


# **Structure, Learning, & Composition: Multitask Reinforcement Learning in Brains and Machines**

*Theodore Harris Moskowitz*

A dissertation submitted in partial fulfillment  
of the requirements for the degree of  
**Doctor of Philosophy**  
of  
**University College London.**

Gatsby Computational Neuroscience Unit  
University College London

April 23, 2024

I, , confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

# Abstract

This thesis centers around different forms of common structure that can be shared across tasks faced by reinforcement learning agents and how these types of structure can be leveraged to both learn new behavioral policies more efficiently and compose existing policies. Specifically, the first part of this thesis is concerned with how agreement among the optimal policies for some group of tasks constitutes a form of behavioral structure. This structure can be used as the basis for a regularized policy optimization approach to speed up policy learning on new tasks. One such approach proves to be an effective model of a number of animal and human behavioral patterns observed in neuroscientific studies of dual process theories of cognition. The second part of this thesis focuses on how consistent environmental transition dynamics across tasks can be exploited by agents to learn state representations which facilitate efficient policy evaluation and composition. In particular, prior work on this topic is extended to include several forms of biologically-inspired, non-Markovian, non-stationary reward functions, with applications to both machine learning and natural behavior.

# Impact Statement

The work presented in this thesis contributes towards the study of sequential decision-making. Its primary focus is on multitask reinforcement learning from a machine learning perspective. However, there are insights for and applications to neuroscience as well.

In terms of machine learning, the study of the convergence properties of KL-regularized policy optimization in a multitask context can provide insight into a wide range of domains where sequential decision-making across tasks is important (e.g., robotics). Additionally, the analysis of KL-regularized policy optimization in a single task context in which the KL cost is computed with respect to a policy that may be “far away” from the action policy provides insight into the training setting for fine-tuning large language models from human feedback. The multitask algorithms developed (Total Variation Policy Optimization and Minimum Description Length Control) can be productively applied in both discrete and continuous control problems. Similarly, the work on state representations which enable policy evaluation and composition for a particular class of non-Markovian, non-stationary rewards can also be impactful. In particular, the first-occupancy representation shows promise as a bonus for intrinsic exploration, as a representation for unsupervised pre-training in unsupervised reinforcement learning, and as the foundation of a planning algorithm to compute shortest paths. The  $\lambda$  representation can be used to help mitigate value overestimation in deep reinforcement learning.

From a neuroscience perspective, Minimum Description Length Control captures many behavioral patterns observed in humans and animals. While this framework is a simplification of control systems in the brain, these results could have implications for our understanding of the theoretical underpinnings of “dual process” cognition. Simi-



larly, the first occupancy and  $\lambda$  representations have applications to the study of spatial navigation, particularly escape behavior and foraging, as well as diminishing marginal utility, among other areas.

# Acknowledgements

The last four years have been a transformative, enriching time in my life. Moving to London and coming to Gatsby was one of the best decisions I've ever made, largely due to the amazing people I've met.

I am immensely grateful to my primary advisor, Maneesh Sahani. He has taught me a tremendous amount about what it means to be a good scientist. I am also very thankful for the freedom he's granted me to pursue a wide variety of research directions. I am also very grateful to my secondary advisor, Matt Botvinick, for his consistent support, insight, and guidance. More broadly, I would also like to thank the community at Gatsby for fostering such a welcoming, friendly environment for research and collaboration. I'd especially like to thank Hugo, Aaditya, Peter, Alex, Samo, Lea, and Michael for all the fun the last few years.

I have been very fortunate to meet and collaborate with many people outside of Gatsby during my PhD. I'd particularly like to thank Tom Zahavy for his incredibly helpful guidance both during and after my internship. I'm also very thankful to DJ, Vivek, Brendan, Seb, and Dave, along with the rest of the friends I met at DeepMind. I also had a lot of fun and was fortunate to work with Aldo Pacchiano, Stephen McAleer, Diana Borsa, Ahmed Touati, and Jack Parker-Holder, among a number of others. I am especially indebted to DJ, Aaditya, and Dan for their incredibly helpful advice and support over the last year.

I am also very thankful to my non-ML London friends for all the good times and welcome distractions, particularly Matt, Franco, Rebecca, Fraser, Vaish, and Jahaan.

I am also thankful to Larry Abbott, Ashok Litwin-Kumar, and Jonathan Pillow for introducing me to research and for their support and mentorship.

I am infinitely grateful to my parents for their unending love and support.

Finally, I'd like to thank Amani for agreeing to marry me, demonstrating how to do a PhD properly, and for putting up with my bullshit.

# Contents

<b>Introduction</b>	<b>35</b>
<b>I Overview</b>	<b>36</b>
<b>2 Reinforcement Learning</b>	<b>38</b>
2.1 Defining the Environment . . . . .	38
2.1.1 Tasks . . . . .	39
2.2 Solving Single-Task RL . . . . .	43
2.2.1 Value-Based Approaches . . . . .	44
2.2.2 Policy-Based Approaches . . . . .	52
2.3 Multitask Reinforcement Learning . . . . .	57
2.4 RL and Neuroscience . . . . .	60
2.4.1 Reward Prediction . . . . .	60
2.4.2 Habits . . . . .	61
<b>I Shared Behaviors</b>	<b>63</b>
<b>3 Towards an Understanding of Default Policies in Multitask Policy Optimization</b>	<b>65</b>
3.1 Introduction . . . . .	65
3.2 Regularized Policy Optimization . . . . .	67
3.3 Related Work . . . . .	69
3.4 A Basic Theory for Default Policies . . . . .	71
3.4.1 Log-barrier regularization . . . . .	71

3.4.2	Regularization with an $\alpha$ -optimal policy	73
3.5	Extension to Multitask Learning	77
3.6	Understanding the Literature	80
3.7	Experiments	81
3.8	Discussion	84
Chapter 3 Appendix		86
3.A	Single-task Analysis	86
3.A.1	State dependent $\lambda$ and $\epsilon$	96
3.B	Multitask Analysis	103
3.C	Experimental Details	109
<b>4</b>	<b>Minimum Description Length Control</b>	<b>110</b>
4.1	Introduction	110
4.2	Setting	112
4.3	The Minimum Description Length Principle	112
4.4	Minimum Description Length Control	114
4.4.1	Motivating the choice of sparsity-inducing priors	115
4.4.2	Performance Analysis	118
4.5	Experiments	120
4.5.1	2D Navigation	121
4.5.2	Continuous Control	122
4.6	Related Work	125
4.7	Conclusion	125
Chapter 4 Appendix		127
4.A	Reinforcement Learning as Inference	127
4.B	Multitask RL Frameworks	128
4.C	Additional Related Work	130
4.D	Motivating the choice of sparsity-inducing priors	131
4.D.1	Correspondence between $p(z)$ and $p(\beta)$	132
4.D.2	MSE risk	132
4.E	Limitations	138

4.F: OCO Background . . . . .	139
4.G: Proofs of Performance Bounds and Additional Theoretical Results . . . . .	139
4.G.3 MDL-C with Persistent Replay . . . . .	143
4.G.4 Comment on Improvement Across Tasks . . . . .	145
4.G.5 Parallel Task Setting . . . . .	146
4.H: Additional Experimental Details . . . . .	147
4.H.6 FourRooms . . . . .	147
4.H.7 DeepMind Control Suite . . . . .	149
4.I: Additional Experimental Results . . . . .	151
4.I.8 FourRooms . . . . .	151
4.I.9 DeepMind Control Suite . . . . .	151
<b>5 A Unified Theory of Dual-Process Control</b>	<b>155</b>
5.1 Introduction . . . . .	155
5.2 General methods: Selection of target phenomena and approach to modeling . . . . .	157
5.3 Results . . . . .	158
5.3.1 Simulation 1: Executive control . . . . .	158
5.3.2 Simulation 2: Reward-based learning . . . . .	162
5.3.3 Simulation 3: Judgment and decision making . . . . .	166
5.3.4 Comparison with Previous Models . . . . .	168
5.4 Discussion . . . . .	175
Chapter 5 Appendix . . . . .	177
5.A: Architecture and Learning Algorithm . . . . .	177
5.B: Simulation 1: Executive control . . . . .	182
5.C: Simulation 2: Reward-based learning . . . . .	184
5.D: Simulation 3: Judgment and decision-making . . . . .	187
<b>II Policy Composition Through Shared Dynamics</b>	<b>192</b>
<b>6 A First-Occupancy Representation for Reinforcement Learning</b>	<b>194</b>

6.1	Introduction	194
6.2	Reinforcement Learning Preliminaries	195
6.3	The First-Occupancy Representation	198
6.4	Experiments	202
6.4.1	The FR as an Exploration Bonus	202
6.4.2	Unsupervised RL with the FF	203
6.4.3	Planning with the FR	206
6.4.4	Escape behavior	209
6.5	Conclusion	210
	Chapter 6 Appendix	212
6.A:	FR recursion	212
6.B:	FRP Algorithm	212
6.C:	Additional Experimental Details	213
6.D:	Additional Proofs	225
6.E:	Explicit Planning	227
6.F:	FRP with Multiple Goals	228
6.G:	Connections to options	228
6.H:	Further connections to related work	229
6.I:	FR vs. SR Visualization	232
<b>7</b>	<b>The <math>\lambda</math>-Occupancy Representation</b>	<b>234</b>
7.1	Diminishing Marginal Utility	236
7.2	The $\lambda$ Representation	236
7.2.1	Continuous State Spaces	239
7.3	Policy Evaluation, Learning, and Composition under DMU	241
7.3.1	Policy Evaluation	241
7.3.2	Policy Learning	242
7.3.3	Policy Composition	243
7.4	Understanding Natural Behavior	245
7.5	Conclusion	246
	Chapter 7 Appendix	248

7.A: Derivation of $\lambda$ R Recursion	248
7.B: Theoretical Analysis	248
7.B.1 Proof of Theorem 7.3.1	252
7.B.2 An Extension of Theorem 7.3.1	257
7.C: An $n$ th Occupancy Representation	261
7.D: Additional Related Work	262
7.E: Further Experimental Details	263
7.E.3 Policy Evaluation	263
7.E.4 Policy Learning	264
7.E.5 Tabular GPI	265
7.E.6 Pixel-Based GPI	266
7.E.7 Continuous Control	267
7.E.8 Learning the $\lambda$ O with FB	267
7.F: Additional Results	270
7.G: Advantage of the Correct $\lambda$	270
7.H: The $\lambda$ Operator	272
7.H.9 Experimental Results with the FB Parameterization	274
7.H.10 $\lambda$ O and the Marginal Value Theorem	275
7.I: SAC	276
7.J: Replenishing Rewards	280
7.K: $\lambda$ vs. $\gamma$	282
7.L: Compute Resources	282
<b>Conclusion</b>	<b>284</b>
<b>8 General Conclusions</b>	<b>285</b>
8.1 Summary	285
8.2 Discussion	286
<b>Bibliography</b>	<b>289</b>



# List of Figures

3.1	As $ \mathcal{A} $ grows, regularizing using $\pi_0$ with larger $d_{TV}(\pi^*(\cdot s), \pi_0(\cdot s))$ will converge to a lower error than log-barrier regularization. In other words, there is a more forgiving margin of error for the default policy.	75
3.2	A tree environment. Each task in the family randomly distributes rewards among leaves marked with a ‘?’ . All other states result in zero reward.	82
3.3	Fixed $\pi_0$ baselines. Results are averaged over 20 seeds, with the shaded region denoting one standard deviation.	82
3.4	Learned $\pi_0$ baselines. Results are averaged over 20 seeds, with the shaded region denoting one standard deviation.	82
3.5	Learned default policies in states $s_1$ and $s_7$ after five tasks. In the simplex for $s_7$ , the marker for TVPO is behind the markers for the other methods.	83
3.6	Delayed training of $\pi_0$ improves performance.	84
4.1	(A) Illustration of a generative model of optimal policy parameters. $\hat{w}_1 = (1 - \beta)w_{11}$ shrinks towards the origin, growing closer to $\bar{w}_1$ than $w_{11}$ . (B) Sparsity-inducing priors over $\beta$ .	117
4.2	MDL-C rapidly adapts to new goal locations (top row) and rule changes (bottom row). All curves represent averages taken over 10 random seeds, with the shading indicating standard error.	121
4.3	MDL-C improves both sequential and parallel learning in continuous control tasks. All curves represent averages taken over 8 random seeds, with the shading indicating standard error. In (b), insets show the improvement of MDL-C as $k$ increases, and in (d), solid curves represent averages over each feature within a category.	123

