly, unsupervised environment discovery [45, 82, 83] aims to efficiently train an agent robust to variations in the environment and can be seen as a more general view of domain randomization. Automatic curriculum learning methods [57, 186, 56, 162, 227, 165, 49,

93] mainly focus on improving an agent’s learning speed and/or performance on a set of desired tasks. Curricula are often generated as distributions that maximize a specific surrogate objective, such as learning progress [15, 162], intermediate task difficulty [56], regret [82], or disagreement between Q -functions [227]. Curriculum generation can also be interpreted as a two-player game [186]. The work by Jiang et al. [83] hints at a link between surrogate objectives and two-player games. Similar to the variety of objectives that the aforementioned algorithms optimize to build a curriculum, their implementations use drastically different approaches to approximate the training distribution for the agent, which is often defined over a continuous space of training tasks. For example, Florensa et al. [56] use a combination of GANs and a replay buffer to represent the task distribution. Portelas et al. [162] use a Gaussian mixture model to approximate the distribution of tasks that promise high learning progress. Jiang et al. [83] use a fixed-size replay buffer to realize an approximate distribution of high-regret tasks, simultaneously encouraging frequent replay of buffered tasks to keep a more accurate estimate of regret.

Opposed to the aforementioned approaches, interpolation-based curriculum RL algorithms formulate the generation of a curriculum as an explicit interpolation between an auxiliary task distribution and a distribution of target tasks [97, 93, 29]. This interpolation is subject to a constraint on the expected agent performance that paces its progress towards the target tasks. As highlighted in Chapter 2, such interpolations can be formally linked to successful curricula in supervised learning [105], the concept of annealing in statistics [147], and homotopic continuation methods in optimization [9]. As for the algorithms based on surrogate objectives, realizations of these interpolation-based curricula inevitably need to rely on approximations such as the restriction to Gaussian distributions in Chapter 2 or approximate update rules enabled by uniform target task distributions [29]. In this chapter, we reveal shortcomings of the aforementioned interpolation-based curriculum RL methods, highlighting how approximations can masquerade issues in the conceptual algorithm formulations. One ingredient to overcome these shortcomings is an explicit notion of task similarity that we formulate as a distance function between tasks. We can then lift this distance function into the space of probability measures using *optimal transport*.

Optimal Transport: Dating back to work by Monge in the 18th century, *optimal transport* has been understood as an important fundamental concept touching upon many fields in both theory and application [158, 32]. In probability theory, optimal transport translates to the so-called Wasserstein metric [86] that compares two distributions under a given metric, allowing, e.g., for the analysis of probabilistic inference algorithms as approximate gradient flows [115] and providing well-defined ways of comparing feature distributions or even graphs in computer vision and machine learning [103, 85, 196]. Gromov-Wasserstein distances [130, 203] even allow comparing distributions across metric spaces, which

has been of use, e.g., in computational biology [44] or imitation learning [55]. In Reinforcement learning, optimal transport has not found widespread application, albeit some interesting works exist. Zhang et al. [225] provide a natural extension of the work by Liu et al. [115] and interpret policy optimization as Wasserstein gradient flows. Metelli, Likmeta, and Restelli [132] use Wasserstein barycenters to propagate uncertainty about value function estimates in a Q -learning approach. In more applied scenarios, optimal transport has been used to regularize RL in sequence generation- [30] or combinatorial optimization problems [62]. In goal-conditioned RL, Wasserstein distances have been previously applied to improve goal generation in the hindsight experience replay framework [169] and to realize well-performing data-driven reward functions by combining them with so-called time-step metrics [48]. Recently, Cho, Lee, and Kim [34] combined the data-driven reward function proposed by Durugkar et al. [48] with a curriculum that, similarly to the work by Ren et al. [169], improves selection of training goals from a buffer of achieved ones. When it comes to building RL curricula over arbitrary MDPs using Optimal Transport, we are only aware of our work [95] at ICML 2022 and the work by Huang et al. [76] at NeurIPS 2022, which we present from a unified perspective and compare in this chapter. In addition to the aforementioned methods in goal-conditioned RL, this chapter emphasizes curriculum reinforcement learning as another promising application domain for optimal transport. An important issue of applied optimal transport is its computational complexity. In Appendix C.1, we discuss the computational aspects of optimal transport in more detail.

3.3. Divergence-Minimizing Curriculum Reinforcement Learning

We refer to Section 1.3 for the introduction of contextual RL as well as Wasserstein distances and Wasserstein barycenters. In this section, we summarize the main algorithmic idea of Chapter 2 in case this chapter has been skipped. On an abstract level, curriculum RL methods can be understood as generating a sequence of task distributions $(p_i: \mathcal{C} \rightarrow \mathbb{R})_i$ under which to train an RL agent by maximizing $J(\pi, p_i)$ w.r.t. π . When chosen appropriately, solving this sequence of optimization problems can yield a policy that performs better on the target distribution $\mu(\mathbf{c})$ than a policy found by maximizing $J(\pi, \mu)$ directly. The benefit of such mediating distributions is particularly obvious in settings where initially random agent behavior is unlikely to observe any meaningful learning signals, such as, e.g. is the case in sparse-reward learning tasks. CRL methods differ in the specification of p_i . Often, the distribution is defined to prioritize tasks that maximize certain surrogate quantities, such as absolute learning progress [162], regret [82] or tasks of intermediate success probability [56]. This chapter focuses on CRL methods that model p_i as the solution to

an optimization problem that aims to minimize a distance or divergence between p_i and μ . We presented one such approach in Chapter 2, defining p_i as the distribution with minimum KL divergence to μ that fulfills a constraint on the expected agent performance

$$\begin{aligned} \min_p D_{\text{KL}}(p(\mathbf{c}) \parallel \mu(\mathbf{c})) \\ \text{s.t. } J(\pi, p) \geq \delta \quad D_{\text{KL}}(p(\mathbf{c}) \parallel q(\mathbf{c})) \leq \epsilon, \end{aligned} \quad (3.1)$$

where δ is the desired level of performance to be achieved by the agent π under $p(\mathbf{c})$ and ϵ limits the maximum KL divergence to the previous context distribution $q(\mathbf{c})=p_{i-1}(\mathbf{c})$. The optimizer of (3.1) balances between tasks likely under the (target) distribution $\mu(\mathbf{c})$ and tasks in which the agent currently obtains large rewards. The KL divergence constraint w.r.t. the previous context distribution $q(\mathbf{c})$ prevents large changes in $p(\mathbf{c})$ during subsequent iterations, avoiding the exploitation of faulty estimates of the agent performance $J(\pi, p)$ from a limited amount of samples. Objective (3.1) can be shown to perform an interpolation between the distributions $p_\eta(\mathbf{c}) \propto \mu(\mathbf{c}) \exp(\eta J(\pi, \mathbf{c}))$ and $q(\mathbf{c})$, given by

$$p_{\alpha, \eta}(\mathbf{c}) \propto (\mu(\mathbf{c}) \exp(J(\pi, \mathbf{c}))^\eta)^\alpha q(\mathbf{c})^{1-\alpha}. \quad (3.2)$$

The two parameters α and η that control the interpolation are the Lagrangian multipliers of the two constraints in objective (3.1). We will later investigate the behavior of this interpolating distribution.

3.4. Curriculum Reinforcement Learning as Constrained Optimal Transport

At this point, we can motivate our approach to curriculum RL by looking at the limitations of Objective 3.1 caused by a) measuring similarity between context distributions via the KL divergence and b) the expected performance constraint used to control the progression towards $\mu(\mathbf{c})$.

3.4.1. Limitations of the KL Divergence

Given the complexity of computing $D_{\text{KL}}(p(\mathbf{c}) \parallel \mu(\mathbf{c}))$ for arbitrary distributions, previous work restricts $\mu(\mathbf{c})$ either to a Gaussian distribution as in Chapter 2 or to be uniform over \mathcal{C} to ease computation and optimization of a weighted KL divergence objective [29]. While empirically successful, these design choices masquerade the pitfalls of the KL divergence

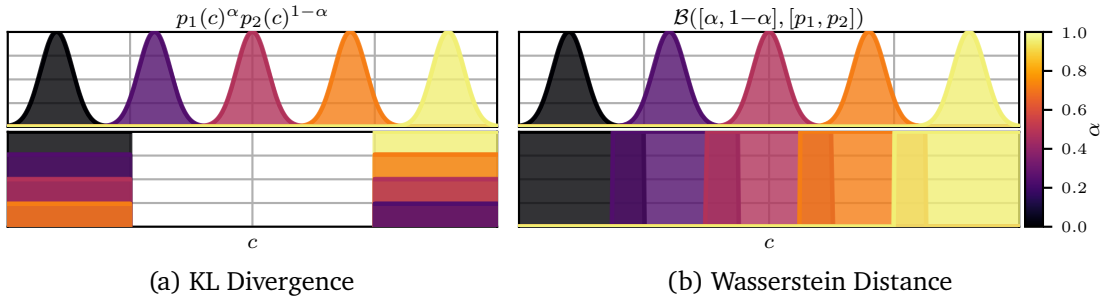


Figure 3.2.: Interpolations generated by optimizing KL Divergence (Objective 3.3) and Wasserstein distance (Objective 1.14) for different values of α (or $\alpha(\epsilon)$ in the case of Objective 3.3). In the top row, $p_1(c)$ and $p_2(c)$ are Gaussian, while in the bottom row, they assign uniform density over different parts of \mathcal{C} .

to measure distributional similarity in a CRL setting, particularly when dealing with a target distribution that does not assign uniform density over all of \mathcal{C} . To demonstrate this issue, we will focus on an interpolation task between two distributions

$$p_1(\mathbf{c})^{\alpha(\epsilon)} p_2(\mathbf{c})^{1-\alpha(\epsilon)} = \arg \min_{p \in \{q \mid D_{\text{KL}}(q \parallel p_2) \leq \epsilon\}} D_{\text{KL}}(p \parallel p_1), \quad (3.3)$$

which corresponds to a version of Objective (3.1) without a constraint on the expected agent performance. Figure 3.2a demonstrates the sensibility of this interpolation to the parametric representation of the distributions $\mu(\mathbf{c})$ and $q(\mathbf{c})$. While for Gaussian distributions, interpolations of the form $p_1(\mathbf{c})^\alpha p_2(\mathbf{c})^{1-\alpha}$ gradually shift density in a metric sense, this behavior is all but guaranteed for non-Gaussian distributions. The interpolation between two uniform distributions with quasi-limited support¹ in the bottom row of Figure 3.2a displaces density from contexts \mathbf{c} to contexts \mathbf{c}' with large Euclidean distance $\|\mathbf{c} - \mathbf{c}'\|_2$. In settings in which the Euclidean distance between contexts \mathbf{c}_1 and \mathbf{c}_2 is a good indicator for the similarity between $\mathcal{M}(\mathbf{c}_1)$ and $\mathcal{M}(\mathbf{c}_2)$, the observed ignorance of the KL divergence w.r.t. the underlying geometry of the context space leads to curricula with “jumps” in task similarity. We can easily convince ourselves that such jumps are not a hypothetical problem by recalling that neural network-based policies $\pi(\mathbf{a} \mid \mathbf{s}, \mathbf{c}) = \mathbf{f}_\theta(\mathbf{s}, \mathbf{c})$ tend to gradually change their behavior with increasing Euclidean distance to \mathbf{c} .

¹We ensure a negligible positive probability density across all of \mathcal{C} to allow for the computation of KL divergences.

At this point, we can leverage the notion of optimal transport to explicitly encode the similarity of two tasks $\mathcal{M}(\mathbf{c})$ and $\mathcal{M}(\mathbf{c}')$ via a metric $d(\mathbf{c}, \mathbf{c}')$ and realize the interpolation between distributions on the resulting metric space as Wasserstein barycenters (Eq. 1.14). As we see in Figure 3.2b, this explicit notion of task similarity allows to generate interpolations that are stable across changes in the parameterization of context distributions and interpolate between arbitrary distributions that are not absolutely continuous w.r.t. each other. Consequently, the optimization problem

$$\min_p \mathcal{W}_2(p, \mu) \quad \text{s.t.} \quad J(\pi, p) \geq \delta \quad (3.4)$$

is a promising approach to leverage optimal transport in curriculum RL. We iterate on this candidate in the next section by investigating the role of the expected performance constraint when generating curricula for reinforcement learning agents.

3.4.2. Challenges of Expected Performance Constraints

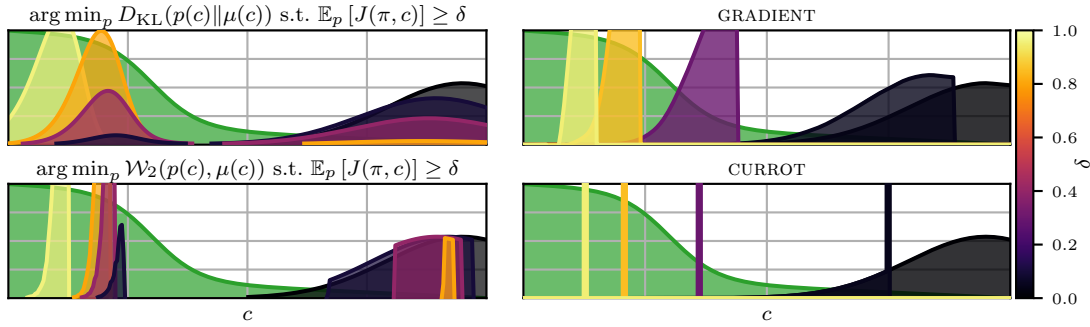
The SPRL objective (3.1) controls the interpolation speed between the initial- and target task distribution by the expected performance of the current agent under the chosen context distribution $J(\pi, p)$. As detailed in Chapter 2, this expected performance constraint allows for establishing a connection to self-paced learning for supervised learning tasks [105, 131]. While this formal connection is interesting in its own right, we show in Figure 3.3a that the expected performance constraint in SPRL can lead to encoding both too simple and too complex tasks, given the current agent capabilities. Furthermore, using Wasserstein distances in Objective (3.4) does not resolve this issue. In Figure 3.3a, both methods encode tasks with very high and very low agent return to fulfill the expected performance constraint, sidestepping the goal of encoding tasks of intermediate difficulty. At this point, we can propose our algorithm **CURROT** as well as a recent algorithm proposed by Huang et al. [76] – called **GRADIENT** – as two ways of resolving the observed interpolation issue:

1. **CURROT** restricts the support of $p(\mathbf{c})$ to those contexts $\mathbf{c} \in \mathcal{C}$ that fulfill the performance constraint $J(\pi, \mathbf{c}) \geq \delta$. We refer to this set as $\mathcal{V}(\pi, \delta) = \{\mathbf{c} | \mathbf{c} \in \mathcal{C}, J(\pi, \mathbf{c}) \geq \delta\}$. With this notation, the restricted optimization can be written as

$$\min_p \mathcal{W}_2(p, \mu) \quad \text{s.t.} \quad p(\mathcal{V}(\pi, \delta)) = 1. \quad (3.5)$$

Putting the constraint in words, we require that all probability density of p is assigned to the contexts that satisfy the performance constraint.

2. **GRADIENT** restricts the interpolation to follow the barycentric interpolation (1.14) between the initial- and target context distribution $p_\alpha(\mathbf{c}) = \mathcal{B}_2([1-\alpha, \alpha], [p_0(\mathbf{c}), \mu(\mathbf{c})])$.



(a) Expected Performance Constraints

(b) Investigated Approaches

Figure 3.3.: Left: Interpolations using KL divergence (top) and Wasserstein distance (bottom) subject to an expected performance constraint with different threshold values δ . Right: Interpolations generated by GRADIENT (Eq. 3.6, top) and CURROT (Eq. 3.5, bottom) for different threshold values δ . For both plots, the performance $J(\pi, c)$ is visualized in green.

This restriction prevents the problematic behavior shown in Figure 3.3a while still allowing to adjust α using an expected performance constraint

$$\max_{\alpha \in [0,1]} \alpha \text{ s.t. } J(\pi, p_\alpha) \geq \delta. \quad (3.6)$$

As shown in Figure 3.3b, both methods avoid the behavior generated by Objective (3.4), resulting in an interpolation that gradually deforms the distribution in a metric sense with changing agent competence. In the remainder of this chapter, we benchmark exact and approximate versions of these algorithms to understand their behavior. The first observation in this regard is that the curriculum of GRADIENT is predetermined by the given metric $d(c_1, c_2)$ as well as the target- and initial distribution, $\mu(c)$ and $p_0(c)$. The agent performance only influences how fast the curriculum proceeds towards $\mu(c)$. On the other hand, CURROT reshapes the curriculum based on the current agent performance to avoid sampling contexts with a performance lower than the threshold δ . Figure 3.3b shows that this reshaping places all probability density on the border of the desired agent performance δ until reaching regions of non-zero probability density under $\mu(c)$. At this point, the curriculum matches the target density in those parts of \mathcal{C} , in which the performance constraint is fulfilled, and concentrates all remaining density on the boundaries of agent capability. This behavior is similar to those CRL methods that combine task-prioritization with a replay buffer to prevent catastrophic forgetting, such as GOALGAN or PLR [56, 82]. To the best of our knowledge, such behavior has not yet been motivated by a first-principle optimization objective in the context of curriculum RL.

3.5. Approximate Algorithms for Discrete- and Continuous Context Spaces

Objectives (3.5) and (3.6) face challenges in more realistic application scenarios with either large discrete- or continuous context spaces due to two reasons:

1. We do not have access to the expected performance $J(\pi, \mathbf{c})$ of an agent π in context \mathbf{c} but can only estimate it from observed training episodes.
2. Computing Wasserstein barycenters for arbitrary continuous- or discrete distributions in non-Euclidean spaces can quickly become intractably expensive.

In the following sections, we address the above problems to benchmark CURROT and GRADIENT in non-trivial experimental settings.

3.5.1. Approximate Wasserstein Barycenters

Before branching into the description of the two algorithms, we first describe a particle-based approximation to the computation of Wasserstein Barycenters which allows to cheaply approximate Barycenters for the GRADIENT algorithm in large discrete state-spaces and is essential for the approximate implementation of the CURROT algorithm. For approximating a Barycenter $p_\alpha = \mathcal{B}([1-\alpha, \alpha], [p_0, \mu])$, we first sample a set of N particles from $\mu(\mathbf{c})$ and $p_0(\mathbf{c})$ to form the empirical distributions

$$\begin{aligned}\hat{\mu}(\mathbf{c}) &= \frac{1}{N} \sum_{n=1}^N \delta_{\mathbf{c}_{\mu,n}}(\mathbf{c}), \quad \mathbf{c}_{\mu,n} \sim \mu(\mathbf{c}) \\ \hat{p}_0(\mathbf{c}) &= \frac{1}{N} \sum_{n=1}^N \delta_{\mathbf{c}_{p_0,n}}(\mathbf{c}), \quad \mathbf{c}_{p_0,n} \sim p_0(\mathbf{c}),\end{aligned}\tag{3.7}$$

where $\delta_{\mathbf{c}_{\text{ref}}}(\mathbf{c})$ represents a Dirac distribution centered at \mathbf{c}_{ref} . Due to the discrete nature of $\hat{\mu}(\mathbf{c})$ and $\hat{p}_0(\mathbf{c})$, the coupling $\phi(\mathbf{c}_1, \mathbf{c}_2)$ reduces to a permutation $\phi \in \text{Perm}(N)$, which assigns the particles between \hat{p}_0 and $\hat{\mu}$ [158, Section 2.3]. With that, the computation of $\mathcal{W}_2(\hat{p}_0, \hat{\mu})$ reduces to

$$\min_{\phi \in \text{Perm}(N)} \left(\frac{1}{N} \sum_{n=1}^N d(\mathbf{c}_{p_0,n}, \mathbf{c}_{\mu,\phi(n)})^2 \right)^{\frac{1}{2}}.\tag{3.8}$$

Since a permutation is a special case of a coupling [158, Section 2.3], we overload the meaning of ϕ to be either a permutation or coupling, depending on the number of arguments. With today’s computing hardware, assignment problems like (3.8) can be solved on a single CPU core in less than a second for N in the hundreds, which is typically enough to represent the context distributions². Given this optimal assignment, we then compute the Fréchet mean for each particle pair

$$\mathbf{c}_{\alpha,n} = \arg \min_{\mathbf{c} \in \mathcal{C}} (1-\alpha)d(\mathbf{c}, \mathbf{c}_{p_0,n})^2 + \alpha d(\mathbf{c}, \mathbf{c}_{\mu,\phi(n)})^2 \quad (3.9)$$

to form the barycenter $\hat{p}_\alpha(\mathbf{c}) = \frac{1}{N} \sum_{n=1}^N \delta_{\mathbf{c}_{\alpha,n}}(\mathbf{c})$. While certainly less efficient than specialized routines for Barycenter computations in Euclidean Spaces, such as e.g. the GeomLoss library [54], the presented approach is useful when dealing with large discrete spaces. In this case, faithful Barycenter computations require to work with the full distance matrix. Assuming a discrete context space of size S and neglecting the cost of computing the optimal assignment, the approximate barycenter computation requires $O(N^2 + 2NS)$ evaluations of the distance function. Even computing the $\frac{S(S+1)}{2}$ entries of the entire distance matrix required for a single step in the Sinkhorn algorithm quickly becomes more expensive if $S \gg N$. Additionally, reducing the Barycenter computation to an optimization problem over individual particles easily allows to incorporate additional constraints that are required by CURROT (3.5).

3.5.2. Approximate GRADIENT

Huang et al. [76] propose to compute barycenters between $p_0(\mathbf{c})$ and $\mu(\mathbf{c})$ for discrete steps of size ϵ . Starting from $\alpha=0$, the agent trains for M episodes on tasks sampled from the current distribution. If the average episodic return $\frac{1}{M} \sum_{m=1}^M R_m$ is greater or equal to δ , α is increased by ϵ and the distribution is set to be the Wasserstein barycenter for the updated value of α .

This step-wise increase of α avoids the explicit optimization over α and, with that, the need to estimate the performance of the current policy π for a given context \mathbf{c} . Having laid out a way of computing approximate Barycenters in the previous section, we can summarize our implementation of GRADIENT in Algorithm 3.

3.5.3. Approximate CURROT

As for the GRADIENT algorithm, we make use of an empirical distribution $\hat{p}(\mathbf{c})$ to represent the context distribution $p(\mathbf{c})$ (see Eq. 3.7). Unlike for GRADIENT, there is no possibility to

²In our experiments, we use less than a thousand particles in all experiments

Algorithm 3 Approximate GRADIENT

Input: Initial context dist. $p_0(\mathbf{c})$, target context dist. $\mu(\mathbf{c})$, metric $d(\mathbf{c}_1, \mathbf{c}_2)$, performance bound δ , step size ϵ

Initialize: $\alpha = 0$

while True **do**

 Compute $\hat{p}_\alpha(\mathbf{c}) = \frac{1}{N} \sum_{n=1}^N \delta_{\mathbf{c}_{\alpha,n}}(\mathbf{c})$ (Eq. (3.8) and (3.9))

Agent Improvement:

 Sample contexts $\mathbf{c}_m \sim \hat{p}_\alpha(\mathbf{c})$, $m \in [1, M]$

 Train policy π under \mathbf{c}_m and observe episodic rewards $R_m = \sum_{t=0}^{\infty} \gamma^t r_{\mathbf{c}_m}(\mathbf{s}_t, \mathbf{a}_t)$, $m \in [1, M]$

Context Distribution Update:

if $\frac{1}{M} \sum_{m=1}^M R_m \geq \delta$ **then**

 Advance interpolation $\alpha = \min(\alpha + \epsilon, 1)$

end if

end while

side-step the estimation of $J(\pi, \mathbf{c})$ for CURROT, and any estimator of $J(\pi, \mathbf{c})$ will inevitably make mistakes. The mistakes will be particularly big for contexts \mathbf{c} with a considerable distance to those contexts sampled under the current training distribution $p(\mathbf{c})$. To avoid exploiting such erroneous performance predictions, we introduce a trust-region constraint similar to the seminal SPRL objective (3.1) into CURROT

$$\min_p \mathcal{W}_2(p, \mu) \quad \text{s.t.} \quad p(\mathcal{V}(\pi, \delta))=1 \quad \mathcal{W}_2(p, q) \leq \epsilon, \quad (3.10)$$

which limits the Wasserstein distance between the current- and next context distribution $q(\mathbf{c})$ and $p(\mathbf{c})$. We realize the performance estimator using Nadaraya-Watson kernel regression [142, 208] with a squared exponential kernel

$$\hat{J}(\pi, \mathbf{c}) = \frac{\sum_{l=1}^L K_h(\mathbf{c}, \mathbf{c}_l) R_l}{\sum_{l=1}^L K_h(\mathbf{c}, \mathbf{c}_l)}, \quad K_h(\mathbf{c}, \mathbf{c}_l) = \exp\left(-\frac{d(\mathbf{c}, \mathbf{c}_l)^2}{2h^2}\right).$$

This estimator does not rely on gradient-based updates and requires no architectural choices except for the lengthscale h , consequently not complicating the application of the overall algorithm. We postpone the discussion of this lengthscale parameter h until after we have discussed the approximate optimization of Objective (3.10) and first focus on the choice of dataset $\mathcal{D} = \{(\mathbf{c}_l, R_l) | l \in [1, L]\}$ used to build the kernel regressor.

We create the dataset from two buffers \mathcal{D}_+ and \mathcal{D}_- , each of size N . These two buffers are updated with the results of policy rollouts $(\mathbf{c}, R_{\mathbf{c}})$ during agent training, where

$R_{\mathbf{c}} = \sum_{t=0}^{\infty} \gamma^t r_{\mathbf{c}}(\mathbf{s}_t, \mathbf{a}_t)$. While \mathcal{D}_- is simply a circular buffer that keeps the most recent N rollouts with $R_{\mathbf{c}}$ below the performance threshold δ , \mathcal{D}_+ contains contexts \mathbf{c} for which $R_{\mathbf{c}} \geq \delta$. However, \mathcal{D}_+ is updated differently if full. Once full, we interpret the samples in \mathcal{D}_+ as an empirical distribution $\hat{p}_+(\mathbf{c})$ and select rollouts from the union of \mathcal{D}_+ and the set of new rollouts above the performance threshold δ to minimize $\mathcal{W}_2(\hat{p}_+, \hat{\mu})$. This optimal selection can be computed with a generalized version of the optimal assignment problem (3.8), where \hat{p}_+ is represented by N_+ particles and $\hat{\mu}$ is represented by N particles with $N_+ \geq N$. The generalized problem then produces a selection of N particles to represent \hat{p}_+ which minimize the resulting distance $\mathcal{W}(\hat{p}_+, \hat{\mu})$. We can hence interpret $\hat{p}_+(\mathbf{c})$ as a conservative solution to the CURROT objective (3.5). The solution is conservative since the particles are obtained from past iterations and may exceed the performance threshold δ by some margin, hence not targeting the exact border of the performance threshold.

To more precisely target this border of agent competence, we proceed as follows: First, we solve an assignment problem between $\hat{p}(\mathbf{c})$ and $\hat{p}_+(\mathbf{c})$ to obtain pairs $(\mathbf{c}_{p,n}, \mathbf{c}_{p_+, \phi(n)})$. We then reset $\mathbf{c}_{p,n} = \mathbf{c}_{p_+, \phi(n)}$ for those contexts $\mathbf{c}_{p,n}$ with $\hat{J}(\pi, \mathbf{c}_{p,n}) < \delta$. Next, we again sample an empirical target distribution $\hat{\mu}(\mathbf{c})$ and solve an assignment problem between the updated empirical distribution $\hat{p}(\mathbf{c})$ and $\hat{\mu}(\mathbf{c})$ to obtain context pairs $(\mathbf{c}_{p,n}, \mathbf{c}_{\mu, \phi(n)})$. We then solve the following optimization problem for each pair to obtain the particles for the new empirical context distribution

$$\arg \min_{\mathbf{c} \in \mathcal{C}} d(\mathbf{c}, \mathbf{c}_{\mu, \phi(n)}) \quad \text{s.t.} \quad \hat{J}(\pi, \mathbf{c}) \geq \delta \quad d(\mathbf{c}, \mathbf{c}_{p,n}) \leq \epsilon. \quad (3.11)$$

Note that the restriction $d(\mathbf{c}, \mathbf{c}_{p,n}) \leq \epsilon$ ensures that $\mathcal{W}_2(\hat{p}, \hat{q}) \leq \epsilon$ while de-coupling the optimization for the individual particles. We use a simple approximate optimization scheme that samples a set of candidate contexts around $\mathbf{c}_{p,n}$ and selects the candidate that minimizes the distance to $\mathbf{c}_{\mu, \phi(n)}$ while fulfilling the performance constraint. In the continuous Euclidean settings, we uniformly sample candidates in the half sphere of contexts that make an angle of less than 90 degrees with the descent direction $\mathbf{c}_{p,n} - \mathbf{c}_{\mu, \phi(n)}$. In discrete context spaces, we evaluate all contexts in the trust region. If even after resetting $\mathbf{c}_{p,n} = \mathbf{c}_{p_+, \phi(n)}$, no candidate satisfies the performance threshold, and hence Objective (3.11) is infeasible, we set $\mathbf{c}_{p,n}$ to the candidate with maximum performance in the ϵ -ball. Having defined Objective (3.11), we can discuss the lengthscale parameter h of the Nadaraya-Watson estimator. Given that the purpose of the estimator is to capture the trend in the ϵ -ball around a particle $\mathbf{c}_{p,n}$, we simply set the lengthscale to 0.3ϵ . This choice ensures that the two-times standard deviation interval of the squared-exponential kernel K_h centered on $\mathbf{c}_{p,n}$ covers the trust region.

A final detail of the approximate CURROT algorithm is its behavior, if the agent performance on the initial distribution $\hat{p}_0(\mathbf{c})$ is below the performance threshold δ . In this case, we use

Algorithm 4 Approximate CURROT

Input: Initial context dist. $p_0(\mathbf{c})$, target context dist. $\mu(\mathbf{c})$, metric $d(\mathbf{c}_1, \mathbf{c}_2)$, performance bound δ , distance bound ϵ
Initialize: $\hat{p}(\mathbf{c}) = \frac{1}{N} \sum_{n=1}^N \delta_{\mathbf{c}_{p_0, n}}(\mathbf{c})$, $\mathbf{c}_{p_0, n} \sim p_0(\mathbf{c})$
while True **do**
 Agent Improvement:
 Sample contexts $\mathbf{c}_m \sim \hat{p}(\mathbf{c})$, $m \in [1, M]$
 Train policy π under \mathbf{c}_m . Observe returns $R_m = \sum_{t=0}^{\infty} \gamma^t r_{\mathbf{c}_m}(\mathbf{s}_t, \mathbf{a}_t)$, $m \in [1, M]$
 Context Distribution Update:
 Update buffers \mathcal{D}_+ and \mathcal{D}_- with $\{(\mathbf{c}_m, R_m) | m \in [1, M]\}$
 Estimate $\hat{J}(\pi, \mathbf{c}) \approx J(\pi, \mathbf{c})$ from \mathcal{D}_+ and \mathcal{D}_-
 Update $\hat{p}(\mathbf{c})$ via Eq. (3.11) and $\hat{J}(\pi, \mathbf{c})$, $\hat{p}(\mathbf{c})$, $\hat{\mu}(\mathbf{c})$
end while

a simple randomized search method to find areas of \mathcal{C} in which the agent achieves returns above δ . This search procedure is detailed in Appendix C.2. The approach without this search procedure is summarized in Algorithm 4.

3.6. Experiments

To demonstrate the behavior of the introduced algorithms CURROT and GRADIENT, we benchmark the algorithms in different environments that feature discrete- and continuous context spaces with Euclidean- and non-Euclidean distance metrics. We furthermore evaluate both the exact approaches as well as their approximate implementations. To highlight the benefits of the proposed approach over currently popular CRL methods, we compare against a range of baselines. More precisely, we evaluate ALP-GMM [162], GOALGAN [56], PLR [82], VDS [227] and ACL [64] in addition to a random curriculum and training directly on $\mu(\mathbf{c})$ (referred to as Default). Details of the experiments, such as hyperparameters and employed RL algorithms, can be found in Appendix C.3.³

3.6.1. E-Maze Environment

To investigate CURROT and GRADIENT without relying on approximations and highlight the effect of the chosen distance metric, we start the experiments with the environment

³Code of the conference version [95] is publicly available under <https://github.com/pscl1nk/currot>. Code for the complete chapter will be made available upon its acceptance in a journal.

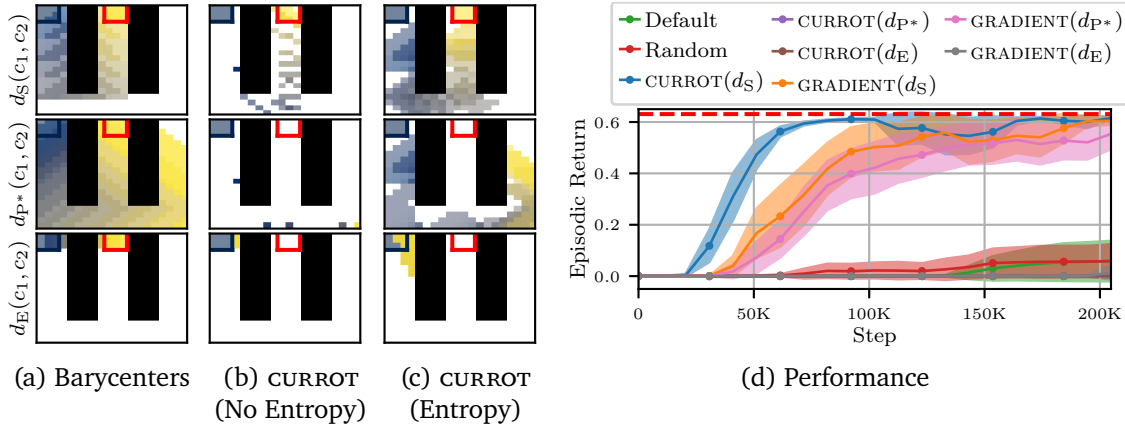


Figure 3.4.: (a) Visualizations of barycenters between initial- and target task distribution for the shortest-path distance d_S , performance pseudo-distance d_{P^*} and Euclidean distance d_E . Brighter colors correspond to distributions generated at later stages of the interpolation. The states covered by initial- and target task distributions are highlighted by the blue and red lines. (b) CURROT sampling distribution without entropy regularization. (c) CURROT sampling distribution for $H_{LB}=2$. (d) Expected return on the target task distribution $\mu(c)$ in the E-Maze environment achieved by CURROT and GRADIENT. The shaded area corresponds to two times the standard error (computed from 20 seeds). The red dotted line represents the maximum possible reward achievable on $\mu(c)$.

shown in Figure 3.4. In this sparse-reward environment that is represented by a 20×20 grid, an agent is tasked to reach a goal position by moving around an elongated wall (black tiles in Figure 3.4). The curricula for this task control the goal position to be reached via the context c . We investigate three different distance functions of \mathcal{C} in this environment:

- A Euclidean distance $d_E(c_1, c_2) = \|\mathbf{r}(c_1) - \mathbf{r}(c_2)\|$ based on representations $\mathbf{r}(c) \in \mathbb{R}^3$ of the discrete contexts which encode the two-dimensional goal position as well as the height (walls have a height of 200 and regular tiles a height of zero).
- A shortest-path distance $d_S(c_1, c_2)$ computed using the Dijkstra algorithm. The search graph for the Dijkstra algorithm is built by connecting neighboring contexts using the previously defined Euclidean distance.
- A pseudo-metric investigated by Huang et al. [76] that is based on the optimal policy’s absolute difference in expected return $d_{P^*}(c_1, c_2) = |J_{\pi^*}(c_1) - J_{\pi^*}(c_2)|$. Opposed to the metrics d_E and d_S , this pseudo-metric can assign $d_{P^*}(c_1, c_2) = 0$ for $c_1 \neq c_2$.

While the definition of Wasserstein barycenters is not entirely rigorous for the pseudo-metric d_{p^*} , the introduced approximate algorithms can still operate on it without problems. Huang et al. [76] also investigated this pseudo-metric for the current policy π , leading to a different metric in each algorithm iteration. We investigate this interesting concept in Appendix C.3.2 to keep the chapter short and consistent with the previous sections that assumed a fixed distance. Figure 3.4 visualizes the barycentric interpolations generated by d_E , d_S , and d_{p^*} . Looking at Figure 3.4, we can already anticipate a detrimental effect of the Euclidean metric d_E on the generation of the curriculum. The visualization of d_{p^*} indicates a weakness of purely performance-based metrics since a similar expected return for c_1 and c_2 does not guarantee similar outcomes of actions in the two contexts. We visualize the expected return for different curricula in Figure 3.4d. As we can see, CURROT and GRADIENT can significantly improve performance over both a purely random- as well as no curriculum. However, the performance gains are highly dependent on an appropriate choice of metric. While both CURROT and GRADIENT show strong performance for d_S , CURROT’s performance diminishes for d_{p^*} , and none of the two methods can make the agent proficient on $\mu(c)$ when using d_E . Figure 3.4b shows interpolations generated by CURROT for the investigated metrics. We see that the interpolating distributions of

Table 3.1.: Final agent performance of CURROT and GRADIENT on $\mu(c)$ in the E-Maze environment for varying amounts of entropy regularization (λ and H_{LB}). Mean and standard error are computed from 20 seeds.

CURROT				
H_{LB}	0.	0.5	1.0	2.0
d_S	0.62±0	0.61±0	0.53±0.04	0.58±0.03
d_{p^*}	0±0	0.45±0.06	0.38±0.06	0.42±0.06
d_E	0±0	0±0	0±0	0±0

GRADIENT				
λ	0.	10^{-8}	10^{-4}	10^{-2}
d_S	0.60±0.01	0.56±0.04	0.62±0.00	0.60±0.01
d_{p^*}	0.55±0.03	0.48±0.05	0.45±0.05	0.30±0.06
d_E	0.01±0.01	0.03±0.03	0.03±0.03	0.01±0.01

CURROT can collapse to a Dirac distribution for d_S and d_{p^*} . As discussed in Section 3.5, Huang et al. [76] proposed using an entropy-regularized version of optimal transport due to its computational speed. Given that we solve Objectives (3.5) and (3.6) analytically, we can investigate the effect of entropy-regularization not with respect to computational speed but to performance. In Table 3.1, we show the final agent performance when using entropy-regularized transport plans for GRADIENT as well as a lower bound H_{LB} on the entropy of the generated task distributions for CURROT. The detailed formulations of these variants are provided in Appendix C.3.2. As the results show, entropy regularization can benefit CURROT. The visualizations in Figure 3.4c indicate that this benefit arises from avoiding the aggressive targeting of contexts right at the edge of the performance constraint that we can see in Figures 3.1, 3.3b, and 3.4b. In the case of the pseudo distance d_{p^*} , the more diverse tasks sampled from $p(c)$ sometimes allowed the agent to generalize enough to solve tasks sampled from $\mu(c)$. For GRADIENT, we cannot see significant performance gains but can observe that a too-high entropy regularization in combination with d_{p^*} diminished performance. Given that for an adequate metric (i.e., d_S), the observed performance is stable across different amounts of entropy regularization, we do not further explore this avenue in the following experiments.

3.6.2. Unlock-Pickup Environment

In the following environment, we aim to benchmark approximate implementations of CURROT and GRADIENT for large discrete context spaces and demonstrate that appropriate distances for non-trivial context spaces can be designed by hand. In Figure 3.5a, we visualize the unlock-pickup environment from the Minigrid environment collection [33] that we chose for this investigation. To master this environment, the agent must pick up a key, unlock a door and eventually pick up a box in the room that has just been unlocked. We define a curriculum by controlling the starting state of an episode via the context c , i.e., controlling the position of the box, key, agent, and door, as well as the state of the door (whether closed or open). As detailed in Appendix C.3.3, this task parameterization results in 81.920 tasks to compile a curriculum from. The initial context distribution is defined to encode states in which the agent is directly in front of the box, similar to the bottom-right image in Figure 3.5a. Starting from this initial distribution, the learning algorithm needs to generate a curriculum that ultimately allows the agent to reach and pick up the box from a random position in the left room with a closed door. As we show in Appendix C.3.3, it is possible to define a so-called highway distance function [13] between contexts that properly takes the role of the door and its interaction with the key into account, without relying on a planning algorithm like in the previous environment. We use this distance function in the following evaluations.

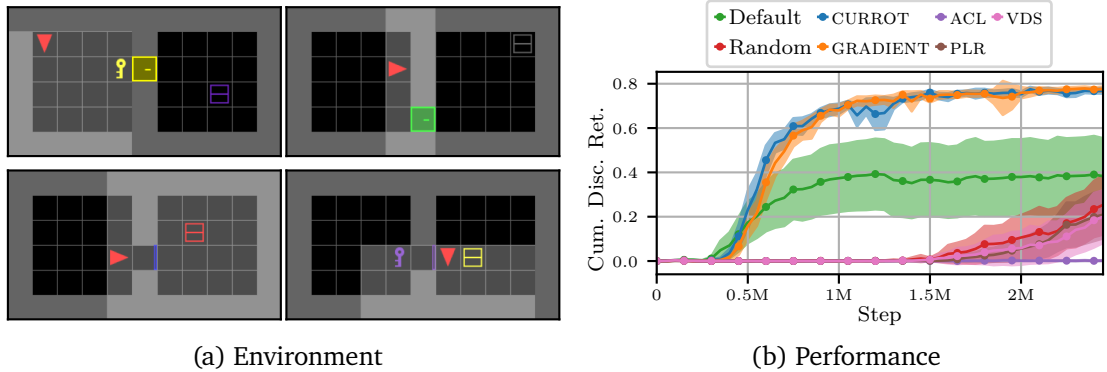


Figure 3.5.: (a) The Unlock-Pickup environment, in which an agent needs to pick up the box in the right room by unlocking the door. After reset, the agent is randomly placed in the left room not carrying the key (top left image). After picking up the key (top right), the door can be unlocked (bottom left) to move to the box (bottom right). The door-, box- and key positions as well as their colors vary across environment resets. The agent receives a partial view of the world (highlighted rectangle) that is blocked by walls and closed doors. (b) Episodic return on the target task distribution $\mu(c)$ in the Unlock-Pickup environment for different curricula. The shaded area corresponds to two times the standard error computed from 20 seeds.

In addition to the approximate versions of CURROT and GRADIENT, we evaluate PLR, VDS, and ACL on this task. We do not evaluate SPRL, ALP-GMM, and GOALGAN since those algorithms have been designed for continuous and Euclidean context spaces by, e.g., leveraging Gaussian distributions, kd-trees, or Gaussian sampling noise. The evaluation results in Figure 3.5b show that CURROT and GRADIENT consistently allow mastering the target tasks (a cumulative discounted return of $0.75 \approx 0.99^{28}$ is obtained by solving a task in 28 steps). For both CURROT and GRADIENT, each of the 20 runs led to a well-performing policy, and we can barely see any difference in learning speed between the approaches. Learning directly on the target task distribution allows mastering the environment in some runs while failing to do so in others due to the high dependence on collecting enough positive reward signals at the beginning of learning. These two outcomes lead, on average, to a lower performance compared to CURROT and GRADIENT. Finally, we see that all baseline curriculum methods learn slower than directly learning on the target task distribution $\mu(c)$, with ACL not producing policies that collect any reward on the target tasks. Given the successful application of PLR in the Procgen benchmark, which features a diverse set of Arcade game levels with highly distinct visual observations, we wish to

discuss the observed low performance of PLR here in more detail. As we show in Appendix C.3.3, PLR indeed samples contexts occurring under $\mu(c)$ with at least 7% in each run. Furthermore, in about half of the runs, the agent also learns to solve those target tasks that are replayed by PLR at some point in the curriculum. However, these replayed target tasks only make up a small fraction of the total number of target tasks, resulting in low performance on all of $\mu(c)$. The absence of a notion of target distribution for PLR seems to lead to ineffective use of samples w.r.t improving performance on the target. This lack of target distribution causing problems will be a re-occurring theme for the subsequent experiments.

3.6.3. Point-Mass Environment

In this environment, in which a point-mass agent must pass through a narrow gate to reach a goal position opposite a wall (Figure 3.6), we benchmark our approximate implementations of CURROT and GRADIENT in continuous settings. The context $c \in \mathbb{R}^2$ controls the position and width of the gate that the agent needs to pass. This environment has been introduced with the SPRL algorithm in Chapter 2 with a Gaussian target distribution that essentially encodes one narrow gate requiring the agent to detour before reaching the target position. Combined with a dense reward based on the Euclidean distance to the goal, the target task is subject to a prominent local minimum that simply moves the agent close to the wall without passing through. We extend this task with a bi-modal target distribution that challenges SPRL’s Gaussian restriction that – as we discussed – is required for it to work properly. As seen in Figures 3.6 and 3.7, CURROT and GRADIENT generate curricula that target both modes of the distribution and allow learning a proficient policy on all of $\mu(c)$. We can also observe the influence of the initial search for feasible contexts in CURROT, which leads to an initial focus of $p(c)$ on tasks with a large gate before matching $\mu(c)$. GRADIENT starts interpolating directly from the initial context distribution, leading to more spread-out samples in $p(c)$ throughout the curriculum. As we show in Appendix

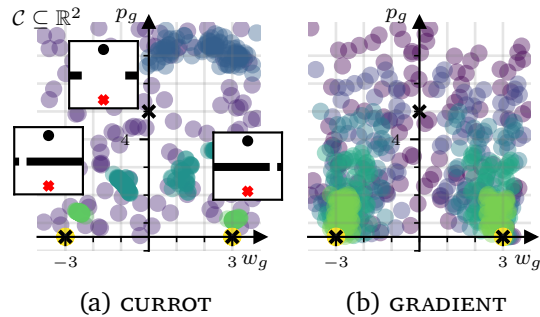


Figure 3.6.: The point-mass environment with its two-dimensional context space. The target distribution $\mu(c)$ encodes the two gates with width $w_g=0.5$, in which the agent (black dot) is required to navigate through a narrow gate at different positions to reach the goal (red cross). The colored dots visualize a curriculum generated by CURROT and GRADIENT for this environment.

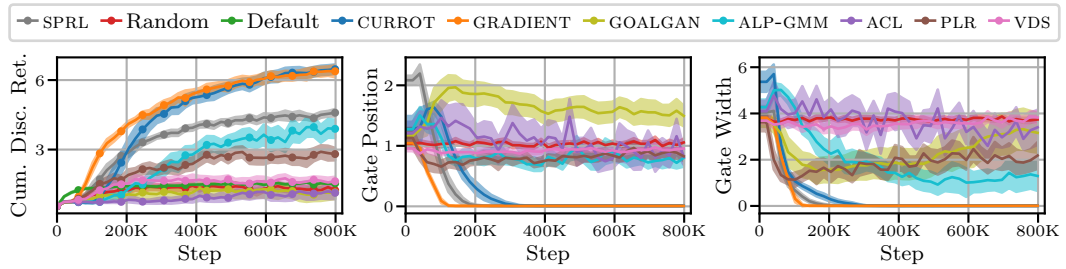


Figure 3.7.: Left: Discounted cumulative return over learning epochs obtained in the point mass environment under different curricula as well as baselines that sample tasks uniformly from all of \mathcal{C} (Random) or $\mu(\mathbf{c})$ (Default). Middle and Right: Median minimum distance to the target contexts of $\mu(\mathbf{c})$ for the two dimensions of the context space (i.e., gate position and -width). Mean and two-times standard error intervals are computed from 20 seeds.

C.3.4, the Gaussian restriction of `SPRL`'s context distribution leads to $p(\mathbf{c})$ matching only one of the modes of $\mu(\mathbf{c})$, resulting in a lower average reward on $\mu(\mathbf{c})$ compared to `CURROT` and `GRADIENT`. We additionally visualize summary statistics for the other CRL methods in Figure 3.7, showing that they result in a less targeted sampling of contexts likely under $\mu(\mathbf{c})$. This observation, in combination with the lower performance compared to `CURROT` and `GRADIENT`, once more emphasizes the importance of embedding a notion of target distribution in CRL algorithms.

3.6.4. Sparse Goal-Reaching Environment

We next turn to a sparse-reward, goal-reaching environment in which an agent needs to reach a desired position with high precision (Figure 3.9). Such environments have, e.g., been investigated by Florensa et al. [56]. The context $\mathbf{c} \in \mathcal{C} \subseteq \mathbb{R}^3$ of this environment encodes the 2D goal position as well as the allowed tolerance for reaching the goal. This parameterization results in both infeasible tasks being part of \mathcal{C} (unreachable regions) as well as tasks that are solely meant to be stepping stones to more complicated ones (low-precision tasks). Given that the agent is ultimately tasked to reach as many goals as possible with the highest precision, i.e., the lowest tolerance, the target distribution $\mu(\mathbf{c})$ is a uniform distribution on a 2D slice of \mathcal{C} in which the tolerance of each context is minimal. The walls in the environment (Figure 3.9) render many tasks encoded by $\mu(\mathbf{c})$ infeasible, requiring the curriculum to identify the feasible subspace of tasks to achieve a good learning performance. Figure 3.8 shows that `CURROT` results

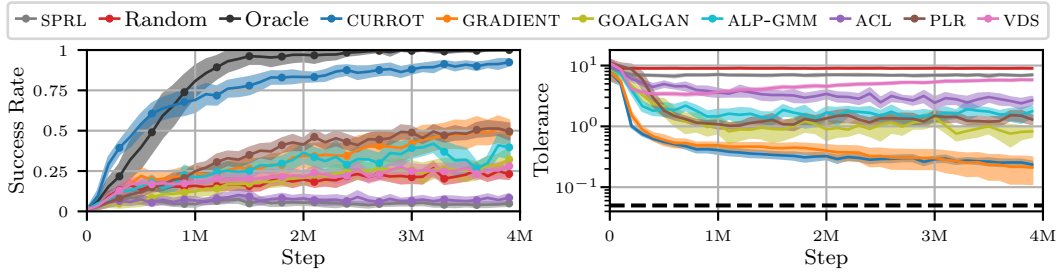


Figure 3.8.: Success rate on the feasible subspace of \mathcal{C} (left) and median goal tolerance (right) for different CRL methods in the SGR environment. We also include an oracle baseline that only samples the feasible tasks in the context space \mathcal{C} . For both plots, mean and two-times standard error intervals are computed from 20 runs.

in the best learning performance across all evaluated CRL methods. Only an oracle, which trains the learning agent only on the feasible subspace of high-precision tasks, can reach higher precision. The evolution of the task tolerances shown in Figure 3.8 highlights that CURROT and GRADIENT continuously increase the precision with which the goals must be reached. The baseline CRL methods lack focus on the tasks encoded by $\mu(c)$, sampling tasks with comparatively high tolerance even towards the end of training. Interestingly, SPRL does not progress to high-precision tasks but continues to sample tasks of high tolerance in later training epochs. As we show in Appendix C.3.5, this behavior is caused by the

Gaussian context distribution of SPRL converging to a quasi-uniform distribution over \mathcal{C} . Otherwise, SPRL would not be able to cover the non-Gaussian target distribution of feasible high-precision tasks without encoding many infeasible tasks. Figure 3.9 shows the evolution of particles for runs of CURROT and GRADIENT. CURROT gradually decreases the goal tolerance over epochs, starting from contexts that are close to the initial position of the agent. Interestingly, it retains higher tolerance contexts located in the walls of the environment even in later epochs due to the trade-off between sampling high-precision

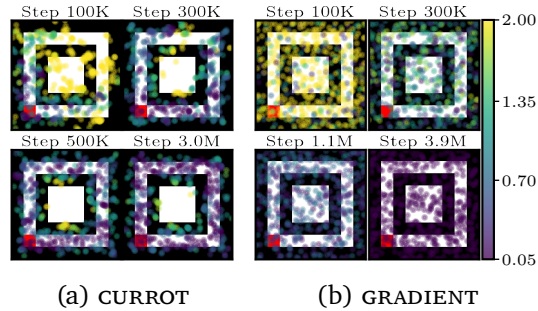


Figure 3.9.: Curricula generated by CURROT and GRADIENT in the spare goal-reaching (SGR) environment at different epochs. The starting area of the agent is highlighted in red. Walls are shown in black. The position of the samples encodes the goal to be reached while the color encodes the goal tolerance.

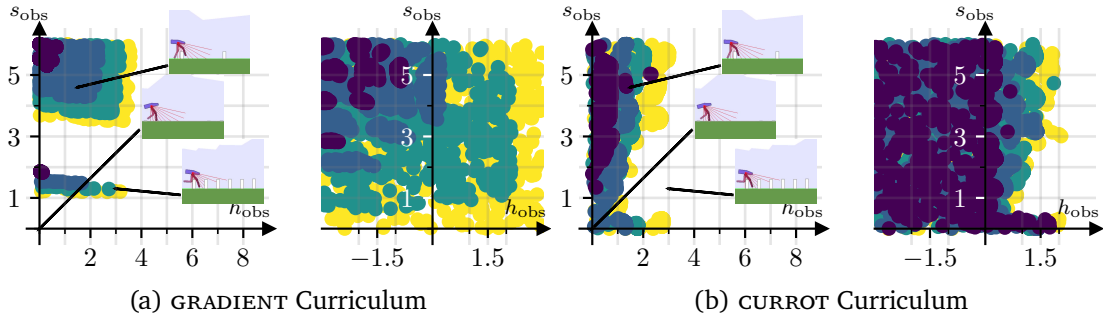


Figure 3.10.: Sampling distribution of GRADIENT and CURROT on the *teach my agent* benchmark in the *no expert knowledge* setting in task spaces with *mostly infeasible*- (left) and *mostly trivial* (right) tasks. The small images visualize the obstacles encoded by the corresponding contexts. For environment details, please see [171]. Brighter colors indicate tasks at later epochs of training. The yellow dots represent the samples from the last generated distribution.

tasks and covering all goal positions. The pre-determined interpolation of GRADIENT cannot adjust to the infeasible parts of the context space and, with that, reduces to a curriculum that shrinks the tolerance interval $[0.05, t_{ub}]$ by reducing the tolerance t_{ub} . Consequently, an increase in precision goes hand-in-hand with an increasing number of infeasible tasks on which the agent is trained, slowing down learning and resulting in a significant performance gap between CURROT and GRADIENT in this environment.

3.6.5. Teach My Agent

In this final evaluation environment, a bipedal agent must learn to maneuver over a track of evenly spaced obstacles of a specified height (see Figure 3.10). The environment is a modified bipedal walker environment introduced by Portelas et al. [162] and extended by Romac et al. [171] in which the spacing and height of obstacles is controlled by the context $\mathbf{c} \in \mathbb{R}^2$. The evaluations by Romac et al. [171] demonstrated poor performance of SPRL, often performing statistically significantly worse than a random curriculum. Given that both CURROT and GRADIENT can be seen as improved versions of SPRL that – among other improvements – explicitly take the geometry of the context space into account, we are interested in whether they can improve upon SPRL. Consequently, we revisit two learning scenarios investigated by Romac et al. [171], in which CRL methods demonstrated a substantial benefit over random sampling: a setting in which most tasks of the context space are infeasible due to large obstacles and a setting in which most tasks of the context

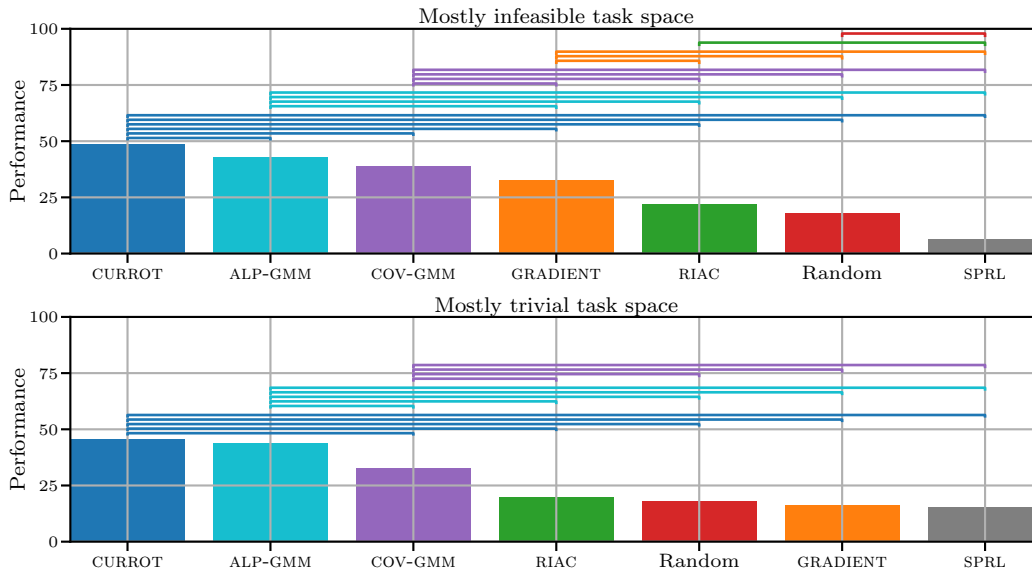


Figure 3.11.: Performance (in percentage of solved tasks) in the *Teach My Agent* benchmark in the *no expert knowledge* setting. The baseline results are taken from [171], and only CURROT and GRADIENT are evaluated by us. Statistics have been computed from 32 seeds. Horizontal lines between connecting two methods indicate statistically significant different performances according to Welch’s t-test with $p < 0.05$.

space are trivially solvable. Both scenarios lead to slow learning progress when choosing tasks randomly due to frequently encountering too complex or too simple learning tasks. Given that both the initial- and target distribution is uniform over the context space \mathcal{C} , we extend the GRADIENT algorithm with the initial feasible context search used in CURROT (see Appendix C.2). Otherwise, GRADIENT would simply result in uniform sampling of \mathcal{C} . Figure 3.11 visualizes the performance of CURROT and GRADIENT in comparison to other CRL methods that were already evaluated by Romac et al. [171]. We see that CURROT achieves the best performance in all environments, in one case performing statistically significantly better than ALP-GMM, the best method evaluated in [171]. We also see that the extended version of GRADIENT can improve upon a random curriculum in the “mostly infeasible” scenario while performing insignificantly worse than a random curriculum in the “mostly trivial” scenario. Figure 3.10 can help shed some light on the observed performance difference between CURROT and GRADIENT. For the “mostly trivial” scenario, GRADIENT consistently arrives at sampling from the uniform $\mu(\mathbf{c})$, whereas CURROT focuses

on the contexts at the border of agent competence. For the “mostly infeasible” scenario, the pre-determined interpolation of GRADIENT can fail to encode feasible learning tasks, ultimately leading to a lower overall performance than CURROT.

Summarizing, the experimental results underline that empirically successful curricula can be generated by framing CRL as an interpolation between context distributions. The leap in performance between GRADIENT and CURROT compared to SPRL and the performance differences between GRADIENT and CURROT underline the impact of design choices, such as the distributional measure of similarity and the way of incorporating performance constraints, on the final algorithm performance. However, when chosen correctly, these curricula exhibit strong performance and allow for guiding training towards tasks of interest specified via $\mu(c)$. Especially this last aspect can allow for more flexibility in the curriculum design, as it is possible to define auxiliary task parameterizations without jeopardizing learning progress toward tasks of interest. We saw an example of this trade-off in the sparse goal-reaching environment, where the additional precision parameter boosted the performance of CURROT while diminishing the performance of other baselines.

3.7. Conclusion

In this chapter, we framed curriculum reinforcement learning as an interpolation between distributions of initial- and target tasks. We demonstrated that the lack of an explicit notion of task similarity in combination with an expected performance constraint makes existing approaches highly dependent on the parameterization of the interpolating task distribution. We avoided these pitfalls by explicitly encoding task similarity via an optimal transport formulation, and by restricting the generated task distributions to only encode tasks that satisfy a specified performance threshold. The resulting method called CURROT led to good performance in experiments due to its focus on tasks at the performance threshold and the adaptive nature of the curriculum. Contrasting our approach to a recently proposed method that generates curricula via Wasserstein barycenters between initial- and target task distributions [76], we saw that the more adaptive nature of our formulation resulted in better performance when facing learning settings with infeasible target tasks. For the future, we believe that the precise notion of task similarity via the distance $d(c_1, c_2)$ can prove beneficial in advancing the understanding of curriculum RL. We already saw that an appropriate definition of task similarity is key to successful curriculum learning. We believe that distances learned from experience, which encode a form of intrinsic motivation, will significantly advance these methods by merging the strong empirical results of intrinsic motivation in open-ended learning scenarios [206] with the targeted learning achieved by CURROT and GRADIENT.

4. Tracking Control for a Spherical Pendulum via Curriculum Reinforcement Learning

Reinforcement Learning (RL) allows learning non-trivial robot control laws purely from data. However, many successful applications of RL have relied on ad-hoc regularizations, such as hand-crafted curricula, to regularize the learning performance. In this chapter, we pair the CURROT algorithm for automatically building curricula with RL on massively parallelized simulations to learn a tracking controller for a spherical pendulum on a robotic arm via RL. Through an improved optimization scheme that better respects the non-Euclidean task structure, we allow the method to reliably generate curricula of trajectories to be tracked, resulting in faster and more robust learning compared to an RL baseline that does not exploit this form of structured learning. The learned policy matches the performance of an optimal control baseline on the real system, demonstrating the potential of curriculum RL to jointly learn state estimation and control for non-linear tracking tasks.

4.1. Introduction

Due to a steady increase in available computation over the last decades, reinforcement learning (RL) [189] has been applied to increasingly challenging learning tasks both in simulated [134, 182] and robotic domains [111, 8, 173]. Learning control of non-trivial systems via reinforcement learning (RL) is particularly appealing when it is required to deal with partially observable systems, high-dimensional observations such as images, or if quick generalization to multiple related tasks is desired. In this chapter, we provide another demonstration of the potential of reinforcement learning to find solutions to a non-trivial control task that has, to the best of our knowledge, not been tackled using learning-based methods. More precisely, we focus on the tracking control of a spherical pendulum attached to a four-degrees-of-freedom Barrett Whole Arm Manipulator (WAM) [194], as shown in Figure 4.1. The partial observability of the system arising from access to only positional information paired with an inherently unstable,

underactuated system and non-trivial kinematics results in a challenge for modern reinforcement learning algorithms. When applying reinforcement learning to increasingly demanding learning tasks, different strategies for improving learning performance, such as guiding learning through highly shaped and -informative reward functions, have evolved.

In this chapter, we improve the training performance of the learning agent via curricula, i.e., tailored sequences of learning tasks that adapt the environment’s complexity to the capability of the learning agent. For the considered tracking task, we adapt the complexity via the target trajectories that are to be tracked by the controller, starting from small deviations from an initial position and progressing to a set of eight-shaped target trajectories. Scheduling the complexity of the learning tasks is subject to ongoing research [146], and solutions to this problem are motivated from different perspectives, such as two-player games [186] or the maximization of intrinsic motivation [15]. In this chapter, we generate the curriculum of tasks using the CURROT algorithm introduced in Chapter 3, which defines the curriculum as a constrained interpolation between an initial- and desired distribution of training tasks and is well-suited to our goal of directing learning to a set of target trajectories. The applications of CURROT have so far relied on training tasks that can be represented in a low-dimensional vector space. In our investigations, we will create a curriculum over desired trajectories, a high-dimensional space of learning tasks, allowing us to benchmark the robustness of the CURROT algorithm to high-dimensional task representations.

We will demonstrate that the sampling-based optimization scheme of CURROT that drives the evolution of the learning tasks faces challenges in high-dimensional settings. Furthermore, the default assumption of a Euclidean distance on the vector space of learning tasks can lead to curricula that do not facilitate learning. Addressing both pitfalls, we obtain robust convergence to the target distribution of tasks, resulting in a tracking controller that can be applied to the real system.

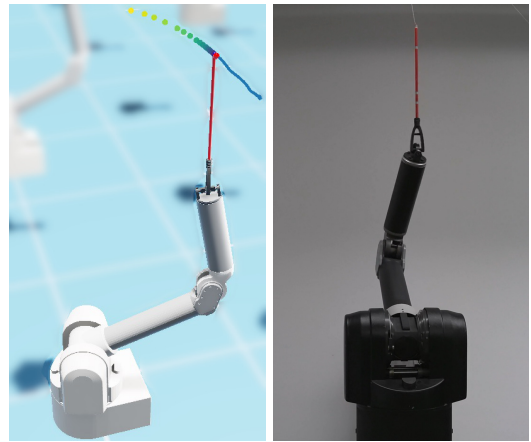


Figure 4.1.: An image of our simulation (left) and robot environment (right) of the spherical pendulum tracking task. The pendulum is mounted to a Barrett WAM robotic arm and is tracked by an Optitrack system. In the left image, the colored dots visualize the upcoming target trajectory to be followed, and the blue line visualizes the achieved trajectory.

Contributions: We demonstrate a simulation-based approach for learning tracking controllers for an underactuated, partially observable, and highly unstable non-linear system that directly transfer to reality. Our approach includes a curriculum reinforcement learning method that reliably works with high-dimensional task spaces equipped with Mahalanobis distances, such as trajectories, commonly encountered in robotics. Through ablations, we confirm the robustness of our method and provide insights into the importance of the policy structure for generalization in tracking tasks.

4.2. Related Work

As of today, there exist many demonstrations of applying reinforcement learning (RL) to real-world robotic problems, ranging from locomotion [173, 31, 114] to object manipulation [65, 111, 116], where the RL agents need to process high-dimensional observations, such as images [111] or grids of surface height measurements [173] in order to produce appropriate actions. The RL agent typically controls the robot via desired joint positions [173, 114], joint position deltas [111], joint velocities [65], or even joint torques [114]. Depending on the application scenario, actions are restricted to a manifold of save actions [65, 116].

Spherical Pendulum: Inverted pendulum systems have been investigated since the 1960s [118] as an archetype of an inherently unstable system and are a long-standing evaluation task for reinforcement learning algorithms [181], with swing-up and stabilization tasks successfully solved on real systems via RL [100, 120]. Other learning-based approaches tune linear quadratic regulators (LQRs) and PID controllers in a data-driven manner to successfully stabilize an inverted pendulum mounted on a robotic arm [126, 47]. The extension of the one-dimensional inverted pendulum task to two dimensions has been widely studied in the control community, resulting in multiple real-world applications in which the pendulum has been mounted either to an omnidirectional moving base [87, 88], a platform driven via leading screws [223], a SCARA robotic arm [185], or a seven degrees-of-freedom collaborative robotic arm [205]. The controllers for these systems were synthesized either via linear controller design in task space [185], a time-variant LQR around pre-planned trajectories [205], linear output regulation [88], sliding-mode control [87], or feedback linearization [223]. In these approaches, the control laws assumed observability of the complete state, requiring specially designed pendulum systems featuring joint encoders or magneto-resistive sensors and additional processing logic to infer velocities.

In this chapter, we learn tracking control of a spherical pendulum on a robotic arm from position-only observations via reinforcement learning. To the best of our knowledge, this

has not yet been achieved, and we believe that the combination of non-trivial kinematics, underactuation, and partial observability is an excellent opportunity to demonstrate the capabilities of modern deep RL agents.

Curriculum Reinforcement Learning: The complexity of this learning task provides an opportunity to utilize methods from the field of curriculum reinforcement learning [146]. These methods improve the learning performance of RL agents in various application scenarios [182, 8, 173] by adaptively modifying environment aspects of a contextual- [67] or, more generally, a configurable Markov Decision Process [133]. As do their application scenarios, motivations for and realizations of these algorithms differ widely, e.g., in the form of two-player games [186, 45], approaches that maximize intrinsic motivation [15, 162], or as interpolations between task distributions [29, 95]. We will focus on the CURROT algorithm, which we introduced in Chapter 3 and belongs to the last category of approaches. It is well suited for our goal of learning to track a specific set of target trajectories and has so far been applied to rather low-dimensional settings, allowing us to extend its application scenarios to the high-dimensional space of trajectories faced here.

4.3. Reinforcement Learning System

This section describes the trajectory tracking task and its simulation in IsaacSim [150]. Further, we re-state the main ideas of the curriculum learning approach from Chapter 3, which we utilize to speed up learning in this environment.

4.3.1. Simulation Environment and Policy Representation

As shown in Figure 4.1, we aim to learn a tracking task of a spherical pendulum that is mounted on a four-degrees-of-freedom Barrett Whole Arm Manipulator (WAM) [194] via a 3D printed universal joint¹. The robot can be approximately modeled as a rigid body system

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} = \mathbf{c}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}) + \boldsymbol{\tau}_{\text{pad}} \quad (4.1)$$

with six degrees of freedom $\mathbf{q} = [\mathbf{q}_w \ \mathbf{q}_p] \in \mathbb{R}^6$ that represent the joint positions of the Barrett WAM (\mathbf{q}_w) and the pendulum (\mathbf{q}_p), and four control signals $\boldsymbol{\tau} \in \mathbb{R}^4$ that drive the joints of the Barrett WAM, where $\boldsymbol{\tau}_{\text{pad}} = [\boldsymbol{\tau} \ 0 \ 0]$ appends the (always zero) controls for the non-actuated universal joint of the spherical pendulum. The universal joint does

¹We designed the universal joint such that it has a large range of motion. Furthermore, the use of skateboard bearings resulted in low joint friction.

not possess any encoders, and we can infer the state of the pole only through position measurements provided by an OptiTrack system [160] at 120 Hz. Hence, albeit the Barrett WAM can be controlled at 500 Hz and delivers updates on its joint positions at the same frequency, we run the control law only at 125 Hz due to the OptiTrack frequency. In the following, we denote a variable’s value at a discrete time index as x_t and the value at arbitrary continuous time as $x(t)$. We learn a tracking control law for following desired trajectories $\gamma:[t_s, t_e] \mapsto \mathbb{R}^3$ of the pendulum tip from a fixed initial configuration $\mathbf{q}_{w,0}$. The control law generates torques on top of a gravity compensation term $\mathbf{g}(\mathbf{q}_w)$ using a history of positional observations, torques, and information about the desired trajectory γ

$$\begin{aligned} \boldsymbol{\tau}_t &= \boldsymbol{\pi}(\mathbf{O}_t, \mathbf{A}_t, \mathbf{T}_t) + \mathbf{g}(\mathbf{q}_{w,t}) & \mathbf{O}_t &= \{\mathbf{o}_{t-i} | i \in [0, K-1]\} \\ \mathbf{T}_t &= \{(\gamma_{t+\Delta_i}, \dot{\gamma}_{t+\Delta_i}) | i \in [1, L]\} & \mathbf{A}_t &= \{\boldsymbol{\tau}_{t-i} | i \in [1, K]\}, \end{aligned} \quad (4.2)$$

where $K=15$, $L=20$, and the Δ_i ’s are spread out over the interval $[0, 1.04]$ (Figure 4.2) to capture both the immediately upcoming positions and velocities of $\gamma(t)$ as well as the future behavior of the trajectory. An observation \mathbf{o}_t is given by the joint position of the Barrett WAM $\mathbf{q}_{w,t}$ as well as a three-dimensional unit vector $\mathbf{x}_{p,t} \in \mathbb{R}^3$ that represents the orientation of the pole (Figure 4.2). In simulation, we compute this vector using the difference between the pendulum tip $\mathbf{x}_{\text{tip},t}$ and the pendulum base $\mathbf{x}_{\text{base},t}$. In the real system, we compute this vector from OptiTrack measurements of four points on the pendulum. We reconstruct neither the pendulum joint positions $\mathbf{q}_{p,t}$ nor the joint velocities $\dot{\mathbf{q}}_t$ as this information is implicitly contained in the observation- and action histories \mathbf{O}_t and \mathbf{A}_t .