4.4	To test the effect of information asymmetry on its on performance, we trained a variant of MANUALIA in which we withheld the input features that MDL-C learned to gate out (Fig. 4.3) in addition to the task ID feature. We call this modified method MANUALIA+. Average performance is plotted above over 10 seeds, with the shading representing one unit of standard error. We can see that while MANUALIA+ narrowly outperforms MANUALIA, the performance gains of MDL-C can't solely be ascribed to effective information asymmetry. . . . .	124
4.5	Heatmaps of $\text{KL}[\pi_\theta(\cdot s), \pi_w(\cdot s)] \forall s \in \mathcal{S}$ for RPO and $\text{KL}[\pi_\theta(\cdot s), \pi_{\bar{w}}(\cdot s)] \forall s \in \mathcal{S}$ , where $\bar{w} = \mathbb{E}_\nu w$ for MDL-C, averaged over all possible goal states. The RPO default policy nearly perfectly matches the control policy, while the MDL-C default policy diverges most strongly from the control policy at the doorways. This is because the direction chosen by the policy in the doorways is highly goal-dependent. Because the MDL-C default policy learns to ignore the goal feature, it's roughly uniform in the doorways, whereas the control policy is highly deterministic, having access to the goal feature. . . . .	152
4.6	Without a sparse prior, RPO does not learn to ignore spurious input features. . . . .	153
4.7	MDL-C's learned $\alpha$ s in the DMC sequential setting. Because $\alpha$ tends to decay, the control policy is able to specialize to the current task later in training. Results averaged over eight random seeds; error shading denotes standard error. . . . .	153
4.8	Test reward on each individual task in the walker domain over the course of parallel task training. Average performance is plotted above over 10 seeds, with the shading representing one unit of standard error. We can see the biggest performance difference on walker, run, the most challenging task. . . . .	153

4.9	Test reward on each individual task in the cartpole domain over the course of parallel task training. Average performance is plotted above over 10 seeds, with the shading representing one unit of standard error. Interestingly, unlike in the sequential learning setting, joint training seems to impede performance on <code>swingup_sparse</code> , with no method succeeding.	154
-----	--	-----

5.1	<p><b>A.</b> Ciaramelli (2008) reported that damage to another (orbitofrontal) region of PFC impaired navigation to novel goals, both in the laboratory and an ecological study. In unsuccessful trials patients frequently navigated to familiar goal locations. Performance improved when patients were given frequent reminders of the goal or were asked to verbally rehearse the goal, but not when the goal reminder was replaced by an uninformative stimulus (<i>Warning</i>). <b>B.</b> In a modified navigation task only two goals were cued, one (blue <math>G</math>) occurring more frequently during training than the other (red <math>G</math>). When the infrequent goal is cued at test, the intact MDL-C agent navigates successfully to it from any start state (see blue example trajectories). When <math>RNN_{\pi}</math> is ablated, the agent ignores the instruction cue and navigates to the more frequent goal (pink trajectories). See Methods for simulation details. <b>C.</b> By inserting a gating layer over input features within <math>RNN_{\pi_0}</math> (see Methods), we can directly read out which information is processed by that pathway. The plot shows attention weights for the three input features in the navigation task referenced in Figure 1. Over the course of the initial training block, <math>RNN_{\pi_0}</math> learns to ignore the current goal cue.</p>	159
-----	--	-----

5.2 **A.** Policies for  $RNN_{\pi}$  (top) and  $RNN_{\pi_0}$  (bottom) for the stimuli shown, in word-reading ( $WR$ ) and color-naming ( $CN$ ) contexts. Response probabilities are shown for the response *red*, complementary to (unshown) probabilities for the alternative *blue* response. **B.** When the MDL-C agent is trained on the Stroop task (see Methods),  $RNN_{\pi_0}$  learns to ignore both the task cue and the stimulus color, attending only to word identity. **C.** Left: KL divergence between  $\pi$  and  $\pi_0$  for the four trial types shown in panel A. Right: Corresponding reaction times (see Methods). **D.** When trained on the Stroop task and then given a choice between blocks of color-naming trials that involve either high or low proportions of incongruent stimuli (see Methods), the MDL-C agent displays a preference for less frequent incongruence, paralleling the demand-avoidance effect seen in human decision making. . . . . 160

5.3 **A.** Structure of the two-step task as introduced by ?. Choice occurs at Stage 1. The value of  $p$  varies over time, and so must be inferred by the participant. Following subsequent research, the version employed in our experiments additionally included explicitly cued reversals in the structure of transitions from Stage 1 to Stage 2. See Methods for full details. **B.** Classical behavioral signatures of model-free (left) and model-based (center) performance in the two-step task. Adapted from Miller et al. (2016a), the plots show logistic regression weights quantifying the influence of two factors on the probability of repeating on the index trial the same first-stage action selected on the previous trial: (1) whether reward was received or omitted on the previous trial, and (2) whether the previous trial featured a transition from stage 1 to 2 that was high-probability (*common*) or low (*uncommon*). The right panel shows a hybrid pattern, similar to that reported in the classic study by (?). **C.** Left: Two-step behavior of MDL-C, reflecting policy  $\pi$ . Right: Influence of the past on policy  $\pi_0$ . **D.** Same as Panel D but with different weighting of terms in the MDL-C objective (see Methods and compare panel C, right). . . . . 163

5.4 **A.** Logistic regression weights showing the influence on the current action of reward contingent on choice (reward seeking), previous choices (perseveration), and reward independent of choice (main effect of outcome) of MDL-C and a standard RL agent on the drifting two-armed bandit task from Miller et al. (2018). MDL-C displays a stronger tendency towards perseveration, reminiscent of rats trained on the same task. **B.** Left: Simulation of contingency degradation from Miller et al. (2019). The longer the training phase (x axis), the longer lever-pressing persists after reward is discontinued (red). Right: Corresponding behavior from MDL-C, also showing the effect of ablating  $\pi_0$ . . . . . 165

5.5 **A.** Heuristic one-reason decision making (left) and richer compensatory decision making (right) in a multi-attribute choice task, from Binz et al. (2022). Gini coefficients, on the y axis, capture the degree to which decisions depend on one feature (higher values, with asymptotic maximum of one) versus all features evenly (zero), with references for one-reason decision making (*single cue*) and a fully compensatory strategy (*equal weighting*) indicated. Data points for each trial correspond to observations from separate simulation runs. Human participants in the study displayed both patterns of behavior, depending on the task conditions. **B.** Behavior of MDL-C in the task from Binz et al. (2022), under conditions where human participants displayed one-reason decision making. **C.** Behavior of  $\pi_0$  (left) and  $\pi$  (right) when the KL penalty for divergence between the two policies is reduced (see Methods). **D.** In the simulation from panel C, the divergence between policies is increased when the agent emits a non-heuristic decision. . . . . 167

§.6	<b>A.</b> Top: Behavioral data from the modified Stroop task studied by MacLeod and Dunbar (1988). Early in training, shape-naming responses were both slower than color-naming responses and more affected by stimulus congruence, consistent with shape-naming being the relatively 'controlled' response and color-naming relatively 'automatic.' With extensive training, the pattern flipped, with shape-naming becoming faster than color-naming and less affected by stimulus congruence. Bottom: Under training conditions mimicking the experimental study, MDL-C displayed a similar pattern of behavior, with a significant main effect of task and a significant interaction between task and trial-type ( $p \leq 0.05$ ) at both 0 trials and 44,000 trials. <b>B.</b> Zero-shot Stroop performance in MDL-C and an unregularized baseline model (see Methods). Top: Color-naming accuracy on incongruent Stroop stimuli, after training only with neutral stimuli (see main text and Methods). Bottom: KL divergence between action probability distributions under two conditions, (1) presentation of incongruent Stroop stimuli, and (2) presentation of Stroop stimuli with the word identity input masked out. MDL-C shows significantly lower divergence, indicating that the control policy attends less to the task-irrelevant factor — i.e., MDL-C is more robust to distractors — despite never having been trained on incongruent stimuli. . . . .	169
§.7	Two-step results from full hyperparameter sweep described in Methods, with $\alpha = 0.05$ . Format as in Fig. §.3 in the main text. . . . .	189
§.8	Two-step results from full hyperparameter sweep described in Methods, with $\alpha = 0.1$ . The boxed plot appears in Fig. §.3 in the main text. Format as in Fig. §.3 in the main text. . . . .	190
§.9	Two-step results from full hyperparameter sweep described in Methods, with $\alpha = 0.2$ . The boxed plot appears in Fig. §.3E in the main text. Format as in Fig. §.3 in the main text. . . . .	191

6.1	<b>The FR is higher for shorter paths.</b> (a-c) A 2D gridworld and fixed policies. (d) The FR from $s_0$ to $s_g$ is higher for $\pi_2$ , but the SR is lower. (e) SR-GPI with the SR picks $\pi_1$ , while FR-GPI selects $\pi_2$ .	200
6.2	<b>The FF facilitates accurate policy evaluation and selection.</b> Shading denotes 1 SE over 20 seeds.	203
6.3	<b>The FR enables efficient planning.</b> (a-d) A 2D gridworld with start ( $s_0$ ) and goal( $s_g$ ) states, along with three fixed policies. (e) GPI follows $\pi_1$ . (f) Planning with the FR enables the construction of a shorter path.	204
6.4	<b>APF accelerates convergence in robotic reaching.</b> Shading denotes 1 SE over 10 seeds.	205
6.5	<b>FRP interpolates between GPI and model-based DP.</b> Shading represents 1 SE.	208
6.6	<b>FRP induces realistic escape behavior.</b>	209
6.7	<b>FF and SF learning curves for continuous MountainCar.</b> Results averaged over 20 runs. Shading represents one standard deviation.	213
6.8	<b>Tabular environments for exploration.</b> The tuples marking each transition denote (action id(s); probability; reward). In RiverSwim, the agent starts in either state 1 or state 2 with equal probability, while for SixArms the agent always starts in state 0.	213
6.9	<b>Network architecture for DQN + FF and DQN + SF</b>	214
6.10	<b>Exploration with function approximation.</b> (Top left) Visualization of the DeepSea environment, credit to <a href="#">Osband et al. (2020)</a> . (Top right) DQN + FF significantly outperforms standard DQN and DQN + SF. (Bottom) Different runs across problem sizes.	216
6.11	<b>The FF is robust to feature dimensionality.</b> FF and SF representation strengths for difference feature dimensionalities between the start and goal locations for an example goal in continuous MountainCar. The vertical dashed line marks the power of the optimal policy. We can see that for all but the coarsest feature representation, the FF is highest for the policy closest to the optimal.	219

6.12	<b>FOURROOMS learning curves.</b> FOURROOMS base policies learning curves (average L2 norm of TD errors over 10 runs; shaded area is one standard deviation); top row is for FRs, bottom is for SRs. . . . .	223
6.13	<b>Implicit planning output.</b> (Left) The planning policies $\pi^F(s)$ that the agent will elect to follow in each state en route to the goal (see Fig. 6.3(a)). Arrows denote the action taken by the chosen policy in each state. (Middle) The (row, column) subgoals for each state $s^F(s)$ . (Right) The state space $\mathcal{S}$ , for reference.	223
6.14	<b>Exploration and escape</b> (a) A sample trajectory from the “exploration phase” starting from the shelter. (b) Because the agent starts from the shelter during exploration, the first time it is tested starting from the top of the grid, its FR for the down policy for that state is still at initialization. (c) After updating its FR during testing and further exploration, the FR for the down policy from the start state is accurate, stopping at the barrier. (d) We can see that if we were to use the SR instead, the value in the state above the wall would accumulate when it gets stuck. . . . .	223
6.15	<b>Escape learning curves.</b> Learning curves (norms of TD errors) for the first exploration phase, the first escape trial, and the second exploration phase for the “down” policy. The vertical dotted lines in the escape trial mark the time step at which the agent encounters the barrier. This causes a temporary jump in the TD errors, as representation learning did not reflect the wall at this point. The top row consists of FR results and the bottom row is from SRs, averaged over 10 runs. The shading represents one standard deviation. . . . .	225
6.16	<b>An SR cannot effectively escape under the same conditions as an FR agent.</b> . . . . .	225