We learn π via the *proximal policy optimization* (PPO) algorithm implemented in the *RL Games* library [123]. This choice is motivated by our use of the IsaacSim simulation environment [150], which allows us to simulate a large number of environments in parallel on a single GPU². The chosen PPO implementation is designed to leverage this

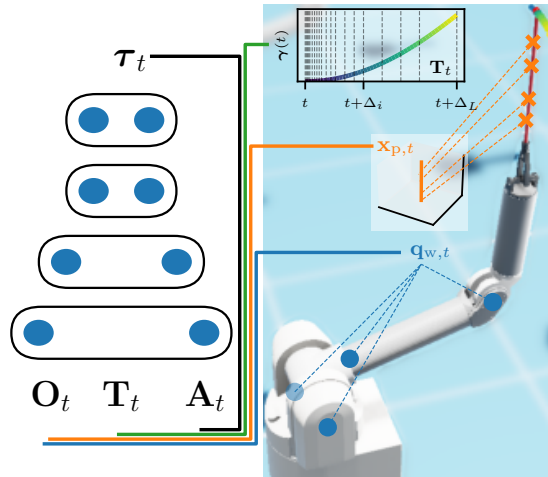


Figure 4.2.: The policy (a feedforward network with $[1024, 512, 256, 256]$ -dimensional hidden layers) observes a history \mathbf{O}_t of joint positions $\mathbf{q}_{w,t}$ and pole directions $\mathbf{x}_{p,t}$, a history \mathbf{A}_t of past actions $\boldsymbol{\tau}_t$, and a lookahead \mathbf{T}_t of the trajectory $\gamma(t)$ to be followed.

²We used 2048 parallel environments for learning.

parallel simulation during training. Screenshots of the simulation environment are shown in Figures 4.1 and 4.2. The trajectories evolve over a total duration of 12 seconds, resulting in $12 \cdot 125 = 1500$ steps per episode. The reward function at a given time-step t mainly penalizes tracking failures and additionally regularizes excessive movement of the robot

$$r(\mathbf{q}_t, \boldsymbol{\tau}_t) = \begin{cases} -\frac{\alpha}{1-\gamma}, & \text{if tipped}(\mathbf{q}_t) \\ 1 - 1000 \|\gamma_t - \mathbf{x}_{\text{tip},t}\|_2^2 - 1e^{-1} \|\dot{\mathbf{q}}_{w,t}\|_2^2 & \\ -1e^{-1} \|\mathbf{q}_{w,t} - \mathbf{q}_{w,0}\|_2^2 - 1e^{-3} \|\boldsymbol{\tau}_t\|_2^2, & \text{else.} \end{cases} \quad (4.3)$$

The function $\text{tipped}(\mathbf{q}_t)$ returns true if either $|\mathbf{q}_{p,t}| \geq 0.5\pi$ or if the z -coordinate of the pendulum tip $\mathbf{x}_{\text{tip},t}$ is less than five centimeters above the z -coordinate of the pendulum base $\mathbf{x}_{\text{base},t}$. The episode ends if $\text{tipped}(\mathbf{q}_t)$ evaluates to true. The large amplification of the tracking error is required since $\|\gamma_t - \mathbf{x}_{\text{tip},t}\|_2$ is measured in meters. With the chosen amplification, a tracking error of three centimeters leads to a penalty of -0.9 . In our experiments, we use a discount factor of $\gamma = 0.992$.

4.3.2. Facilitating Sim2Real Transfer

To enable successful transfer from simulation to reality, we first created a rigid-body model of the Barrett WAM (Eq. 4.1) based on the kinematic and inertial data sheets from Barrett Technology [194] in the MuJoCo physics simulator [195]. We chose the MuJoCo simulator for initial investigations since it allows us to more accurately model the actuation of the Barrett WAM via tendons and differentials³. Opposed to the simplified model (4.1), this more faithful model of the Barrett WAM requires an extended state space $\mathbf{q}_{\text{ext}} = [\mathbf{q}_w \ \mathbf{q}_r \ \mathbf{q}_p] \in \mathbb{R}^{10}$, in which the joint- \mathbf{q}_w and rotor positions \mathbf{q}_r of the Barrett WAM are coupled via tendons that transfer the torques generated at the rotors to the joints (and vice versa). The joint encoders of the WAM are located at the rotors, and hence, we can only observe \mathbf{q}_r , which may differ from \mathbf{q}_w depending on the stiffness of the tendons. During our initial evaluations, we found that modeling this discrepancy between measured- and real joint positions as well as delayed actions (approximated as an exponential filter)

$$\tilde{\boldsymbol{\tau}}_t = \boldsymbol{\omega} \odot \tilde{\boldsymbol{\tau}}_{t-1} + (\mathbf{1} - \boldsymbol{\omega}) \odot \boldsymbol{\tau}_t, \quad \boldsymbol{\omega} \in [0, 1]^4, \quad (4.4)$$

where \odot represents the element-wise multiplication of vectors and $\mathbf{1}$ is a vector of all ones, were required to achieve stable behavior of the learned policy on the real system. When not modeling these effects, the actions generated by the learned policies resulted in unstable

³IsaacSim also has support for tendon modeling. However, this support is significantly more restricted at the moment, preventing to recreate the tendon structure of the Barrett WAM in simulation.

feedback loops. A final extension to the model is given by simulating a Stribeck-like behavior of friction by compensating the coulomb friction modeled by MuJoCo

$$\tilde{\boldsymbol{\tau}}_{a,t} = \tilde{\boldsymbol{\tau}}_t + \mathbf{c} \odot \tanh(\boldsymbol{\beta} \odot \dot{\mathbf{q}}_{w,t}), \quad (4.5)$$

where $\mathbf{c} \in \mathbb{R}_{\geq 0}^4$ is the coefficient of coulomb friction simulated by MuJoCo and $\boldsymbol{\beta} \in \mathbb{R}_{\geq 0}^4$ models the reduction of this friction due to movement. Having completed our model, we then adjusted the tendon stiffness, rotor armature, damping, coulomb friction \mathbf{c} , as well as $\boldsymbol{\omega}$ and $\boldsymbol{\beta}$ using trajectories from the real system.

Given the lack of possibilities to model the tendon drives of the Barret WAM in IsaacSim, we simulate the robot without tendons and model the discrepancies between \mathbf{q}_r observed by the policy and \mathbf{q}_w by a simple spring-damper model

$$\ddot{\mathbf{q}}_r = \mathbf{K}_P(\mathbf{q}_w - \mathbf{T}_q \mathbf{q}_r) + \mathbf{K}_D(\dot{\mathbf{q}}_w - \mathbf{T}_q \dot{\mathbf{q}}_r) + \mathbf{T}_\tau \tilde{\boldsymbol{\tau}}, \quad (4.6)$$

where $\mathbf{T}_q, \mathbf{T}_\tau \in \mathbb{R}^{4 \times 4}$ model the transformation of joint position and -torques via the tendons and $\mathbf{K}_P, \mathbf{K}_D \in \mathbb{R}^{4 \times 4}$ model the spring-damper properties of the tendons.

Given the policy’s reliance on Optitrack measurements of the pendulum, which are exchanged over the network, we measured the time delays arising from the communication over the network stack. We then modeled these delays in the simulation, as detailed in Appendix D.2.

During learning, we randomize the link masses within 75% and 125% of their nominal values and randomize damping and coulomb friction within 50% and 150% of their nominal values. Additionally, we add zero-mean Gaussian distributed noise with a standard deviation of 0.005 to the actions generated by the agent, which are normalized between -1 and 1 . The observations are corrupted by uniform noise within $[-0.01, 0.01]$. Finally, the amount of action delay is also randomized by sampling the elements of $\boldsymbol{\omega}$ from $[0.5, 0.9]$, and $\boldsymbol{\beta}$ is set to zero 25% of the time and sampled from $[0, 100]$ otherwise.

4.3.3. Trajectory Representations

We represent the target trajectories $\gamma: [t_s, t_e] \mapsto \mathbb{R}^3$ via a constrained three-dimensional LTI system that is driven by a sequence of jerks (time-derivatives of accelerations)

$$\forall t \in [t_s, t_e] : \gamma(t) \in \mathcal{P} \quad (4.7)$$

$$\forall t \in [t_s, t_e] : \left\| \frac{d^3}{dt^3} \gamma(t) \right\|_2 \leq j_{UB} \quad (4.8)$$

$$\gamma(t_s) = \gamma(t_e) \quad \dot{\gamma}(t_s) = \dot{\gamma}(t_e) = 0 \quad \ddot{\gamma}(t_s) = \ddot{\gamma}(t_e) = 0 \quad (4.9)$$

with a convex set $\mathcal{P} \subset \mathbb{R}^3$ of allowed positions. We model the LTI system as three individual triple integrator models. For simplicity of exposition, we focus on only one of the three systems, i.e., $\gamma: [t_s, t_e] \mapsto \mathbb{R}$. The full system is obtained by simple “concatenation” of three copies of the following system

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}u(t) \quad \mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (4.10)$$

with $x_i(t) = \frac{d^{i-1}}{dt^{i-1}}\gamma(t)$ and $u(t) = \frac{d^3}{dt^3}\gamma(t)$. To represent the trajectories as some finite-dimensional vectors $\mathbf{u} \in \mathbb{R}^K$, we assume that the control trajectory of jerks $u(t)$ is piecewise constant

$$u(t) = \sum_{k=1}^K u_k 1_k(t), \quad 1_k(t) = \begin{cases} 1, & \text{if } t_{k-1} \leq t < t_k \\ 0, & \text{else} \end{cases}$$

with $t_0 = t_s$ and $t_K = t_e$. This assumption allows us to represent $\mathbf{x}(t)$ at time t as a linear combination of the initial system state and the piece-wise constant jerks

$$\mathbf{x}(t) = \Phi(t_s, t)\mathbf{x}(t_s) + \underbrace{\left[\psi(t_s, t_1, t) \quad \psi(t_1, t_2, t) \quad \dots \quad \psi(t_{K-1}, t_K, t) \right]}_{\Psi(t) \in \mathbb{R}^{3 \times K}} \underbrace{\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_K \end{bmatrix}}_{\mathbf{u} \in \mathbb{R}^K}. \quad (4.11)$$

We derive Φ and ψ in Appendix D.3. With the closed-form solution (4.11), we can rewrite Constraint (4.9) as a system of three linear equations

$$\mathbf{x}(t_e) = \Phi(t_s, t_e)\mathbf{x}(t_s) + \Psi(t_e)\mathbf{u} \Leftrightarrow \mathbf{x}(t_e) - \Phi(t_s, t_e)\mathbf{x}(t_s) = \Psi(t_e)\mathbf{u} \Leftrightarrow \mathbf{0} = \Psi(t_e)\mathbf{u}. \quad (4.12)$$

Since our initial state $\mathbf{x}(t_s)$ is defined as $\mathbf{x}(t_s) = [\gamma(t_s) \ 0 \ 0]$, the form of $\Phi(t_s, t_e)$ implies $\mathbf{x}(t_e) - \Phi(t_s, t_e)\mathbf{x}(t_s) = \mathbf{0}$. We can hence represent all trajectories that fulfill Constraint (4.9) in a $(K-3)$ -dimensional basis of the kernel $\ker(\Psi(t_e))$. We refer to vectors in this kernel as $\tilde{\mathbf{u}} \in \mathbb{R}^{K-3}$. The two remaining constraints (4.7) and (4.8) specify a convex set in $\ker(\Psi(t_e))$. As described in the next section, generating a curriculum over trajectories will require sampling in an ϵ -ball around a given kernel element $\tilde{\mathbf{u}}$ in the convex set of constraints, which we perform using simple rejection sampling.

4.3.4. Curriculum Reinforcement Learning

By now, we can represent target trajectories $\gamma(t)$ via a vector $\mathbf{c} = [\tilde{\mathbf{u}}_1 \ \tilde{\mathbf{u}}_2 \ \tilde{\mathbf{u}}_3] \in \mathcal{C} \subseteq \mathbb{R}^{3(K-3)}$ that encodes the trajectory behavior in the three spatial dimensions. We will treat $\gamma(t)$ and \mathbf{c} interchangeably for the remainder of this chapter. We are interested in learning a policy π that performs well on a target distribution $\mu(\gamma) = \mu(\mathbf{c})$ of trajectories. To facilitate learning, we use the curriculum method `CURROT` developed in Chapter 3, whose main algorithmic aspects we summarize in this section. `CURROT` creates a curriculum of task distributions $p_i(\mathbf{c})$ by iteratively minimizing their Wasserstein distance $\mathcal{W}_2(p, \mu)$ to the target distribution $\mu(\mathbf{c})$ under a given distance function $d(\mathbf{c}_1, \mathbf{c}_2)$ and subject to a performance constraint

$$\arg \min_p \mathcal{W}_2(p, \mu) \quad \text{s.t.} \quad p(\mathcal{V}(\pi, \delta)) = 1, \quad (4.13)$$

where the set $\mathcal{V}(\pi, \delta) = \{\mathbf{c} \in \mathcal{C} \mid J(\pi, \mathbf{c}) \geq \delta\}$ is the set of contexts $\mathbf{c} \in \mathcal{C}$ in which the agent achieves a performance $J(\pi, \mathbf{c}) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{q}_t, \boldsymbol{\tau}_t) \right]$ of at least δ . We refer to Chapter 3 for the precise definition and derivation of the algorithm and, for brevity, only state the resulting algorithm. The task distribution $p_i(\mathbf{c})$ is represented by a set of N particles, i.e. $\hat{p}_i(\mathbf{c}) = \frac{1}{N} \sum_{n=1}^N \delta_{\mathbf{c}_{p_i, n}}(\mathbf{c})$ with $\delta_{\mathbf{c}_{\text{ref}}}(\mathbf{c})$ being the Dirac distribution centered on \mathbf{c}_{ref} . Each particle is updated by minimizing the distance $d(\mathbf{c}, \mathbf{c}_{\mu, \phi(n)})$ to a target particle $\mathbf{c}_{\mu, \phi(n)}$

$$\min_{\mathbf{c} \in \mathcal{C}} d(\mathbf{c}, \mathbf{c}_{\mu, \phi(n)}) \quad \text{s.t.} \quad \hat{J}(\pi, \mathbf{c}) \geq \delta \quad d(\mathbf{c}, \mathbf{c}_{p_i, n}) \leq \epsilon, \quad (4.14)$$

where $\hat{J}(\pi, \mathbf{c})$ is a prediction of $J(\pi, \mathbf{c})$ using Nadaraya-Watson kernel regression [142]

$$\hat{J}(\pi, \mathbf{c}) = \frac{\sum_{l=1}^L K_h(\mathbf{c}, \mathbf{c}_l) J_l}{\sum_{l=1}^L K_h(\mathbf{c}, \mathbf{c}_l)}, \quad K_h(\mathbf{c}, \mathbf{c}_l) = \exp\left(-\frac{d(\mathbf{c}, \mathbf{c}_l)^2}{2h^2}\right). \quad (4.15)$$

The N target particles $\mathbf{c}_{\mu, \phi(n)}$ are sampled from $\mu(\mathbf{c})$ and the permutation $\phi \in \text{Perm}(N)$ assigning them to $\mathbf{c}_{p_i, n}$ is obtained by minimizing an assignment problem

$$\mathcal{W}_2(\hat{p}_i, \hat{\mu}) = \min_{\phi \in \text{Perm}(N)} \left(\frac{1}{N} \sum_{n=1}^N d(\mathbf{c}_{p_i, n}, \mathbf{c}_{\mu, \phi(n)})^2 \right)^{\frac{1}{2}}. \quad (4.16)$$

The parameter ϵ in (4.14) limits the displacements of the particles within one update step, preventing the exploitation of faulty performance estimates $\hat{J}(\pi, \mathbf{c})$. The kernel bandwidth h is set to a fraction of ϵ , e.g., $h = 0.3\epsilon$ in Chapter 3, given its purpose to capture the trend of $J(\pi, \mathbf{c})$ within the trust region around $\mathbf{c}_{p_i, n}$. The L contexts \mathbf{c}_l and episodic returns J_l

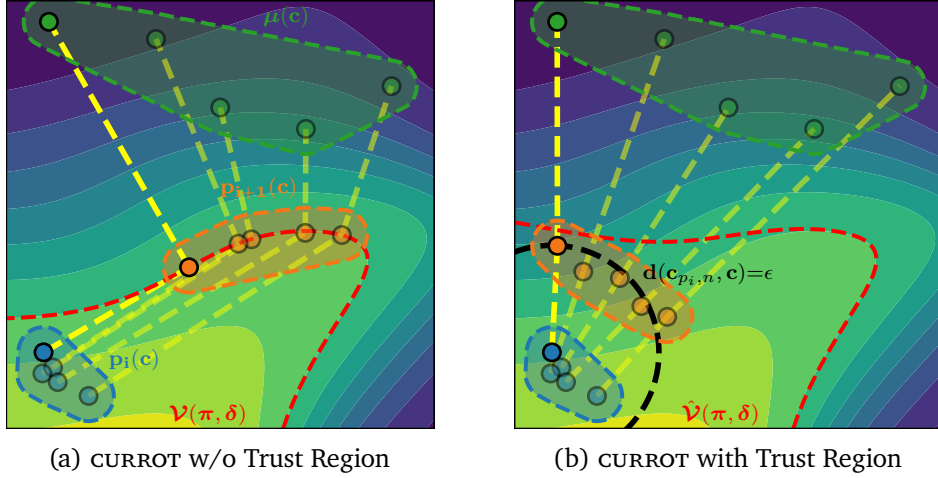


Figure 4.3.: Task sampling scheme used by CURROT. (Left) A particle-based representation $\hat{p}_i(\mathbf{c})$ of the task distribution $p_i(\mathbf{c})$ is updated to minimize the Wasserstein distance $\mathcal{W}_2(\hat{p}_i, \hat{\mu})$ while keeping all particles in the feasible set $\mathcal{V}(\pi, \delta)$ of tasks in which agent π achieves a performance of at least δ . The yellow lines indicate which particles of \hat{p}_i have been matched to $\hat{\mu}$ to compute $\mathcal{W}_2(\hat{p}_i, \hat{\mu})$. (Right) In practice, CURROT needs to rely on an approximation $\hat{\mathcal{V}}(\pi, \delta)$ of $\mathcal{V}(\pi, \delta)$, which is why a trust region $d(\mathbf{c}_{p_i,n}, \mathbf{c}) \leq \epsilon$ is introduced to avoid the overly greedy exploitation of approximation errors. The indicated trust region belongs to the non-opaque particle.

used for predicting the agent performance are stored in two buffers, whose update rules we define in Chapter 3. Figure 4.3 shows a schematic visualization of CURROT. If we can optimize $d(\mathbf{c}_{p_i,n}, \mathbf{c}_{\mu, \phi(n)})$ to zero for each particle in each iteration, we essentially sample from $\mu(\mathbf{c})$. A crucial ingredient in the CURROT algorithm is the distance function $d(\mathbf{c}_1, \mathbf{c}_2)$ that expresses the (dis)similarity between two learning tasks. In Chapter 3, d has been assumed to be the Euclidean distance in continuous space. A critical part of our experimental investigation of the benefit of curricula for learning tracking control will be the comparison of the Euclidean distance between context vectors, \mathbf{c}_1 and \mathbf{c}_2 , and a Mahalanobis distance [122]. In the following section, we describe this distance and other improvements that we benchmark in the experimental section.

4.4. Improved Curriculum Generation

The CURROT algorithm has so far been evaluated in rather low-dimensional scenarios, with two- or three-dimensional context spaces \mathcal{C} that lend themselves to a Euclidean

interpretation. In this section, we describe technical adjustments of the CURROT algorithm that improve the creation of curricula over trajectories, i.e., over a high-dimensional context space \mathcal{C} with a more intricate metric structure.

4.4.1. Affine Metrics

In Chapter 3, we evaluated the CURROT algorithm under the assumption of a Euclidean metric

$$d(\mathbf{c}_1, \mathbf{c}_2) = \|\mathbf{c}_1 - \mathbf{c}_2\|_2 = \sqrt{(\mathbf{c}_1 - \mathbf{c}_2)^T (\mathbf{c}_1 - \mathbf{c}_2)}$$

in continuous context spaces \mathcal{C} . For our trajectory representation, this corresponds to a Euclidean distance between elements in $\ker(\Psi(t_e))$. However, according to Eq. (4.11), we know that the difference between two (one-dimensional) LTI system states is given by

$$\mathbf{x}_1(t) - \mathbf{x}_2(t) = \Psi(t)(\mathbf{u}_1 - \mathbf{u}_2).$$

This observation allows us to compute the similarity of the trajectories $\gamma_1(t)$, $\gamma_2(t)$ generated by \mathbf{c}_1 , \mathbf{c}_2 via a Mahalanobis distance

$$d_{\Psi}(\mathbf{c}_1, \mathbf{c}_2) = \sqrt{(\mathbf{c}_1 - \mathbf{c}_2)^T \mathbf{A} (\mathbf{c}_1 - \mathbf{c}_2)}, \quad \mathbf{A} = \mathbf{\Gamma}_3^T \begin{bmatrix} \Psi_3(t_s) \\ \Psi_3(t_1) \\ \vdots \\ \Psi_3(t_e) \end{bmatrix}^T \begin{bmatrix} \Psi_3(t_s) \\ \Psi_3(t_1) \\ \vdots \\ \Psi_3(t_e) \end{bmatrix} \mathbf{\Gamma}_3,$$

where $\Psi_3(t) = \text{blkdiag}(\{\Psi(t)\}_{n=1}^3)$ and $\mathbf{\Gamma}_3 = \text{blkdiag}(\{\mathbf{\Gamma}\}_{n=1}^3)$ are block diagonal matrices. We denote $\mathbf{\Gamma} \in \mathbb{R}^{K \times K-3}$ as the matrix that maps the elements $\tilde{\mathbf{u}} \in \ker(\Psi(t_e))$ to jerk sequences \mathbf{u} . The Mahalanobis distance can be computed with no change to the algorithm by whitening the contexts \mathbf{c} and computing the Euclidean distance in the whitened space.

4.4.2. Sampling-Based Optimization

The optimization of Objective (4.14) in Chapter 3 is carried out in parallel for all particles $\mathbf{c}_{p_i,n}$ by uniformly sampling contexts in an n -dimensional ϵ -half ball

$$B_{\geq 0}^n(\mathbf{c}_{p_i,n}, \epsilon) = \{\mathbf{c} \mid \|\mathbf{c} - \mathbf{c}_{p_i,n}\|_2 \leq \epsilon \wedge \langle \mathbf{c} - \mathbf{c}_{p_i,n}, \mathbf{c}_{\mu,\phi(n)} \rangle \geq 0\}$$

around $\mathbf{c}_{p_i,n}$ and selecting the sample with minimum distance to $\mathbf{c}_{\mu,\phi(n)}$ that fulfills the performance constraint. $\langle \cdot, \cdot \rangle$ denotes the dot product. In higher dimensions, this sampling

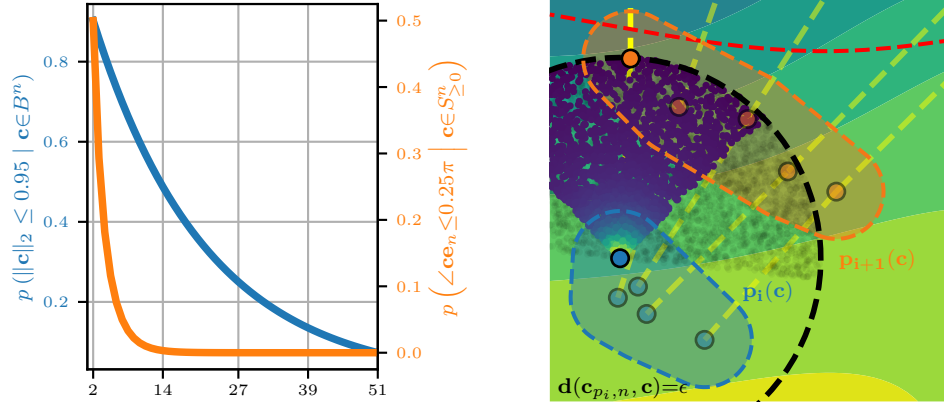


Figure 4.4.: (Left) In higher dimensions, the Euclidean norm of a vector $\|\mathbf{c}\|_2$ in the n -dimensional ball B^n increasingly converges to one. The angle $\angle \mathbf{c}\mathbf{e}_n$ between a context \mathbf{c} in the n -dimensional half sphere $S_{\geq 0}^n$ and any fixed n -dimensional vector \mathbf{e}_n is with increasing certainty larger than $45^\circ = 0.25\pi$. (Right) We adapt the sampling-based optimization of Objective (4.14) to sample those unit vectors that make an angle of less than 45° with a descent direction and scale them uniformly in $[0, \epsilon]$. Unlike the default CURROT sampling scheme (black samples), this sampling scheme (colored dots, color indicates density) more robustly finds descent directions in high-dimensional tasks.

scheme faces two problems. Firstly, the mass of a ball is increasingly concentrated on the surface for higher dimensions, resulting in samples that are increasingly concentrated at the border of the trust region. Secondly, the chance of sampling a context \mathbf{c} for which $d(\mathbf{c}, \mathbf{c}_{\mu, \phi(n)}) < d(\mathbf{c}, \mathbf{c}_{p_i, n})$ decreases dramatically for higher dimensions as soon as $\|d(\mathbf{c}_{p_i, n}, \mathbf{c}_{\mu, \phi(n)})\| \leq \epsilon$. To remedy both problems, we first sample unit vectors that make an angle of less than $\theta = 0.25\pi$ with the descent direction $\mathbf{c}_{\mu, \phi(n)} - \mathbf{c}_{p_i, n}$. Such unit vectors can be sampled using, e.g., the sampling scheme described in [12]. We then scale these search direction vectors by a scalar that we uniformly sample from the interval $[0, \epsilon]$. Figure 4.4 contrasts the new sampling scheme with the one introduced in Chapter 3.

4.4.3. Tracking Metrics Other than Reward

The constraint $p(\mathcal{V}(\pi, \delta))=1$ in Objective (4.13) is controlling the curriculum's progression towards $\mu(\mathbf{c})$ by preventing it from sampling contexts in which the agent does not fulfill a performance threshold δ . We generalize this constraint to define \mathcal{V} based on an arbitrary function $M(\pi, \mathbf{c}) \in \mathbb{R}$ obtained from a rollout of the policy π in a context \mathbf{c} . We hence define

$\mathcal{V}(\pi, \delta) = \{\mathbf{c} \in \mathcal{C} | M(\pi, \mathbf{c}) \geq \delta\}$. The same Nadaraya-Watson kernel regression introduced in Section 4.3.4 can approximate $M(\pi, \mathbf{c})$. In our setting, the increased flexibility enables restricting training to those trajectories for which the agent can stabilize the pendulum throughout almost the whole episode, i.e., almost all of the 1500 episode steps. Encoding this restriction via a fixed lower bound on the episode return is hard to achieve due to, e.g., regularizing terms on the joint velocities and the penalty for non-precise tracking of $\gamma(t)$. These terms can result in highly differing returns for episodes in which the agent stabilized the pendulum throughout the episode.

4.4.4. GPU Implementation

The experiments in Chapter 3 relied on an implementation of CURROT in NumPy [71] and SciPy [204], computing the assignment to the target distribution particles using the SciPy-provided linear sum assignment solver. Given the large number of parallel simulations that we utilize, this application of CURROT needed to work with a large number of particles N and contexts for performance prediction L . We hence created a GPU-based implementation using PyTorch [156]. To solve the assignment problem (4.16), we implemented a default auction algorithm [22] using the PyKeOps library [28], which provides highly efficient CUDA routines for reduction operations on large arrays. We also use the PyKeOps library for the Nadaraya-Watson kernel regression.⁴

4.5. Experiments

In this section, we answer the following questions by evaluating the described learning system in simulation as well as in the real system:

- Do curricula stabilize or speed up learning in the trajectory tracking task?
- How do the proposed changes to the CURROT algorithm alter the generated curriculum and its benefit on the learning agent?
- Does the behavior learned in simulation transfer to the real system?

The experiment requires the agent to track eight-shaped trajectories projected onto a sphere (Figure 4.5). The target distribution $\mu(\gamma)$ of tasks encodes eight-shaped trajectories whose maximal distance to the starting position is 0.36-0.4m in the x -dimension and 0.18-0.2m in the y -dimension. We choose the z -coordinate of the trajectory such that the trajectory has a constant distance to the first joint of the Barrett WAM, i.e., moves on a

⁴Code for this chapter will be made available upon its acceptance in a journal.

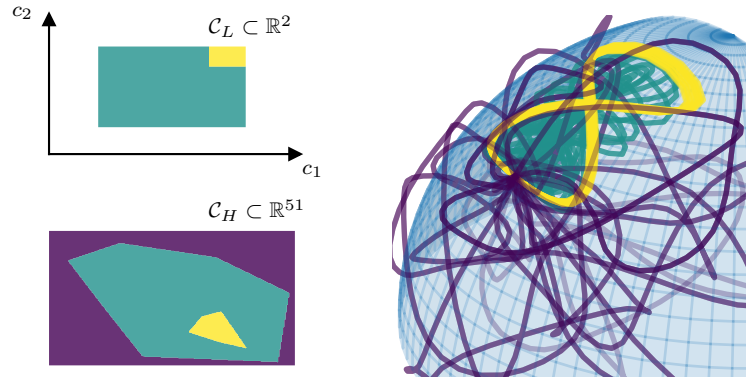


Figure 4.5.: A visualization of the eight-shaped target trajectories $\gamma(t)$ (in yellow) that the learning agent is required to track in our experiments. The trajectories are projected onto a dome that is centered around the robot. Due to the particular shape of the trajectories, we can represent them via both a low- and high-dimensional parametric description, providing the possibility to test how CURROT scales to high-dimensional context representations. Compared to the low-dimensional representation, eight-shaped trajectories are only a small part of the full, high-dimensional context space.

sphere centered on this joint. We chose this particular task since the trajectories encoded by the target distributions $\mu(\gamma)$ can, in addition to the parameterization via jerks, be parameterized in a two-dimensional parameter space. This dual parameterization enables us to benchmark the CURROT algorithm in low- and high-dimensional task parameterizations. For the two-dimensional representation, we represent the maximum distance in x - and y -dimension when generating curricula in the two-dimensional context space $\mathcal{C}_L \subset \mathbb{R}^2$. When representing trajectories via jerks, we compose the jerk sequence \mathbf{u} of $K=20$ constant segments evenly spread in the interval $[1, 10.5]$. The first- and last second of each trajectory is always stationary at $\mathbf{x}(t_s)$. Hence, the actual movement happens within $[1, 11]$. Due to constraint (4.9) of starting and ending in $\mathbf{x}(t_s)$, the parameterization reduces to 17 dimensions for each task space dimension, i.e., $\mathcal{C}_H \subset \mathbb{R}^{51}$. For building the curricula, we define the set $\mathcal{V}(\pi, \delta)$ to contain those trajectories for which the policy manages to keep the pendulum upright for at least 1400 steps, i.e., those trajectories which fully complete their movements during the lifetime of the agent (remember that the policy is stationary for the last second, i.e., the last 125 out of 1500 steps).

We ablate the default PPO learner as well as four ablations of the CURROT method introduced in Section 4.3.4

- CURROT: The default algorithm, as introduced in Chapter 3, using our GPU-based implementation and using $M(\pi, \mathbf{c})$ instead of $J(\pi, \mathbf{c})$ to define $\mathcal{V}(\pi, \delta)$.

- CURROT_L : The default algorithm exploiting the low-dimensional parameterization of the target trajectories to generate curricula in \mathbb{R}^2 instead of \mathbb{R}^{51} .
- CURROT_A : A variation of CURROT that uses the metric d_Ψ to capture the dissimilarity between the generated trajectories rather than the context variables.
- CURROT_{AO} : The version of CURROT that combines the use of d_Ψ with improvements to the sampling-based optimization of Objective (4.14).

For all curricula, we choose the trust region parameter ϵ of Objective (4.14) as in Chapter 3, i.e., setting $\epsilon \approx 0.05 \max_{\mathbf{c}_1, \mathbf{c}_2 \in \mathcal{C}} d(\mathbf{c}_1, \mathbf{c}_2)$. All curricula train on an initial distribution $p_0(\mathbf{c})$ of trajectories that barely deviate from the starting position until $\mathbb{E}_{p_0} [\hat{M}(\pi, \mathbf{c})] \geq \delta$, at which point the methods start updating the context distribution. All methods train for 262 million learning steps, where a policy update is performed after 64 environment steps, resulting in $64 \cdot 2048 = 131072$ samples generated between a policy update.

4.5.1. Quantitative Results

Figure 4.6 shows the performance of the learned policies. More precisely, we show the average tracking error during the agent’s lifetime and the number of completed trajectory steps on $\mu(\gamma)$. While the tracking errors of the investigated methods behave similarly, the curricula shorten the required training iterations until tracking complete target trajectories. The results indicate that by first focusing on trajectories that can be tracked entirely and gradually transforming them into more complicated ones, we avoid repeated sampling of initial parts of the trajectory due to system resets once the pendulum falls over.

We additionally ablate the results over the penalty term α that the agent receives when the pendulum topples over (Eq. 4.3). Figure 4.6 shows that its influence on the learning speed of the agent is limited, as the epochs required by PPO to track the target trajectories completely stays relatively constant even when increasing α by a factor of three.

For the curriculum methods themselves, we can make two observations. First, in the high-dimensional context space $\mathcal{C}_H \subset \mathbb{R}^{51}$, all curricula learn the task reliably with comparable learning speed. Second, operating in the low-dimensional context space $\mathcal{C}_L \subset \mathbb{R}^2$ does not lead to faster learning. Both results surprised us since, in the high-dimensional context space, we expected that exploiting the structure of the context space via $d_\Psi(\mathbf{c}_1, \mathbf{c}_2)$ and the improved optimization would significantly improve the curricula. Furthermore, we expected the low-dimensional representation $\mathcal{C}_L \subset \mathbb{R}^2$ to ease the performance estimation $\hat{J}(\pi, \mathbf{c})$ via kernel regression since we can more densely populate the context space \mathcal{C}_L with samples of the current agent performance. The following section highlights why the observed performance did not behave according to our expectations.

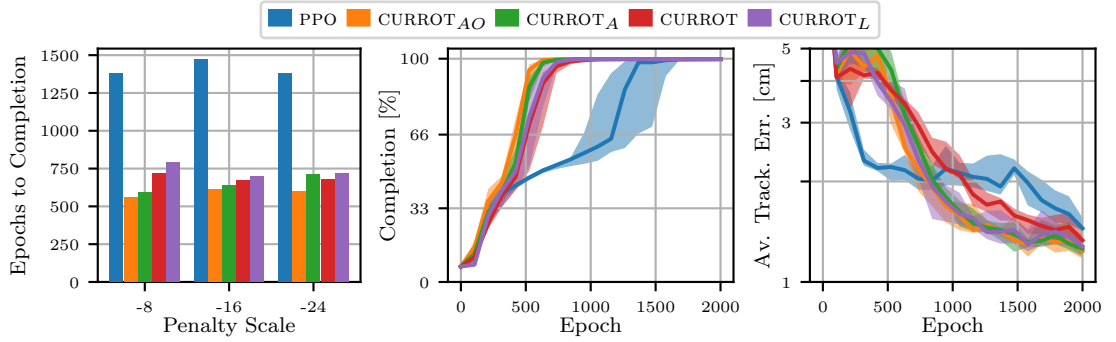


Figure 4.6.: (Left) Ablation over different tipping penalties α and their effect on the required number of epochs until successfully completing the target trajectories. (Middle) Completion rate (i.e. fraction of maximum steps per episode) over epochs for $\alpha = -8$ for different learning methods. (Right) Average tracking error achieved during the agent lifetime over epochs for different learning methods. Thick lines represent the median, and the shaded area represents the interquartile range. Statistics are computed from 10 seeds.

4.5.2. Qualitative Analysis of Generated Curricula

To better understand the dynamics of the generated curricula, we visualize the generated trajectories throughout different learning epochs and the evolution of the Wasserstein distance $\mathcal{W}_2(p_i, \mu)$ in Figures 4.7 and 4.8. Focusing on the evolution of Wasserstein distances shown in Figure 4.8, we can see that only CURROT_{AO} and CURROT_L can converge to the target distribution, achieving zero Wasserstein distance. CURROT and CURROT_A do not converge to $\mu(c)$ after initially exhibiting fast progression towards $\mu(c)$ but slowing down as the Wasserstein distance approaches the value of the trust region parameter ϵ . This slowing-down behavior is precisely due to the naive sampling in the half-ball of the default CURROT algorithm, which we discussed in Section 4.3.4. If the target contexts are well outside the trust region, even samples that do make an angle larger than 0.25π with the descent direction $\mathbf{c}_{\mu, \phi(n)} - \mathbf{c}_{p_i, n}$ decrease the distance to the target $\mathbf{c}_{\mu, \phi(n)}$. Once the target contexts are on or within the boundary of the trust region, the effect visualized in Figure 4.4 takes place, preventing further approach to the target samples. As shown in Figure 4.7, the effect of the resulting bias on the generated trajectories greatly depends on the chosen metric. By measuring dissimilarity via the Euclidean distance between contexts \mathbf{c}_1 and \mathbf{c}_2 , CURROT generates trajectories that behave entirely differently from the target trajectories during the initial and later stages of training. Incorporating domain knowledge via the Mahalanobis distance d_{Ψ} allows CURROT_A to generate trajectories with similar qualitative

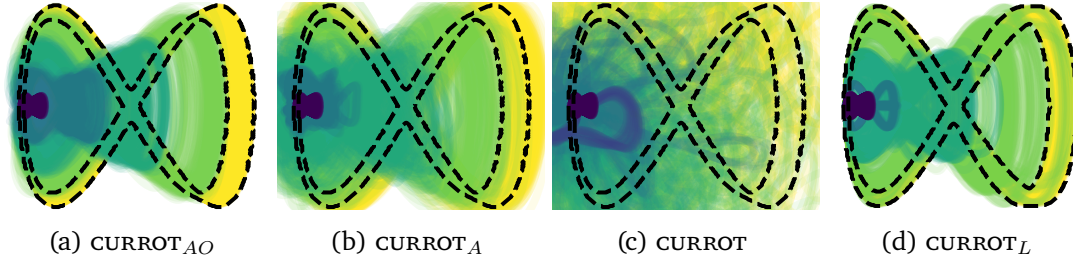


Figure 4.7.: Evaluation of training distributions p_i for different ablations of CURROT. Brighter colors indicate later iterations. The black dotted line indicates the "boundaries" of the support of the target distribution. Note that the distributions have been projected onto a 2D plane, omitting the z-coordinate.

behavior to the target trajectories throughout the whole learning process.

While underlining the importance of the proposed improvements in CURROT_{AO}, the high performance achieved by CURROT and CURROT_A, despite the potentially strong dissimilarity in generated trajectories, stresses a critical observation: The success of a curriculum is inherently dependent on the generalization capability of the learning agent. By conditioning the policy behavior on limited-time lookahead windows of the target trajectory \mathbf{T}_t , the learning agent seems capable of generalizing well to unseen trajectories as long as those trajectories visit similar task-space positions as the trajectories in the training distribution. Consequently, the failure of CURROT to generate trajectories of similar shape to those in $\mu(\gamma)$ is compensated for by the generalization capabilities of the learning agent. All in all, the results indicate that convergence of $p_i(c)$ to $\mu(c)$ is only a sufficient condition for good agent performance on $\mu(c)$, but not a necessary one.

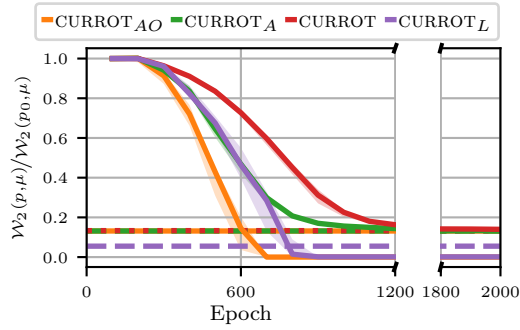


Figure 4.8.: Evolution of Wasserstein distance $\mathcal{W}_2(p, \mu)$ compared to the initial distance over learning epochs for different variants of CURROT. The dashed horizontal lines represent $\frac{\epsilon}{\mathcal{W}_2(p_0, \mu)}$, i.e., the fraction between the trust region ϵ for Objective (4.14) and the initial Wasserstein distance $\mathcal{W}_2(p_0, \mu)$. Thick lines represent the medians, and shaded areas visualize the interquartile range. Statistics are computed from 10 seeds.

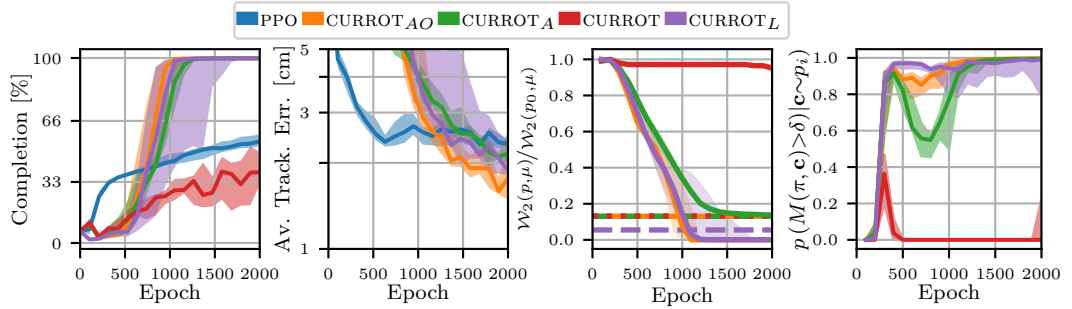


Figure 4.9.: (Left) Completion rate (i.e. fraction of maximum steps per episode) over epochs for $\alpha = -8$ for different learning methods. (Mid-Left) Average tracking error achieved during the agent lifetime over epochs for different learning methods. (Mid-Right) Wasserstein distance between training- and target distribution over epochs. (Right) Percentage for which $M(\pi, \mathbf{c}) \geq \delta$ on the training distribution $p_i(\mathbf{c})$ over epochs. We show median and interquartile ranges that are computed from 10 seeds.

4.5.3. Alternative Trajectory Representation

The surprising effectiveness of CURROT, despite its ignorance of the context space structure, led us to conclude that the policy structure leads to rather strong generalization capabilities of the agent, concealing shortcomings of the generated curricula. We changed the policy architecture to test this hypothesis, replacing the trajectory lookahead \mathbf{T}_t simply by the contextual parameter $\mathbf{c} \in \mathbb{R}^{51}$ and the current time index $t \in \mathbb{R}$. While this representation still contains all required information about the desired target position $\gamma(t)$ at time step t , it does not straightforwardly allow the agent to exploit common subsections of two different trajectories. Figure 4.9 visualizes the results of this experiment. Comparing Figures 4.6 and 4.9, we see that the different context representation slows down the learning progress of all curricula and consequently leads to higher tracking errors after 2000 epochs. We also see that the lookahead \mathbf{T}_t benefits learning with PPO, as with the new context representation, none of the 10 seeds learn to complete the trajectory within 2000 epochs. More importantly, we see how the inadequate metric of CURROT now leads to a failure in the curriculum generation, with $\mathcal{W}_2(p_i, \mu)$ staying almost constant for the entire 2000 epochs as the agent struggles to solve the tasks in the curriculum. This failure to generate tasks of adequate complexity is also shown in Figure 4.9, where we visualize the percentage of tasks \mathbf{c} sampled by the curriculum for which $M(\pi, \mathbf{c}) \geq \delta$. As we can see, this percentage drops to zero under CURROT once the algorithm starts updating $p_0(\mathbf{c})$. The other curricula maintain a non-zero success percentage. We can also observe a

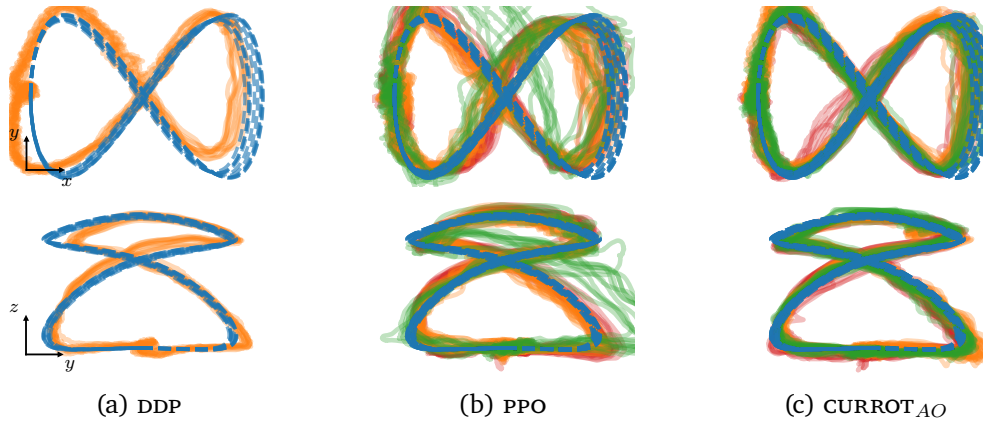


Figure 4.10.: Generated trajectories on the real robot. We visualize projections to the xy - (top) and yz -plane (bottom) to highlight the three-dimensional nature of the trajectory. The reference trajectories are shown in blue. Other colors indicate trajectories that have been generated by the different (learned) controllers. For `PPO` and `CURROTAO`, we evaluate the three best-performing seeds, indicated by colors.

pronounced drop in success rate for `CURROTA`, which is not present for `CURROTAO`. This more stable performance of `CURROTAO` may result from the more targeted sampling in the approximate update step of the context distribution, resulting in more similar trajectories. Apart from this ablation, we also performed experiments for increasing context space dimensions, generating curricula in up to 399-dimensional context spaces. The results in Appendix D.1 show that `CURROTAO` generates beneficial curricula across all investigated dimensions and trajectory representations, while the curricula of `CURROTA` become less effective in higher dimensions for the alternative trajectory representation presented in this section. For `CURROT`, the resulting picture stays unchanged with poor observed performance for the alternative trajectory representation regardless of the context space dimension.

4.5.4. Real Robot Results

To assess transferability to the real world and put the achieved results into perspective, we evaluated the three best policies learned with `PPO` and `CURROTAO` for $\alpha = -8$ on the real robot and compared them to an optimal control baseline. We evaluated each seed on 10 trajectories sampled from $\mu(\gamma)$. The target trajectories are shown in Figure 4.10, and Figure 4.11 shows snapshots of the policy execution on the real system. Given the architectural simplicity of the agent policy, it was easy to embed it in a C++-based ROS

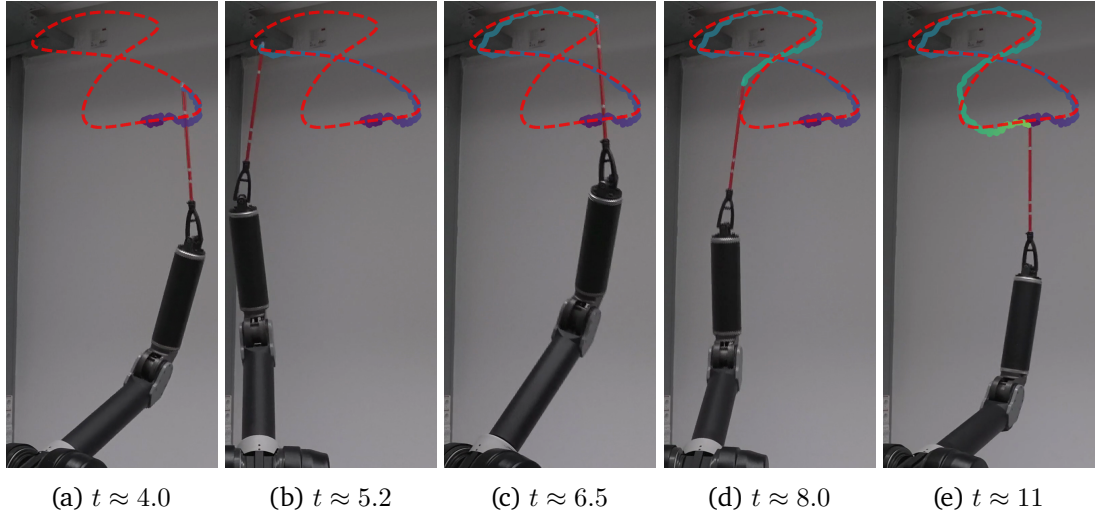


Figure 4.11.: Snapshots of the policy learned with CURROT_{AO} during execution on the real robot. The dotted red line visualizes the target trajectory to be tracked by the policy. The generated trajectory is visualized by the colored line, where brighter colors indicate later time-steps.

[164] controller using the Eigen library, receiving the pole information from Optitrack via UDP packets. The execution time of the policy network was less than a millisecond and hence posed no issue for our target control frequency of 125 Hz. Given that the pole starts in an upright position during training, we attached a thread to the tip of the pendulum to stabilize the pendulum via a pulley system before starting the controller. We then simultaneously release the thread and start the controller. Given the negligible weight of the thread, we did not observe any interference with the pole. To ensure safety during the policy execution, we first executed the policies in a MuJoCo simulation embedded in the ROS ecosystem. We monitored the resulting minimum- and maximum joint positions \mathbf{q}_{\min} and \mathbf{q}_{\max} , and defined a safe region \mathcal{S} , which the agent is not supposed to leave during execution on the real system

$$\mathcal{S} = [\bar{\mathbf{q}} - 1.25(\mathbf{q}_{\max} - \mathbf{q}_{\min}), \bar{\mathbf{q}} + 1.25(\mathbf{q}_{\max} - \mathbf{q}_{\min})],$$

where $\bar{\mathbf{q}} = 1/2(\mathbf{q}_{\min} + \mathbf{q}_{\max})$. Inspired by the results of Vu, Hartl-Nesic, and Kugi [205], we decided to compare the results of PPO and CURROT_{AO} to an optimal control baseline that we obtained by computing a time-varying linear feedback controller using the differential dynamic programming (DDP) algorithm implemented in the Crocoddyl library [128]. At the convergence of DDP, we can obtain a time-varying linear controller from the internally

computed linearization of the dynamics on the optimal trajectory. We use the same cost function as for the reinforcement learning agent, simply removing the penalty term for tipping the pendulum, as the gradient-based DDP does not run into danger of tipping the pendulum. The obtained time-varying controller requires access to full state information, i.e., position and velocity of the robot and pole, which we infer using a high-gain non-linear observer [89], whose gains we tuned on the real system using the synthesized controller

Table 4.1.: Mean and standard deviation of tracking errors achieved with different controllers. We evaluate both on a ROS-embedded MuJoCo simulation (Sim) and the real robot (Real). For PPO and CURROT_{AO}, the color of the seeds corresponds to the trajectories shown in Figure 4.10. In each row, statistics are computed from 20 seeds.

DDP				
Gain	Sim		Real	
	Completion	Error [cm]	Completion	Error [cm]
High	1.00	0.75±0.02	-	-
Low	1.00	2.58±0.11	1.00	3.11±0.12

CURROT _{AO}				
Seed	Sim		Real	
	Completion	Error [cm]	Completion	Error [cm]
2	1.00	1.98±0.06	1.00	2.30±0.16
4	1.00	1.80±0.07	1.00	2.44±0.09
5	1.00	1.85±0.07	1.00	2.10±0.13
Avg.	1.00	1.88±0.10	1.00	2.28±0.19

PPO				
Seed	Sim		Real	
	Completion	Error [cm]	Completion	Error [cm]
1	1.00	3.10±0.14	1.00	2.82±0.20
5	1.00	2.30±0.08	1.00	2.71±0.20
8	1.00	2.15±0.07	0.55	4.18±0.66
Avg.	1.00	2.52±0.43	0.85	3.07±0.68

to achieve the best tracking performance. We performed the tracking experiments twice on different days, obtaining 20 trajectories per seed and method, from which we can compute statistics. Figure 4.10 visualizes the result of the policy rollouts on the real system, and Table 4.1 provides quantitative data. As shown in Figure 4.10, the policies learned with CURROT_{AO} seem to track the reference trajectories more precisely than the other methods. This impression is backed up by the data in Table 4.1, where the average tracking performance of both DDP and PPO on the real robot is about 35% worse than that of CURROT_{AO} . Comparing the results of the ROS-embedded MuJoCo simulation and the execution on the real robot, we see that, on average, the performance on the real system is about 20% worse across all methods. Regarding reliability, one out of the three policies learned with PPO did not reliably perform the tracking task, as it left the safe region \mathcal{S} during execution. We can observe a significantly worse real robot tracking performance for this particular seed in Figure 4.10. Looking at the DDP results again, we see a distinction between high and low gains. The high gain setting corresponds to precisely using reward function 4.3, resulting in the best tracking performance across all methods in simulation. However, the high gains of the generated time-varying linear feedback controllers resulted in unstable behavior in the real system. To obtain stable controllers in the real system, we needed to increase the regularization of actions, position, and velocity by a factor of 30. We assume that more sophisticated methods that better account for uncertainty in the model parameters could further improve performance. Since our baseline only aims to put the learned behavior into perspective, we did not explore such advanced methods. Instead, we interpreted the results as evidence that deep RL-based methods can learn precise control of highly unstable systems comparable to classical control methods. Looking back at Figure 4.6, we see that the tracking errors in the ROS-embedded MuJoCo simulation in Table 4.1 are slightly worse than the error observed in Isaac Sim, where PPO consistently achieved a tracking error of less than 2cm, and CURROT_{AO} consistently achieved a tracking error of less than 1.5cm. This performance gap may be caused by our approximate modeling of actuation delay and tendons, and we expect additional efforts on system identification, modeling, and domain randomization to close this gap.

4.6. Conclusion

We presented an approach that learns a tracking controller for an inverted spherical pendulum mounted to a four-degrees-of-freedom Barret Whole Arm Manipulator. We showed that increasingly available massively parallel simulators allow off-the-shelf reinforcement learning algorithms paired with curricula to reliably learn this non-trivial partially observable control task across policy- and task-space representations. Our evalu-

ations of curricula and their effect on learning success showed multiple interesting results. Apart from confirming the sample-complexity benefit of learning the tracking task via curricula, we showed that a) the generation of curricula is possible in high-dimensional context spaces and b) that high-dimensionality does not need to make the curriculum generation less efficient. However, we also saw that the very structure of our learning agent was a significant factor in the robustness of the generated curricula, allowing it to track target trajectories that are significantly different from the trajectories encountered in the curricula. These findings motivate future investigations into the interplay between agent generalization and curricula. From a technical point of view, we demonstrated the importance of appropriately encoding the structure of the context space \mathcal{C} via the distance function of CURROT, particularly when the generalization capability of the agent is limited. An interesting next step is to generalize CURROT to work with arbitrary Riemannian manifolds. On the robotic side, we see much potential in applications to other robotic tasks, e.g., locomotion problems. On this particular setup, investigating the control of a non-rigidly attached inverted pendulum would allow us to tackle more complicated movements that, e.g., require the robot to thrust the pendulum into the air and catch it again. Furthermore, a non-rigidly attached pendulum would pose an additional challenge for modeling the system in simulation and deriving controllers using optimal control, as contact friction becomes essential to balancing the non-rigidly attached pendulum.

5. Conclusion and Future Work

In this thesis, we tackled the problem of slow learning and convergence to undesirable behavior in the framework of reinforcement learning. We modeled the problem as learning to solve a distribution of target tasks $\mu(\mathcal{M})$ and proposed methods that generate a sequence of training task distributions $p(\mathcal{M})$ to increase the robustness of learning on the target distribution $\mu(\mathcal{M})$. These methods are often referred to as curriculum reinforcement learning. However, as we have shown throughout this thesis, they can also be seen as approximate implementations of continuation- or annealed inference methods.

We represented the sequence of task distributions $p(\mathcal{M})$ by a sequence of distributions over *contexts* $p(\mathbf{c})$, where the context $\mathbf{c} \in \mathcal{C}$ parameterizes a contextual MDP $\mathcal{M}(\mathbf{c})$. While the experiments in this thesis showed that this formulation is flexible enough to accommodate different scenarios, they also reminded us that we must carefully consider the interpolating behavior of $p(\mathbf{c})$. In a sense, the importance of a gradual change in task complexity along the interpolation $p(\mathbf{c})$ has already been emphasized in early works on curriculum reinforcement learning, e.g., by Asada et al. [11].

A core observation of this thesis is that optimal transport provides a well-suited mechanism for specifying the behavior of $p(\mathbf{c})$, and, with that, a gradual change in task complexity by lifting a distance metric on the learning tasks $\mathcal{M}(\mathbf{c})$ into a distance on the space of distributions over learning tasks. In this thesis's final chapter, we demonstrated that an approach generating intermediate task distributions using notions of optimal transport can scale to problems with high dimensional task spaces \mathcal{C} and enable robust learning of non-trivial tracking behavior on a four-degrees-of-freedom robotic arm.

Throughout the thesis, we showed various successful applications of the developed methods in different prototypical evaluation tasks. However, as we already stated in Chapter 1, there is no guarantee that the presented methods provide a benefit over regular learning in any given task. Instead, these methods should rather be seen as ways of stabilizing learning if standard algorithms learn in an unacceptably slow or unstable fashion. Compared to alternative curriculum reinforcement learning algorithms, the methods developed in this thesis emphasize an explicit notion of task similarity and target task distribution, leading to empirical advantages over existing curriculum reinforcement learning methods if the problem structure at hand violates their implicit assumptions.

We believe and hope that the explicit notion of assumptions in our algorithms will allow for future investigations that ultimately result in a more solid theoretical foundation of the presented methods. A first step towards this goal may be to investigate under which conditions the interpolations $p(c)$ contain a connected solution path from the initial distribution $p_0(c)$ to the target distribution $\mu(c)$ by establishing similar results as probability-one homotopies [35]. Such results would facilitate a better understanding of the situations in which curriculum reinforcement learning can bring a benefit in the first place since disconnected solution paths imply that the policy needs to change abruptly at some point during the curriculum.

Next, our evaluated algorithms rely on approximate or restricted representations of the distributions they are manipulating, as well as estimates of the agent performance over the context space made from a limited and biased set of samples. In Chapter 3, we witnessed the importance of these approximations, showing that parametric restrictions are an important ingredient of the SPRL algorithm. In Chapter 4, we demonstrated that the approximate optimization of CURROT requires a careful treatment in higher dimensions. Therefore, once a better formal understanding of the presented algorithms has been established, an important next step is to see which theoretical guarantees can be preserved in the face of approximations necessary for practice.

The task-space metric $d(\mathcal{M}_1, \mathcal{M}_2)$ introduced in Chapter 3 provides another opportunity for future investigations. We empirically showed that the choice of metric significantly affects the performance of the resulting curriculum, both in a positive and negative sense. Given the empirical importance of a “good” task-space metric, the question arises whether a universally “good” task-space metric exists. In Appendix C.3.2, we investigated a general-purpose (pseudo-)metric initially proposed by Huang et al. [76], which led to a desirable explorative behavior of the generated curriculum in the investigated environment. Understanding whether the observed behavior is simply an artifact of the specific learning task or whether this (pseudo-)metric is a first step towards universally “good” metrics for curriculum reinforcement learning seems a promising next step. The results of such investigations may indeed be connected to the first question of whether we can create curricula that correspond to probability-one homotopies, which guarantee a connected solution path to a local optimum of the target problem.

With few or no prior assumptions, it seems reasonable to assume that general-purpose metrics would need to be constructed or refined with the help of data generated by learning agents, connecting this metric learning problem to the curriculum design problem investigated by Narvekar, Sinapov, and Stone [144], which is modeled as a higher-level *curriculum MDP*. In this framework, approximate methods allowed to learn adaptive curricula purely from data [145, 222]. Hence, efforts to learn a metric structure over learning tasks from data should leverage the insights gained in these works.

Finally, the problem of transferring behavior between two tasks, \mathcal{M}_1 and \mathcal{M}_2 , has been mostly ignored, relying on appropriate choices of the state-, action- and context space \mathcal{S} , \mathcal{A} , and \mathcal{C} to allow transfer to happen implicitly in the chosen parametric policy. We already saw in Chapter 4 that this ignorance can make the success of curricula dependent on these choices. More importantly, such an approach is not suitable in more general settings when, e.g., state- or action-spaces change [192]. In these more general settings, the transfer of behavior is expected to play a vital role in the success of curricula and consequently needs to be accounted for explicitly.

6. Contribution Statements

Albeit summarizing the findings of my scientific work, this thesis has not been written from the perspective of a group (using *We* instead of *I*) by accident. Research is only possible with collaboration, so I wish to disentangle the individual contributions that have led to the papers presented in this thesis.

6.1. Contributions to Chapter 2

The work leading to this chapter started with the SPRL algorithm that I developed during my Master’s thesis under the supervision of Hany Abdulsamad and Boris Belousov. We started the thesis without having a specific idea other than generalizing solutions between learning tasks, and I believe that their continuing support and the time they invested in our discussions tremendously helped me progress towards the final idea behind SPRL. After completing my thesis, I worked on generalizing the initial algorithm to broader application scenarios and establishing the connection to self-paced learning [105], during which I collaborated with Carlo D’Eramo. His expertise in deep reinforcement learning was a great help in setting up the additional experiments. Finally, all authors of this work contributed to its writing.

6.2. Contributions to Chapter 3

The motivation for this work emerged during the Master’s thesis of Haoyi Yang, whom I supervised starting in 2021. Under my lead, we first aimed to remove the parametric restriction of SPRL while staying in the variational inference (VI) framework. While we could show some successes, the formulation and results of the created approach were not fully satisfactory. This dissatisfaction led us to investigate optimal transport approaches, for which Haoyi could do some very first investigations towards the end of his thesis. Convinced by his early results, I performed more research on the weaknesses of the VI-based approaches and how to best integrate the idea of optimal transport into our current algorithmic framework, resulting in both the analysis of the weaknesses of the

KL-based interpolations in Chapter 3 as well as the CURROT algorithm with its features like the Nadaraya-Watson regression and enforcing the performance constraint on individual contexts instead of in expectation. I have performed all the evaluations and comparisons. Carlo D’Eramo and Joni Pajarinen supported my work on the topic through discussions and helped in writing the submitted manuscripts.

6.3. Contributions to Chapter 4

The idea for applying curriculum RL for the tracking task described in this chapter came up while supervising a Bachelor thesis by Jonathan Kinzel, who designed a stabilizing controller for a spherical pendulum mounted on a Barrett WAM. To better understand the more intricate tracking task, Kai Ploeger and I supervised Florian Wolf as an integrated project student, who created a high-precision tracking controller in simulation via model-predictive control.

Based on these successes, I saw a chance to learn tracking control using curriculum RL, simultaneously putting CURROT to the test in high-dimensional context spaces. I have performed the training in simulation and the investigations in simulation and on the robot. Florian helped implement an optimal control baseline for the robot to compare the RL agent against. For the real-world set-up, I could leverage some basic infrastructure for communicating with the Optitrack system our group had set up in previous projects. The ROS stack for the Barrett WAM developed by Kai Ploeger and Alap Kshirsagar tremendously accelerated the robot experiments. I wrote the remaining domain-specific code of the real-world system, e.g., for starting the controller and initializing the experiment.

A. Appendix to Chapter 1

This short appendix serves to precisely define the Markov Decision Process (MDP) \mathcal{M} that was used in Section 1.1.2 to showcase an MDP whose corresponding RL objective (1.8) was equivalent to a function $f(a)$ that only depends on an action $a \in \mathcal{A} \subseteq \mathbb{R}$.

The MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, p, r, p_0 \rangle$ of interest can be constructed from a state space with two states, i.e. $\mathcal{S} = \{s_+, s_-\}$ and an initial state distribution $p_0(s) = \delta_{s_+}(s)$ that assigns all probability to s_+ . The reward function is given by

$$r(s, a) = \begin{cases} f(a), & \text{if } s = s_+, \\ 0, & \text{else.} \end{cases}$$

and the transition probabilities $p(s'|s, a) = \delta_{s_-}(s')$ make the agent transition into s_- and stay in this state regardless of the action. With these definitions, the expected reward reduces to a myopic one-step objective

$$\begin{aligned} J(\pi, \mathcal{M}) &= \mathbb{E}_{p_0(s_0), p(s_{i+1}|s_i, a_i), \pi(a_i|s_i)} \left[\sum_{i=0}^{\infty} \gamma^i r(s_i, a_i) \right] \\ &= \mathbb{E}_{p_0(s_0), \pi(a_0|s_0)} [r(s_0, a_0)] \\ &= \mathbb{E}_{\pi(a_0|s_+)} [r(s_+, a_0)]. \end{aligned}$$

If we optimize over a deterministic policy that always generates the same action, i.e. $\pi_\theta(a|s) = \delta_\theta(a)$, it follows immediately that optimizing $J(\pi_\theta, \mathcal{M})$ is equivalent to optimizing $f(\theta)$ w.r.t. θ . If we chose a Gaussian policy $\pi_\theta(a|s) = \mathcal{N}(a|\mu, \sigma^2)$ with $\theta = [\mu \ \sigma] \in \mathbb{R}^2$, we obtain the objective from Section 1.1.2.

B. Appendix to Chapter 2

B.1. Proof of Theorem 1

We begin by restating the theorem from the main text

Theorem 1. *Alternatingly solving*

$$\min_{\boldsymbol{\theta}, \boldsymbol{\nu}} \mathbb{E}_{p(c|\boldsymbol{\nu})} [l(\mathbf{x}_c, y_c, \boldsymbol{\theta})] + \alpha D_{\text{KL}}(p(c|\boldsymbol{\nu}) \parallel \mu(c))$$

w.r.t. $\boldsymbol{\theta}$ and $\boldsymbol{\nu}$ is a majorize-minimize scheme applied to the regularized objective

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{\mu(c)} \left[\alpha \left(1 - \exp \left(-\frac{1}{\alpha} l(\mathbf{x}_c, y_c, \boldsymbol{\theta}) \right) \right) \right].$$

Proof. To prove the theorem, we make use of the result established by Meng, Zhao, and Jiang [131] that optimizing the SPL objective alternatingly w.r.t. $\boldsymbol{\nu}$ and $\boldsymbol{\theta}$

$$\boldsymbol{\nu}^*, \boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\nu}, \boldsymbol{\theta}} r(\boldsymbol{\theta}) + \sum_{i=1}^N (\nu_i l(\mathbf{x}_i, y_i, \boldsymbol{\theta}) + f(\alpha, \nu_i)), \quad \alpha > 0. \quad (2.1)$$

is a majorize-minimize scheme applied to the objective

$$\min_{\boldsymbol{\theta}} r(\boldsymbol{\theta}) + \sum_{i=1}^N F_{\alpha}(l(\mathbf{x}_i, y_i, \boldsymbol{\theta})), \quad F_{\alpha}(l(\mathbf{x}_i, y_i, \boldsymbol{\theta})) = \int_0^{l(\mathbf{x}_i, y_i, \boldsymbol{\theta})} \nu^*(\alpha, \iota) d\iota. \quad (2.3)$$

Based on this result, the proof of Theorem 1 requires three steps: **First**, we need to show that the function

$$f_{\text{KL},i}(\alpha, \nu) = \alpha \nu (\log(\nu) - \log(\mu(c=i))) - \alpha \nu, \quad (2.6)$$

is a valid self-paced regularizer for objective (2.1) and that the corresponding objective (2.3) has the form of the second objective in Theorem 1. **Second**, we need to show the

equivalence between the SPL objective (2.1) and the probabilistic objective (2.5) for the regularizer $f_{\text{KL},i}$. **Finally**, we need to show that objective (2.5) corresponds to the first objective in Theorem 1 when using $f_{\text{KL},i}$. We begin by restating the axioms of self-paced regularizers defined by [80] to prove the first of the three points. Again making use of the notation $\nu^*(\alpha, l) = \arg \min_{\nu} \nu l + f(\alpha, \nu)$, these axioms are

1. $f(\alpha, \nu)$ is convex w.r.t. ν
2. $\nu^*(\alpha, l)$ is monotonically decreasing w.r.t. l and it holds that $\lim_{l \rightarrow 0} \nu^*(\alpha, l) = 1$ as well as $\lim_{l \rightarrow \infty} \nu^*(\alpha, l) = 0$
3. $\nu^*(\alpha, l)$ is monotonically decreasing w.r.t. α and it holds that $\lim_{\alpha \rightarrow \infty} \nu^*(\alpha, l) \leq 1$ as well as $\lim_{\alpha \rightarrow 0} \nu^*(\alpha, l) = 0$.