6.17	<b>SR vs. FR visualization</b> The SRs and FRs from the start state for the policies in Fig. Fig. 6.3. For the SRs (Fig. 6.17, top row), we can see that states that are revisited (or in which the policy simply stays) are more highly weighted, while for the FRs (Fig. 6.17, bottom row), the magnitude of $F(s_0, s')$ is higher for states $s'$ that are closer along the path taken by the policy. . . . .	233
7.1	Diminishing rewards. . . . .	234
7.2	<b>The <math>\lambda</math>R interpolates between the FR and the SR.</b> We visualize the $\lambda$ R of the bottom left state for the depicted policy for $\lambda \in \{0.0, 0.5, 1.0\}$ . . . . .	239
7.3	<b>The <math>\lambda</math>R is required for accurate policy evaluation.</b> Policy evaluation of the policy depicted in Fig. 7.2 using dynamic programming, tabular TD learning, and $\lambda$ F TD learning produces the most accurate value estimates when using the $\lambda$ R with $\lambda = \lambda_{true}$ . Results are averaged over three random seeds. Shading indicates standard error. . . . .	242
7.4	<b>The <math>\lambda</math>R is required for strong performance.</b> We apply a tabular $Q$ -learning-style algorithm and deep actor-critic algorithm to policy optimization in the TwoRooms domain. The blue locations indicate reward, and the black triangle shows the agent's position. Results are averaged over three random seeds. Shading indicates standard error. . . . .	242
7.5	<b>Tabular GPI.</b> (Left) Average returns obtained by agents performing GPE+GPI using $\lambda$ R with $\lambda \in \{0.0, 0.5, 1.0\}$ over 50 episodes. Error bars indicate standard error. (Right) Sample trajectories. Agents with $\lambda$ set too high overstay in rewarding states, and those with $\lambda$ too low leave too early. . . . .	244
7.6	<b>Pixel-Based GPI.</b> Performance is strongest for agents using the correct $\lambda = 0.5$ . PCA on the learned features in each underlying environment state shows that the $\lambda$ Fs capture the value-conditioned structure of the environment. . . . .	245

7.7	<b><math>\lambda</math>O trained via FB. a)</b> $\lambda$ values of two agents in FourRooms, one which learns $\lambda$ and one which does not. <b>b)</b> Performance of the two agents from (a). Learning $\lambda$ improves performance. <b>c)</b> Reward structure and starting state of the asymmetric environment. <b>d)</b> Trajectory of an agent with $\lambda = 1$ . The optimal strategy is to reach the high reward state and exploit it <i>ad infinitum</i> . <b>e)</b> Trajectory of an agent with $\lambda = 0.1$ . The optimal strategy is to exploit each reward state for a few time steps before moving to the next reward state. . . . .	247
7.8	<b>Learning curves for <math>\lambda</math>F policy evaluation.</b> Results are averaged over three runs, with shading indicating one unit of standard error. . . . .	267
7.9	<b>A simple grid and several policies.</b> . . . . .	270
7.II	<b>Convergence of dynamic programming on FourRooms with and without stochastic transitions.</b> . . . . .	270
7.12	<b>GPI with noisy transitions in FourRooms.</b> To verify that performance was maintained even with stochastic transitions, we added a 20% probability that a given action would result in a random transition to neighboring state. Results are consistent with Fig. 7.5, indicating the having the correct value of $\lambda$ produces better performance. . . . .	270
7.10	<b>Visualizing the SR, the <math>\lambda</math>R and the FR.</b> We can see that the $\Phi_1^\pi$ is equivalent to the SR and $\Phi_0^\pi$ is equivalent to the FR, with intermediate values of $\lambda$ providing a smooth transition between the two. . . . .	271
7.13	<b>A 3-state toy environment.</b> . . . . .	272
7.14	<b>Performance of the <math>\lambda</math>O-FB with two values of <math>\lambda</math>.</b> Results averaged over six seeds and 10 episodes per seed. Error bars indicate standard error. . . . .	274

7.15	<b>Analysis of MVT-like behavior of <math>\lambda</math>O-FB.</b> a) Three environments with equal start state and structure but different distances between reward states. b) Difference between the agent's leave times and MVT-predicted leave times for $\gamma = 0.99$ , with discounting taken into account. The agent on average behaves similar to the discounted MVT. c) Difference between the agent's leave times and MVT-predicted leave times for $\gamma = 1.0$ , i.e., with no discounting taken into account. The agent on average behaves similar to the MVT. Results for (b) and (c) are averaged over three seeds. Error bars indicate standard error.	276
7.16	<b><math>\lambda</math>-SAC (1 critic) matches the performance of SAC (2 critics) by adapting <math>\lambda</math> online.</b>	279
7.17	<b><math>\lambda</math>-SAC matches performance while saving in computational cost.</b>	279
7.18	<b>Visualizing three different replenishment schemes.</b> For all schemes, $\bar{r}(s) = 1$ and visits to $s$ are at $t = 2, 5$ . (Left) The time elapsed reward with $\lambda = 0.5$ ; (Middle) The eligibility trace reward with $\lambda_r = \lambda_d = 0.5$ ; (Right) The total time reward with $\lambda_d = 0.5, \lambda_r = 0.9$ .	281
7.19	<b>Performance on TwoRooms with replenishing rewards.</b> Return is averaged over five runs, with shading indicating one unit of standard error.	282

# List of Tables

4.1	Correspondence between $p(z^2)$ and $p(\beta)$ . . . . .	132
4.2	DM control suite hyperparameters, used for all experiments. *In the parallel setting, $\alpha$ was simply set to 0.1 for methods with learned default policies. . . . .	151
4.3	FourRooms: Average cumulative regret across 8 random seeds in phase 2 of the goal change and contingency change experiments for each method. $\pm$ values are standard error. . . . .	151
4.4	DM Control Suite, Sequential: Average cumulative regret across 8 random seeds in the sequential setting. $\pm$ values are standard error. . . . .	152
4.5	DM Control Suite, Parallel: Average cumulative regret across 8 random seeds in the parallel task setting. $\pm$ values are standard error. . . . .	152
5.1	Hyperparameters for each task. *The Two-Step task settings are described in greater detail below. . . . .	182
6.1	Exploration results. $\pm$ values denote 1 SE across 100 trials. . . . .	202
6.2	RIVERSWIM-N results. $\pm$ values denote 1 SE across 100 trials. . . . .	214
6.3	Hyperparameter settings for the DEEPSEA experiment . . . . .	217
6.4	Overview of basic points of comparison between the FR/FF, SR/SF (Dayan, 1993; Barreto et al., 2017), DDL (Hartikainen et al., 2020), TDMs (Pong et al., 2018), and DG (Kaelbling, 1993). . . . .	230
7.1	<b><math>\lambda</math>O-FB hyperparameters.</b> . . . .	275
7.2	Hyperparameters for SAC Mujoco experiments. *Only applicable to $\lambda$ -SAC . . . . .	279

## UCL Research Paper Declaration Form: referencing the doctoral candidate's own published work(s) (I)

1. **1. For a research manuscript that has already been published** (if not yet published, please skip to section 2):

- (a) **What is the title of the manuscript?** Towards an Understanding of Default Policies in Multitask Policy Optimization
- (b) **Please include a link to or doi for the work:** <https://proceedings.mlr.press/v151/moskovitz22a.html>
- (c) **Where was the work published?** International Conference on Artificial Intelligence and Statistics
- (d) **Who published the work?** Proceedings of Machine Learning Research (PMLR)
- (e) **When was the work published?** 2022
- (f) **List the manuscript's authors in the order they appear on the publication:** Ted Moskovitz, Michael Arbel, Jack Parker-Holder, Aldo Pacchiano
- (g) **Was the work peer reviewed?** Yes
- (h) **Have you retained the copyright?** Yes
- (i) **Was an earlier form of the manuscript uploaded to a preprint server (e.g. medRxiv)? If 'Yes', please give a link or doi** Yes: <https://arxiv.org/abs/2111.02994>

If 'No', please seek permission from the relevant publisher and check the box next to the below statement:

☐ *I acknowledge permission of the publisher named under 1d to include in this thesis portions of the publication named as included in 1c.*

2. **For a research manuscript prepared for publication but that has not yet been published** (if already published, please skip to section 3):

- (a) **What is the current title of the manuscript?**

- (b) **Has the manuscript been uploaded to a preprint server 'e.g. medRxiv'?**  
**If 'Yes', please give a link or doi:**
- (c) **Where is the work intended to be published?**
- (d) **List the manuscript's authors in the intended authorship order:**
- (e) **Stage of publication:**
3. **For multi-authored work, please give a statement of contribution covering all authors** (if single-author, please skip to section 4):
- Ted Moskovitz: conceived of project, ran all experiments, obtained majority of theoretical results, wrote the paper
  - Michael Arbel: contributed to theoretical results, paper writing
  - Jack Parker-Holder: provided code template for binary tree environment, contributed to paper writing
  - Aldo Pacchiano: contributed to theoretical results, paper writing, project guidance
4. **In which chapter(s) of your thesis can this material be found?** Chapter 3

**e-Signatures confirming that the information above is accurate** (this form should be co-signed by the supervisor/ senior author unless this is not appropriate, e.g. if the paper was a single-author work):

**Candidate:** Theodore Moskovitz

**Date:** April 8, 2024

**Supervisor/Senior Author signature** (where appropriate): 

**Date:** April 23, 2024

## UCL Research Paper Declaration Form: referencing the doctoral candidate's own published work(s) (2)

1. **1. For a research manuscript that has already been published** (if not yet published, please skip to section 2):

- (a) **What is the title of the manuscript?** Minimum Description Length Control
- (b) **Please include a link to or doi for the work:** <https://openreview.net/forum?id=oX3tGygjW1q>
- (c) **Where was the work published?** International Conference on Learning Representations
- (d) **Who published the work?** International Conference on Learning Representations
- (e) **When was the work published?** 2023
- (f) **List the manuscript's authors in the order they appear on the publication:** Ted Moskovitz, Ta-Chu Kao, Maneesh Sahani, Matthew Botvinick
- (g) **Was the work peer reviewed?** Yes
- (h) **Have you retained the copyright?** Yes
- (i) **Was an earlier form of the manuscript uploaded to a preprint server (e.g. medRxiv)? If 'Yes', please give a link or doi** Yes: <https://arxiv.org/abs/2207.08258>

If 'No', please seek permission from the relevant publisher and check the box next to the below statement:

☐ *I acknowledge permission of the publisher named under 1d to include in this thesis portions of the publication named as included in 1c.*

2. **For a research manuscript prepared for publication but that has not yet been published** (if already published, please skip to section 3):

- (a) **What is the current title of the manuscript?**


- (b) **Has the manuscript been uploaded to a preprint server 'e.g. medRxiv'?**  
**If 'Yes', please give a link or doi:**
- (c) **Where is the work intended to be published?**
- (d) **List the manuscript's authors in the intended authorship order:**
- (e) **Stage of publication:**
3. **For multi-authored work, please give a statement of contribution covering all authors** (if single-author, please skip to section 4):
- Ted Moskovitz: conceived of study, co-developed MDL-C algorithm, implemented code and ran experiments, performed theoretical analysis of core method, wrote the paper
  - Calvin Kao: contributed to theoretical analysis, figure generation, and paper writing
  - Maneesh Sahani: contributed to project guidance, method analysis, and paper writing
  - Matt Botvinick: contributed to project guidance and paper writing, co-developed MDL-C algorithm

4. **In which chapter(s) of your thesis can this material be found?** Chapter 4

**e-Signatures confirming that the information above is accurate** (this form should be co-signed by the supervisor/ senior author unless this is not appropriate, e.g. if the paper was a single-author work):

**Candidate:** Ted Moskovitz

**Date:** April 8, 2024

**Supervisor/Senior Author signature** (where appropriate): 

**Date:** April 23, 2024



## UCL Research Paper Declaration Form: referencing the doctoral candidate's own published work(s) (3)

1. **1. For a research manuscript that has already been published** (if not yet published, please skip to section 2):

- (a) **What is the title of the manuscript?**
- (b) **Please include a link to or doi for the work:**
- (c) **Where was the work published?**
- (d) **Who published the work?**
- (e) **When was the work published?**
- (f) **List the manuscript's authors in the order they appear on the publication:**
- (g) **Was the work peer reviewed?**
- (h) **Have you retained the copyright?**
- (i) **Was an earlier form of the manuscript uploaded to a preprint server (e.g. medRxiv)? If 'Yes', please give a link or doi**

If 'No', please seek permission from the relevant publisher and check the box next to the below statement:

☐ *I acknowledge permission of the publisher named under 1d to include in this thesis portions of the publication named as included in 1c.*

2. **For a research manuscript prepared for publication but that has not yet been published** (if already published, please skip to section 3):


- (a) **What is the current title of the manuscript?** A Normative Theory of Dual Process Control
- (b) **Has the manuscript been uploaded to a preprint server 'e.g. medRxiv'?**  
If 'Yes', please please give a link or doi: Yes: <https://arxiv.org/abs/2211.07036>
- (c) **Where is the work intended to be published?** PLOS Computational Biology

- (d) **List the manuscript's authors in the intended authorship order:** Ted Moskovitz, Kevin Miller, Maneesh Sahani, Matt Botvinick
- (e) **Stage of publication:** Under Review
3. **For multi-authored work, please give a statement of contribution covering all authors** (if single-author, please skip to section 4):
- Ted Moskovitz: Co-conceived of work, implemented code and ran all experiments, co-wrote paper
  - Kevin Miller: Helped select experimental simulations, guided project development, contributed to paper writing
  - Maneesh Sahani: Helped guide project and paper writing
  - Matt Botvinick: Co-conceived of work, guided project development, co-wrote paper
4. **In which chapter(s) of your thesis can this material be found?** Chapter 5

**e-Signatures confirming that the information above is accurate** (this form should be co-signed by the supervisor/ senior author unless this is not appropriate, e.g. if the paper was a single-author work):

**Candidate:** Ted Moskovitz

**Date:** April 8, 2024

**Supervisor/Senior Author signature** (where appropriate): 

**Date:** April 23, 2024

## UCL Research Paper Declaration Form: referencing the doctoral candidate's own published work(s) (4)

1. **1. For a research manuscript that has already been published** (if not yet published, please skip to section 2):

- (a) **What is the title of the manuscript?** A First Occupancy Representation for Reinforcement Learning
- (b) **Please include a link to or doi for the work:** <https://openreview.net/forum?id=JBAZe2yN6Ub>
- (c) **Where was the work published?** The International Conference of Learning Representations
- (d) **Who published the work?** The International Conference of Learning Representations
- (e) **When was the work published?** 2022
- (f) **List the manuscript's authors in the order they appear on the publication:** Ted Moskowitz, Spencer Wilson, Maneesh Sahani
- (g) **Was the work peer reviewed?** Yes
- (h) **Have you retained the copyright?** Yes
- (i) **Was an earlier form of the manuscript uploaded to a preprint server (e.g. medRxiv)?** If 'Yes', please give a link or doi Yes: <https://arxiv.org/abs/2109.13863>

If 'No', please seek permission from the relevant publisher and check the box next to the below statement:

☐ *I acknowledge permission of the publisher named under 1d to include in this thesis portions of the publication named as included in 1c.*

2. **For a research manuscript prepared for publication but that has not yet been published** (if already published, please skip to section 3):

- (a) **What is the current title of the manuscript?**

- (b) **Has the manuscript been uploaded to a preprint server 'e.g. medRxiv'?**  
**If 'Yes', please give a link or doi:**
- (c) **Where is the work intended to be published?**
- (d) **List the manuscript's authors in the intended authorship order:**
- (e) **Stage of publication:**

3. **For multi-authored work, please give a statement of contribution covering all authors** (if single-author, please skip to section 4):


- Ted Moskovitz: co-conceived of project, derived core theoretical results, coded and ran all experiments, wrote the paper
- Spencer Wilson: assisted with figure creation, paper writing
- Maneesh Sahani: co-conceived of project, contributed to algorithm development, theoretical results, project direction, paper writing

4. **In which chapter(s) of your thesis can this material be found?** Chapter 6

**e-Signatures confirming that the information above is accurate** (this form should be co-signed by the supervisor/ senior author unless this is not appropriate, e.g. if the paper was a single-author work):

**Candidate:** Ted Moskovitz

**Date:** April 8, 2024

**Supervisor/Senior Author signature** (where appropriate): 

**Date:** April 23, 2024

## UCL Research Paper Declaration Form: referencing the doctoral candidate's own published work(s) (5)

1. **For a research manuscript that has already been published** (if not yet published, please skip to section 2):

- (a) **What is the title of the manuscript?** A State Representation for Diminishing Rewards
- (b) **Please include a link to or doi for the work:** <https://openreview.net/forum?id=7Uix1eQZ8z>
- (c) **Where was the work published?** Neural Information Processing Systems
- (d) **Who published the work?** Curran Associates
- (e) **When was the work published?** 2023
- (f) **List the manuscript's authors in the order they appear on the publication:** Ted Moskovitz, Samo Hromadka, Ahmed Touati, Diana Borsa, Maneesh Sahani
- (g) **Was the work peer reviewed?** Yes
- (h) **Have you retained the copyright?** Yes
- (i) **Was an earlier form of the manuscript uploaded to a preprint server (e.g. medRxiv)? If 'Yes', please give a link or doi** Yes: <https://arxiv.org/abs/2309.03710>

If 'No', please seek permission from the relevant publisher and check the box next to the below statement:

☐ *I acknowledge permission of the publisher named under 1d to include in this thesis portions of the publication named as included in 1c.*

2. **For a research manuscript prepared for publication but that has not yet been published** (if already published, please skip to section 3):

- (a) **What is the current title of the manuscript?**
- (b) **Has the manuscript been uploaded to a preprint server 'e.g. medRxiv'? If 'Yes', please give a link or doi:**

(c) **Where is the work intended to be published?**

(d) **List the manuscript's authors in the intended authorship order:**

(e) **Stage of publication:**

3. **For multi-authored work, please give a statement of contribution covering all authors** (if single-author, please skip to section 4):


- Ted Moskovitz: conceived of project, contributed to theoretical results, implemented major experiments, wrote the paper
- Samo Hromadka: contributed to theoretical results, implemented  $\lambda$  operator experiment, assisted with paper writing
- Ahmed Touati: contributed to theoretical results, paper writing
- Diana Borsa: helped guide project, contributed to paper writing
- Maneesh Sahani: guided project, contributed to theoretical results, paper writing

4. **In which chapter(s) of your thesis can this material be found?** Chapter 7

**e-Signatures confirming that the information above is accurate** (this form should be co-signed by the supervisor/ senior author unless this is not appropriate, e.g. if the paper was a single-author work):

**Candidate:** Ted Moskovitz

**Date:** April 8, 2024

**Supervisor/Senior Author signature** (where appropriate): 

**Date:** April 23, 2024

# Introduction

## Chapter 1

# Overview

A longstanding goal of artificial intelligence is to build generalist agents: those which can learn and adapt to accomplish a variety of goals in their environments. This problem is especially challenging in part because it demands rational decisions from bounded decision-makers. In other words, any agent must decide on its next action given only finite computational resources. If that were not the case, it could simply build a perfect simulator of the world and examine the outcome of every available course of action to inform its choice. Alongside this effort, understanding the flexibility of human and animal intelligence is also a key object of study in neuroscience.

Although their end goals differ, both disciplines are interested in understanding this aspect of intelligence:

*How can bounded agents learn and act efficiently in a seemingly unbounded world?*

The answer to this question, and the subject of this thesis, is that doing so requires that the environment and the goals which the agent seeks to accomplish must contain patterns—that is, they must have some form of underlying *structure*. For example, the effects of our actions on the world around us are usually fairly consistent from day to day. Such structure allows decision-makers to make predictions and effectively shrink the problems they face so that they're solvable. While structure in the world represents opportunity for an agent to problem-solve, the agent must still be able to actually use this structure to influence its decision-making.

Accordingly, this thesis will discuss work which has centered around two different forms of structure which may be shared across the tasks animals and agents must perform



and the ways in which it can be leveraged:

1. Part I: Shared behaviors (Chapters 3 to 5): Often, different tasks we perform require us to repeat the same actions or sequences of actions. For example, a jogger might navigate to a different landmark within the local park each day, but they might repeat the same route everyday to get to the park itself. The work in this section studies how such consistently useful behaviors can be leveraged to accelerate learning on new tasks (e.g., run to new locations in the park). Results demonstrate that a simple computational framework built around this intuition can account for many behavioral results from cognitive neuroscience in cognitive control, reward-based learning, and judgment and decision-making. This section draws from the following works: Moskowitz et al. (2022a, 2023a, 2022b).
2. Part II: Shared dynamics (Chapters 6 and 7): The local dynamics of the world around us are usually relatively stable<sup>1</sup>. Walking west down Howland Street from Tottenham Court Road will bring one to the Sainsbury Wellcome Centre, and is unlikely to one day suddenly start teleporting PhD students to midtown Manhattan instead. Understanding of this consistency allows us to compose previously learned behaviors more efficiently, because our beliefs about the effects of those behaviors on the world can generally be relied upon. The work described in this section draws from Moskowitz et al. (2022c, 2024).

Reinforcement learning (RL) provides a powerful, general framework for reasoning about sequential decision-making, and forms the foundation for the work described. Relevant background is provided in Chapter 2. This thesis concludes with a discussion and speculation regarding future work (Chapter 8).

---

<sup>1</sup>This can be understood as the *perceived* dynamics of the world relevant to decision-making, and not, strictly speaking, the laws of physics themselves.

## Chapter 2

# Reinforcement Learning

In reinforcement learning (RL), an agent learns how to act within its environment over multiple time steps in order to maximize its reward on a given task or tasks. To formalize this problem setting, we require a mathematical specification of both the environment and the agent. Once these are established, we'll cover several foundational algorithms for solving RL problems which form the basis for many of the original contributions of this thesis. Throughout this and subsequent sections, when the dimensionality of a variable (i.e., whether it is a scalar, a vector, a matrix, or a tensor) is unspecified, the default will be to use scalar notation (e.g.,  $x$ ). For a much more comprehensive treatment of RL, see [Sutton and Barto \(2018a\)](#); [Agarwal et al. \(2021\)](#).

### 2.1 Defining the Environment

To facilitate efficient learning, there are several common assumptions made about the structure of the world and the process which generates the observations received by the agent. Chief among these is that the world can be divided into distinct states  $\mathcal{S} = \{s\}$  such that the current state  $s_t$  at any given time  $t$ , in conjunction with the agent's action  $a_t$ —chosen from within a set of available actions  $\mathcal{A}$ —comprises a sufficient statistic for the following state. This is the *Markov assumption*: that the transition dynamics which determine the next state are independent of previous interactions with the environment given the current state and action. In other words, there exists a transition kernel  $P : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{P}(\mathcal{S})$  such that  $P(s_{t+1}|s_t, a_t) = P(s_{t+1}|h_t)$ , where  $h_t \triangleq (s_0, a_0, \dots, s_t, a_t)$  is the *history* up to time  $t$ . The agent's method for choosing

an action is called its *policy*, denoted by  $\pi$ . In a Markovian environment, the policy need not depend on the full history, and can instead be conditioned only on the current state  $\pi : \mathcal{S} \rightarrow \mathcal{P}(\mathcal{A})$ . We call this type of policy *stationary*. When the state and action spaces are finite, we can describe the expected transition probabilities under a policy  $\pi$  using a  $|\mathcal{S}| \times |\mathcal{S}|$  matrix  $P^\pi$  such that  $P_{s,s'}^\pi = p^\pi(s'|s) \triangleq \sum_{a \in \mathcal{A}} P(s'|s, a)\pi(a|s)$ . Note that we have allowed the policy to depend directly on the environment state. This is valid if we make a second common assumption: that the underlying state of the world is *fully observable*. In this case, the agent directly observes the environment state, and the model of the world is called a *controlled Markov process* (CMP; [Abel et al., 2021](#)), defined formally as a tuple  $\mathcal{E} \triangleq (\mathcal{S}, \mathcal{A}, P)$ . If this assumption is not made, we consider the problem to be *partially observable*. In this case, the state is not directly revealed to the agent and instead there is some unknown distribution over possible observations for each state. We can define a *partially observable CMP* (POCMP) as a tuple  $\mathcal{E} \triangleq (\mathcal{S}, \mathcal{A}, P, \mathcal{O}, f)$  where  $\mathcal{S}, \mathcal{A}$ , and  $P$  are as defined for a CMP, with  $\mathcal{O}$  the observation space and  $f : \mathcal{S} \rightarrow \mathcal{P}(\mathcal{O})$  the observation function. The transitions and rewards are *not* Markov in the observations in this setting, making it substantially more challenging. Unless otherwise noted, we will default to the fully observable setting.