It is important to note that, due to the term $\mu(c=i)$ in (2.6), there is now an individual regularizer $f_{\text{KL},i}$ for each sample. This formulation is in line with the theory established by Meng, Zhao, and Jiang [131] and simply corresponds to an individual regularizer $F_{\alpha,i}$ for each sample in (2.3). Inspecting the second derivative of $f_{\text{KL},i}$ w.r.t. ν , we see that $f_{\text{KL},i}(\alpha, \nu)$ is convex w.r.t. ν . Furthermore, the solution to the SPL objective (2.1)

$$\nu_{\text{KL},i}^*(\alpha, l) = \mu(c=i) \exp\left(-\frac{1}{\alpha}l\right) \quad (\text{B.1})$$

fulfills above axioms except for $\lim_{l \rightarrow 0} \nu_{\text{KL},i}^*(\alpha, l) = 1$, since $\lim_{l \rightarrow 0} \nu_{\text{KL},i}^*(\alpha, l) = \mu(c=i)$. However, we could simply remove the log-likelihood term $\log(\mu(c=i))$ from $f_{\text{KL},i}(\alpha, \nu_i)$ and pre-weight each sample with $\mu(c=i)$, which would yield exactly the same curriculum while fulfilling all axioms. We stick to the introduced form, as it eases the connection of $f_{\text{KL},i}$ to the KL divergence between $p(c|\nu)$ and $\mu(c)$. Given that we have ensured that $f_{\text{KL},i}$ is a valid self-paced regularizer, we know that optimizing the SPL objective (2.1) under $f_{\text{KL},i}$ corresponds to employing the non-convex regularizer

$$F_{\text{KL},\alpha,i}(l(\mathbf{x}_i, y_i, \boldsymbol{\theta})) = \int_0^{l(\mathbf{x}_i, y_i, \boldsymbol{\theta})} \nu_{\text{KL},i}^*(\alpha, \nu) d\nu = \mu(c=i) \alpha \left(1 - \exp\left(-\frac{1}{\alpha}l(\mathbf{x}_i, y_i, \boldsymbol{\theta})\right)\right). \quad (\text{B.2})$$

Put differently, optimizing the SPL objective (2.1) with $r(\boldsymbol{\theta}) = 0$ under $f_{\text{KL},i}$ corresponds to optimizing

$$\min_{\boldsymbol{\theta}} \sum_{i=1}^N F_{\text{KL},\alpha,i}(l(\mathbf{x}_i, y_i, \boldsymbol{\theta})) = \min_{\boldsymbol{\theta}} \mathbb{E}_{\mu(c)} \left[\alpha \left(1 - \exp\left(-\frac{1}{\alpha}l(\mathbf{x}_c, y_c, \boldsymbol{\theta})\right)\right) \right],$$

as stated in Theorem 1. As a next step, we notice that entries in the optimal ν for a given θ and α in the probabilistic SPL objective (2.5) are proportional to $\nu_{\text{KL},i}^*(\alpha, l)$ in (B.1), where the factor of proportionality Z simply rescales the variables $\nu_{\text{KL},i}^*$ so that they fulfill the normalization constraint of objective (2.5). Since

$$\mathbb{E}_{p(c|\nu)} [f(\mathbf{x}_c, y_c, \theta)] = \sum_{i=1}^N \nu_i f(\mathbf{x}_i, y_i, \theta)$$

by definition of $p(c|\nu)$ introduced in Section 2.4, we see that consequently the only difference between (2.1) and (2.5) for this particular regularizer is a different weighting of the regularization term $r(\theta)$ throughout the iterations of SPL. More precisely, $r(\theta)$ is weighted by the aforementioned factor of proportionality Z . Since $r(\theta) = 0$ in Theorem 1, SPL (2.1) and the probabilistic interpretation (2.5) introduced in Chapter 2 are exactly equivalent, since a constant scaling does not change the location of the optima w.r.t θ in both (2.1) and (2.5). Consequently, we are left with proving that the PSPL objective (2.5) under $f_{\text{KL},i}$ and $r(\theta) = 0$ is equal to the first objective in Theorem 1. This reduces to proving that $\sum_{i=1}^N f_{\text{KL},i}(\alpha, \nu_i)$ is equal to the KL divergence between $p(c|\nu)$ and $\mu(c)$. Remembering $p(c=i|\nu) = \nu_i$, it follows that

$$\begin{aligned} \sum_{i=1}^N f_{\text{KL},i}(\alpha, \nu_i) &= \alpha \sum_{i=1}^N p(c=i|\nu) (\log(p(c=i|\nu)) - \log(\mu(c=i))) - \alpha \sum_{i=1}^N p(c=i|\nu) \\ &= \alpha D_{\text{KL}}(p(c=i|\nu) \parallel \mu(c=i)) - \alpha. \end{aligned}$$

The removal of the sum in the second term is possible because $\sum_{i=1}^N p(c=i|\nu) = 1$ per definition of a probability distribution. Since the constant value α does not change the optimization w.r.t. ν , this proves the desired equivalence and with that Theorem 1. \square

B.2. Self-Paced Episodic Reinforcement Learning Derivations

This appendix serves to highlight some important details regarding the derivation of the weights (2.12) and (2.13) as well as the dual objective (2.14). The most notable detail is the introduction of an additional distribution $q(\mathbf{c})$ that takes the role of the marginal $\int q(\theta, \mathbf{c}) d\theta$ as well as the regularization of this additional distribution via a KL divergence constraint w.r.t. to the previous marginal $p(\mathbf{c}) = \int p(\theta, \mathbf{c}) d\theta$. This yields the following

objective

$$\begin{aligned}
& \max_{q(\boldsymbol{\theta}, \mathbf{c}), q(\mathbf{c})} \mathbb{E}_{q(\boldsymbol{\theta}, \mathbf{c})} [r(\boldsymbol{\theta}, \mathbf{c})] - \alpha D_{\text{KL}}(q(\mathbf{c}) \parallel \mu(\mathbf{c})) \\
& \text{s.t. } D_{\text{KL}}(q(\boldsymbol{\theta}, \mathbf{c}) \parallel p(\boldsymbol{\theta}, \mathbf{c})) \leq \epsilon & \int q(\boldsymbol{\theta}, \mathbf{c}) \, d\mathbf{c} \, d\boldsymbol{\theta} = 1 \\
& D_{\text{KL}}(q(\mathbf{c}) \parallel p(\mathbf{c})) \leq \epsilon & \int q(\mathbf{c}) \, d\mathbf{c} = 1 \\
& \int q(\boldsymbol{\theta}, \mathbf{c}) \, d\boldsymbol{\theta} = q(\mathbf{c}) \quad \forall \mathbf{c} \in \mathcal{C}.
\end{aligned}$$

However, these changes are purely of technical nature as they allow to derive numerically stable weights and duals. It is straightforward to verify that $D_{\text{KL}}(q(\boldsymbol{\theta}, \mathbf{c}) \parallel p(\boldsymbol{\theta}, \mathbf{c})) \leq \epsilon$ implies $D_{\text{KL}}(q(\mathbf{c}) \parallel p(\mathbf{c})) \leq \epsilon$. Hence, the constraint $\int q(\boldsymbol{\theta}, \mathbf{c}) \, d\boldsymbol{\theta} = q(\mathbf{c})$ guarantees that a solution $q(\boldsymbol{\theta}, \mathbf{c})$ to above optimization problem is also a solution to (2.11). The dual as well as the weighted updates now follow from the Lagrangian

$$\begin{aligned}
\mathcal{L}(q, V, \eta_q, \eta_{\tilde{q}}, \lambda_q, \lambda_{\tilde{q}}) &= \mathbb{E}_{q(\boldsymbol{\theta}, \mathbf{c})} [r(\boldsymbol{\theta}, \mathbf{c})] - \alpha D_{\text{KL}}(q(\mathbf{c}) \parallel \mu(\mathbf{c})) \\
&+ \eta_q (\epsilon - D_{\text{KL}}(q(\boldsymbol{\theta}, \mathbf{c}) \parallel p(\boldsymbol{\theta}, \mathbf{c}))) + \lambda_q \left(1 - \int q(\boldsymbol{\theta}, \mathbf{c}) \, d\mathbf{c} \, d\boldsymbol{\theta} \right) \\
&+ \eta_{\tilde{q}} (\epsilon - D_{\text{KL}}(q(\mathbf{c}) \parallel p(\mathbf{c}))) + \lambda_{\tilde{q}} \left(1 - \int q(\mathbf{c}) \, d\mathbf{c} \right) \\
&+ \int V(\mathbf{c}) \left(\int q(\boldsymbol{\theta}, \mathbf{c}) \, d\boldsymbol{\theta} - q(\mathbf{c}) \right) \, d\mathbf{c}. \tag{B.3}
\end{aligned}$$

Note that we slightly abuse notation and overload the argument q in the definition of the Lagrangian. The update equations (2.12) and (2.13) follow from the two conditions $\frac{\partial \mathcal{L}}{\partial q(\boldsymbol{\theta}, \mathbf{c})} = 0$ and $\frac{\partial \mathcal{L}}{\partial q(\mathbf{c})} = 0$. Inserting (2.12) and (2.13) into equation (B.3) then allows to derive the dual (2.14). We refer to [201] for detailed descriptions on the derivations in the non-contextual setting, which however generalize to the one investigated here.

B.3. Regularized Policy Updates

In order to enforce a gradual change in policy and context distribution not only during the computation of the weights via equations (2.12) and (2.13) but also during the actual inference of the new policy and context distribution, the default weighted linear regression and weighted maximum likelihood objectives need to be regularized. Given a data set of N weighted samples

$$D = \{(w_i^x, w_i^y, \mathbf{x}_i, \mathbf{y}_i) | i = 1, \dots, N\},$$

with $\mathbf{x}_i \in \mathbb{R}^{d_x}$, $\mathbf{y}_i \in \mathbb{R}^{d_y}$, the task of fitting a joint-distribution

$$q(\mathbf{x}, \mathbf{y}) = q_y(\mathbf{y}|\mathbf{x})q_x(\mathbf{x}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\phi(\mathbf{x}), \Sigma_y)\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_x, \Sigma_x)$$

to D while limiting the change with regards to a reference distribution

$$p(\mathbf{x}, \mathbf{y}) = p_y(\mathbf{y}|\mathbf{x})p_x(\mathbf{x}) = \mathcal{N}(\mathbf{y}|\tilde{\mathbf{A}}\phi(\mathbf{x}), \tilde{\Sigma}_y)\mathcal{N}(\mathbf{x}|\tilde{\boldsymbol{\mu}}_x, \tilde{\Sigma}_x),$$

with feature function $\phi : \mathbb{R}^{d_x} \mapsto \mathbb{R}^o$, can be expressed as a constrained optimization problem

$$\begin{aligned} & \max_{\mathbf{A}, \Sigma_y, \boldsymbol{\mu}_x, \Sigma_x} \sum_{i=1}^N (w_i^x \log(q_x(\mathbf{x}_i)) + w_i^y \log(q_y(\mathbf{y}_i|\mathbf{x}_i))) \\ & \text{s.t. } D_{\text{KL}}(p \parallel q) \approx \frac{1}{N} \sum_{i=1}^N D_{\text{KL}}(p_y(\cdot|\mathbf{x}_i) \parallel q_y(\cdot|\mathbf{x}_i)) + D_{\text{KL}}(p_x \parallel q_x) \leq \epsilon. \end{aligned}$$

Note that we employ the reverse KL divergence in the constraint as this is the only form that allows for a closed form solution w.r.t. the parameters of the Gaussian distribution. Due to the unimodal nature of Gaussian distributions as well as the typically small value of ϵ this is a reasonable approximation. Since the distributions p_x , p_y , q_x and q_y are Gaussians, the KL divergences can be expressed analytically. Setting the derivative of the Lagrangian with respect to the optimization variables to zero yields to following expressions of the

optimization variables in terms of the multiplier η and the samples from D

$$\mathbf{A} = \left[\sum_{i=1}^N \left(w_i \mathbf{y}_i + \frac{\eta}{N} \tilde{\mathbf{A}} \phi(\mathbf{x}_i) \right) \phi(\mathbf{x}_i)^T \right] \left[\sum_{i=1}^N \left(w_i + \frac{\eta}{N} \right) \phi(\mathbf{x}_i) \phi(\mathbf{x}_i)^T \right]^{-1},$$

$$\boldsymbol{\Sigma}_y = \frac{\sum_{i=1}^N w_i \Delta \mathbf{y}_i \Delta \mathbf{y}_i^T + \eta \tilde{\boldsymbol{\Sigma}}_y + \frac{\eta}{N} \Delta \mathbf{A} \sum_{i=1}^N \phi(\mathbf{x}_i) \phi(\mathbf{x}_i)^T \Delta \mathbf{A}^T}{\sum_{i=1}^N w_i + \eta},$$

$$\boldsymbol{\mu}_x = \frac{\sum_{i=1}^N w_i \mathbf{x}_i + \eta \tilde{\boldsymbol{\mu}}_x}{\sum_{i=1}^N w_i + \eta},$$

$$\boldsymbol{\Sigma}_x = \frac{\sum_{i=1}^N w_i (\mathbf{x}_i - \boldsymbol{\mu}_x) (\mathbf{x}_i - \boldsymbol{\mu}_x)^T + \eta \left(\tilde{\boldsymbol{\Sigma}}_x + (\boldsymbol{\mu}_x - \tilde{\boldsymbol{\mu}}_x) (\boldsymbol{\mu}_x - \tilde{\boldsymbol{\mu}}_x)^T \right)}{\sum_{i=1}^N w_i + \eta},$$

with $\Delta \mathbf{y}_i = \mathbf{y}_i - \mathbf{A} \phi(\mathbf{x}_i)$ and $\Delta \mathbf{A} = \mathbf{A} - \tilde{\mathbf{A}}$. Above equations yield a simple way of enforcing the KL bound on the joint distribution: Since η is zero if the constraint on the allowed KL divergence is not active, \mathbf{A} , $\boldsymbol{\Sigma}_y$, $\boldsymbol{\mu}_x$ and $\boldsymbol{\Sigma}_x$ can be first computed with $\eta = 0$ and only if the allowed KL divergence is exceeded, η needs to be found by searching the root of

$$f(\eta) = \epsilon - \frac{1}{N} \sum_{i=1}^N D_{\text{KL}}(p_y(\cdot | \mathbf{x}_i) \| q_y(\cdot | \mathbf{x}_i)) + D_{\text{KL}}(p_x \| q_x),$$

where q_y and q_x are expressed as given by above formulas and hence implicitly depend on η . As this is a one-dimensional root finding problem, simple algorithms can be used for this task.

B.4. Experimental Details

This section is composed of further details on the experiments in sections 2.6 and 2.7, which were left out in the main chapter to improve readability. The details are split between the episodic- and step-based scenarios as well as the individual experiments conducted in them.

To conduct the experiments, we use the implementation of ALP-GMM, GOALGAN and SAGG-RIAC provided in the repositories accompanying the papers from Florensa et al. [56] and

Portelas et al. [162] as well as the CMA-ES implementation from Hansen, Akimoto, and Baudis [68]. The employed hyperparameters are discussed in the corresponding sections. Conducting the experiments with SPRL and SPDL, we found that restricting the standard deviation of the context distribution $p(\mathbf{c}|\nu)$ to stay above a certain lower bound σ_{LB} helps to stabilize learning when generating curricula for narrow target distributions. This is because the Gaussian distributions have a tendency to quickly reduce the variance of the sampling distribution in this case. In combination with the KL divergence constraint on subsequent context distributions, this slows down progression towards the target distribution. Although we could enforce aforementioned lower bound via constraints on the distribution $p(\mathbf{c}|\nu)$, we simply clip the standard deviation until the KL divergence w.r.t. the target distribution $\mu(\mathbf{c})$ falls below a certain threshold $D_{\text{KL,B}}$. This threshold was chosen such that the distribution with the clipped standard deviation roughly “contains” the mean of target distribution within its standard deviation interval. The specific values of $D_{\text{KL,B}}$ and σ_{LB} are listed for the individual experiments.

B.4.1. Episodic Setting

For the visualization of the success rate as well as the computation of the success indicator for the GOALGAN algorithm, the following definition is used: An experiment is considered successful, if the distance between final- and desired state (\mathbf{s}_f and \mathbf{s}_g) is less than a given threshold τ

$$\text{Success}(\boldsymbol{\theta}, \mathbf{c}) = \begin{cases} 1, & \text{if } \|\mathbf{s}_f(\boldsymbol{\theta}) - \mathbf{s}_g(\mathbf{c})\|_2 < \tau, \\ 0, & \text{else.} \end{cases}$$

For the Gate and reacher environment, the threshold is fixed to 0.05, while for the ball-in-a-cup environment, the threshold depends on the scale of the cup and the goal is set to

	ϵ	n_{SAMPLES}	BUFFER SIZE	ζ	K_α	σ_{LB}	$D_{\text{KL,B}}$
GATE “GLOBAL”	0.25	100	10	0.002	140	-	-
GATE “PRECISION”	0.4	100	10	0.02	140	-	-
REACHER	0.5	50	10	0.15	90	[0.005 0.005]	20
BALL-IN-A-CUP	0.35	16	5	3.0	15	0.1	200

Table B.1.: Important parameters of SPRL and C-REPS in the conducted experiments. The meaning of the symbols correspond to those presented in the algorithm from the main text and introduced in this appendix.

	δ_{NOISE}	$n_{\text{ROLLOUT}_{\text{GG}}}$	n_{GOALS}	n_{HIST}
GATE "GLOBAL"	0.05	5	100	500
GATE "PRECISION"	0.05	5	100	200
REACHER	0.1	5	80	300
BALL-IN-A-CUP	0.05	3	50	120

Table B.2.: Important parameters of GOALGAN and SAGG-RIAC in the conducted experiments. The meaning of the symbols correspond to those introduced in this appendix.

be the center of the bottom plate of the cup. The policies are chosen to be conditional Gaussian distributions $\mathcal{N}(\boldsymbol{\theta} | \mathbf{A}\phi(\mathbf{c}), \boldsymbol{\Sigma}_{\boldsymbol{\theta}})$, where $\phi(\mathbf{c})$ is a feature function. SPRL and C-REPS both use linear policy features in all environments.

In the reacher and the ball-in-a-cup environment, the parameters $\boldsymbol{\theta}$ encode a feed-forward policy by weighting several Gaussian basis functions over time

$$\mathbf{u}_i(\boldsymbol{\theta}) = \boldsymbol{\theta}^T \boldsymbol{\psi}(t_i), \quad \boldsymbol{\psi}_j(t_i) = \frac{b_j(t_i)}{\sum_{l=1}^L b_l(t_i)}, \quad b_j(t_i) = \exp\left(-\frac{(t_i - c_j)^2}{2L}\right),$$

where the centers c_j and length L of the basis functions are chosen individually for the experiments. With that, the policy represents a Probabilistic Movement Primitive [154], whose mean and covariance matrix are progressively shaped by the learning algorithm to encode movements with high reward.

In order to increase the robustness of SPRL and C-REPS while reducing the sample complexity, an experience buffer storing samples of recent iterations is used. The size of this buffer dictates the number of past iterations, whose samples are kept. Hence, in every iteration, C-REPS and SPRL work with $N_{\text{SAMPLES}} \times \text{BUFFER SIZE}$ samples, from which only N_{SAMPLES} are generated by the policy of the current iteration.

As the employed CMA-ES implementation only allows to specify one initial variance for all dimensions of the search distribution, this variance is set to the maximum of the variances contained in the initial covariance matrices used by SPRL and C-REPS.

For the GOALGAN algorithm, the percentage of samples that are drawn from the buffer containing already solved tasks is fixed to 20%. The noise added to the samples of the GAN δ_{NOISE} and the number of iterations that pass between the training of the GAN $n_{\text{ROLLOUT}_{\text{GG}}}$ are chosen individually for the experiments.

The SAGG-RIAC algorithm requires, besides the probabilities for the sampling modes which are kept as in the original paper, two hyperparameters to be chosen: The maximum number of samples to keep in each region n_{GOALS} as well as the maximum number of

recent samples for the competence computation n_{HIST} . Tables B.1 and B.2 show the aforementioned hyperparameters of C-REPS, SPRL, GOALGAN and SAGG-RIAC for the different environments.

Point-Mass Experiment

The linear system that describes the behavior of the point-mass is given by

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} 5 \\ -1 \end{bmatrix} + \mathbf{u} + \boldsymbol{\delta}, \quad \boldsymbol{\delta} \sim \mathcal{N}(\mathbf{0}, 2.5 \times 10^{-3} \mathbf{I}).$$

The point-mass is controlled by two linear controllers

$$\mathbf{C}_i(x, y) = \mathbf{K}_i \begin{bmatrix} x_i - x \\ y_i - y \end{bmatrix} + \mathbf{k}_i, \quad i \in [1, 2], \quad \mathbf{K}_i \in \mathbb{R}^{2 \times 2}, \quad \mathbf{k}_i \in \mathbb{R}^2, \quad x_i, y_i \in \mathbb{R},$$

where x is the x -position of the point-mass and y its position on the y -axis. The episode reward exponentially decays with the final distance to the goal. In initial iterations of the algorithm, the sampled controller parameters sometimes make the control law unstable, leading to very large penalties due to large actions and hence to numerical instabilities in SPRL and C-REPS because of very large negative rewards. Because of this, the reward is clipped to always be above 0.

Table B.1 shows that a large number of samples per iteration for both the “global” and “precision” setting are used. This is purposefully done to keep the influence of the sample size on the algorithm performance as low as possible, as both of these settings serve as a first conceptual benchmark of our algorithm.

Figure B.1 helps in understanding why SPRL drastically improves upon C-REPS especially in the “precision” setting even with this large amount of samples. For narrow gates, the reward function has a local maximum which tends to attract both C-REPS and CMA-ES, as the chance of sampling a reward close to the true maximum is very unlikely. By first training on contexts in which the global maximum is more likely to be observed and only gradually moving towards the desired contexts, SPRL avoids this sub-optimal solution.

Reacher Experiment

In the reacher experiment, the ProMP encoded by the policy π has 20 basis functions of width $L = 0.03$. The centers are evenly spread in the interval $[-0.2, 1.2]$ and the time interval of the movement is normalized to lie in the interval $[0, 1]$ when computing the activations of the basis functions. Since the robot can only move within the xy -plane, $\boldsymbol{\theta}$

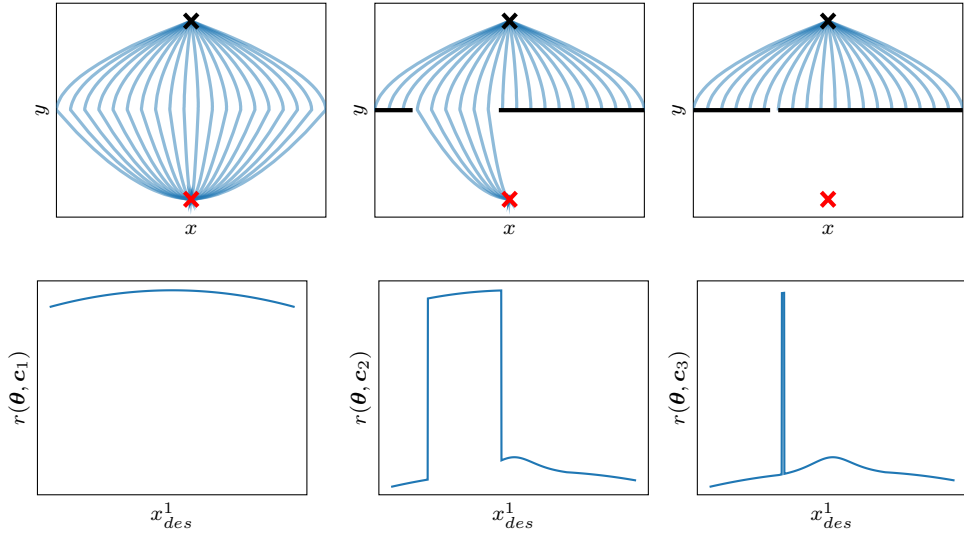


Figure B.1.: The columns show visualizations of the point-mass trajectories (upper plots) as well as the obtained rewards (lower plots) in the point-mass task, when the desired position of the first controller is varied while all other parameters are kept fixed such that a stable control law is obtained. In every column, the gate is positioned at $x = 4.0$ while the size of it varies from 20 (left), over 3 (middle) to 0.1 (right).

is a 40-dimensional vector. As in the previous experiment, the episode reward decays exponentially with the final distance to the goal. As we can see in Table B.1, the number of samples in each iteration was decreased to 50, which in combination with the increased dimensionality of θ makes the task more challenging.

As in the step-based setting, the PPO results are obtained using the version from the `Stable Baselines` library [73]. A step-based version of the reacher experiment is used, in which the reward function is given by

$$r(\mathbf{s}, \mathbf{a}) = \exp\left(-2.5\sqrt{(x - x_g)^2 + (y - y_g)^2}\right),$$

where $\mathbf{s} = (x \ \dot{x} \ y \ \dot{y})$ is the position and velocity of the end-effector, $\mathbf{a} = (a_x \ a_y)$ the desired displacement of the end-effector (just as in the regular reacher task from the OpenAI Gym simulation environment) and x_g and y_g is the x - and y - position of the goal. When an obstacle is touched, the agent is reset to the initial position. This setup led to the best performance of PPO, while resembling the structure of the episodic learning task used

by the other algorithms (a version in which the episode ends as soon as an obstacle is touched led to a lower performance of PPO).

To ensure that the poor performance of PPO is not caused by an inadequate choice of hyperparameters, PPO was run on an easy version of the task in which the two obstacle sizes were set to 0.01, where it encountered no problems in solving the task.

Every iteration of PPO uses 3600 environment steps, which corresponds to 24 trajectory executions in the episodic setting. PPO uses an entropy coefficient of 10^{-3} , $\gamma = 0.999$ and $\lambda = 1$. The neural network that learns the value function as well as the policy has two dense hidden layers with 164 neurons and tanh activation functions. The number of minibatches is set to 5 while the number of optimization epochs is set to 15. The standard deviation in each action dimension is initialized to 1, giving the algorithm enough initial variance, as the actions are clipped to the interval $[-1, 1]$ before being applied to the robot.

Ball-in-a-Cup Experiment

For the ball-in-a-cup environment, the 9 basis functions of the ProMP are spread over the interval $[-0.01, 1.01]$ and have width $L = 0.0035$. Again, the time interval of the movement is normalized to lie in the interval $[0, 1]$ when computing the basis function activations. The ProMP encodes the offset of the desired position from the initial position. By setting the first and last two basis functions to 0 in each of the three dimensions, the movement always starts in the initial position and returns to it after the movement execution. All in all, θ is a 15-dimensional vector. The reward function is defined as

$$r(\theta, \mathbf{c}) = \begin{cases} 1 - 0.07\theta^T\theta & , \text{ if successful} \\ 0 & , \text{ else} \end{cases} .$$

This encodes a preference over movements that deviate as little as possible from the initial position while still solving the task.

Looking back at Table B.1, the value of ζ stands out, as it is significantly higher than in the other experiments. We suppose that such a large value of ζ is needed because of the shape of the reward function, which creates a large drop in reward if the policy is sub-optimal. Because of this, the incentive required to encourage the algorithm to shift probability mass towards contexts in which the current policy is sub-optimal needs to be significantly higher than in the other experiments.

After learning the movements in simulation, the successful runs were executed on the real robot. Due to simulation bias, just replaying the trajectories did not work satisfyingly. At this stage, we could have increased the variance of the movement primitive and re-trained on the real robot. As sim-to-real transfer is, however, not the focus of this chapter, we

	K_α	ζ	K_{OFFSET}	V_{LB}	n_{STEP}	σ_{LB}	$D_{\text{KL},B}$
POINT-MASS (TRPO)	20	1.6	5	3.5	2048	[0.2 0.1875 0.1]	8000
POINT-MASS (PPO)	10	1.6	5	3.5	2048	[0.2 0.1875 0.1]	8000
POINT-MASS (SAC)	25	1.1	5	3.5	2048	[0.2 0.1875 0.1]	8000
ANT (PPO)	15	1.6	10	600	81920	[1 0.5]	11000
BALL-CATCHING (TRPO)	70	0.4	5	42.5	5000	-	-
BALL-CATCHING* (TRPO)	0	0.425	5	42.5	5000	-	-
BALL-CATCHING (PPO)	50	0.45	5	42.5	5000	-	-
BALL-CATCHING* (PPO)	0	0.45	5	42.5	5000	-	-
BALL-CATCHING (SAC)	60	0.6	5	25	5000	-	-
BALL-CATCHING* (SAC)	0	0.6	5	25	5000	-	-

Table B.3.: Hyperparameters for the SPDL algorithm per environment and RL algorithm. The asterisks in the table mark the ball-catching experiments with an initialized context distribution.

decided to manually adjust the execution speed of the movement primitive by a few percent, which yielded the desired result.

B.4.2. Step-Based Setting

The parameters of SPDL for different environments and RL algorithms are shown in Table B.3. Opposed to the sketched algorithm in the main chapter, we specify the number of steps n_{STEP} in the environment between context distribution updates instead of the number of trajectory rollouts. The additional parameter K_{OFFSET} describes the number of RL algorithm iterations that take place before SPDL is allowed to change the context distribution. We used this in order to improve the estimate regarding task difficulty, as for completely random policies, task difficulty is not as apparent as for slightly more structured ones. This procedure corresponds to providing parameters of a minimally pre-trained policy as θ_0 in the algorithm sketched in the main chapter. We selected the best ζ for every RL algorithm by a simple grid-search in an interval around a reasonably working parameter that was found by simple trial and error. For the point-mass environment, we only tuned the hyperparameters for SPDL in the experiment with a three-dimensional context space and reused them for the two-dimensional context space.

Since the step-based algorithm makes use of the value function estimated by the individual RL algorithms, particular regularizations of RL algorithms can affect the curriculum. SAC, for example, estimates a “biased” value function due to the employed entropy regularization. This bias caused problems for our algorithm when working with the α -heuristic

based on V_{LB} . Because of this, we simply replace the value estimates for the contexts by their sample return when working with SAC and V_{LB} . This is an easy way to obtain an unbiased, yet noisier estimate of the value of a context. Furthermore, the general advantage estimation (GAE) employed by TRPO and PPO can introduce bias in the value function estimates as well. For the ant environment, we realized that this bias is particularly large due to the long time horizons. Consequently, we again made use of the sample returns to estimate the value functions for the sampled contexts. In all other cases and environments, we used the value functions estimated by the RL algorithms.

For ALP-GMM we tuned the percentage of random samples drawn from the context space p_{RAND} , the number of policy rollouts between the update of the context distribution n_{ROLLOUT} as well as the maximum buffer size of past trajectories to keep s_{BUFFER} . For each environment and algorithm, we did a grid-search over

$$(p_{\text{RAND}}, n_{\text{ROLLOUT}}, s_{\text{BUFFER}}) \in \{0.1, 0.2, 0.3\} \times \{25, 50, 100, 200\} \times \{500, 1000, 2000\}.$$

For GOALGAN we tuned the amount of random noise that is added on top of each sample δ_{NOISE} , the number of policy rollouts between the update of the context distribution n_{ROLLOUT} as well as the percentage of samples drawn from the success buffer p_{SUCCESS} . For each environment and algorithm, we did a grid-search over

$$(\delta_{\text{NOISE}}, n_{\text{ROLLOUT}}, p_{\text{SUCCESS}}) \in \{0.025, 0.05, 0.1\} \times \{25, 50, 100, 200\} \times \{0.1, 0.2, 0.3\}.$$

The results of the hyperparameter optimization for GOALGAN and ALP-GMM are shown in Table B.4.

Since for all environments, both initial- and target distribution are Gaussians with independent noise in each dimension, we specify them in Table B.5 by providing their mean μ and the vector of standard deviations for each dimension δ . When sampling from a Gaussian, the resulting context is clipped to stay in the defined context space.

The experiments were conducted on a computer with an AMD Ryzen 9 3900X 12-Core Processor, an Nvidia RTX 2080 graphics card and 64GB of RAM.

Point-Mass Environment

The state of this environment is comprised of the position and velocity of the point-mass $s = [x \ \dot{x} \ y \ \dot{y}]$. The actions correspond to the force applied in x- and y-dimension $\mathbf{a} = [F_x \ F_y]$. The context encodes position and width of the gate as well as the dynamic friction coefficient of the ground on which the point-mass slides $\mathbf{c} = [p_g \ w_g \ \mu_k] \in [-4, 4] \times [0.5, 8] \times [0, 4] \subset \mathbb{R}^3$.

	p_{RAND}	$n_{\text{ROLLOUT}_{\text{AG}}}$	s_{BUFFER}	δ_{NOISE}	$n_{\text{ROLLOUT}_{\text{GG}}}$	p_{SUCCESS}
POINT-MASS 3D (TRPO)	0.1	100	1000	0.05	200	0.2
POINT-MASS 3D (PPO)	0.1	100	500	0.025	200	0.1
POINT-MASS 3D (SAC)	0.1	200	1000	0.1	100	0.1
POINT-MASS 2D (TRPO)	0.3	100	500	0.1	200	0.2
POINT-MASS 2D (PPO)	0.2	100	500	0.1	200	0.3
POINT-MASS 2D (SAC)	0.2	200	1000	0.025	50	0.2
ANT (PPO)	0.1	50	500	0.05	125	0.2
BALL-CATCHING (TRPO)	0.2	200	2000	0.1	200	0.3
BALL-CATCHING (PPO)	0.3	200	2000	0.1	200	0.3
BALL-CATCHING (SAC)	0.3	200	1000	0.1	200	0.3

Table B.4.: Hyperparameters for the ALP-GMM and GOALGAN algorithm per environment and RL algorithm. The abbreviation AG is used for ALP-GMM, while GG stands for GOALGAN.

The dynamics of the system are defined by

$$\begin{pmatrix} \dot{x} \\ \ddot{x} \\ \dot{y} \\ \ddot{y} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & -\mu_k & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -\mu_k \end{pmatrix} \mathbf{s} + \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} \mathbf{a}.$$

The x - and y - position of the point-mass is enforced to stay within the space $[-4, 4] \times [-4, 4]$. The gate is located at position $[p_g, 0]$. If the agent crosses the line $y = 0$, we check whether its x -position is within the interval $[p_g - 0.5w_g, p_g + 0.5w_g]$. If this is not the case, we stop the episode as the agent has crashed into the wall. Each episode is terminated after a maximum of 100 steps. The reward function is given by

$$r(\mathbf{s}, \mathbf{a}) = \exp(-0.6 \|\mathbf{o} - [x \ y]\|_2),$$

where $\mathbf{o} = [0 \ -3]$, $\|\cdot\|_2$ is the L2-Norm. The agent is always initialized at state $\mathbf{s}_0 = [0 \ 0 \ 3 \ 0]$.

For all RL algorithms, we use a discount factor of $\gamma = 0.95$ and represent policy and value function by networks using two hidden layers with 64 neurons and tanh activations. For TRPO and PPO, we take 2048 steps in the environment between policy updates.

For TRPO we set the GAE parameter $\lambda = 0.99$, leaving all other parameters to their implementation defaults.

For PPO we use GAE parameter $\lambda = 0.99$, an entropy coefficient of 0 and disable the

	μ_{INIT}	δ_{INIT}	μ_{TARGET}	δ_{TARGET}
POINT-MASS	[0 4.25 2]	[2 1.875 1]	[2.5 0.5 0]	[0.004 0.00375 0.002]
ANT	[0 8]	[3.2 1.6]	[-8 3]	[0.01 0.005]
BALL-CATCHING	[0.68 0.9 0.85]	[0.03 0.03 0.3]	[1.06 0.85 2.375]	[0.8 0.38 1]

Table B.5.: Mean and standard deviation of target and initial distributions per environment.

clipping of the value function objective. The number of optimization epochs is set to 8 and we use 32 mini-batches. All other parameters are left to their implementation defaults. For SAC, we use an experience-buffer of 10000 samples, starting learning after 500 steps. We use the soft Q-Updates and update the policy every 5 environment steps. All other parameters were left at their implementation defaults.

For SPRL, we use $K_\alpha = 40$, $K_{\text{OFFSET}} = 0$, $\zeta = 2.0$ for the 3D- and $\zeta = 1.5$ and 2D case. We use the same values for σ_{LB} and $D_{\text{KL}_{\text{LB}}}$ as for SPDL (Table B.3). Between updates of the episodic policy, we do 25 policy rollouts and keep a buffer containing rollouts from the past 10 iterations, resulting in 250 samples for policy- and context distribution update. The linear policy over network weights is initialized to a zero-mean Gaussian with unit variance. We use polynomial features up to degree two to approximate the value function. For the allowed KL divergence, we observed best results when using $\epsilon = 0.5$ for the weight computation of the samples, but using a lower value of $\epsilon = 0.2$ when fitting the parametric policy to these samples. We suppose that the higher value of ϵ during weight computation counteracts the effect of the buffer containing policy samples from earlier iterations.

Looking at Figure B.2, we can see that depending on the learning algorithm, ALP-GMM, GOALGAN, and a random curriculum allowed to learn policies that sometimes are able to pass the gate. However, in other cases, the policies crashed the point-mass into the wall. Opposed to this, directly training on the target task led to policies that learned to steer the point-mass very close to the wall without crashing (which is unfortunately hard to see in the plot). Reinvestigating the above reward function, this explains the lower reward of GOALGAN compared to directly learning on the target task, as a crash prevents the agent from accumulating positive rewards over time. SPDL learned more reliable and directed policies across all learning algorithms.

Ant Environment

As mentioned in the main chapter, we simulate the ant using the Isaac Gym simulator [151]. This allows to speed up training time by parallelizing the simulation of policy rollouts on the graphics card. Since the Stable-Baselines implementation of TRPO and SAC

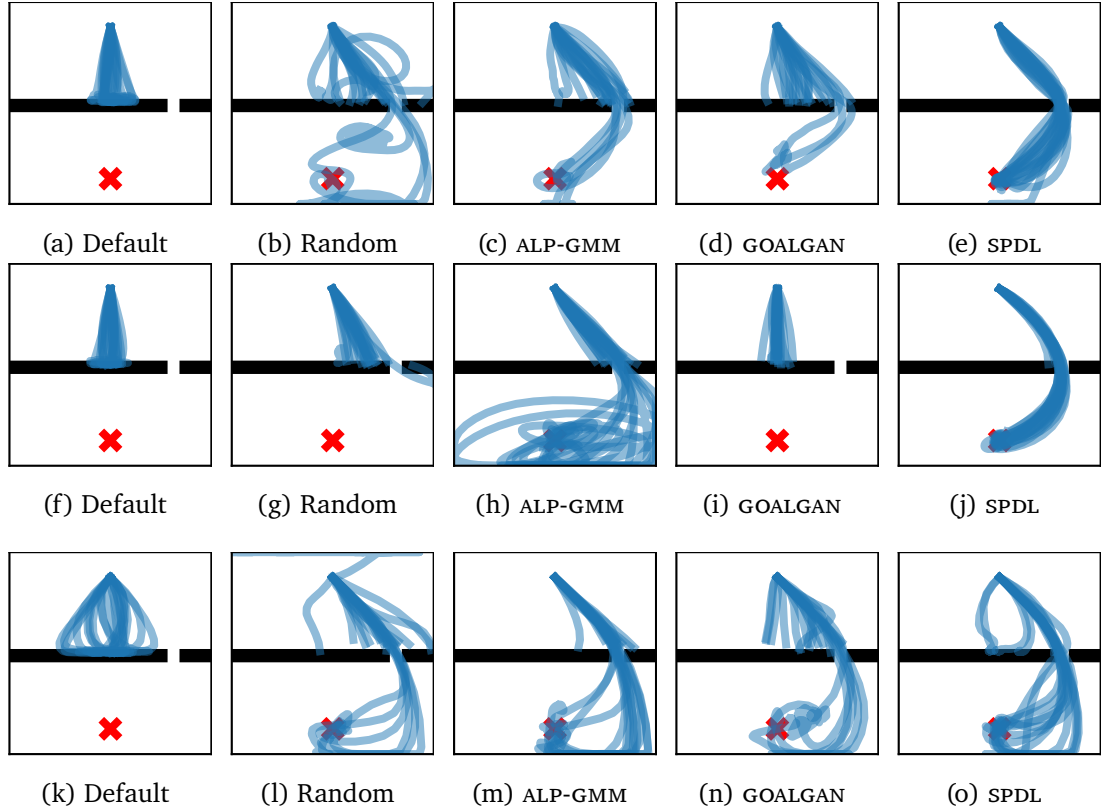


Figure B.2.: Visualizations of policy rollouts in the point-mass environment (three context dimensions) with policies learned using different curricula and RL algorithms. Each rollout was generated using a policy learned with a different seed. The first row shows results for TRPO, the second for PPO and the third shows results for SAC.

do not support the use of vectorized environments, it is hard to combine Isaac Gym with these algorithms. Because of this reason, we decided not to run experiments with TRPO and SAC in the ant environment.

The state $s \in \mathbb{R}^{29}$ is defined to be the 3D-position of the ant’s body, its angular and linear velocity as well as positions and velocities of the 8 joints of the ant. An action $a \in \mathbb{R}^8$ is defined by the 8 torques that are applied to the ant’s joints.

The context $c = [p_g \ w_g] \in [-10, 10] \times [3, 13] \subset \mathbb{R}^2$ defines, just as in the point-mass environment, the position and width of the gate that the ant needs to pass.

The reward function of the environment is computed based on the x -position of the ant’s

center of mass c_x in the following way

$$r(\mathbf{s}, \mathbf{a}) = 1 + 5 \exp(-0.5 \min(0, c_x - 4.5)^2) - 0.3 \|\mathbf{a}\|_2^2.$$

The constant 1 term was taken from the OpenAI Gym implementation to encourage the survival of the ant [27]. Compared to the OpenAI Gym environment, we set the armature value of the joints from 1 to 0 and also decrease the maximum torque from 150Nm to 20Nm, since the values from OpenAI Gym resulted in unrealistic movement behavior in combination with Isaac Gym. Nonetheless, these changes did not result in a qualitative change in the algorithm performances.

With the wall being located at position $x=3$, the agent needs to pass it in order to obtain the full environment reward by ensuring that $c_x \geq 4.5$.

The policy and value function are represented by neural networks with two hidden layers of 64 neurons each and \tanh activation functions. We use a discount factor $\gamma = 0.995$ for all algorithms, which can be explained due to the long time horizons of 750 steps. We take

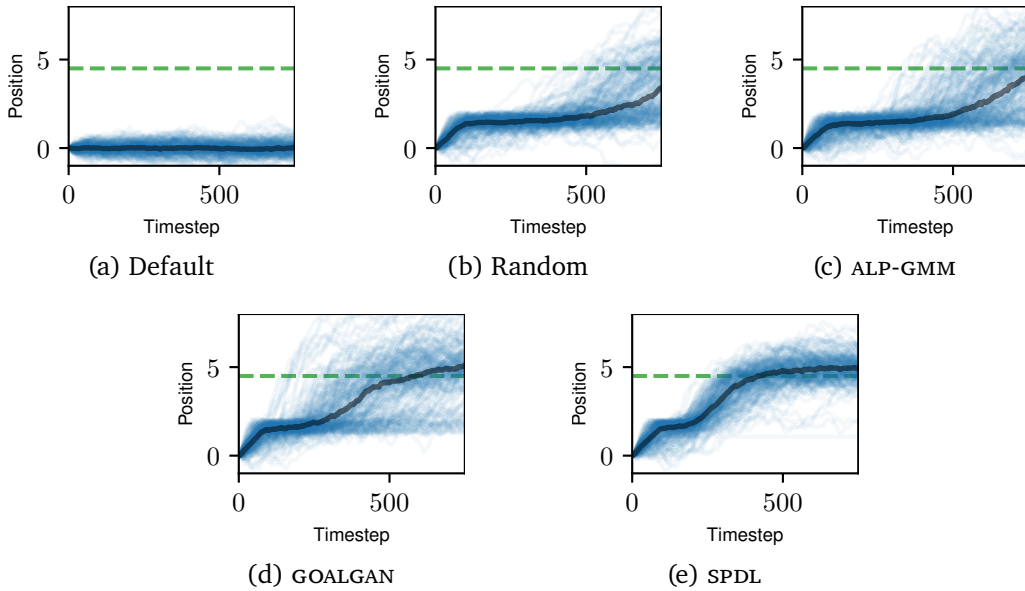


Figure B.3.: Visualizations of the x -position during policy rollouts in the ant environment with policies learned using different curricula. The blue lines correspond to 200 individual trajectories and the thick black line shows the median over these individual trajectories. The trajectories were generated from 20 algorithms runs, where each final policy was used to generate 10 trajectories.

81920 steps in the environment between a policy update. This was significantly sped-up by the use of the Isaac Gym simulator, which allowed to simulate 40 environments in parallel on a single GPU.

For PPO, we use an entropy coefficient of 0 and disable the clipping of the value function objective. All other parameters are left to their implementation defaults. We disable the entropy coefficient as we observed that for the ant environment, PPO still tends to keep around 10 – 15% of its initial additive noise even during late iterations.

Investigating Figure B.3, we see that both SPDL and GOALGAN learn policies that allow to pass the gate. However, the policies learned with SPDL seem to be more reliable compared to the ones learned with GOALGAN. As mentioned in the main chapter, ALP-GMM and a random curriculum also learn policies that navigate the ant towards the goal in order to pass it. However, the behavior is less directed and less reliable. Interestingly, directly learning on the target task results in a policy that tends to not move in order to avoid action penalties. Looking at the main chapter, we see that this results in a similar reward compared to the inefficient policies learned with ALP-GMM and a random curriculum.

Ball-Catching Environment

In the final environment, the robot is controlled in joint space via the desired position for 5 of the 7 joints. We only control a subspace of all available joints, since it is not necessary for the robot to leave the "catching" plane (defined by $x = 0$) that is intersected by each ball. The actions $\mathbf{a} \in \mathbb{R}^5$ are defined as the displacement of the current desired joint position. The state $\mathbf{s} \in \mathbb{R}^{21}$ consists of the positions and velocities of the controlled joints, their current desired positions, the current three-dimensional ball position and its linear velocity.

As previously mentioned, the reward function is sparse,

$$r(\mathbf{s}, \mathbf{a}) = 0.275 - 0.005\|\mathbf{a}\|_2^2 + \begin{cases} 50 + 25(\mathbf{n}_s \cdot \mathbf{v}_b)^5, & \text{if ball caught} \\ 0, & \text{else} \end{cases},$$

only giving a meaningful reward when catching the ball and otherwise just a slight penalty on the actions to avoid unnecessary movements. In the above definition, \mathbf{n}_s is a normal vector of the end effector surface and \mathbf{v}_b is the linear velocity of the ball. This additional term is used to encourage the robot to align its end effector with the curve of the ball. If the end effector is e.g. a net (as assumed for our experiment), the normal is chosen such that aligning it with the ball maximizes the opening through which the ball can enter the net.

The context $c = [\phi, r, d_x] \in [0.125\pi, 0.5\pi] \times [0.6, 1.1] \times [0.75, 4] \subset \mathbb{R}^3$ controls the target

ball position in the catching plane, i.e.

$$\mathbf{p}_{\text{des}} = [0 \quad -r \cos(\phi) \quad 0.75 + r \sin(\phi)].$$

Furthermore, the context determines the distance in x -dimension from which the ball is thrown

$$\mathbf{p}_{\text{init}} = [d_x \ d_y \ d_z],$$

where $d_y \sim \mathcal{U}(-0.75, -0.65)$ and $d_z \sim \mathcal{U}(0.8, 1.8)$ and \mathcal{U} represents the uniform distribution. The initial velocity is then computed using simple projectile motion formulas by requiring the ball to reach \mathbf{p}_{des} at time $t = 0.5 + 0.05d_x$. As we can see, the context implicitly controls the initial state of the environment.

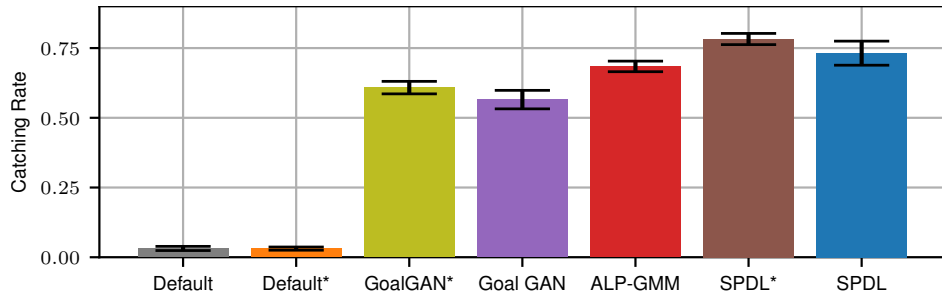
The policy and value function networks for the RL algorithms have three hidden layers with 64 neurons each and tanh activation functions. We use a discount factor of $\gamma = 0.995$. The policy updates in TRPO and PPO are done after 5000 environment steps.

For SAC, a replay buffer size of 100,000 is used. Due to the sparsity of the reward, we increase the batch size to 512. Learning with SAC starts after 1000 environment steps. All other parameters are left to their implementation defaults.

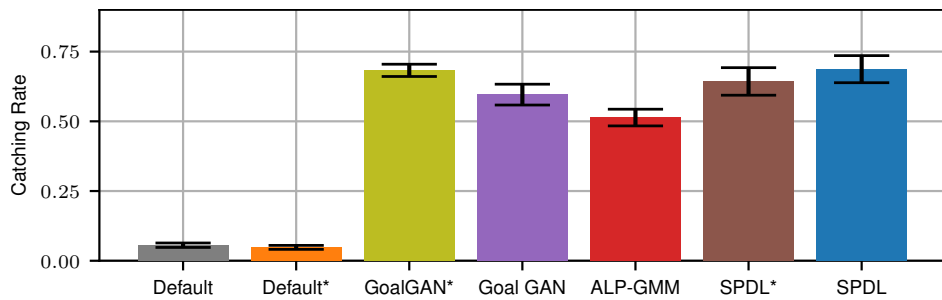
For TRPO we set the GAE parameter $\lambda = 0.95$, leaving all other parameters to their implementation defaults.

For PPO we use a GAE parameter $\lambda = 0.95$, 10 optimization epochs, 25 mini-batches per epoch, an entropy coefficient of 0 and disable the clipping of the value function objective. The remaining parameters are left to their implementation defaults.

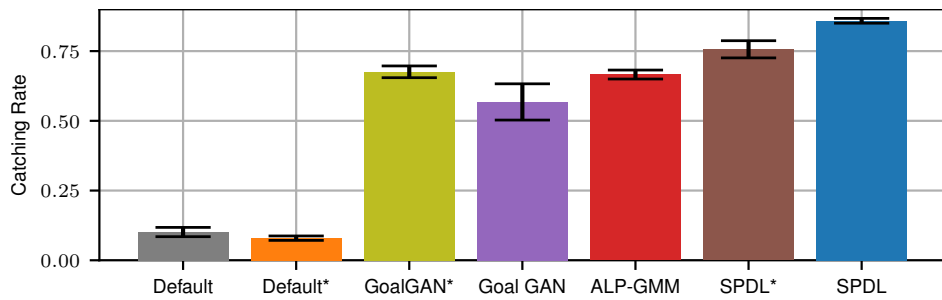
Figure B.4 visualizes the catching success rates of the learned policies. As can be seen, the performance of the policies learned with the different RL algorithms achieve comparable catching performance. Interestingly, SAC performs comparable in terms of catching performance, although the average reward of the final policies learned with SAC is lower. This is to be credited to excessive movement and/or bad alignment of the end effector with the velocity vector of the ball.



(a) SAC



(b) TRPO



(c) PPO

Figure B.4.: Mean catching rate of the final policies learned with different curricula and RL algorithms on the ball catching environment. The mean is computed from 20 algorithm runs with different seeds. For each run, the success rate is computed from 200 ball-throws. The bars visualize the estimated standard error.

C. Appendix to Chapter 3

C.1. Computational Cost of Optimal Transport

The benefits of optimal transport (OT), such as explicitly incorporating a ground distance on the sample space, come at the price of a relatively high computational burden caused by the need to solve an optimization problem to compute the Wasserstein distance between two distributions. In practice, OT problems in continuous spaces (such as some of the context spaces investigated in this chapter) are often reduced to linear assignment problems between sets of particles. Such assignment problems can be exactly solved with variations of the Hungarian algorithm with a time complexity of $\mathcal{O}(n^3)$ [84]. While this polynomial complexity ultimately leads to prohibitive runtimes for large n , we can typically avoid this problem for curriculum RL. Given the often moderate dimensionality of the chosen context spaces, a few hundred particles are typically sufficient to represent the context distributions. In our experiments, we used less than 500 particles in the continuous environments and 640 particles for the discrete unlock-pickup environment, leading to observed solving times of less than 200ms with the `linear_sum_assignment` function of the SciPy library [204] on an AMD Ryzen 9 3900X. Since the `CURROT` and `GRADIENT` algorithms solve, at most, three OT problems per context distribution update, the computational costs of OT are relatively small for the investigated environments.

Furthermore, approximations have emerged to tackle problems that require a large number of particles. For example, the `GeomLoss` library [53], which we use in the `GRADIENT` implementations for continuous Euclidean spaces, implements a variant of entropy-regularized OT that has brought down the computation time of OT for sets of hundreds of thousands of samples to seconds on high-end GPUs [54]. So-called sliced Wasserstein distances [25, 102] approximately solve the given OT problem by solving M OT problems in 1-D subspaces, reducing the time complexity to $\mathcal{O}(Mn \log(n))$, where typically $M \ll n$. Finally, neural function approximators have been employed e.g. to speed up the computation of Wasserstein distances by learning a metric embedding from data [38] or enable to computation of regularized free-support Wasserstein barycenters by approximating the dual potentials [112]. Consequently, we see opportunities to significantly increase the number of particles via such approximate approaches.

C.2. CURROT Search for Feasible Contexts

As detailed in Section 3.5, the initial context distribution $p_0(\mathbf{c})$ may be uninformed and consequently lead to sampling many learning tasks for which the agent performance is below δ . In such scenarios, we initiate a search procedure for tasks in which the current agent achieves a performance at least δ of as long as $\bar{R} = \frac{1}{M} \sum_{m=1}^M R_m < \delta$. We terminate this search procedure as soon as $\bar{R} \geq \delta$. During this search, \mathcal{D}_+ contains the best-encountered samples, and \mathcal{D}_- is empty. When a batch of M new episodes arrives, we add those episodes whose return is at least as large as the median return in \mathcal{D}_+ to the buffer – and for each new episode added, remove the worst performing episode. The search distribution is a (truncated) Gaussian mixture model

$$p_{\text{search}}(\mathbf{c}) = \sum_{i=1}^{N_{\mathcal{D}}} w_i \mathcal{N}(\mathbf{c} | \mathbf{c}_i, \sigma_i^2 \mathbf{I})$$

with weights w_i and variances σ_i^2 defined via the minimum return observed over all episodes R_{\min} and the median performance of the buffered episodes R_{med}

$$w_i \propto \max(0, R_{\mathbf{c}_i} - R_{\text{med}}), \quad \sigma_i = \max\left(10^{-3}, 2 \frac{\delta - R_{\mathbf{c}_i}}{\delta - R_{\min}}\right).$$

For simplicity of exposition, we assume that $\mathcal{C} = [0, 1]^d$, i.e., that the context space is a d -dimensional hyper-cube of edge-length one. Consequently, a context \mathbf{c} with a return of R_{\min} will have a standard deviation of two in each dimension, which, in combination with the Gaussian being truncated, leads to spread-out sampling across the hyper-cube. If the dimensions of \mathcal{C} are scaled differently, a simple re-scaling is sufficient to use the above sampling procedure. Note that we only required the search procedure in the point-mass and teach my agent environments, as in the other environments $p_0(\mathbf{c})$ provided enough successful initial episodes. For discrete context spaces, the search distribution would need to be adapted, e.g., by defining a uniform distribution over all contexts \mathbf{c} with a distance $d(\mathbf{c}, \mathbf{c}_i)$ less than or equal to a threshold that is similarly scaled as the variance σ_i^2 .

C.3. Experimental Details

This section discusses hyperparameters and additional details of the conducted experiments that could not be provided in the main text due to space limitations. For all experiments except the *teach my agent* benchmark, we used RL algorithms from the `Stable Baselines 3` library [73]. For *teach my agent*, we use the SAC implementation provided with the benchmark.

ENV.	SPRL				CURROT		GRADIENT	
	δ	ϵ	σ_{LB}	$D_{\text{KL}_{\text{LB}}}$	δ	ϵ	δ	ϵ
SPARSE GOAL-REACHING	0.6	.25	-	-	0.8	1.2	0.6	0.05
POINT MASS	4	.25	[.2 .1875]	8000	4	0.7	3.0	0.2
UNLOCK-PICKUP	-	-	-	-	0.6	3	0.6	0.05
TEACH MY AGENT	-	-	-	-	180	0.5 0.4	180	0.05

Table C.1.: Hyperparameters of SPRL, CURROT, and GRADIENT in the different learning environments. The ϵ parameter of CURROT is computed according to the procedure described in appendix C.3. We do not provide *teach my agent* parameters for SPRL as we rely on the results reported by [171]. We also do not evaluate SPRL in the unlock-pickup environment since SPRL is designed for continuous context spaces.

C.3.1. Algorithm Hyperparameters

The main parameters of SPRL, CURROT, and GRADIENT all factor into one parameter δ corresponding to the performance constraint and one parameter ϵ controlling the interpolation speed. We did not perform an extensive hyperparameter search for these parameters but used their interpretability to select appropriate parameter regions to search in. The

ENV.	ALP-GMM			GOALGAN		
	p_{RAND}	n_{ROLLOUT}	s_{BUFFER}	δ_{NOISE}	n_{ROLLOUT}	p_{SUCCESS}
SPARSE GOAL-REACHING	.2	200	500	.1	200	.2
POINT MASS	.1	100	500	.1	200	.2
UNLOCK-PICKUP	-	-	-	-	-	-

ENV.	PLR			VDS			ACL	
	ρ	β	p	LR	n_{EP}	n_{BATCH}	η	ϵ
SPARSE GOAL-REACHING	.45	.15	.55	5×10^{-4}	10	80	0.05	0.2
POINT MASS	.15	.45	.85	10^{-3}	3	20	0.025	0.2
UNLOCK-PICKUP	.45	.45	.55	10^{-3}	5	20	0.025	0.1

Table C.2.: Hyperparameters of the investigated baseline algorithms in the different learning environments, as described in Appendix C.3.

performance parameter δ was chosen by evaluating values around 50% of the maximum reward. This approach resulted in a search over $\delta \in \{3, 4, 5\}$ for the point-mass environment and $\delta \in \{0.4, 0.6, 0.8\}$ for the sparse goal-reaching and unlock-pickup environment. For the *teach my agent* experiments, we evaluated $\delta \in \{140, 160, 180\}$ for CURROT and GRADIENT. We did not evaluate SPRL in the *teach my agent* experiment since we took the results from Romac et al. [171]. We evaluated GRADIENT for $\epsilon \in [0.05, 0.1, 0.2]$. For SPRL, we initialized ϵ with a value of 0.05 used in the initial experiments in Chapter 2. However, we realized that larger values slightly improved performance. For CURROT, the value of ϵ depends on the magnitude of the distances d and hence changes per experiment. For CURROT, the parameter ϵ is set to around 5% of the maximum distance between any two points in the context space, also evaluating a slightly larger and smaller value. When targeting narrow target distributions, we introduced a lower bound on the standard deviation σ_{lb} of the context distribution of SPRL in Chapter 2. This lower bound needs to be respected until the KL divergence w.r.t. $\mu(\mathbf{c})$ falls below a threshold D_{KL} , as otherwise, the variance of the context distribution may collapse too early, causing the KL divergence constraint on subsequent distributions to only allow for minimal changes to the context distribution. This detail again highlights the benefit of Wasserstein distances, as they are not subject to such subtleties due to their reliance on a chosen metric. Table C.1 shows the parameters of CURROT, GRADIENT, and SPRL for the different environments.

For ALP-GMM, the relevant hyperparameters are the percentage of random samples drawn from the context space p_{rand} , the number of completed learning episodes between the update of the context distribution $n_{rollout}$, and the maximum buffer size of past trajectories to keep s_{buffer} . Similar as in Chapter 2, we chose them by a grid-search over $(p_{rand}, n_{rollout}, s_{buffer}) \in \{0.1, 0.2, 0.3\} \times \{50, 100, 200\} \times \{500, 1000, 2000\}$.

For GOALGAN, we tuned the amount of random noise that is added on top of each sample δ_{noise} , the number of policy rollouts between the update of the context distribution $n_{rollout}$ as well as the percentage of samples drawn from the success buffer $p_{success}$ via a grid search over $(\delta_{noise}, n_{rollout}, p_{success}) \in \{0.025, 0.05, 0.1\} \times \{50, 100, 200\} \times \{0.1, 0.2, 0.3\}$.

For ACL, the continuous context spaces of the environments need to be discretized, as the algorithm is formulated as a bandit problem. The Exp3.S bandit algorithm that ultimately realizes the curriculum requires two hyperparameters to be chosen: the scale factor for updating the arm probabilities η and the ϵ parameter of the ϵ -greedy exploration strategy. We combine ACL with the absolute learning progress (ALP) metric also used in ALP-GMM and conducted a hyperparameter search over $(\eta, \epsilon) \in \{0.05, 0.1, 0.2\} \times \{0.01, 0.025, 0.05\}$. Hence, contrasting ACL and ALP-GMM sheds light on the importance of exploiting the continuity of the context space. For ACL, the absolute learning progress in a context \mathbf{c} can be estimated by keeping track of the last reward obtained in the bin of \mathbf{c} (note that we discretize the context space) and then computing the absolute difference between the

return obtained from the current policy execution and the stored last reward. We had numerical issues when implementing the ACL algorithm by Graves et al. [64] due to the normalization of the ALPs via quantiles. Consequently, we normalized via the maximum and minimum ALP seen over the entire history of tasks.

For `PLR`, the staleness coefficient ρ , the score temperature β , and the replay probability p need to be chosen. We did a grid search over $(\rho, \beta, p) \in \{0.15, 0.3, 0.45\} \times \{0.15, 0.3, 0.45\} \times \{0.55, 0.7, 0.85\}$ and chose the best configuration for each environment.

For `VDS`, the parameters for the training of the Q -function ensemble, i.e., the learning rate lr , the number of epochs n_{ep} and the number of mini-batches n_{batch} , need to be chosen. Just as for `PLR`, we conducted a grid search over $(lr, n_{ep}, n_{batch}) \in \{10^{-4}, 5 \times 10^{-4}, 10^{-3}\} \times \{3, 5, 10\} \times \{20, 40, 80\}$. The parameters of all employed baselines are given in Table C.2. We now continue with the description of experimental details for each environment.

C.3.2. E-Maze Environment

The xy -coordinates of the representatives

$$\mathbf{r}(c) = [x, y, z]^T \in \mathbb{R}^3$$

of a context c form a grid on $[-1, 1] \times [-1, 1]$ and, as mentioned in the main chapter, $z=200$ for walls and $z=0$ for all other cells. The four actions {up, down, left, right} lead to a transition to the corresponding neighboring cell with a probability of 0.9, if the neighboring cell has the same height, and 0 if not. Upon reaching the desired state (controlled by the context c), the agent observes a reward of value one, and the episode terminates. In this environment, we use `PPO` with $\lambda = 0.99$ and all other parameters left to the implementation defaults of the `Stable Baselines 3` library. For solving objectives (3.5) and (3.6), we make use of the `linprog` function from the `SciPy` library [204].

Current Agent Performance as a Distance: In the main text, we have investigated the pseudo-distance

$$d_{p^*}(c_1, c_2) = |J(\pi^*, c_1) - J(\pi^*, c_2)| \tag{C.1}$$

that defines the similarity of contexts based on the absolute performance difference of the optimal policy in the contexts c_1 and c_2 . While d_{p^*} only performed slightly worse than the more informed distance d_S for `GRADIENT`, it could only provide meaningful performance for `CURROT` if combined with entropy regularization. However, Huang et al. [76] also investigated a pseudo-distance function that computes the similarity of two contexts based on the *current* policy π

$$d_p(c_1, c_2) = |J(\pi, c_1) - J(\pi, c_2)|, \tag{C.2}$$

leading to a distance function that changes in each iteration. As we show in Figure C.1b, this distance, while still leading to slower learning for CURROT compared to d_S , leads to stable learning across different levels of entropy regularization without any prior environment knowledge. Figure C.1a shows multiple curricula that have been generated by CURROT and GRADIENT. Particularly for CURROT, we can see fairly diverse curricula, which sometimes target all three corridors at once (top middle) and sometimes even back track out of the right-most corridor into the remaining two (top right). We see the good performance and the diverse behavior as indicators for the potential of general purpose distance metrics that encode some form of implicit exploration, calling for future investigations to better understand their mechanics. Furthermore, computational aspects arise with the use of such metrics, since for the case of d_p , robust and efficient versions for estimating $J(\pi, c)$ need to be devised.

Entropy-Regularized CURROT and GRADIENT: As discussed in Section 3.6.1, we benchmark versions of GRADIENT and CURROT in which we introduce different forms of entropy regularization. For GRADIENT, we recreate the implementation by Huang et al. [76] by using optimal transport formulations that regularize the entropy of the transport plan ϕ [19, 39]

$$\mathcal{W}_{p,\lambda}(p_1, p_2) = \left(\inf_{\phi \in \Phi(p_1, p_2)} \mathbb{E}_\phi [d(\mathbf{c}_1, \mathbf{c}_2)^p] - \lambda H(\phi) \right)^{1/p}, \quad (\text{C.3})$$

with the constraint set $\Phi(p_1, p_2)$ defined as in Section 1.3 and the entropy $H(p)$ of a distribution p over a sample space \mathcal{X} defined as $H(p) = -\int_{\mathbf{x} \in \mathcal{X}} p(\mathbf{x}) \log(p(\mathbf{x}))$. Note that Huang et al. [76] chose these formulations for computational speed rather than curriculum performance. This formulation allows for a straightforward adaptation of the GRADIENT objective to incorporate entropy-regularization

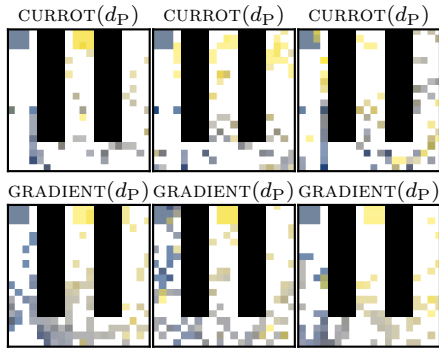
$$\max_{\alpha \in [0,1]} \alpha \quad \text{s.t.} \quad J(\pi, p_{\alpha,\lambda}) \geq \delta \quad (\text{C.4})$$

$$p_{\alpha,\lambda}(c) = \arg \min_p \alpha \mathcal{W}_{2,\lambda}(p, \mu) + (1 - \alpha) \mathcal{W}_{2,\lambda}(p, p_0). \quad (\text{C.5})$$

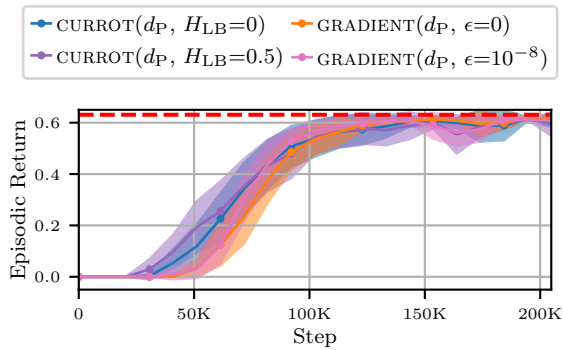
For the CURROT algorithm, we choose a more direct form of regularization and directly constrain the entropy of the interpolating distribution p

$$\begin{aligned} \min_p \quad & \mathcal{W}_2(p, \mu) \\ \text{s.t.} \quad & p(\mathcal{V}(\pi, \delta)) = 1 \quad H(p) \geq H_{\text{LB}}. \end{aligned} \quad (\text{C.6})$$

The above entropy regularized objectives are not linear programs anymore, and we hence solve the (convex) objectives with the CVXPY library [46].



(a) Curricula for Distance d_p



(b) Performance of Curricula using d_p

Figure C.1.: a) Interpolations generated by CURROT and GRADIENT in different runs for the current performance-based distance $d_p(c_1, c_2)$. Brighter colors indicate later iterations. b) Expected return on the target task distribution $\mu(c)$ in the E-Maze environment achieved by CURROT and GRADIENT under varying entropy regularizations for the current performance-based distance d_p . The shaded area corresponds to two times the standard error (computed from 20 seeds). The red dotted line represents the maximum possible reward achievable on $\mu(c)$.

C.3.3. Unlock-Pickup Environment

We use the Unlock-Pickup environment from the Minigrad library [33]. We do not change the behavior of the environment and only remove the additional discounting that occurs within the environment, as the environment does not reveal the current timestep to the agent, which, combined with an internally discounted reward, leads to non-Markovian behavior. As stated in the main chapter, the context c controls the initial state of the environment by specifying the position of the agent, key, and box as well as the position and state of the door (i.e., open or closed). We use the DQN algorithm since the extremely sparse nature of the environment favors RL algorithms with a replay buffer. Compared to the default parameters of the DQN algorithm, we only increase the exploration rate from 0.05 to 0.1 and also increase the batch size to 256. We train the Q -network every fourth step, updating the target network with a Polyak update with $\tau = 0.005$ in each step.

The Q -network is realized by encoding the image observation with a convolutional neural network with three convolutions of kernel size $(2, 2)$, ReLU activations after each convolution, and a max-pool operation with kernel size $(2, 2)$ after the first convolution and ReLU operation. We do not use information about the agent orientation or the textual task description, as both are not strictly necessary for our environment. The convolutional

network has 32-dimensional hidden layers. The output of the convolutional encoder is 64-dimensional, which is then further processed by two fully connected layers with 64 dimensions and ReLU activations before being reduced to the Q -values for the seven actions available in the environment.

As briefly mentioned in the main chapter, the target distribution $\mu(c)$ is a uniform distribution over all those contexts in which the agent is in the left room with a closed door and does not hold the key. The initial state distribution contains one context for each box position in the right room in which the agent is positioned directly next to the box.

Distance Function As discussed in Section 3.6.2, a context c controls the starting state of the environment, which is defined by

- the agent position $\text{ap} : \mathcal{C} \mapsto [1, 9] \times [1, 4]$
- the key position $\text{kp} : \mathcal{C} \mapsto [1, 9] \times [1, 4]$
- the box position $\text{bp} : \mathcal{C} \mapsto [6, 9] \times [1, 4]$
- the position of the door in the wall $\text{dp} : \mathcal{C} \mapsto [1, 4]$
- the state of the door $\text{ds} : \mathcal{C} \mapsto \{\text{open}, \text{closed}\}$.

The images of the individual functions that access the state information of a context are motivated by the two rooms $R_1 = [1, 4] \times [1, 4]$ and $R_2 = [6, 9] \times [1, 4]$ that make up the environment. Consequently, the agent and the key can be placed in both rooms, whereas the box can only be placed in R_2 . The wall that separates the rooms occupies tiles in $W(c) = \{(5, y) \mid y \in [1, 4], y \neq \text{dp}(c)\}$. Due to this wall, we restrict the context space \mathcal{C} such that it does not contain contexts in which the agent or key is located in the wall, i.e., $\text{ap}(c) \notin W(c)$ and $\text{kp}(c) \notin W(c)$. Additionally, we only allow placing the agent and key in R_2 if the door is open. Formally, this requires $\text{ap}(c) \geq 4 \Rightarrow \text{ds}(c) = \text{open}$ and $\text{kp}(c) \geq 4 \Rightarrow \text{ds}(c) = \text{open}$. Finally, neither key nor agent can be at the same position as the box, i.e., $\text{ap}(c) \neq \text{bp}(c)$ and $\text{kp}(c) \neq \text{bp}(c)$. With these restrictions, we arrive at the 81.920 individual contexts mentioned in Section 3.6.2.

Note that the distance function between contexts reasons both about state changes that can be achieved in an episode, such as moving between agent positions, and ones that can't, such as moving the box. Moving boxes is impossible since the episode terminates successfully when the agent picks up the box. Hence, a distance function that is purely based on state transitions would neglect certain similarities between contexts in this environment.

We define the distance function $d_{\text{base}}(c_1, c_2)$ function via representatives $r(c)$, i.e.

$$d(c_1, c_2) = \begin{cases} d_{\text{base}}(c_1, r(c_1)) + d_{\text{base}}(r(c_1), r(c_2)) \\ \quad + d_{\text{base}}(r(c_2), c_2), & \text{if } \text{ds}(c_1) \neq \text{ds}(c_2) \\ d_{\text{base}}(c_1, c_2), & \text{else.} \end{cases} \quad (\text{C.7})$$

Such distances are also known as highway distances [13]. The mapping $r : \mathcal{C} \mapsto \mathcal{C}$ from a context c to its representative $r(c)$ ensures that the agent is standing right in front of the open door with the key in its hand, i.e., $\text{ds}(r(c)) = \text{open}$, and $\text{ap}(r(c)) = \text{kp}(r(c)) = [4, \text{dp}(c)]$, while ensuring that $\text{dp}(r(c)) = \text{dp}(c)$ and $\text{bp}(r(c)) = \text{bp}(c)$.

The base distance $d_{\text{base}}(c_1, c_2)$ encodes the cost of moving both key and agent from their positions in c_1 to those in c_2 (via d_{ka}) as well as the cost of equalizing the box positions between the contexts (via the L1 distance)

$$d_{\text{base}}(c_1, c_2) = \begin{cases} d_{\text{ka}}(c_1, c_2) + \|\text{bp}(c_1) - \text{bp}(c_2)\|_1, \\ \quad \text{if } \text{dp}(c_1) = \text{dp}(c_2) \\ \infty, & \text{else.} \end{cases} \quad (\text{C.8})$$

We see that we render contexts with different door positions incomparable to ease the definition of the distance function. The key-agent distance is defined on top of an object distance $d_{\text{obj,dp}}$ that is conditioned on a door position dp

$$d_{\text{ka}}(c_1, c_2) = \begin{cases} d_{\text{obj,dp}(c_1)}(\text{ap}(c_1), \text{ap}(c_2)), & \text{if } \text{kp}(c_1) = \text{kp}(c_2) \\ d_{\text{obj,dp}(c_1)}(\text{ap}(c_1), \text{kp}(c_1)) \\ \quad + d_{\text{obj,dp}(c_1)}(\text{kp}(c_1), \text{kp}(c_2)) \\ \quad + d_{\text{obj,dp}(c_1)}(\text{ap}(c_2), \text{kp}(c_2)), & \text{else.} \end{cases} \quad (\text{C.9})$$

Note that we can simply take $\text{dp}(c_1)$ since we know that $\text{dp}(c_1) = \text{dp}(c_2)$. The object distance is defined as the L1 distance between the two objects if they are in the same room and incorporates the detour caused by passing through the door in the wall if not

$$d_{\text{obj,dp}}(\mathbf{p}_1, \mathbf{p}_2) = \begin{cases} \|\mathbf{p}_1 - \mathbf{p}_2\|_1, & \text{if } p_{1,0} \leq 4 \Leftrightarrow p_{2,0} \leq 4 \\ \|\mathbf{p}_1 - [5, \text{dp}]\|_1 + \|[5, \text{dp}] - \mathbf{p}_2\|_1, & \text{else.} \end{cases} \quad (\text{C.10})$$

We ensured that the resulting distance $d(c_1, c_2)$ fulfills all axioms of a valid distance function, i.e. $d(c_1, c_2) \geq 0$, $d(c_1, c_2) = 0 \Leftrightarrow c_1 = c_2$, $d(c_1, c_2) = d(c_2, c_1)$, and $d(c_1, c_3) \leq d(c_1, c_2) + d(c_2, c_3)$ via brute-force computations. Note that the in-comparability of contexts with different door positions effectively splits the context space into four disjoint sets (for the four different door positions) that cannot be compared. Hence, we must only ensure these axioms within the four disjoint sets separately.