### 2.1.1 Tasks

Now that we’ve defined the environment, we can specify what we mean by the concept of a “task.” Informally, we will call a task a pairing of an environment with a starting state or states, a performance criterion (delineating behaviors which mark success and failure), and an effective time span over which the task must be completed.

An easily overlooked but important aspect of a task is its starting point: the initial state or states in which the agent must begin to make decisions. We model this property with an *initial state distribution*  $\rho \in \mathcal{P}(\mathcal{S})$ , so that the agent begins a given task in a state  $s_0 \sim \rho(\cdot)$ . In the case that there is a single initial state, for example,  $\rho$  is simply a Dirac delta distribution. The significance of this distribution is in large-part dependent on the manner in which the agent’s experience is modeled. If the agent’s experience is treated as one continuous stream of interaction with its environment—that is,  $s_0$  is sampled from  $\rho$  exactly once and then the agent is left to accumulate as much reward as it can forever—

then the importance of  $\rho$  is directly tied to the connectedness of the environment. If there is some policy for which the induced Markov chain of environment interactions is ergodic, then all states can be visited infinitely often and the initial state  $s_0$  is less significant. If this is not the case, then  $s_0$  determines which subset of the environment the agent is able to reach, which often has a significant impact on its ability to accumulate reward. However, a much more common modeling assumption is that the agent's experience while mastering a task is broken up into discrete chunks termed *episodes*. At the beginning of each episode, a new initial state is drawn from  $\rho$ , and the agent is allowed a certain number of time steps (which may be infinite) to accomplish its task, after which the episode ends and the agent resets to a new initial state drawn from  $\rho$ .

The performance criterion in all RL problems is the reward function  $r$ . In the general case, the reward is given as  $r : \mathcal{H} \rightarrow \mathbb{R}$ , a mapping from histories to scalar values—higher values are better, and lower are worse. However, if the Markov assumption holds, we can define  $r$  over some subset of  $\{s_t, a_t, s_{t+1}\}$  (i.e., we have either  $r(s_t)$ ,  $r(s_t, a_t)$ , or  $r(s_t, a_t, s_{t+1})$ ). For simplicity, this thesis will default to using  $r(s_t, a_t)$  unless otherwise noted, and will frequently abbreviate the reward at time  $t$  to  $r_t$ . The agent's goal is to maximize the amount of reward it collects over its lifespan (or the duration of the task).

The time span—or *horizon*—over which the task must be completed, which we denote by  $H$ , can be either finite or infinite. In the finite horizon case, the objective that the agent is trying to maximize is simply the expected total reward, which is termed the *value*:

$$V^\pi \triangleq \mathbb{E}_{\pi, P, \rho} \left[ \sum_{t=0}^{H-1} r(s_t, a_t) \right] = \mathbb{E}_{s_0 \sim \rho} \mathbb{E}_{\pi, P} \left[ \sum_{t=0}^{H-1} r(s_t, a_t) \mid s_0 \right]. \quad (2.1)$$

Generalizing, the mapping from states to the cumulative reward that the agent is expected to achieve is called the *value function*  $V : \mathcal{S} \rightarrow \mathbb{R}$ :

$$V^\pi(s) \triangleq \mathbb{E}_{\pi, P} \left[ \sum_{k=t}^{H-1} r(s_{t+k}, a_{t+k}) \mid s_t = s \right]. \quad (2.2)$$

We can also define the *action-value* or “Q” function:

$$Q^\pi(s, a) \triangleq \mathbb{E}_{\pi, P} \left[ \sum_{k=t}^{H-1} r(s_{t+k}, a_{t+k}) \mid s_t = s, a_t = a \right], \quad (2.3)$$

where  $V^\pi(s) = \mathbb{E}_{a \sim \pi(\cdot|s)} Q^\pi(s, a)$ . In the remainder of the text, we will frequently abbreviate  $\mathbb{E}_{\pi, P, \rho}$  to  $\mathbb{E}_\pi$  where the meaning is clear. The *return* of a trajectory  $\tau = (s_0, a_0, s_1, a_1, \dots, s_{H-1}, a_{H-1})$  is given by

$$R(\tau) \triangleq \sum_{t=0}^{H-1} r(s_t, a_t). \quad (2.4)$$

The value can then be written as the expected return under a fixed policy,  $V^\pi = \mathbb{E}_{\tau \sim \mathbb{P}^\pi(\cdot)} R(\tau)$ , where  $\mathbb{P}^\pi(\tau)$  is the probability of a trajectory, given in the Markovian case by

$$\mathbb{P}^\pi(\tau) = \rho(s_0) \prod_{t=0}^{H-1} \pi(a_t|s_t) P(s_{t+1}|s_t, a_t). \quad (2.5)$$

Another way to write the value is as the expected reward under the *cumulative, discounted state-action occupancy measure*  $d^\pi \in \mathcal{P}(\mathcal{S} \times \mathcal{A})$ . That is,  $V^\pi = \mathbb{E}_{s, a \sim d^\pi} r(s, a)$ , where

$$d^\pi(s) = \frac{1}{H} \sum_{t=0}^{H-1} \mathbb{P}^\pi(s_t = s), \quad d^\pi(s, a) = \pi(a|s) d^\pi(s). \quad (2.6)$$

For concision, from here onward we will refer to both  $d^\pi(s)$  and  $d^\pi(s, a)$  interchangeably as a policy’s *occupancy measure*, with the precise meaning implied by context. Overall, in the finite horizon case, a task is the combination of the initial state distribution, reward function, and the horizon, and is called a finite horizon *Markov decision process* (MDP [Puterman, 1994](#)), described by the tuple  $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, \rho, r, H)$ .

The infinite horizon case, however, presents potential problems because summing rewards over infinitely long trajectories can frequently lead to infinite values. In this case,

---

<sup>1</sup>The term “trajectory” is subtly different from “history” — the history refers to the entirety of the agent’s experience (e.g., across episodes), while a trajectory only consists of the current episode. In a continual/non-episodic setting, these terms mean the same thing.

a policy which earns the agent 1/10 reward units every step will have the same value as a policy which nets 100 reward units per step. One way to address this is to change the definition of the value so that it measures a policy's *average* reward per step:

$$V_{avg}^{\pi} = \lim_{H \rightarrow \infty} \frac{1}{H} \mathbb{E}_{\pi} \sum_{t=0}^{H-1} r(s_t, a_t). \quad (2.7)$$

This is known as an average-reward MDP [Sutton and Barto \(2018a\)](#). However, a more common approach is to make the agent “shortsighted.” Specifically, the agent is endowed with a temporal discount factor  $\gamma \in [0, 1)$  such that it attempts to maximize its *discounted return*, given by

$$R_{\gamma}(\tau) \triangleq \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t). \quad (2.8)$$

Then as long as the rewards are bounded (i.e.,  $r(s, a) \in [r_{min}, r_{max}]$  for all  $s, a \in \mathcal{S} \times \mathcal{A}$ ), the value is as well:

$$V_{\gamma}^{\pi} \in \left[ \frac{r_{min}}{1 - \gamma}, \frac{r_{max}}{1 - \gamma} \right].$$

Note that this is equivalent to introducing an “effective” horizon on the problem  $H_{\gamma} = 1/(1 - \gamma)$  over which events are able to influence the agent's decision-making. This can be seen by noting that the value is bounded between  $H_{\gamma}r_{min}$  and  $H_{\gamma}r_{max}$ , as if the agent were getting either  $r_{min}$  or  $r_{max}$  reward every step of a finite trajectory of length  $H_{\gamma}$ . In this case, the horizon is defined implicitly via the discount factor  $\gamma$ , and we can define an infinite horizon MDP as a tuple  $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, \rho, r, \gamma)$ . Here, the discounted occupancy measure is defined as

$$d^{\pi}(s) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \mathbb{P}^{\pi}(s_t = s), \quad (2.9)$$

where the factor  $1 - \gamma$  ensures proper normalization. One subtlety worth noting is that the myopia created by the discount factor  $\gamma$  could perhaps be better described as an attribute of the agent rather than the task (especially when modeling animal behavior).

Nevertheless, it is still standard to include  $\gamma$  as part of the task specification rather than as a parameter of the agent due to the way it implicitly defines the task horizon. An important practical point is that while a task may formally be defined as infinite-horizon, in experiments, there is always an upper limit on the number of allowed time steps. Therefore, even when a discount factor is used, there is always additionally an  $H$  (which may not equal  $H_\gamma$ ) used in practice as well. In the remainder of the text, We'll drop the subscript  $\gamma$  (e.g., in  $V_\gamma$ ) when the context is clear.

## 2.2 Solving Single-Task RL

Given a task, the agent's goal is to find a policy which is *optimal*: one which achieves a value at least as high as any other. Such a policy is called an *optimal policy*  $\pi^*$ , and is formally defined as a policy  $\pi'$  for which  $V^{\pi'} \geq V^\pi \forall \pi$ . The agent is then meant to solve the optimization problem

$$\max_{\pi \in \Pi} V^\pi, \quad (2.10)$$

where  $\Pi$  is the set of possible policies. When solving MDPs,  $\Pi$  is usually taken to be the set of stationary policies. To begin with, we'll assume a finite number of states and actions—a setting often called *tabular*. We will then discuss infinite and continuous state and action spaces. There are a number of possible approaches to solving Eq. (2.10). Historically, one major division has been between *model-free* (MF) and *model-based* (MB) control. In MF learning, the agent attempts to learn an optimal policy via trial-and-error, learning good and bad actions directly from experience. In MB approaches, the agent learns a model of the task—classically, the MDP dynamics  $P$  and reward function  $r$ —and uses this model to improve its policy either by simulating experience in this imagined world or via *dynamic programming*, which is discussed in more detail below. While these approaches are canonically seen as distinct, there is also a long history of methods which combine them in various ways (Sutton, 1990). From a biological perspective, this makes sense, as it is clear that humans and animals make use of both world models and trial-and-error experience to update their behavior. Indeed, a core technique underpinning the methods introduced in Part 2 of this thesis blends the distinction between MF and

MB learning. Regardless of whether the agent uses a MF or MB approach (or something in-between), the precise method of improving a policy still requires specification. We describe the primary two algorithm families below.

### 2.2.1 Value-Based Approaches

Value-based methods for solving Eq. (2.10) do so indirectly, by exploiting an important property of MDPs known as the *Bellman optimality equations* (Bellman, 1957; Agarwal et al., 2021). We say that a matrix  $Q \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{A}|}$  satisfies the *Bellman optimality equations* if

$$Q(s, a) = r(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s, a)} \left[ \max_{a' \in \mathcal{A}} Q(s', a') \right] \quad \forall s, a \in \mathcal{S} \times \mathcal{A}. \quad (2.11)$$

For any  $Q \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{A}|}$ ,  $Q = Q^*$ , where  $Q^*$  denotes the  $Q$ -values of an optimal policy, if and only if  $Q$  satisfies the Bellman optimality equations. Furthermore, the deterministic policy defined by  $\pi(s) \in \operatorname{argmax}_{a \in \mathcal{A}} Q^*(s, a)$  is an optimal policy.

The Bellman optimality equations imply that if we are able to identify the optimal action-values  $Q^*$ , then we immediately get an optimal policy simply by acting greedily in each state with respect to  $Q^*$ . Value-based methods rely on this fact and aim to find the optimal policy indirectly, by first identifying  $Q^*$ . There are two foundational value-based policy improvement approaches: *value iteration* and *policy iteration* (Bellman, 1957).