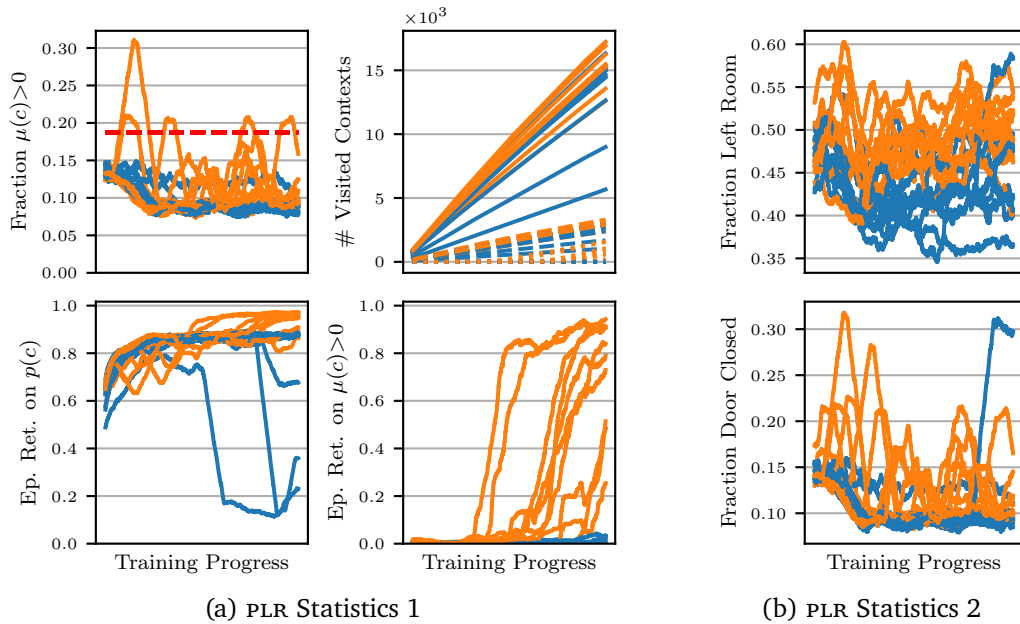


Figure C.2.: (a) Statistics of the PLR curricula in the unlock-pickup environment over training progress. Top left: Fraction of contexts sampled by PLR that are also sampled by the target context distribution $\mu(c)$. The red dashed line indicates the fraction of target samples generated by a random curriculum. Top right: Number of unique contexts (solid lines), unique target contexts (dashed lines), and unique solved target contexts (dotted lines) sampled by PLR at least once. Bottom left: Performance on the PLR curriculum. Bottom right: Performance in those contexts of the curriculum, which are also sampled by the target context distribution $\mu(c)$ (i.e., on the fraction indicated in the top left). (b) Fraction of contexts in the PLR curricula in which the agent is placed in the left room (left) and in which the door is closed (right) at the start of the episode. A closed door implies that the agent is located in the left room, hence a more strict condition. The color again indicates runs with high- (orange) and low performance (blue) on $p(c)$.

PLR Performance: As mentioned in Section 3.6.2, Figure C.2a shows statistics of the PLR curricula. We can see that throughout most PLR curricula, the chance of sampling a target context stays relatively constant, even though the number of distinct sampled contexts and the number of distinct sampled target contexts continuously grows. We also see that the agent receives a positive learning signal on $p(c)$ in all runs of PLR. Additionally, we see that the prioritization by PLR suppresses contexts from $\mu(c)$ since a purely random

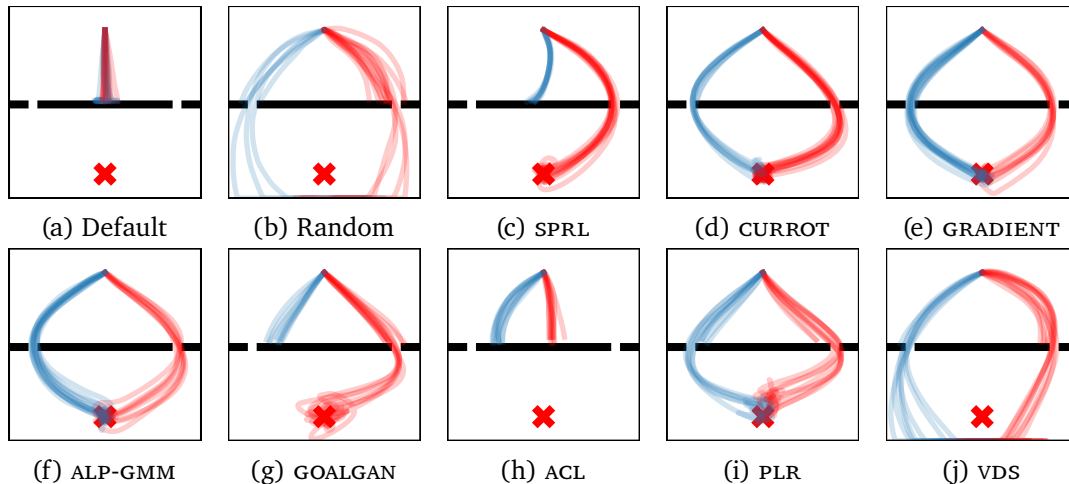


Figure C.3.: Final trajectories generated by the different investigated curricula in the point mass environment. The color encodes the context: Blue represents gates positioned at the left and red at the right.

curriculum would sample a target context 18.75% of the time. In about half of the runs (orange lines), the agent learned to solve some of the target tasks, although this fraction is rather low (there are 15.360 target tasks). Interestingly, this increase in proficiency on tasks from $\mu(c)$ does not go hand-in-hand with a consistently increased sampling rate of target tasks. However, as we see in Figure C.2b there seems to be a tendency of `PLR` runs that are more successful on $\mu(c)$ to sample more contexts in which the agent is located in the left room at the beginning of the episode. Generally speaking, Figures C.2a and C.2b show that `PLR` prioritized specific contexts over others. However, either due to the missing notion of a target distribution or the dependence of `PLR` on the agent’s internal value function (which may be biased and incorrect), the generated curricula did not consistently progress to the most challenging, long-sequence tasks encoded by $\mu(c)$.

C.3.4. Point-Mass Environment

The environment setup is the same as the one investigated in Chapter 2 with the only difference in the target context distributions, which is now defined as a Gaussian mixture

$$\mu(\mathbf{c}) = \frac{1}{2}\mathcal{N}(\mathbf{c}_1, 10^{-4}\mathbf{I}) + \frac{1}{2}\mathcal{N}(\mathbf{c}_2, 10^{-4}\mathbf{I})$$

$$\mathbf{c}_1 = [-3 \ 0.5]^T, \mathbf{c}_2 = [3 \ 0.5]^T.$$

In this environment, we use PPO with 4.096 steps per policy update, a batch size of 128, and $\lambda=0.99$. All other parameters are left to the implementation defaults of the `Stable Baselines 3` implementation.

Figure C.3 shows trajectories generated by agents trained with different curricula in the point-mass environment. We see that directly learning on the two target tasks (Default) prevents the agent from finding the gates in the wall to pass through. Consequently, the agent minimizes the distance to the goal by moving right in front of the wall (but not crashing into it) to accumulate reward over time. We see that random learning indeed generates meaningful behavior. This behavior is, however, not precise enough to pass reliably through the wall. As mentioned in the main chapter, SPRL only learns to pass through one of the gates, as its uni-modal Gaussian distribution can only encode one of the modes of $\mu(c)$ (see Figure C.4 for a visualization). CURROT and GRADIENT learn policies that can pass through both gates reliably, showing that the gradual interpolation towards both target tasks allowed the agent to learn both. ALP-GMM and PLR also learn good policies. The generated trajectories are, however, not as precise as the ones learned with CURROT and GRADIENT and sometimes only solve one of the two tasks reliably. ACL, GOALGAN, and VDS partly create meaningful behavior. However, this behavior is unreliable,

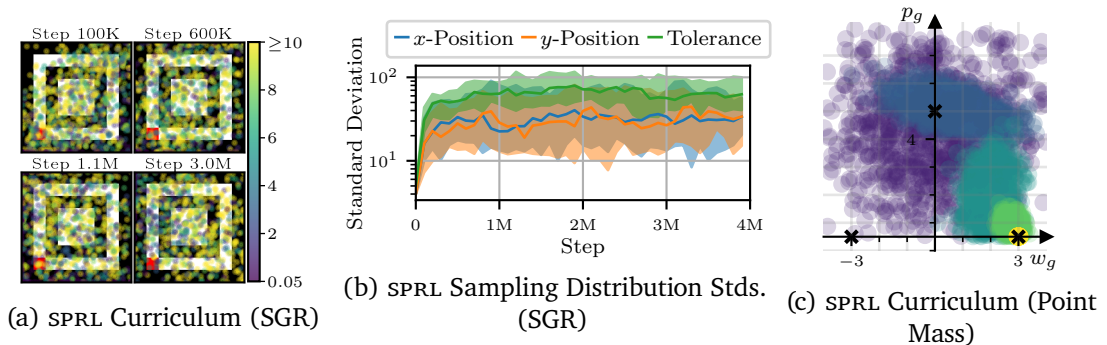


Figure C.4.: a) Visualization of the sampling distribution of SPRL in the sparse goal-reaching (SGR) task. The color of the dots encodes the tolerance of the corresponding contexts, and the position represents the goal to be reached under that tolerance. The walls are shown in black, and the red area visualizes the starting area of the agent. b) 10-, 50- and 90-percentile of the standard deviation of SPRL’s sampling distribution on the sparse goal-reaching task. The statistics have been computed from 20 seeds. c) Sampling distribution of SPRL in the point mass environment for a given seed. The color indicates the iteration, where brighter colors correspond to later iterations.

leading to low returns due to the agent frequently crashing into the wall.

C.3.5. Sparse Goal-Reaching Environment

For the sparse goal-reaching task, the goal can be chosen within $[-9, 9] \times [-9, 9]$, and the allowed tolerance can be chosen from $[0.05, 18]$. Hence, the context space is a three-dimensional cube $\mathcal{C} = [-9, 9] \times [-9, 9] \times [0.05, 18]$. The actually reachable space of positions (and with that goals) is a subset of $[-7, 7] \times [-7, 7]$ due to the “hole” caused by the inner walls of the environment. The target context distribution is a uniform distribution over tasks with a tolerance of 0.05

$$\mu(\mathbf{c}) \propto \begin{cases} 1, & \text{if } c_3 = 0.05, \\ 0, & \text{else.} \end{cases}$$

The state \mathbf{s} of the environment is given by the agent’s x - and y -position. The reward is sparse, only rewarding the agent if the goal is reached. A goal is considered reached if the Euclidean distance between the goal and position of the point mass falls below the tolerance

$$\|\mathbf{s} - [\mathbf{c}_1 \ \mathbf{c}_2]^T\|_2 \leq c_3.$$

The two-dimensional action of the agent corresponds to its displacement in the x - and y - direction. The action is clipped such that the Euclidean displacement per step is no larger than 0.3.

Given the sparse reward of the task, we again use an RL algorithm that utilizes a replay buffer. Since the actions are continuous in this environment, we use SAC instead of DQN. Compared to the default algorithm parameters of `Stable Baselines 3`, we only changed the policy update frequency to 5 environment steps, increased the batch size to 512, and reduced the buffer size to 200,000 steps.

Figure C.4 visualizes the behavior of SPRL in the sparse goal-reaching (SGR). We see that for the SGR environment, SPRL increases the variance of the Gaussian context distribution to assign probability density to the target contexts while fulfilling the expected performance constraint by encoding trivial tasks with high tolerance (Figures C.4a and C.4b). The inferior performance of an agent trained with SPRL compared to one trained with a random curriculum shows that the Gaussian approximation to a uniform distribution is a poor choice for this environment. While it may be possible to find other parametric distributions that are better suited to the particular problem, CURROT flexibly adapts the shape of the distribution without requiring any prior choices.

C.3.6. Teach My Agent

As mentioned in the main text, we used the environment and SAC learning agent implementation provided by Romac et al. [171]. We only interfaced CURROT and GRADIENT to the setup they provided, allowing us to reuse the baseline evaluations provided by Romac et al. [171]. The two settings (*mostly infeasible* and *mostly trivial*) differ in the boundaries of their respective context spaces. The *mostly infeasible* setting encodes tasks with a stump height in $[0, 9]$ and -spacing in $[0, 6]$. The *mostly trivial* setting keeps the same boundaries for the stump spacing while encoding stumps with a height in $[-3, 3]$. Since a stump with negative height is considered not present, half of the context space of the *mostly trivial* setting does not encode any obstacles for the bipedal walker to master. The initial- and target context distribution $\mu(c)$ is uniform over the respective context space \mathcal{C} for both settings.

D. Appendix to Chapter 4

D.1. High-Dimensional Ablations

To investigate the robustness of the different CURROT versions w.r.t. changes in context space dimensions, we increased the number of sections to represent the jerk trajectory $\mathbf{u}(t)$. We tested three numbers, resulting in 99, 198, and 399 context space dimensions. When increasing the dimension, we observed that the condition number of the whitening matrix for CURROT_{AO} and CURROT_A increased significantly, leading to high-jerk interpolations, which were smooth in position and velocity but exhibited strong oscillations in acceleration. We counteracted this behavior by not only measuring the LTI system state via the matrix \mathbf{A} (Section 4.4.1) but also adding the transform Γ_3 as additional rows to the entries $\Psi_3(t)$ in the definition of \mathbf{A} , where Γ maps the elements of $\ker(\Psi(t_e))$ to piece-wise constant jerk trajectories and Γ_3 is its block-diagonal version as defined in the main chapter. The resulting explicit regularization of the generated jerks prevented the previously observed high jerk interpolations.

Figure D.1 shows the results of the experiments with increasing context space dimensions. The required number of epochs to fully track the target trajectories and the final tracking performance stays almost constant for all methods when using the default trajectory representation, as it allows for good generalization of learned behavior. However, the Wasserstein distances between the final context distribution of the curriculum $p_f(\mathbf{c})$ and $\mu(\mathbf{c})$ increases with the context space dimension for CURROT_A and CURROT.

When using the alternative trajectory representation from Section 4.5.3 (indicated by (JC) in Figure D.1), this increasingly poor convergence to $\mu(\mathbf{c})$ leads to a noticeable performance decrease in the required epochs to completion and final tracking performance for CURROT_A. The performance of CURROT_{AO} decreases only slightly, as the convergence of $p_f(\mathbf{c})$ to $\mu(\mathbf{c})$ seems unaffected by higher-dimensional context spaces. As discussed in the main chapter, CURROT does not allow for good learning with the alternative trajectory representation, rendering the observed tracking performance rather uninformative as they are computed on partially tracked trajectories. The presented results additionally highlight the importance of the improved optimization scheme implemented in CURROT_{AO}, which was not obvious from the experiments in the main chapter.

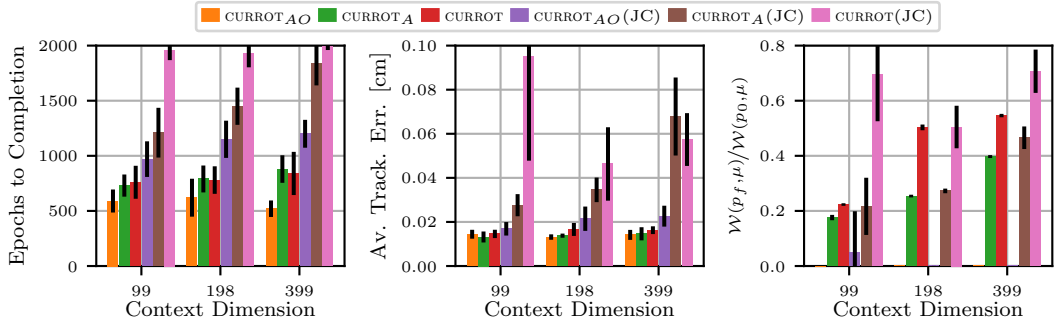


Figure D.1.: Quantitative results for different CURROT versions under increasing dimensions. We show the mean *Epochs To Completion* (left), *Final Average Tracking Error* (middle), and *Final (Normalized) Wasserstein Distance* (right). The error bars indicate the standard error. Statistics are computed from 10 seeds. The abbreviation (JC) stands for experiments in which the agent is given the alternative trajectory representation investigated in Section 4.5.3.

D.2. Modeling Network Communication Delays

Since our measurements indicated a non-negligible chance of delayed network packets, we modeled this effect during training. With the simulation advancing in discrete timesteps, we modeled the network delays in multiples of simulation steps. More formally, the observation of the pendulum $\mathbf{x}_{p,t}$ at time t only becomes available to the agent at time $t + \delta_t$, where $\delta_t \in [0, 1, 2, 3, 4]$. Furthermore, even if $t + i + \delta_{t+i} < t + \delta_t$ for some $i > 0$, the observation at $t + i$ cannot become available before time $t + \delta_t$. We realized this behavior by a FIFO queue, where we sample δ_t upon entry of an observation $\mathbf{x}_{p,t}$.

We also observed packet losses over the network. Since those losses seemed to correlate with packet delays, we, in each timestep, drop the first packet in the queue with a chance of 25%. Hence, the longer the queue is non-empty, i.e., packets are subject to delays, the higher the chance of packets being lost. The probabilities for the delays are given by

$$p(\delta_t) = [0.905 \quad 0.035 \quad 0.02 \quad 0.02 \quad 0.02]_{\delta_t}. \quad (\text{D.1})$$

D.3. Analytic Solution to the LTI System Equations

Given that Constraints (4.7) and (4.8) on the LTI system specify a convex set, which can be relatively easily dealt with, we turn towards Constraint (4.9), for which we need to derive the closed-form solution of the LTI system (4.10)

$$\mathbf{x}(t) = \mathbf{\Phi}(t_s, t)\mathbf{x}(t_s) + \int_{t_s}^t \mathbf{\Phi}(\tau, t)\mathbf{B}u(\tau)d\tau. \quad (\text{D.2})$$

The transition matrix $\mathbf{\Phi}(t_s, t)$ is given by

$$\begin{aligned} \mathbf{\Phi}(t_s, t) &= e^{\mathbf{A}\Delta_s} = \mathbf{I} + \mathbf{A}\Delta_s + \frac{\mathbf{A}^2\Delta_s^2}{2} + \dots + \frac{\mathbf{A}^k\Delta_s^k}{k!} + \dots \\ &= \mathbf{I} + \mathbf{A}\Delta_s + \frac{\Delta_s^2}{2} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & \Delta_s & \frac{\Delta_s^2}{2} \\ 0 & 1 & \Delta_s \\ 0 & 0 & 1 \end{bmatrix}, \end{aligned} \quad (\text{D.3})$$

where $\Delta_s = t - t_s$. We can now turn towards the second term in Equation (D.2). For solving the corresponding integral, we exploit the assumption that the control signal $u(t)$ is piece-wise constant on the intervals $[t_k, t_{k+1})$ with $t_s=t_0 < t_1 < \dots < t_{K-1} < t_K=t_e$. With that, the second term reduces to

$$\int_{t_0}^t \mathbf{\Phi}(\tau, t)\mathbf{B}u(\tau)d\tau = \sum_{k=1}^K u_k \int_{t_{k-1}}^{\min(t_k, t)} \mathbf{\Phi}(\tau, t)\mathbf{B}d\tau. \quad (\text{D.4})$$

We are hence left to solve

$$\begin{aligned} \int_{t_l}^{t_h} \mathbf{\Phi}(\tau, t)\mathbf{B}d\tau &= \int_{t_l}^{t_h} \begin{bmatrix} \frac{(t-\tau)^2}{2} \\ t-\tau \\ 1 \end{bmatrix} d\tau = \begin{bmatrix} \frac{t^2\tau}{2} - \frac{t\tau^2}{2} + \frac{\tau^3}{6} \\ t\tau - \frac{\tau^2}{2} \\ \tau \end{bmatrix} \Big|_{\tau=t_l}^{t_h} \\ &= \begin{bmatrix} \frac{t^2t_h}{2} - \frac{tt_h^2}{2} + \frac{t_h^3}{6} \\ tt_h - \frac{t_h^2}{2} \\ t_h \end{bmatrix} - \begin{bmatrix} \frac{t^2t_l}{2} - \frac{tt_l^2}{2} + \frac{t_l^3}{6} \\ tt_l - \frac{t_l^2}{2} \\ t_l \end{bmatrix} = \begin{bmatrix} \frac{t^2\Delta_{lh}}{2} - \frac{t\tilde{\Delta}_{lh}^2}{2} + \frac{\tilde{\Delta}_{lh}^3}{6} \\ t\Delta_{lh} - \frac{\tilde{\Delta}_{lh}^2}{2} \\ \Delta_{lh} \end{bmatrix} = \boldsymbol{\psi}(t_l, t_h, t), \end{aligned} \quad (\text{D.5})$$

where $\Delta_{lh} = t_h - t_l$ and $\tilde{\Delta}_{lh}^i = (t_h - t_l)^i$. Note that we assume $t_l \leq t_h \leq t$ and otherwise define $\psi(t_l, t_h, t)$ to be zero. With Φ and ψ available, we can rewrite the state \mathbf{x} at t_e as

$$\mathbf{x}(t_e) = \Phi(t_s, t_e)\mathbf{x}(t_s) + \underbrace{\begin{bmatrix} \psi(t_s, t_1, t_e) & \psi(t_1, t_2, t_e) & \dots & \psi(t_{K-1}, t_e, t_e) \end{bmatrix}}_{\Psi(t_e) \in \mathbb{R}^{3 \times K}} \underbrace{\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_K \end{bmatrix}}_{\mathbf{u} \in \mathbb{R}^K} \quad (\text{D.6})$$

and consequently arrive at the following rewritten version of Constraint (4.9)

$$\begin{aligned} \mathbf{x}(t_s) &= \mathbf{x}(t_e) \\ \Leftrightarrow \mathbf{x}(t_s) &= \Phi(t_s, t_e)\mathbf{x}(t_s) + \Psi(t_e)\mathbf{u} \\ \Leftrightarrow (\mathbf{I} - \Phi(t_s, t_e))\mathbf{x}(t_s) &= \Psi(t_e)\mathbf{u}. \end{aligned} \quad (\text{D.7})$$

With $\mathbf{x}(t_s) = [\gamma(t_s) \ 0 \ 0]$, and the particular form of $\Phi(t_s, t_e)$ in Eq. (D.3), we can see that $\Phi(t_s, t_e)\mathbf{x}(t_s) = \mathbf{x}(t_s)$ and hence $(\mathbf{I} - \Phi(t_s, t_e))\mathbf{x}(t_s) = \mathbf{0}$. Consequently, we know that the set of admissible controls \mathbf{u} is given by the kernel $\ker(\Psi(t_e))$.

Bibliography

- [1] Abbas Abdolmaleki et al. “Maximum a posteriori policy optimisation”. In: *International Conference on Learning Representations (ICLR)*. 2018.
- [2] Hany Abdulsamad et al. “A Variational Infinite Mixture for Probabilistic Inverse Dynamics Learning”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2021.
- [3] Hany Abdulsamad et al. “Variational Hierarchical Mixtures for Probabilistic Learning of Inverse Dynamics”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2023, In Press).
- [4] David Abel et al. “On the expressivity of markov reward”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2021.
- [5] Bernardetta Addis, Marco Locatelli, and Fabio Schoen. “Local optima smoothing for global optimization”. In: *Optimization Methods and Software* 20.4-5 (2005), pp. 417–437.
- [6] Alekh Agarwal et al. “Reinforcement learning: Theory and algorithms”. In: *CS Dept., UW Seattle, Seattle, WA, USA, Tech. Rep 32* (2019). URL: https://rltheorybook.github.io/rl_monograph_AJK.pdf.
- [7] Martial Agueh and Guillaume Carlier. “Barycenters in the Wasserstein space”. In: *SIAM Journal on Mathematical Analysis* 43.2 (2011), pp. 904–924.
- [8] Ilge Akkaya et al. “Solving rubik’s cube with a robot hand”. In: *arXiv preprint arXiv:1910.07113* (2019).
- [9] Eugene L Allgower and Kurt Georg. *Introduction to numerical continuation methods*. SIAM, 2003.
- [10] Marcin Andrychowicz et al. “Hindsight experience replay”. In: *Advances in Neural Information Processing Systems (NIPS)*. 2017.
- [11] Minoru Asada et al. “Purposive Behavior Acquisition for a Real Robot by Vision-based Reinforcement Learning”. In: *Machine Learning* 23.2 (1996), pp. 279–303.

-
-
- [12] Abolfazl Asudeh et al. “On Obtaining Stable Rankings”. In: *Proceedings of the VLDB Endowment (PVLDB)* 12.3 (2018), pp. 237–250.
- [13] Eftychia Baikousi, Georgios Rogkakos, and Panos Vassiliadis. “Similarity measures for multidimensional data”. In: *International Conference on Data Engineering (ICDE)*. 2011.
- [14] Adrien Baranes and Pierre-Yves Oudeyer. “Active learning of inverse models with intrinsically motivated goal exploration in robots”. In: *Robotics and Autonomous Systems* 61.1 (2013), pp. 49–73.
- [15] Adrien Baranes and Pierre-Yves Oudeyer. “Intrinsically motivated goal exploration for active motor learning in robots: A case study”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2010.
- [16] Marc Bellemare et al. “Unifying count-based exploration and intrinsic motivation”. In: *Advances in Neural Information Processing Systems (NIPS)*. 2016.
- [17] Richard Bellman. “A Markovian decision process”. In: *Journal of mathematics and mechanics* 6.5 (1957), pp. 679–684.
- [18] Boris Belousov et al. *Reinforcement Learning Algorithms: Analysis and Applications*. Springer International Publishing, 2021.
- [19] Jean-David Benamou et al. “Iterative Bregman projections for regularized transportation problems”. In: *SIAM Journal on Scientific Computing* 37.2 (2015), A1111–A1138.
- [20] Yoshua Bengio et al. “Curriculum learning”. In: *International Conference on Machine Learning (ICML)*. 2009.
- [21] Felix Berkenkamp, Angela P Schoellig, and Andreas Krause. “Safe controller optimization for quadrotors with Gaussian processes”. In: *International Conference on Robotics and Automation (ICRA)*. 2016.
- [22] Dimitri P Bertsekas. “Auction Algorithms.” In: *Encyclopedia of optimization* 1 (2009), pp. 73–77.
- [23] Douglas Blank et al. “Bringing up robot: Fundamental mechanisms for creating a self-motivated, self-organizing architecture”. In: *Cybernetics and Systems* 36.2 (2005), pp. 125–150.
- [24] Josh Bongard and Hod Lipson. “Once more unto the breach: Co-evolving a robot and its simulator”. In: *Conference on Artificial Life (ALIFE)*. 2004.
- [25] Nicolas Bonneel et al. “Sliced and radon wasserstein barycenters of measures”. In: *Journal of Mathematical Imaging and Vision* 51.1 (2015), pp. 22–45.

-
-
- [26] Serena Booth et al. “The perils of trial-and-error reward design: misdesign through overfitting and invalid task specifications”. In: *AAAI Conference on Artificial Intelligence (AAAI)*. 2023.
- [27] Greg Brockman et al. “OpenAI gym”. In: *arXiv preprint arXiv:1606.01540* (2016).
- [28] Benjamin Charlier et al. “Kernel Operations on the GPU, with Autodiff, without Memory Overflows”. In: *Journal of Machine Learning Research (JMLR)* 22.74 (2021), pp. 1–6.
- [29] Jiayu Chen et al. “Variational Automatic Curriculum Learning for Sparse-Reward Cooperative Multi-Agent Problems”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2021.
- [30] Liqun Chen et al. “Sequence generation with optimal-transport-enhanced reinforcement learning”. In: *AAAI Conference on Artificial Intelligence (AAAI)*. 2020.
- [31] Shuxiao Chen et al. “Learning torque control for quadrupedal locomotion”. In: *arXiv preprint arXiv:2203.05194* (2022).
- [32] Yongxin Chen, Tryphon T Georgiou, and Michele Pavon. “Stochastic Control Liaisons: Richard Sinkhorn Meets Gaspard Monge on a Schrödinger Bridge”. In: *SIAM Review (SIREV)* 63.2 (2021), pp. 249–313.
- [33] Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. *Minimalistic Gridworld Environment for Gymnasium*. 2018. URL: <https://github.com/Farama-Foundation/Minigrid>.
- [34] Daesol Cho, Seungjae Lee, and H Jin Kim. “Outcome-directed Reinforcement Learning by Uncertainty & Temporal Distance-Aware Curriculum Goal Generation”. In: *International Conference on Learning Representations (ICLR)*. 2023.
- [35] Shui Nee Chow, John Mallet-Paret, and James A Yorke. “Finding zeroes of maps: homotopy methods that are constructive with probability one”. In: *Mathematics of Computation* 32.143 (1978), pp. 887–899.
- [36] Andrew R Conn, Katya Scheinberg, and Luis N Vicente. *Introduction to Derivative-Free Optimization*. SIAM, 2009.
- [37] Luca Consolini and Mario Tosques. “On the exact tracking of the spherical inverted pendulum via an homotopy method”. In: *Systems & Control Letters* 58.1 (2009), pp. 1–6.
- [38] Nicolas Courty, Rémi Flamary, and Mélanie Ducoffe. “Learning Wasserstein Embeddings”. In: *International Conference on Learning Representations (ICLR)*. 2018.

-
-
- [39] Marco Cuturi. “Sinkhorn distances: Lightspeed computation of optimal transport”. In: *Advances in Neural Information Processing Systems (NIPS)*. 2013.
- [40] Tuan Dam et al. “Generalized Mean Estimation in Monte-Carlo Tree Search”. In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 2020.
- [41] Peter Dayan and Geoffrey E Hinton. “Using expectation-maximization for reinforcement learning”. In: *Neural Computation* 9.2 (1997), pp. 271–278.
- [42] Jonas Degraeve et al. “Magnetic control of tokamak plasmas through deep reinforcement learning”. In: *Nature* 602.7897 (2022), pp. 414–419.
- [43] Marc Peter Deisenroth, Gerhard Neumann, and Jan Peters. “A survey on policy search for robotics”. In: *Foundations and Trends® in Robotics* 2.1–2 (2013), pp. 1–142.
- [44] Pinar Demetci et al. “Gromov-Wasserstein optimal transport to align single-cell multi-omics data”. In: *ICML Workshop on Computational Biology*. 2020.
- [45] Michael Dennis et al. “Emergent complexity and zero-shot transfer via unsupervised environment design”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.
- [46] Steven Diamond and Stephen Boyd. “CVXPY: A Python-embedded modeling language for convex optimization”. In: *Journal of Machine Learning Research* 17.83 (2016), pp. 1–5.
- [47] Andreas Doerr et al. “Model-based policy search for automatic tuning of multivariate PID controllers”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2017.
- [48] Ishan Durugkar et al. “Adversarial intrinsic motivation for reinforcement learning”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2021.
- [49] Theresa Eimer et al. “Self-Paced Context Evaluation for Contextual Reinforcement Learning”. In: *International Conference on Machine Learning (ICML)*. 2021.
- [50] Tom Erez and William D Smart. “What does shaping mean for computational reinforcement learning?” In: *International Conference on Development and Learning (ICDL)*. 2008.
- [51] Hehe Fan et al. “Unsupervised person re-identification: Clustering and fine-tuning”. In: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 14.4 (2018), pp. 1–18.

-
-
- [52] Matthew Fellows et al. “Virel: A variational inference framework for reinforcement learning”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.
- [53] Jean Feydy and Pierre Roussillon. *GeomLoss*. 2019. URL: <https://www.kernel-operations.io/geomloss/index.html> (visited on 06/04/2022).
- [54] Jean Feydy et al. “Interpolating between Optimal Transport and MMD using Sinkhorn Divergences”. In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2019.
- [55] Arnaud Fickinger et al. “Cross-Domain Imitation Learning via Optimal Transport”. In: *International Conference on Learning Representations (ICLR)*. 2022.
- [56] Carlos Florensa et al. “Automatic goal generation for reinforcement learning agents”. In: *International Conference on Machine Learning (ICML)*. 2018.
- [57] Carlos Florensa et al. “Reverse Curriculum Generation for Reinforcement Learning”. In: *Conference on Robot Learning (CoRL)*. 2017.
- [58] Pierre Fournier et al. “Accuracy-based Curriculum Learning in Deep Reinforcement Learning”. In: *arXiv preprint arXiv:1806.09614* (2018).
- [59] Yuzhen Ge et al. “Probability-one homotopy algorithms for full-and reduced-order H_2/H_∞ controller synthesis”. In: *Optimal Control Applications and Methods* 17.3 (1996), pp. 187–208.
- [60] Andrew Gelman and Xiao-Li Meng. “Simulating normalizing constants: From importance sampling to bridge sampling to path sampling”. In: *Statistical Science* 13.2 (1998), pp. 163–185.
- [61] Mohammad Ghavamzadeh et al. “Bayesian Reinforcement Learning: A Survey”. In: *Foundations and Trends® in Machine Learning* 8.5-6 (2015), pp. 359–483.
- [62] Yong Liang Goh et al. “Combining reinforcement learning and optimal transport for the traveling salesman problem”. In: *1st International Workshop on Optimal Transport and Structured Data Modeling*. 2022.
- [63] Matthew M. Graham and Amos J. Storkey. “Continuously tempered hamiltonian monte carlo”. In: *Conference on Uncertainty in Artificial Intelligence (UAI)*. 2017.
- [64] Alex Graves et al. “Automated curriculum learning for neural networks”. In: *International Conference on Machine Learning (ICML)*. 2017.
- [65] Shixiang Gu et al. “Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2017.

-
-
- [66] Tuomas Haarnoja et al. “Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor”. In: *International Conference on Machine Learning (ICML)*. 2018.
- [67] Assaf Hallak, Dotan Di Castro, and Shie Mannor. “Contextual Markov decision processes”. In: *arXiv preprint arXiv:1502.02259* (2015).
- [68] Nikolaus Hansen, Youhei Akimoto, and Petr Baudis. *CMA-ES/pycma on Github*. Feb. 2019. URL: <https://github.com/CMA-ES/pycma>.
- [69] Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. “Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)”. In: *Evolutionary Computation* 11.1 (2003), pp. 1–18.
- [70] Nikolaus Hansen et al. “Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009”. In: *Genetic and Evolutionary Computation Conference (GECCO)*. 2010.
- [71] Charles R. Harris et al. “Array programming with NumPy”. In: *Nature* 585.7825 (Sept. 2020), pp. 357–362.
- [72] Philipp Hennig, Michael A Osborne, and Mark Girolami. “Probabilistic numerics and uncertainty in computations”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 471.2179 (2015), p. 20150142.
- [73] Ashley Hill et al. *Stable Baselines*. <https://github.com/hill-a/stable-baselines>. 2018.
- [74] Rein Houthoofd et al. “Vime: Variational information maximizing exploration”. In: *Advances in Neural Information Processing Systems (NIPS)*. 2016.
- [75] Ronald A Howard. *Dynamic programming and markov processes*. John Wiley, 1960.
- [76] Peide Huang et al. “Curriculum Reinforcement Learning using Optimal Transport via Gradual Domain Adaptation”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2022.
- [77] Max Jaderberg et al. “Reinforcement learning with unsupervised auxiliary tasks”. In: *International Conference on Learning Representations (ICLR)*. 2017.
- [78] F Jalali-Farahani and JD Seader. “Use of homotopy-continuation method in stability analysis of multiphase, reacting systems”. In: *Computers & Chemical Engineering* 24.8 (2000), pp. 1997–2008.
- [79] Lu Jiang et al. “Easy samples first: Self-paced reranking for zero-example multimedia search”. In: *ACM International Conference on Multimedia (MM)*. 2014.

-
-
- [80] Lu Jiang et al. “Self-paced curriculum learning”. In: *AAAI Conference on Artificial Intelligence (AAAI)*. 2015.
- [81] Lu Jiang et al. “Self-paced learning with diversity”. In: *Advances in Neural Information Processing Systems (NIPS)*. 2014.
- [82] Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. “Prioritized Level Replay”. In: *International Conference on Machine Learning (ICML)*. 2021.
- [83] Minqi Jiang et al. “Replay-Guided Adversarial Environment Design”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2021.
- [84] Roy Jonker and Anton Volgenant. “A shortest augmenting path algorithm for dense and sparse linear assignment problems”. In: *Computing* 38.4 (1987), pp. 325–340.
- [85] Kirthevasan Kandasamy et al. “Neural architecture search with bayesian optimisation and optimal transport”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.
- [86] Leonid Kantorovich. “On the transfer of masses (in Russian)”. In: *Doklady Akademii Nauk* 37.2 (1942), pp. 227–229.
- [87] Sho-Tsung Kao, Wan-Jung Chiou, and Ming-Tzu Ho. “Balancing of a spherical inverted pendulum with an omni-directional mobile robot”. In: *International Conference on Control Applications (CCA)*. 2013.
- [88] Sho-Tsung Kao and Ming-Tzu Ho. “Tracking control of a spherical inverted pendulum with an omnidirectional mobile robot”. In: *International Conference on Advanced Robotics and Intelligent Systems (ARIS)*. 2017.
- [89] Hassan K Khalil and Laurent Praly. “High-gain observers in nonlinear feedback control”. In: *International Journal of Robust and Nonlinear Control* 24.6 (2014), pp. 993–1015.
- [90] Chaitanya Kharyal et al. “Do As You Teach: A Multi-Teacher Approach to Self-Play in Deep Reinforcement Learning”. In: *Deep Reinforcement Learning Workshop at NeurIPS*. 2022.
- [91] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. “Optimization by simulated annealing”. In: *Science* 220.4598 (1983), pp. 671–680.
- [92] Pascal Klink and Jan Peters. “Measuring Similarities between Markov Decision Processes”. In: *Multidisciplinary Conference on Reinforcement Learning and Decision Making (RLDM)*. 2019.

-
-
- [93] Pascal Klink et al. “A probabilistic interpretation of self-paced learning with applications to reinforcement learning”. In: *Journal of Machine Learning Research (JMLR)* 22.182 (2021), pp. 1–52.
- [94] Pascal Klink et al. “Boosted Curriculum Reinforcement Learning”. In: *International Conference on Learning Representations (ICLR)*. 2022.
- [95] Pascal Klink et al. “Curriculum Reinforcement Learning via Constrained Optimal Transport”. In: *International Conference on Machine Learning (ICML)*. 2022.
- [96] Pascal Klink et al. “On the Benefit of Optimal Transport for Curriculum Reinforcement Learning”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2023, Submitted).
- [97] Pascal Klink et al. “Self-paced contextual reinforcement learning”. In: *Conference on Robot Learning (CoRL)*. 2020.
- [98] Pascal Klink et al. “Self-paced deep reinforcement learning”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.
- [99] Pascal Klink et al. “Tracking Control for a Spherical Pendulum via Curriculum Reinforcement Learning”. In: *IEEE Transactions on Robotics (TRO)* (2023, Submitted).
- [100] Jens Kober and Jan Peters. “Policy search for motor primitives in robotics”. In: *Advances in Neural Information Processing Systems (NIPS)*. 2008.
- [101] Jens Kober and Jan Peters. “Policy search for motor primitives in robotics”. In: *Advances in Neural Information Processing Systems (NIPS)*. 2009.
- [102] Soheil Kolouri et al. “Generalized sliced wasserstein distances”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.
- [103] Soheil Kolouri et al. “Optimal mass transport: Signal processing and machine-learning applications”. In: *IEEE Signal Processing Magazine* 34.4 (2017), pp. 43–59.
- [104] Solomon Kullback and Richard A Leibler. “On information and sufficiency”. In: *The annals of mathematical statistics* 22.1 (1951), pp. 79–86.
- [105] M Pawan Kumar, Benjamin Packer, and Daphne Koller. “Self-paced learning for latent variable models”. In: *Advances in Neural Information Processing Systems (NIPS)*. 2010.
- [106] Andras Gabor Kupcsik et al. “Data-efficient generalization of robot skills with contextual policy search”. In: *AAAI Conference on Artificial Intelligence (AAAI)*. 2013.

-
-
- [107] Zeb Kurth-Nelson and A David Redish. “Temporal-difference reinforcement learning with distributed representations”. In: *PLOS ONE* 4.10 (2009), pp. 1–19.
- [108] Alessandro Lazaric. “Transfer in reinforcement learning: a framework and a survey”. In: *Reinforcement Learning*. Springer, 2012, pp. 143–173.
- [109] Sergey Levine. “Reinforcement learning and control as probabilistic inference: Tutorial and review”. In: *arXiv preprint arXiv:1805.00909* (2018).
- [110] Sergey Levine et al. “End-to-end training of deep visuomotor policies”. In: *Journal of Machine Learning Research (JMLR)* 17.1 (2016), pp. 1334–1373.
- [111] Sergey Levine et al. “Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection”. In: *International Journal of Robotics Research (IJRR)* 37.4-5 (2018), pp. 421–436.
- [112] Lingxiao Li et al. “Continuous regularized wasserstein barycenters”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.
- [113] Qiyang Li et al. “Understanding the complexity gains of single-task rl with a curriculum”. In: *International Conference on Machine Learning (ICML)*. 2023.
- [114] Zhongyu Li et al. “Robust and versatile bipedal jumping control through multi-task reinforcement learning”. In: *arXiv preprint arXiv:2302.09450* (2023).
- [115] Chang Liu et al. “Understanding and accelerating particle-based variational inference”. In: *International Conference on Machine Learning (ICML)*. 2019.
- [116] Puze Liu et al. “Safe reinforcement learning of dynamic high-dimensional robotic tasks: navigation, manipulation, interaction”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2023.
- [117] Siqi Liu et al. “From Motor Control to Team Play in Simulated Humanoid Football”. In: *arXiv preprint arXiv:2105.12196* (2021).
- [118] Kent H Lundberg and Taylor W Barton. “History of inverted-pendulum systems”. In: *IFAC Proceedings Volumes* 42.24 (2010), pp. 131–135.
- [119] Rui Luo et al. “Thermostat-assisted continuously-tempered Hamiltonian Monte Carlo for Bayesian learning”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.
- [120] Michael Lutter et al. “Continuous-time fitted value iteration for robust policies”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 45.5 (2022), pp. 5534–5548.

-
-
- [121] Marlos C Machado, Marc G Bellemare, and Michael Bowling. “Count-based exploration with the successor representation”. In: *AAAI Conference on Artificial Intelligence (AAAI)*. 2020.
- [122] Prasanta Chandra Mahalanobis. “On the generalized distance in statistics”. In: *Proceedings of the National Institute of Science of India*. 1936.
- [123] Denys Makoviichuk and Viktor Makoviychuk. *rl-games: A High-performance Framework for Reinforcement Learning*. https://github.com/Denys88/rl_games. May 2021.
- [124] Stephan Mandt et al. “Variational tempering”. In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2016.
- [125] Peter Marbach and John N Tsitsiklis. “Simulation-based optimization of Markov reward processes”. In: *IEEE Transactions on Automatic Control (TACON)* 46.2 (2001), pp. 191–209.
- [126] Alonso Marco et al. “Automatic LQR tuning based on Gaussian process global optimization”. In: *IEEE International conference on robotics and automation (ICRA)*. 2016.
- [127] Enzo Marinari and Giorgio Parisi. “Simulated Tempering: A New Monte Carlo Scheme”. In: *Europhysics Letters* 19.6 (1992), pp. 451–458.
- [128] Carlos Mastalli et al. “Crocoddyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2020.
- [129] Francisco S Melo, Sean P Meyn, and M Isabel Ribeiro. “An analysis of reinforcement learning with function approximation”. In: *International Conference on Machine Learning (ICML)*. 2008.
- [130] Facundo Mémoli. “Gromov–Wasserstein distances and the metric approach to object matching”. In: *Foundations of computational mathematics* 11.4 (2011), pp. 417–487.
- [131] Deyu Meng, Qian Zhao, and Lu Jiang. “A theoretical understanding of self-paced learning”. In: *Information Sciences* 414 (2017), pp. 319–328.
- [132] Alberto Maria Metelli, Amarildo Likmeta, and Marcello Restelli. “Propagating uncertainty in reinforcement learning via wasserstein barycenters”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

-
-
- [133] Alberto Maria Metelli, Mirco Mutti, and Marcello Restelli. “Configurable Markov Decision Processes”. In: *International Conference on Machine Learning (ICML)*. 2018.
- [134] Volodymyr Mnih et al. “Human-level control through deep reinforcement learning”. In: *Nature* 518.7540 (2015), pp. 529–533.
- [135] Hossein Mobahi and John Fisher III. “A theoretical analysis of optimization by Gaussian continuation”. In: *AAAI Conference on Artificial Intelligence (AAAI)*. 2015.
- [136] Hossein Mobahi, C Lawrence Zitnick, and Yi Ma. “Seeing through the blur”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012.
- [137] Aditya Modi et al. “Markov decision processes with continuous side information”. In: *International Conference on Algorithmic Learning Theory (ALT)*. 2018.
- [138] Gaspard Monge. *Mémoire sur la théorie des déblais et des remblais*. Imprimerie Royale, 1781.
- [139] Igor Mordatch, Emanuel Todorov, and Zoran Popović. “Discovery of complex behaviors through contact-invariant optimization”. In: *ACM Transactions on Graphics (ToG)* 31.4 (2012), pp. 1–8.
- [140] Alexander Morgan. *Solving polynomial systems using continuation for engineering and scientific problems*. SIAM, 2009.
- [141] Rémi Munos. “Performance bounds in l_p -norm for approximate value iteration”. In: *SIAM journal on control and optimization* 46.2 (2007), pp. 541–561.
- [142] Elizbar A Nadaraya. “On estimating regression”. In: *Theory of Probability & Its Applications* 9.1 (1964), pp. 141–142.
- [143] Abhishek Naik et al. “Discounted reinforcement learning is not an optimization problem”. In: *Optimization Foundations for Reinforcement Learning Workshop at NeurIPS*. 2019.
- [144] Sanmit Narvekar, Jivko Sinapov, and Peter Stone. “Autonomous Task Sequencing for Customized Curriculum Design in Reinforcement Learning.” In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2017.
- [145] Sanmit Narvekar and Peter Stone. “Learning curriculum policies for reinforcement learning”. In: *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 2019.
- [146] Sanmit Narvekar et al. “Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey”. In: *Journal of Machine Learning Research (JMLR)* 21.181 (2020), pp. 1–50.

-
-
- [147] Radford M Neal. “Annealed importance sampling”. In: *Statistics and Computing* 11.2 (2001), pp. 125–139.
- [148] Gerhard Neumann. “Variational inference for policy search in changing situations”. In: *International Conference on Machine Learning (ICML)*. 2011.
- [149] Andrew Y Ng, Daishi Harada, and Stuart Russell. “Policy invariance under reward transformations: Theory and application to reward shaping”. In: *International Conference on Machine Learning (ICML)*. 1999.
- [150] NVIDIA. *Nvidia Isaac Sim*. <https://developer.nvidia.com/isaac-sim>. July 2023.
- [151] Nvidia. *Isaac Gym*. <https://developer.nvidia.com/gtc/2019/video/S9918>. Accessed: 2020-02-06. 2019.
- [152] Pierre-Yves Oudeyer, Frdric Kaplan, and Verena V Hafner. “Intrinsic motivation systems for autonomous mental development”. In: *IEEE Transactions on Evolutionary Computation (TEVC)* 11.2 (2007), pp. 265–286.
- [153] Sinno Jialin Pan and Qiang Yang. “A survey on transfer learning”. In: *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 22.10 (2009), pp. 1345–1359.
- [154] Alexandros Paraschos et al. “Probabilistic movement primitives”. In: *Advances in Neural Information Processing Systems (NIPS)*. 2013.
- [155] Simone Parisi et al. “Reinforcement learning vs human programming in tetherball robot games”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2015.
- [156] Adam Paszke et al. “Pytorch: An imperative style, high-performance deep learning library”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.
- [157] Jan Peters, Katharina Mulling, and Yasemin Altun. “Relative entropy policy search”. In: *AAAI Conference on Artificial Intelligence (AAAI)*. 2010.
- [158] Gabriel Peyré, Marco Cuturi, et al. “Computational optimal transport: With applications to data science”. In: *Foundations and Trends® in Machine Learning* 11.5-6 (2019), pp. 355–607.
- [159] Kai Ploeger, Michael Lutter, and Jan Peters. “High acceleration reinforcement learning for real-world juggling with binary rewards”. In: *Conference on Robot Learning (CoRL)*. 2020.
- [160] Natural Point. *Optitrack*. <https://optitrack.com>. July 2023.
- [161] Ivalylo Popov et al. “Data-efficient deep reinforcement learning for dexterous manipulation”. In: *arXiv preprint arXiv:1704.03073* (2017).

-
-
- [162] Rémy Portelas et al. “Teacher algorithms for curriculum learning of Deep RL in continuously parameterized environments”. In: *Conference on Robot Learning (CoRL)*. 2019.
- [163] Simon JD Prince. *Computer Vision: Models, Learning, and Inference*. Cambridge University Press, 2012.
- [164] Morgan Quigley et al. “ROS: an open-source Robot Operating System”. In: *ICRA Workshop on Open Source Robotics*. 2009.
- [165] Sebastien Racaniere et al. “Automated curricula through setter-solver interactions”. In: *International Conference on Learning Representations (ICLR)*. 2020.
- [166] Jette Randløv and Preben Alstrøm. “Learning to Drive a Bicycle Using Reinforcement Learning and Shaping.” In: *International Conference on Machine Learning (ICML)*. 1998.
- [167] Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. “On stochastic optimal control and reinforcement learning by approximate inference”. In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2013.
- [168] Zhipeng Ren et al. “Self-paced prioritized curriculum learning with coverage penalty in deep reinforcement learning”. In: *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)* 29.6 (2018), pp. 2216–2226.
- [169] Zhizhou Ren et al. “Exploration via hindsight goal generation”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.
- [170] Martin Riedmiller et al. “Learning by playing-solving sparse reward tasks from scratch”. In: *International Conference on Machine Learning (ICML)*. 2018.
- [171] Clément Romac et al. “TeachMyAgent: a Benchmark for Automatic Curriculum Learning in Deep RL”. In: *International Conference on Machine Learning (ICML)*. 2021.
- [172] Philipp Rostalski et al. “Numerical algebraic geometry for optimal control applications”. In: *SIAM Journal on Optimization* 21.2 (2011), pp. 417–437.
- [173] Nikita Rudin et al. “Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning”. In: *Conference on Robot Learning (CoRL)*. 2021.
- [174] Stefan Schaal. “Dynamic movement primitives-a framework for motor control in humans and humanoid robotics”. In: *Adaptive Motion of Animals and Machines*. Springer, 2006, pp. 261–280.
- [175] Tom Schaul et al. “Universal value function approximators”. In: *International Conference on Machine Learning (ICML)*. 2015.

-
-
- [176] Jürgen Schmidhuber. “Curious model-building control systems”. In: *International Joint Conference on Neural Networks (IJCNN)*. 1991.
- [177] John Schulman et al. “Proximal policy optimization algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017).
- [178] John Schulman et al. “Trust region policy optimization”. In: *International Conference on Machine Learning (ICML)*. 2015.
- [179] Matthias Schultheis, Constantin A Rothkopf, and Heinz Koepl. “Reinforcement learning with non-exponential discounting”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2022.
- [180] Matthias Schultheis et al. “Receding Horizon Curiosity”. In: *Conference on Robot Learning (CoRL)*. 2020.
- [181] Oliver G Selfridge, Richard S Sutton, and Andrew G Barto. “Training and Tracking in Robotics”. In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 1985.
- [182] David Silver et al. “Mastering the game of Go with deep neural networks and tree search”. In: *nature* 529.7587 (2016), pp. 484–489.
- [183] David Silver et al. “Mastering the game of Go without human knowledge”. In: *Nature* 550.7676 (2017), p. 354.
- [184] Burrhus Frederic Skinner. *The Behavior of Organisms: An Experimental Analysis*. Appleton-Century, 1938.
- [185] Bernhard Sprenger, Ladislav Kucera, and Safer Mourad. “Balancing of an inverted pendulum with a SCARA robot”. In: *IEEE/ASME Transactions on Mechatronics (TMECH)* 3.2 (1998), pp. 91–97.
- [186] Sainbayar Sukhbaatar et al. “Intrinsic motivation and automatic curricula via asymmetric self-play”. In: *International Conference on Learning Representations (ICLR)*. 2018.
- [187] James S Supancic and Deva Ramanan. “Self-paced learning for long-term tracking”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2013.
- [188] Richard S Sutton. “Learning to predict by the methods of temporal differences”. In: *Machine learning* 3 (1988), pp. 9–44.
- [189] Richard S Sutton and Andrew G Barto. *Introduction to Reinforcement Learning*. MIT Press, 1998.
- [190] Richard S Sutton et al. “Policy gradient methods for reinforcement learning with function approximation”. In: 1999.

-
-
- [191] Haoran Tang et al. “# exploration: A study of count-based exploration for deep reinforcement learning”. In: *Advances in Neural Information Processing Systems (NIPS)*. 2017.
- [192] Matthew E Taylor and Peter Stone. “Transfer learning for reinforcement learning domains: A survey.” In: *Journal of Machine Learning Research (JMLR)* 10.7 (2009), pp. 1633–1685.
- [193] Matthew E Taylor, Peter Stone, and Yaxin Liu. “Transfer Learning via Inter-Task Mappings for Temporal Difference Learning.” In: *Journal of Machine Learning Research* 8.9 (2007).
- [194] Barrett Technology. *Barrett Whole Arm Manipulator*. <https://advanced.barrett.com/wam-arm-1>. July 2023.
- [195] Emanuel Todorov, Tom Erez, and Yuval Tassa. “Mujoco: A physics engine for model-based control”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2012.
- [196] Matteo Togninalli et al. “Wasserstein weisfeiler-lehman graph kernels”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.
- [197] Marc Toussaint and Amos Storkey. “Probabilistic inference for solving discrete and continuous state Markov Decision Processes”. In: *International Conference on Machine Learning (ICML)*. 2006.
- [198] J.N. Tsitsiklis and B. Van Roy. “An analysis of temporal-difference learning with function approximation”. In: *IEEE Transactions on Automatic Control (TACON)* 42.5 (1997), pp. 674–690.
- [199] Georgios Tzannetos et al. “Proximal Curriculum for Reinforcement Learning Agents”. In: *Transactions on Machine Learning Research (TMLR)* (2023). ISSN: 2835-8856.
- [200] Naonori Ueda and Ryohei Nakano. “Deterministic annealing variant of the EM algorithm”. In: *Advances in Neural Information Processing Systems (NIPS)*. 1995.
- [201] Herke Van Hoof, Gerhard Neumann, and Jan Peters. “Non-parametric policy search with limited information loss”. In: *Journal of Machine Learning Research (JMLR)* 18.73 (2017), pp. 1–46.
- [202] Peter JM Van Laarhoven and Emile HL Aarts. “Simulated annealing”. In: *Simulated Annealing: Theory and Applications*. Springer, 1987, pp. 7–15.

-
-
- [203] Cédric Vincent-Cuaz et al. “Semi-relaxed Gromov-Wasserstein divergence and applications on graphs”. In: *International Conference on Learning Representations (ICLR)*. 2022.
- [204] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272.
- [205] Minh Nhat Vu, Christian Hartl-Nesic, and Andreas Kugi. “Fast swing-up trajectory optimization for a spherical pendulum on a 7-dof collaborative robot”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2021.
- [206] Rui Wang et al. “Poet: open-ended coevolution of environments and their optimized solutions”. In: *Genetic and Evolutionary Computation Conference (GECCO)*. 2019.
- [207] Christopher JCH Watkins and Peter Dayan. “Q-learning”. In: *Machine learning* 8 (1992), pp. 279–292.
- [208] Geoffrey S Watson. “Smooth regression analysis”. In: *Sankhyā: The Indian Journal of Statistics, Series A* 26.4 (1964), pp. 359–372.
- [209] Joe Watson et al. “Latent Derivative Bayesian Last Layer Networks”. In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2021.
- [210] Joe Watson et al. “Neural Linear Models with Functional Gaussian Process Priors”. In: *Symposium on Advances in Approximate Bayesian Inference (AABI)*. 2021.
- [211] Layne T Watson. “Numerical linear algebra aspects of globally convergent homotopy methods”. In: *SIAM review* 28.4 (1986), pp. 529–545.
- [212] Layne T Watson. “Theory of globally convergent probability-one homotopies for nonlinear programming”. In: *SIAM Journal on Optimization* 11.3 (2001), pp. 761–780.
- [213] Layne T Watson and Raphael T Haftka. “Modern homotopy methods in optimization”. In: *Computer Methods in Applied Mechanics and Engineering* 74.3 (1989), pp. 289–305.
- [214] Layne T Watson et al. “Algorithm 777: HOMPACT90: A suite of Fortran 90 codes for globally convergent homotopy algorithms”. In: *ACM Transactions on Mathematical Software (TOMS)* 23.4 (1997), pp. 514–549.
- [215] Daphna Weinshall and Dan Amir. “Theory of curriculum learning, with convex loss functions”. In: *Journal of Machine Learning Research (JMLR)* 21.222 (2020), pp. 1–19.

-
-
- [216] Andre Wibisono. “Sampling as optimization in the space of measures: The Langevin dynamics as a composite optimization problem”. In: *Conference on Learning Theory (COLT)*. 2018.
- [217] Ronald J Williams. “Simple statistical gradient-following algorithms for connectionist reinforcement learning”. In: *Machine learning* 8 (1992), pp. 229–256.
- [218] Ronald J Williams and Jing Peng. “Function optimization using connectionist reinforcement learning algorithms”. In: *Connection Science* 3.3 (1991), pp. 241–268.
- [219] Aaron Wilson et al. “Multi-task reinforcement learning: a hierarchical bayesian approach”. In: *International Conference on Machine Learning (ICML)*. 2007.
- [220] Jan Wöhlke, Felix Schmitt, and Herke van Hoof. “A Performance-Based Start State Curriculum Framework for Reinforcement Learning”. In: *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 2020.
- [221] Xiaoxia Wu, Ethan Dyer, and Behnam Neyshabur. “When Do Curricula Work?” In: *International Conference on Learning Representations (ICLR)*. 2021.
- [222] Zifan Xu et al. “Model-Based Meta Automatic Curriculum Learning”. In: *Conference on Lifelong Learning Agents (CoLLAs)*. 2023.
- [223] Rong Yang, Yiu-Yiu Kuen, and Zexiang Li. “Stabilization of a 2-DOF spherical pendulum on xy table”. In: *International Conference on Control Applications (CCA)*. 2000.
- [224] Cun-Hui Zhang. “Nearly unbiased variable selection under minimax concave penalty”. In: *The Annals of Statistics* 38.2 (2010), pp. 894–942.
- [225] Ruiyi Zhang et al. “Policy optimization as wasserstein gradient flows”. In: *International Conference on Machine Learning (ICML)*. 2018.
- [226] Tong Zhang. “Multi-stage convex relaxation for learning with sparse regularization”. In: *Advances in Neural Information Processing Systems (NIPS)*. 2008.
- [227] Yunzhi Zhang, Pieter Abbeel, and Lerrel Pinto. “Automatic curriculum learning through value disagreement”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.
- [228] Brian D Ziebart et al. “Maximum entropy inverse reinforcement learning”. In: *AAAI Conference on Artificial Intelligence (AAAI)*. 2008.

List of Figures

1.1. Curriculum Reinforcement Learning Applications	2
1.2. Local Optima in Policy Gradient Methods	6
1.3. Gaussian Continuations for two Functions	8
2.1. Loss Function Shaping via Self-Paced Regularizers	24
2.2. SPRL Evaluation - Gate Environment (Rewards and Curriculum)	31
2.3. SPRL Evaluation - Gate Environment (Success Rates)	32
2.4. SPRL Evaluation - Reacher Environment (Rewards and Curriculum)	33
2.5. SPRL Evaluation - Reacher Environment (Final Policies)	34
2.6. SPRL Evaluation - Ball-in-a-Cup Environment (Success Rates and Curriculum)	34
2.7. SPDL Evaluation Environments	37
2.8. SPDL Evaluation - Point-Mass Environment (Rewards and Curriculum)	38
2.9. SPDL Evaluation - Ant- and Ball-Catching Environment (Rewards)	40
2.10. SPDL Hyperparameter Comparison - Point-Mass Environment	42
3.1. CURROT Schematic Overview	52
3.2. KL- and Wasserstein Interpolations between Distributions	57
3.3. Interpolations Generated with an Expected Performance Constraint, CURROT, and GRADIENT	59
3.4. CURROT Evaluation - E-Maze Environment	65
3.5. CURROT Evaluation - Unlock-Pickup Environment	68
3.6. CURROT Evaluation - Point-Mass Environment (Curriculum)	69
3.7. CURROT Evaluation - Point-Mass Environment (Rewards)	70
3.8. CURROT Evaluation - Sparse Goal Reaching Environment (Success Rates)	71
3.9. CURROT Evaluation - Sparse Goal Reaching Environment (Curriculum)	71
3.10. CURROT Evaluation - TMA Environment (Curriculum)	72
3.11. CURROT Evaluation - TMA Environment (Success Rates)	73
4.1. Pendulum Tracking Task and Simulation	76
4.2. Pendulum Tracking Policy Architecture	79
4.3. CURROT Sampling Scheme	84

4.4. CURROT in High-Dimensional Context Spaces	86
4.5. Target Tracking Trajectories and Context Spaces	88
4.6. Learned Tracking Performance in Simulation	90
4.7. CURROT Curricula over Tracking Trajectories	91
4.8. CURROT Wasserstein Distances over Epochs	91
4.9. Learned Tracking Performance in Simulation (Policy Structure Ablation)	92
4.10. Tracking Trajectories on the Real Robot	93
4.11. Policy Execution on the Real Robot	94
B.1. SPRL Gate Environment - Local Optimum Visualization	116
B.2. SPDL Evaluation - Point-Mass Environment (Final Policies)	122
B.3. SPDL Evaluation - Ant Environment (Final Policies)	123
B.4. SPDL Evaluation - Ball-Catching Environment (Success Rates)	126
C.1. CURROT Evaluation - E-Maze Environment (Performance Distance)	133
C.2. PLR Curriculum Statistics - Unlock-Pickup Environment	136
C.3. CURROT Evaluation - Point-Mass Environment (Final Policies)	137
C.4. SPRL Curricula - Sparse-Goal Reaching Environment and Point-Mass Environment	138
D.1. CURROT Curricula in High-Dimensional Context Spaces	142

List of Algorithms

1.	Self-Paced Episodic Reinforcement Learning (SPRL)	29
2.	Self-Paced Deep Reinforcement Learning (SPDL)	36
3.	Approximate GRADIENT	62
4.	Approximate CURROT	64

List of Tables

2.1. SPDL Evaluation - Quantitative Results	41
2.2. SPDL α -Schedule Comparison	44
3.1. CURROT Entropy Ablation - E-Maze Environment	66
4.1. Real Robot Tracking Performances	95
B.1. SPRL/C-REPS Hyperparameters	113
B.2. GOALGAN/SAGG-RIAC Hyperparameters	114
B.3. SPDL Hyperparameters	118
B.4. GOALGAN/ALP-GMM Hyperparameters	120
B.5. Target Distributions in the Evaluation Environments	121
C.1. SPRL/CURROT/GRADIENT Hyperparameters	129
C.2. Baseline Algorithm Hyperparameters	129

Publication List

Journal Papers

Pascal Klink, Florian Wolf, Kai Ploeger, Jan Peters, and Joni Pajarinen. “Tracking Control for a Spherical Pendulum via Curriculum Reinforcement Learning”. In: *IEEE Transactions on Robotics (TRO)* (2023, Submitted)

Pascal Klink, Carlo D’Eramo, Jan Peters, and Joni Pajarinen. “On the Benefit of Optimal Transport for Curriculum Reinforcement Learning”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2023, Submitted)

Hany Abdulsamad, Peter Nickl, Pascal Klink, and Jan Peters. “Variational Hierarchical Mixtures for Probabilistic Learning of Inverse Dynamics”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2023, In Press)

Pascal Klink, Hany Abdulsamad, Boris Belousov, Carlo D’Eramo, Jan Peters, and Joni Pajarinen. “A probabilistic interpretation of self-paced learning with applications to reinforcement learning”. In: *Journal of Machine Learning Research (JMLR)* 22.182 (2021), pp. 1–52

Conference Papers

Pascal Klink, Haoyi Yang, Carlo D’Eramo, Jan Peters, and Joni Pajarinen. “Curriculum Reinforcement Learning via Constrained Optimal Transport”. In: *International Conference on Machine Learning (ICML)*. 2022

Pascal Klink, Carlo D’Eramo, Jan Peters, and Joni Pajarinen. “Boosted Curriculum Reinforcement Learning”. In: *International Conference on Learning Representations (ICLR)*. 2022

Hany Abdulsamad, Peter Nickl, Pascal Klink, and Jan Peters. “A Variational Infinite

Mixture for Probabilistic Inverse Dynamics Learning”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2021

Joe Watson, Jihao Andreas Lin, Pascal Klink, and Jan Peters. “Neural Linear Models with Functional Gaussian Process Priors”. In: *Symposium on Advances in Approximate Bayesian Inference (AABI)*. 2021

Joe Watson, Jihao Andreas Lin, Pascal Klink, Joni Pajarinen, and Jan Peters. “Latent Derivative Bayesian Last Layer Networks”. In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2021

Pascal Klink, Carlo D’Eramo, Jan R Peters, and Joni Pajarinen. “Self-paced deep reinforcement learning”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020

Tuan Dam, Pascal Klink, Carlo D’Eramo, Jan Peters, and Joni Pajarinen. “Generalized Mean Estimation in Monte-Carlo Tree Search”. In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 2020

Pascal Klink, Hany Abdulsamad, Boris Belousov, and Jan Peters. “Self-paced contextual reinforcement learning”. In: *Conference on Robot Learning (CoRL)*. 2020

Pascal Klink and Jan Peters. “Measuring Similarities between Markov Decision Processes”. In: *Multidisciplinary Conference on Reinforcement Learning and Decision Making (RLDM)*. 2019

Books

Boris Belousov, Hany Abdulsamad, Pascal Klink, Simone Parisi, and Jan Peters. *Reinforcement Learning Algorithms: Analysis and Applications*. Springer International Publishing, 2021

Curriculum Vitae

Mr. Pascal Klink
Technische Universität Darmstadt
Hochschulstr. 10
64289 Darmstadt, Germany

pascal@robot-learning.de
<https://www.ias.informatik.tu-darmstadt.de/Team/PascalKlink>

Education

- | | |
|---|----------------|
| Ph.D. Student in Computer Science
Intelligent Autonomous Systems Group, Technical University of Darmstadt, Germany
Thesis: Reinforcement Learning Curricula as Interpolations between Task Distributions | 2019 - present |
| M.Sc. Autonomous Systems
Technical University of Darmstadt, Germany
Thesis: Generalization and Transferability in Reinforcement Learning
GPA (German System): 1.02 | 2016-2019 |
| B.Sc. Computer Science
Technical University of Darmstadt, Germany
Thesis: Model Learning for Probabilistic Movement Primitives
GPA (German System): 1.00 | 2013-2016 |
| General University Entrance Qualification
Martin-Luther-Schule Rimbach, Germany
GPA (German System): 1.00 | 2003-2012 |

Work Experience

Research Scientist Intern Amazon Robotics, Berlin, Germany Research on Automation Technology for Amazon Fulfillment Centers	2022-2023
Student Assistant Intelligent Autonomous Systems Group, Technical University of Darmstadt, Germany Development of a robot control architecture for C++ and Python capable of running at kHz control frequencies. Support of research in the field of Machine Learning for Advanced Robotic Motor Skills.	2016-2019
Working Student Bisnode Informatics Deutschland GmbH, Darmstadt, Germany Implementation of RESTful web services using Java, relational-, and document-based databases processing 10,000 - 100,000 datasets per day.	2013-2016

Scholarships and Exchange Terms

Deutschlandstipendium In cooperation with Datenlotsen Informationssysteme GmbH (2013 - 2014) and Bosch Rexroth AG (2014 - 2018).	2013-2018
Exchange Term University of British Columbia, Vancouver	2017-2018

Teaching

Computational Engineering and Robotics TU Darmstadt	2021
Computational Engineering and Robotics TU Darmstadt	2020
Organization of Computing Systems TU Darmstadt	2019

Invited Talks

Technical University of Berlin Berlin, Oliver Brock	2019
ATR Computational Neuroscience Lab Kyoto, Mitsuo Kawato	2019
Preferred Networks Tokyo, Guilherme Maeda	2019
NAIST Robot Learning Lab Nara, Takamitsu Matsubara	2019

Student Supervision

Thesis Supervision

Modeling and Control of a Spherical Pendulum on a 4-DOF Barret WAM Bachelor Thesis of Kinzel, J.	2022
Multi-Object Pose Estimation for Robotic Applications in Cluttered Scenes Master Thesis of Felix, K.	2022
Particle-Based Adaptive Sampling for Curriculum Learning Master Thesis of Carrasco, H.	2022
Variational Inference for Curriculum Reinforcement Learning Master Thesis of Yang, H.	2022
Multi-Instance Pose Estimation for Robot Mikado Bachelor Thesis of Meser, M.	2022
Functional Variational Inference in Bayesian Neural Networks Master Thesis of Lin, J. A.	2021
Variational Inference for Curriculum Reinforcement Learning Master Thesis of Yang, H.	2021

Memory Representations for Partially Observable Reinforcement Learning 2021
Master Thesis of Zhang, Y.

3D Pose Estimation for Robot Mikado 2019
Master Thesis of Tengang, V. M.

Project Supervision

Measuring Task Similarity using Learned Features 2023
Integrated Project of Metternich, H.

Pendulum Acrobatics 2023
Integrated Project of Wolf, F.

Combining Intrinsic Motivation and Self-Paced Reinforcement Learning 2021
Integrated Project of Scheler, U. and Niehues, T.

Multi-Object Pose Estimation for Robot Mikado 2021
Integrated Project of Kaiser, F. and Meser, M. and Sterker, L.

Review Experience

Conferences

International Conference on Intelligent Robots and Systems (**IROS**) 2023
Symposium on Advances in Approximate Bayesian Inference (**AABI**) 2023
Conference on Robot Learning (**CoRL**) 2022
International Conference on Intelligent Robots and Systems (**IROS**) 2022
International Conference on Learning Representations (**ICLR**) 2022
International Conference on Robotics and Automation (**ICRA**) 2022
Neural Information Processing Systems (**NeurIPS**) 2022
Conference on Robot Learning (**CoRL**) 2021
International Conference on Automated Planning and Scheduling (**ICAPS**) 2021
International Conference on Intelligent Robots and Systems (**IROS**) 2021
International Conference on Learning Representations (**ICLR**) 2021
International Conference on Robotics and Automation (**ICRA**) 2021
Neural Information Processing Systems (**NeurIPS**) 2021
Robotics: Science and Systems (**RSS**) 2021

International Conference on Automated Planning and Scheduling (**ICAPS**) 2020

Journals

Robotics and Automation Letters (**RA-L**) 2023
Journal of Machine Learning Research (**JMLR**) 2023
Journal of Machine Learning Research (**JMLR**) 2022
Transaction on Robotics (**T-RO**) 2022
Transaction on Machine Learning Research (**TMLR**) 2022
Robotics and Automation Letters (**RA-L**) 2022