**Value Iteration** Value iteration (VI) is an algorithm for finding the optimal policy which can be derived directly from Eq. (2.11). To see how, we can define the *Bellman optimality operator*  $\mathcal{T}^* : \mathbb{R}^{|\mathcal{S}| \times |\mathcal{A}|} \rightarrow \mathbb{R}^{|\mathcal{S}| \times |\mathcal{A}|}$  as follows:

$$\mathcal{T}^*Q = R + \gamma \mathcal{P} \mathbf{v}_Q, \quad (2.12)$$

where  $R \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{A}|}$  is the matrix of rewards,  $\mathcal{P}$  denotes a  $|\mathcal{S}| \times |\mathcal{A}| \times |\mathcal{S}|$  tensor such that  $\mathcal{P}_{s, a, s'} = P(s'|s, a)$ , and  $(\mathbf{v}_Q)_s = \max_a Q(s, a) \quad \forall s \in \mathcal{S}$  is the vector of values for the greedy policy over  $Q$ . Given this definition, we can rewrite Eq. (2.11) as simply

$$\mathcal{T}^*Q = Q. \quad (2.13)$$



That is,  $Q = Q^*$  if and only if it's a fixed point of the operator  $\mathcal{T}^*$ . VI, then, consists of “simply” applying  $\mathcal{T}^*$  until this fixed point is reached:

$$Q \leftarrow \mathcal{T}^*Q. \quad (2.14)$$

We've put “simply” in quotes because applying the Bellman optimality operator is simple only under the condition that the agent has access to the transition matrix  $P$  and reward function  $r$ , but this is generally not true in RL. The agent must either try to approximate  $P$  and  $r$  (as in MB approaches) or approximate this update by directly collecting experience in the world (as in MF approaches). For now, we can simply verify that in the idealized case (with access to  $P$  and  $r$ ) VI produces  $Q^*$ . Importantly, the Bellman optimality operator is a *contraction*. That is, for any two matrices  $Q, Q' \in \mathbb{R}^{|S| \times |A|}$ ,

$$\|\mathcal{T}^*Q - \mathcal{T}^*Q'\|_\infty \leq \gamma \|Q - Q'\|_\infty,$$

where  $\|\cdot\|_\infty$  denotes the max-norm. This property guarantees that successive applications converge to a fixed point—and by the Banach Fixed Point theorem, this fixed point is unique. In fact, it is the optimal  $Q$  function.

**Policy Iteration** Like value iteration, policy iteration (PI; [Bellman, 1957](#)) improves the agent's policy by leveraging properties of MDPs, and defines its policy implicitly via the action-value estimate. Rather than emerging directly from the Bellman optimality equations, PI follows from a simpler, somewhat surprising property of MDPs: *greedy policy improvement*. The greedy policy improvement property says that for any policy  $\pi$ , the greedy policy with respect to its  $Q$ -values  $\pi'(s) \in \operatorname{argmax}_a Q^\pi(s, a)$  is at least as good as  $\pi$ —that is,  $Q^{\pi'}(s, a) \geq Q^\pi(s, a)$  for all  $s, a \in \mathcal{S} \times \mathcal{A}$ . This fact underpins the intuition behind PI: given our current policy, if we can evaluate it—that is, compute its  $Q$ -values—we can immediately define a better policy by acting greedily with respect to those  $Q$ -values, and then repeat the process until convergence. That is, each iteration  $k$  of PI consists of two steps:

1. *Policy evaluation*: Given  $\pi^{(k)}$ , compute  $Q^{(k)}$ .

2. *Policy improvement*:  $\pi^{(k+1)}(s) \in \operatorname{argmax}_s Q^{(k)}(s, a) \forall s, a \in \mathcal{S} \times \mathcal{A}$ .

The question, then, is how to perform policy evaluation. It turns out that we can exploit the same recursive property of the optimal value function which underlies the Bellman optimality equations. This analogous relationship is called the *Bellman consistency equations*. That is,  $V^\pi$  and  $Q^\pi$  obey the following relationship:

$$V^\pi(s) = \mathbb{E}_{a \sim \pi(\cdot|s)} Q^\pi(s, a) \quad (2.15)$$

$$Q^\pi(s, a) = r(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s, a)} V^\pi(s') \quad (2.16)$$

for all states and actions. This recursive form follows directly from the definition of the action-value function and the Markov property of the transition dynamics. Bellman consistency allows us to define the *Bellman evaluation operator*  $\mathcal{T}^\pi$ , analogous to  $\mathcal{T}^*$ :

$$\mathcal{T}^\pi \mathbf{q} \triangleq \mathbf{r} + \gamma P^\pi \mathbf{q}, \quad (2.17)$$

where  $\mathbf{q}$  is  $Q$  flattened into an  $|\mathcal{S}||\mathcal{A}|$ -element vector and we overload notation so that  $P^\pi$  is the  $|\mathcal{S}||\mathcal{A}| \times |\mathcal{S}||\mathcal{A}|$  matrix with  $P_{sa, s'a'}^\pi = \pi(a'|s')P(s'|s, a)$ . We can slightly abuse notation and write  $\mathcal{T}^\pi Q$  instead of  $\mathcal{T}^\pi \mathbf{q}$  with the understanding that  $Q$  can be flattened and then reshaped back into a matrix. Similar to the Bellman evaluation operator,  $\mathcal{T}^\pi$  is a contraction and successive applications cause a matrix  $Q$  to converge to  $Q^\pi$ .

**Soft Policy Iteration** While all MDPs admit deterministic optimal policies, a common issue when using approximate methods for large-scale problems is that the policy will collapse to a (near-)deterministic function prematurely, before it has reached optimality. This is often the case when policy optimization overfits to a local maximum in the value function. To prevent this, it's common to add a regularization term which encourages the policy to remain stochastic. The maximum entropy RL (Ziebart, 2010) objective is given by

$$\mathcal{J}(\pi) = \mathbb{E}_{s, a \sim d^\pi} [r(s, a) + \alpha \mathbf{H}[\pi(\cdot|s)]] , \quad (2.18)$$

where  $\alpha$  is a temperature parameter which determines the strength of the regularization, with the standard RL objective recovered as  $\alpha \rightarrow 0$ . This objective results in Bellman consistency equations which are analogous to those which hold for standard MDPs:

$$Q_H^\pi(s, a) = r(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s, a)} V_H^\pi(s')$$

where  $V_H^\pi(s) = \mathbb{E}_{a \sim \pi(\cdot|s)} [Q_H^\pi(s, a) - \alpha \log \pi(a|s)]$ .

This relationship allows for the definition of a *soft Bellman evaluation operator*  $\mathcal{T}_H^\pi$ :

$$\mathcal{T}_H^\pi \mathbf{q} \triangleq \mathbf{r} + \gamma P^\pi(\mathbf{q} - \alpha \log \boldsymbol{\pi}), \quad (2.19)$$

where  $\boldsymbol{\pi}$  is the policy flattened into an  $|\mathcal{S}||\mathcal{A}|$ -vector and  $\log(\cdot)$  is applied element-wise. Repeatedly applying  $\mathcal{T}_H^\pi$  induces convergence to  $Q_H^\pi$  (Haarnoja et al., 2018).

To perform a policy improvement step with this objective in the tabular setting, we can solve the following constrained optimization problem:

$$\max_{\boldsymbol{\pi}} V_H^\pi \quad \text{s.t.} \quad \sum_a \pi(a|s) = 1 \quad \forall s \in \mathcal{S}.$$

To do this, we can use Lagrangian relaxation:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\pi}, \boldsymbol{\lambda}) &= V_H + \sum_s \boldsymbol{\lambda}_s \left( \sum_a \pi(a|s) - 1 \right) \\ &= \boldsymbol{\pi}^\top [\mathbf{q} - \alpha \log \boldsymbol{\pi}] + \sum_s \boldsymbol{\lambda}_s \left( \sum_a \pi(a|s) - 1 \right), \end{aligned}$$

where  $\boldsymbol{\lambda}_1, \dots, \boldsymbol{\lambda}_{|\mathcal{S}|}$  are Lagrange multipliers for each state. Taking the gradient with respect to the policy and setting the result equal to zero:

$$\begin{aligned} \nabla_{\boldsymbol{\pi}} \mathcal{L}(\boldsymbol{\pi}, \boldsymbol{\lambda}) &= \mathbf{q} - \alpha - \alpha \log \boldsymbol{\pi} + \boldsymbol{\lambda} = 0 \\ \Rightarrow \pi(a|s) &= \exp(Q^\pi(s, a)/\alpha - 1 + \boldsymbol{\lambda}_s/\alpha) = \exp(Q^\pi(s, a)/\alpha) \exp(\boldsymbol{\lambda}_s/\alpha - 1), \end{aligned}$$

where we slightly abuse notation and use  $\boldsymbol{\lambda}$  to denote a  $|\mathcal{S}||\mathcal{A}|$ -length vector with each

Lagrange multiplier  $\lambda_s$  repeated  $|\mathcal{A}|$  times. The gradient with respect to the Lagrange multiplier  $\lambda_s$  is  $\sum_a \pi(a|s) - 1$ , which reflects that the policy must be a valid probability distribution in each state. Using this, we get

$$\begin{aligned} \sum_a \exp(Q^\pi(s, a)/\alpha) \exp(\lambda_s/\alpha - 1) &= 1 \\ \Rightarrow \exp(\lambda_s/\alpha - 1) &= \frac{1}{\sum_a \exp(Q^\pi(s, a)/\alpha)}. \end{aligned}$$

Plugging this in, the improved soft policy is given by

$$\pi^{(k+1)}(a|s) = \frac{\exp(Q^{\pi^{(k)}}(s, a)/\alpha)}{\sum_{a'} \exp(Q^{\pi^{(k)}}(s, a')/\alpha)} = \frac{1}{Z(s)} \exp(Q^{\pi^{(k)}}(s, a)/\alpha), \quad (2.20)$$

where  $Z(s) = \sum_{a'} \exp(Q^{\pi^{(k)}}(s, a')/\alpha)$ . Analogously to standard PI, soft PI converges to an optimal policy with respect to the regularized objective (Haarnoja et al., 2018).

**Model-Free Learning** As noted above, in standard RL settings, the agent is not given access to  $P$  and  $r$ . Nonetheless, it can approximate Bellman updates via sampled experience in the environment.

One family of such approaches falls under the heading of *temporal difference* (TD) learning. There are a wide variety of TD algorithms, but we'll briefly describe the foundational approaches here. *Q-learning* (Watkins and Dayan, 1992), in its most basic form, is a sample-based approximation of VI, where given an observed tuple of experience  $(s_t, a_t, r_t, s_{t+1})$ , the agent updates its  $Q$ -values as follows:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \eta_t \delta_t^Q, \quad (2.21)$$

where  $\eta_t$  is the learning rate at time  $t$  and

$$\delta_t^Q = r_t + \gamma \max_{a \in \mathcal{A}} Q_t(s_{t+1}, a) - Q_t(s_t, a_t) \quad (2.22)$$

is the  $Q$ -learning TD error. Observe that  $r_t + \gamma \max_{a \in \mathcal{A}} Q_t(s_{t+1}, a)$  is a single-sample approximation of  $\mathcal{T}^* Q_t$ , and so the  $Q$ -learning error is zero when  $Q = \mathcal{T}^* Q$ . If all state-action pairs are visited infinitely often and the learning rate obeys the *Robbins-Monro*

conditions:

$$\sum_{t=0}^{\infty} \eta_t = \infty, \quad \sum_{t=0}^{\infty} \eta_t^2 < \infty$$

then  $Q_t$  is guaranteed to converge to  $Q^*$  as  $t \rightarrow \infty$  (Watkins and Dayan, 1992).

TD methods can also be used for evaluation. In fact, unless otherwise specified “TD learning” typically refers to methods used for this purpose. The simplest approach is the one-step sample-based analogue of the Bellman evaluation operator:

$$\delta_t^{\text{TD}} = r_t + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t), \quad (2.23)$$

which is used to update the value function just as in Eq. (2.21). Note that the only difference between this approach and  $Q$ -learning is that the TD target in  $Q$ -learning is obtained by the greedy policy over the current  $Q$ -values (which is often different from the policy used to actually take actions—a setting known as *off-policy* learning), while the evaluation target is obtained by sampling from the policy used for acting (known as *on-policy* learning). Unsurprisingly given their similarity, TD learning for evaluation converges under the same conditions as  $Q$ -learning. An analogous TD update can be derived for soft policy evaluation.

Single-step TD methods like this are useful because that they allow the agent to update its value estimates every time step. Additionally, because the learning target is formed from a single step of experience, it generally has low variance, offering a more stable target for learning. However, TD learning is also biased, as the target is formed from a bootstrapped estimate of the value function, e.g.,  $r_t + \gamma Q_t(s_{t+1}, a_{t+1}) \approx Q^\pi(s_t, a_t)$ . To obtain an unbiased target for learning, the agent can instead allow the agent to “roll out” full episodes of experience and simply average the resulting returns. This is known as *Monte-Carlo* (MC) value estimation, and there are a wide variety of methods which fall under this heading. Unlike TD learning, however, MC methods typically suffer from high-variance and require the agent to experience a full episode before learning, which also makes them incompatible with non-episodic tasks. There are a number of approaches which blend TD and MC learning in an effort to reap the benefits of both methods, such

as  $n$ -step returns and  $\text{TD}(\lambda)$  (Sutton and Barto, 2018a).

**Function Approximation** When the state space and/or action space are large or continuous, it becomes impractical to use a tabular representation for value functions. In this case, we must approximate the true value function by optimizing within a—typically parametric—function class. In deep RL, this function class is a neural network architecture (Goodfellow et al., 2016). In this setting, all theoretical guarantees go out the window, but strong empirical performance is usually achievable. There are many value-based approaches in deep RL, but for brevity we’ll only describe the basic analogue of VI/ $Q$ -learning here.

Deep  $Q$ -learning is implemented most commonly in the form of the *deep  $Q$  network* (DQN; Mnih et al., 2015) algorithm, which first drew widespread attention to deep RL for its success on Atari games. A naïve implementation of deep  $Q$ -learning would simply involve parameterizing the  $Q$ -function as a neural network  $Q_\theta : \mathcal{S} \rightarrow \mathbb{R}^{|\mathcal{A}|}$  with parameters  $\theta$  and training it via online stochastic gradient descent (SGD) on the mean-squared TD error. (With regard to notation, we’ll use both  $Q_\theta(s, a)$  and  $Q(s, a; \theta)$  interchangeably.) However, this presents several difficulties.

First, because the  $Q$ -function is used in both the TD target and to evaluate the current state, naïve differentiation will backpropagate through both, which is typically high variance in practice and breaks the IID<sup>2</sup> assumption upon which SGD relies. To address this, Mnih et al. (2015) introduced the idea of *target networks*: a copy of the  $Q$ -function is kept with “frozen” parameters which are not differentiable, which we denote with as  $\theta^-$ . This target network is used to compute the TD target, with the frozen parameters either reset to the current (updated) parameter values every few iterations or via an exponential moving average. This both provides a more stable target for learning and prevents the issues associated with backpropagating through a bootstrapped target.

The second challenge is also rooted in the non-stationarity of RL—as the  $Q$ -function (and therefore the policy) is updated, the distribution of the observed data changes, which again violates the IID assumption underlying SGD. Updating the  $Q$ -function online only compounds this issue. To address this, Mnih et al. (2015) exploited

---

<sup>2</sup>IID stands for “independent and identically distributed.”

the off-policy nature of  $Q$ -learning, collecting hundreds of thousands of transition tuples  $(s_t, a_t, r_t, s_{t+1})$  into a *replay buffer*  $\mathcal{B}$  (Lin, 1992). Rather than update the  $Q$ -function online, they instead sample minibatches of size  $B$  uniformly from  $\mathcal{B}$  and use these to train the network. The intuition is that doing so provides a more stable data distribution for learning, and averaging over many samples provides a more accurate, lower-variance gradient update. It also allows the agent to reuse its previous experience, rather than throwing it away after only using it for a single weight update.

These two innovations—target networks and use of a replay buffer—are now ubiquitous in deep RL, and there are many variations. Put together, the learning update for DQN (ignoring the use of another deep learning optimizer like Adam (Kingma and Ba, 2014)) is given by

$$\theta_{t+1} = \theta_t - \eta_t \nabla_{\theta} \frac{1}{B} \sum_{b=1}^B \frac{1}{2} (r_t^b + \gamma \max_a Q_{\theta^-}(s_{t+1}^b, a) - Q_{\theta}(s_t^b, a_t^b))^2.$$

One subtle, but important implementation detail here is that such an approach is feasible only when the action space  $\mathcal{A}$  is finite. While there are a number of methods which attempt to adapt  $Q$ -learning style updates to problems which require continuous control (Xiong et al., 2018; Gu et al., 2016; Seyde et al., 2022), policy-based methods (Section 2.2.2) are typically preferred in that setting. Policy evaluation can be performed by minimizing a loss analogous to DQN's:

$$\theta_{t+1} = \theta_t - \eta_t \nabla_{\theta} \frac{1}{B} \sum_{b=1}^B \frac{1}{2} (r_t^b + \gamma Q_{\theta^-}(s_{t+1}^b, a_{t+1}) - Q_{\theta}(s_t^b, a_t^b))^2,$$

where  $a_{t+1} \sim \pi_t(\cdot | s_{t+1}^b)$ . While approximate policy iteration methods can be implemented this way, they are most commonly applied to continuous control tasks (i.e., those for which  $\mathcal{A}$  is continuous), where they form the basis for several state-of-the-art algorithms. In this setting, a direct parameterization of the policy is required. These approaches are discussed in more detail in the following section.

### 2.2.2 Policy-Based Approaches

In contrast to value-based methods, which encode a policy implicitly via  $Q$ -values, policy-based approaches take a more direct approach to solving Eq. (2.10). The immediate question when optimizing over  $\pi$  is how  $\pi$  is encoded—in other words, its parameterization.

**Policy Parameterizations** In policy-based approaches, the agent seeks to find the best parameters  $\theta$  within some particular function class  $\Theta$ , framing the RL problem as

$$\max_{\theta \in \Theta} V^{\pi_\theta}. \quad (2.24)$$

When  $\mathcal{S}$  and  $\mathcal{A}$  are finite, we can use a tabular parameterization,  $\theta \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{A}|}$ . A tabular representation can either be *direct*, with  $\pi(a|s) = \theta_{s,a}$  or *softmax*, with

$$\pi_\theta(a|s) = \frac{\exp(\theta_{s,a})}{\sum_{a' \in \mathcal{A}} \exp(\theta_{s,a'})}. \quad (2.25)$$

Both of these parameterizations are called *complete*, as they are able to represent any stationary policy for finite  $\mathcal{S}$  and  $\mathcal{A}$ . When the state space is prohibitively large, but the action space is finite, we can use a *neural softmax policy*, of the form

$$\pi_\theta(a|s) = \frac{\exp(f_\theta(s, a))}{\sum_{a' \in \mathcal{A}} \exp(f_\theta(s, a'))}, \quad (2.26)$$

where  $f_\theta$  is a neural network. This policy class may not be complete. If the action space is continuous, it's common to instantiate the policy as a particular distribution, with, e.g., a neural network outputting the parameters of that distribution class. For example, it is common to have a Gaussian policy:

$$\pi_\theta(a|s) = \mathcal{N}(a; f_\theta(s), \Sigma). \quad (2.27)$$

Note that here we've written the neural network as only parameterizing the mean of the distribution, with  $\Sigma$  a constant, but it is also common to have the network produce both the mean and the variance.

There are a diversity of algorithms for optimizing policies. Two that are of particular importance, both within RL and to this thesis specifically, are *direct policy search* methods



and *approximate policy iteration* algorithms.

### 2.2.2.1 Direct Policy Search

**Policy Gradients** Given a parameterization, we need to derive a learning rule for the policy. The most direct approach is to proceed by gradient ascent on  $V^{\pi_\theta}$  (often written as  $V^\pi$  for brevity). Despite the fact that  $V^\pi$  is non-concave in  $\pi$ —and so therefore gradient ascent is in general not guaranteed to reach a global optimum—such *policy gradient* methods are highly popular in practice and lie at the root of many state-of-the-art algorithms (Silver et al., 2016; Schulman et al., 2017; Hafner et al., 2023; Abdolmaleki et al., 2018; Haarnoja et al., 2018). Throughout this section, we’ll assume rewards are bounded between 0 and 1 and state and action spaces are finite for convenience, though all derivations below are easily translated to different reward bounds and continuous  $\mathcal{S}$  and  $\mathcal{A}$ . To obtain estimators for the gradient of the value with respect to the policy, we can write

$$\begin{aligned}
\nabla_\theta V^\pi &= \mathbb{E}_{s_0 \sim \rho(\cdot)} \left[ \sum_{a_0} \pi_\theta(a_0|s_0) Q^\pi(s_0, a_0) \right] \\
&= \mathbb{E}_\rho \left[ \sum_{a_0} Q^\pi(s_0, a_0) \nabla_\theta \pi_\theta(a_0, s_0) + \pi_\theta(a_0|s_0) \nabla_\theta Q^\pi(s_0, a_0) \right] \\
&= \mathbb{E}_\rho \left[ \sum_{a_0} \pi_\theta(a_0|s_0) Q^\pi(s_0, a_0) \nabla_\theta \log \pi_\theta(a_0|s_0) \right. \\
&\quad \left. + \sum_{a_0} \pi_\theta(a_0|s_0) \nabla_\theta \left( r_0 + \gamma \sum_{s_1} P(s_1|s_0, a_0) V^\pi(s_1) \right) \right] \\
&= \mathbb{E}_\rho \left[ \sum_{a_0} \pi_\theta(a_0|s_0) Q^\pi(s_0, a_0) \nabla_\theta \log \pi_\theta(a_0|s_0) \right. \\
&\quad \left. + \gamma \sum_{a_0, s_1} \pi_\theta(a_0|s_0) P(s_1|s_0, a_0) \nabla_\theta V^\pi(s_1) \right] \\
&= \mathbb{E}_{\tau \sim \mathbb{P}^{\pi_\theta}(\cdot)} [Q^\pi(s_0, a_0) \nabla_\theta \log \pi_\theta(a_0|s_0)] + \gamma \mathbb{E}_{\tau \sim \mathbb{P}^{\pi_\theta}(\cdot)} [\nabla_\theta V^\pi(s_1)] \\
&\stackrel{(i)}{=} \mathbb{E}_{\mathbb{P}^{\pi_\theta}} [Q^\pi(s_0, a_0) \nabla_\theta \log \pi_\theta(a_0|s_0)] + \gamma \mathbb{E}_{\mathbb{P}^{\pi_\theta}} [Q^\pi(s_1, a_1) \nabla_\theta \log \pi_\theta(a_1|s_1)] + \dots \\
&= \mathbb{E}_{\mathbb{P}^{\pi_\theta}} \left[ \sum_{t=0}^{\infty} \gamma^t Q(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t|s_t) \right] \tag{2.28}
\end{aligned}$$

$$= \frac{1}{1-\gamma} \mathbb{E}_{s \sim d^\pi(\cdot)} \mathbb{E}_{a \sim \pi_\theta(\cdot|s)} [Q^\pi(s, a) \nabla_\theta \log \pi_\theta(a|s)], \tag{2.29}$$

**Algorithm 1** Basic policy gradient algorithm

- 
- 1: **Input** MDP  $M$ , policy class  $\Theta$
  - 2: initialize  $\theta^{(0)} \in \Theta$
  - 3: **for** iteration  $k = 0, 1, 2, \dots$  **do**
  - 4:   sample a trajectory:

$$\tau = (s_0, a_0, s_1, \dots) \sim \mathbb{P}^{\pi_{\theta^{(k)}}}(\cdot) = \rho(s_0) \prod_{t=0}^{\infty} P(s_{t+1}|s_t, a_t) \pi_{\theta^{(k)}}(a_t|s_t)$$

- 5:   update parameters:

$$\theta^{(k+1)} = \theta^{(k)} + \eta \widehat{\nabla V^{\pi_{\theta}}}$$

where

$$\widehat{\nabla V^{\pi_{\theta}}} = \sum_{t=0}^{\infty} \gamma^t \widehat{Q^{\pi_{\theta}}}(s_t, a_t) \nabla \log \pi_{\theta}(a_t|s_t),$$

with  $\widehat{Q^{\pi_{\theta}}}(s_t, a_t) = \sum_{t'=t}^{\infty} \gamma^{t'-t} r(s_{t'}, a_{t'})$

- 6: **end for**
- 

where (i) is a result of applying the previous lines of the derivation recursively to  $\nabla_{\theta} V^{\pi}(s_1)$ . Eq. (2.28) gives us an estimator for the gradient that we can compute from full trajectories (where the trajectory length is finite in practice), and Eq. (2.29) provides an estimator obtainable just from state-action pairs. Intuitively, we can view updating the policy using either estimator as increasing the log-probability of actions with higher values. A simple policy gradient approach using Eq. (2.28) is illustrated in Algorithm 1, where  $\hat{\cdot}$  is used to denote an empirical estimate. Note this approach implicitly assumes the use of a policy class such as softmax policies which permits unconstrained gradient updates. In contrast, using a direct tabular representation would require the use of projected gradient ascent to ensure that the policy remained a valid probability distribution in each state.

In Algorithm 1 the action-value is simply a Monte Carlo estimate obtained by computing the return (this update is the REINFORCE estimator (Williams, 1992)), but in more sophisticated policy gradient algorithms, the value function of the current policy

is also parameterized and learned, e.g., via TD methods. Such approaches are known as *actor-critic* algorithms, so-named because the two central components used by the agent are an *actor* (the policy) which takes actions in the environment and a *critic* (the value function) which evaluates the quality of the actor's behavior and guides its learning.

In practice, the policy gradient estimators in Eq. (2.28) and Eq. (2.29) are often high-variance, which makes learning challenging. One common way to address this issue without adding bias is to make use of the fact that subtracting any action-independent quantity from the  $Q$ -value does not affect the expected value of the gradient. More precisely, let  $b : \mathcal{S} \rightarrow \mathbb{R}$  be some state-dependent function, and let  $g$  denote the policy gradient estimator in Eq. (2.29) (though the following derivation is equally applicable to Eq. (2.28)). Then we can see that

$$\begin{aligned}
& \frac{1}{1-\gamma} \mathbb{E}_{d^\pi} \mathbb{E}_\pi \left[ (Q^\pi(s, a) - b(s)) \nabla_\theta \log \pi_\theta(a|s) \right] \\
&= \frac{1}{1-\gamma} \mathbb{E}_{d^\pi} \mathbb{E}_\pi [Q^\pi(s, a) \nabla_\theta \log \pi_\theta(a|s)] - \mathbb{E}_{d^\pi} \mathbb{E}_\pi [b(s) \nabla_\theta \log \pi_\theta(a|s)] \\
&= g - \sum_s d^\pi(s) \sum_a \pi_\theta(a|s) b(s) \nabla_\theta \log \pi_\theta(a|s) \\
&= g - \sum_s d^\pi(s) b(s) \sum_a \pi_\theta(a|s) \frac{1}{\pi_\theta(a|s)} \nabla_\theta \pi_\theta(a|s) \\
&= g - \sum_s d^\pi(s) b(s) \nabla_\theta \underbrace{\sum_a \pi_\theta(a|s)}_{=1} \\
&= g,
\end{aligned}$$

as desired. While  $b(s)$  can be any state-dependent, action-independent function, the most common choice is the value function  $V^\pi(s)$ . The quantity  $Q^\pi(s, a) - V^\pi(s)$  is ubiquitous in RL, and is called the *advantage function*, denoted  $A^\pi(s, a)$ . Intuitively,  $A^\pi(s, a)$  expresses how much more (or less) cumulative reward that action  $a$  is expected to lead the policy to starting from  $s$  compared to the average behavior under  $\pi$ . Conveniently, because of the Bellman consistency relationship (Eq. (2.15)), to estimate advantages, the agent only needs to maintain a state-value function  $V^\pi(s)$ , because  $r_t + \gamma V^\pi(s_{t+1})$  is an estimator for  $Q^\pi(s_t, a_t)$ . Replacing  $Q^\pi(s, a)$  with  $A^\pi(s, a)$  to

perform policy gradients forms the basis of the *advantage actor-critic* (A2C; Mnih et al., 2016) algorithm.

### 2.2.2.2 Approximate Policy Iteration

While policy iteration can be approximated with neural networks in a value-based way when the action space is finite, deep RL approaches to PI are most commonly applied to continuous control problems (Abdolmaleki et al., 2018; Haarnoja et al., 2018). There are a number of algorithms within this family, but one worth highlighting here—both for its strong performance in practice and use in later chapters—is *soft actor-critic* (SAC; Haarnoja et al., 2018). SAC is an off-policy approach designed to approximate soft PI (Ziebart, 2010). Policy evaluation is performed by minimizing the soft policy evaluation loss

$$\mathcal{L}(\phi) = \frac{1}{B} \sum_{b=1}^B \frac{1}{2} \left( r_b + \gamma \mathbb{E}_{a' \sim \pi(\cdot|s'_b)} [Q_{\phi^-}(s'_b, a') - \alpha \log \pi(a'|s'_b)] - Q_{\phi}(s_b, a_b) \right)^2,$$

where  $\{(s_b, a_b, r_b, s'_b)\}_{b=1}^B$  is a minibatch of experience sampled from a replay buffer  $\mathcal{B}$  and  $\phi$  are the parameters of the  $Q$ -function. Policy improvement is carried out by minimizing the KL divergence between the policy and an approximation of the tabular softmax policy from the improvement step of soft policy iteration:

$$\mathcal{L}(\theta) = \frac{1}{B} \sum_{b=1}^B \text{KL} \left[ \pi_{\theta}(\cdot|s_b); \frac{1}{Z_{\phi}} \exp(Q_{\phi}(s_b, \cdot)/\alpha) \right].$$

Practical implementations of SAC also frequently adapt the entropy weight  $\alpha$  online, increasing it if the policy entropy falls below a chosen threshold and decreasing it otherwise (similar to a Lagrange multiplier).

These approaches form the basis for the algorithms used throughout this thesis. While there are more advanced techniques than the methods presented in later chapters build from directly, these are covered where necessary so as to present them in the most relevant context. We now present a brief overview of multitask RL and in particular previous approaches to leveraging commonalities across tasks.

## 2.3 Multitask Reinforcement Learning

So far, we have only described RL in the context of a single task—that is, there is only one MDP with which the agent need concern itself. Is this setting enough to build and understand generalist agents and to model natural behavior? One could imagine defining a single reward function that covers all possible goals an agent might wish to accomplish. Indeed, in RL, the *reward hypothesis* postulates that “all of what we mean by goals and purposes can be well thought of as maximization of the expected value of the cumulative sum of a received scalar signal (reward).” (Sutton, 2004; Silver et al., 2021; Bowling et al., 2023). However, an essential idea underpinning this thesis—and *multitask RL* (MTRL) more broadly—is that it’s more useful for an agent to decompose such a global reward function into distinct rewards which measure success for distinct goals. On a daily basis, we’re more often concerned with solving immediate problems like “What should I make for dinner?” than making every action only by taking into account its long-term impact on our lives. Making decisions this way would require excessive computation and ignore the compositional structure of decision-making. The lives of humans and animals are built in many ways around routines: we brush our teeth, follow a familiar route to the grocery store, and go to work everyday. A failure to leverage this repetition to make decisions would be horribly inefficient—in an information-theoretic sense, behavior is highly compressible, and we’d like to be able to take advantage of that. Additionally, solving short-horizon tasks is significantly easier than long-term tasks. This can be seen in the required sample complexities of value and policy iteration, which reduce as the discount factor is lowered (Sutton and Barto, 2018b).

I define MTRL as any RL setting in which the agent’s overall objective is decomposed into multiple tasks or goals. The two primary parts of this thesis are primarily concerned with how, given some distribution over a set of tasks, agents can leverage similarities across them to learn and/or compose policies. Before proceeding, however, we’ll briefly provide an overview of the two most common ways in which task distributions are used to train MTRL agents, as well as discuss connections with related sub-fields in the literature.

### 2.3.0.1 The Parallel Task Setting

Consider a set of tasks (MDPs)  $\mathcal{M} = \{M\}$  which may be infinite in size. The agent’s goal is to maximize its average value across these tasks  $\mathbb{E}V_M$ , where the expectation is taken with respect to a *task distribution*  $\mathbb{P}_{\mathcal{M}}(M)$ . When trained on a single stream of experience (i.e., on one machine), a new task is sampled at the beginning of every episode, making the task presentation more aptly called “interleaved” than parallel. (However, the experience data used to update the agent is often pooled across tasks.) If training is distributed across multiple machines, the agent can employ multiple actors to collect experience across tasks in parallel. Commonly, each task is associated with a particular input feature  $g \in \mathcal{G}$ , where  $\mathcal{G}$  is the space of possible goals, which indicates which task (goal) has been sampled. The idea in this case is that similarities among tasks in  $\mathcal{M}$  enable the agent to generalize to previously unseen MDPs drawn from the same distribution. One type of parallel MTRL is parallel *meta-RL*. In this setting, the agent trains on each sampled task for only a few episodes total with the goal of improving few-shot performance and is “meta-tested” on a set of held-out tasks (Finn et al., 2017; Yu et al., 2019).

### 2.3.0.2 The Sequential Task Setting

In the *sequential* task setting (Moskovitz et al., 2022a; Pacchiano et al., 2022), tasks are sampled one at a time  $M_k \sim \mathbb{P}_{\mathcal{M}}$ , with the sampling process either independent or conditioned on previous tasks and performance (e.g.,  $\mathbb{P}_{\mathcal{M}}(M_k | M_{k-1})$ ) and the agent trained on each until convergence, defined as reaching within a certain threshold  $\epsilon > 0$  of optimal performance, or until a maximum number of environment interactions is reached. Sequential MTRL can be seen as a special case of *continual RL* (Abel et al., 2023; Khetarpal et al., 2022), which demands no particular compositional structure of the overall reward and instead emphasizes the need for never-ending adaptation. In sequential MTRL, the agent’s goal is simply to achieve a “good enough” policy as soon as possible for each task it faces, and its learning process may terminate if it finds a sufficiently good policy or set of policies. This setting is itself a generalization of *transfer learning*, in which the agent is trained on a single *source task* and then evaluated/fine-tuned on another *target* task. Frequently, the tasks are such that it is relatively computationally cheap to generate a lot of experience in the source task, with the hope that relatively little adaptation is required to

perform well in the target environment (assuming that the source task is similar in some sense to the target task). One particular type of transfer learning in RL is known as *unsupervised RL* (Jaderberg et al., 2016; Laskin et al., 2021; Strouse et al., 2021), in which an agent attempts to learn useful behaviors and/or representations by pre-training in an environment without extrinsic rewards before being fine-tuned using reward. The idea in this case is that learning about the structure of the environment and how to traverse it enables the agent to adapt more efficiently once assigned a task.

### 2.3.0.3 Types of Structure in MTRL

There are a number of different assumptions made in MTRL to simplify learning and make the task distribution more compressible. The primary structure underpinning all of MTRL, as noted previously, is *hierarchy*. That is, that the overall objective can be decomposed into multiple tasks. In the sequential setting, it's also often assumed that these tasks must be solved in a particular order (Singh, 1992). In both the sequential and parallel settings, another common assumption is that the behaviors needed to solve the different tasks in  $\mathcal{M}$  are similar in some way, enabling the agent to learn policies that are useful across tasks (Galashov et al., 2019a; Teh et al., 2017a; Tirumala et al., 2019, 2020a). This form of structure only requires that the different tasks share the same state and action spaces, and is explored in greater depth in Part One of this thesis. Another common assumption is that the environment (CMP) is conserved across tasks—that is, that the state space, action space, and transition dynamics (and, frequently, the discount factor) are constants, with variation limited to the reward function (Barreto et al., 2017, 2020; Ma et al., 2020; Vértés and Sahani, 2019; Zahavy et al., 2021). This setting is explored in Part Two. Other structural assumptions include the smooth/slow or step-wise variation of task elements over time (Khetarpal et al., 2022), low-rank environment transition dynamics decodable from a set of shared features (Agarwal et al., 2020a; Pacchiano et al., 2022; Cheng et al., 2022; Agarwal et al., 2022), and similarity in optimal value functions across tasks (Schaul et al., 2015; Borsa et al., 2018).

## 2.4 RL and Neuroscience

RL has been connected to neuroscience and psychology since its inception. In fact, the work of pioneers in the field like Richard Sutton and Andrew Barto (Sutton and Barto, 2018a) was largely inspired by theories of animal learning and behaviorism advanced by B.F. Skinner and his study of operant conditioning (Skinner, 1965). There are too many such connections to enumerate here, but we'll briefly address a few of the ideas most relevant to this thesis, with more detail provided as needed in subsequent chapters.

### 2.4.1 Reward Prediction

Perhaps the canonical example which embodies the relationship between the brain and RL is the study of reward prediction. One of the earliest models of animal reward prediction is the *Rescorla-Wagner model* (Wagner and Rescorla, 1972; Dayan and Abbott, 2005), which uses a single-step reward prediction error with a linear predictor

$$\theta_{t+1} = \theta_t + \eta \delta_t x \quad \text{where} \quad \delta = r_t - \theta_t x \quad (2.30)$$

to model the average reward signal associated with a stimulus  $x$  (where  $\theta$  is the model parameter). This is simply a one-step TD evaluation update for a linear model of a single-step ( $\gamma = 0$ ) task. One of the most striking findings supporting the connection between RL theory and the brain is that of Schultz (1998). Recording from the ventral tegmental area (VTA) in the midbrain of monkeys trained to respond in different ways to varying stimuli in order to obtain rewards, Schultz (1998) found that the firing rate of dopaminergic neurons (those which produce the neurotransmitter dopamine) aligned almost exactly with the TD reward prediction error  $\delta = r + \gamma V(s') - V(s)$ . That is, when reward exceeded the expected amount, firing would spike, and when it fell short, there would be a drop in the firing rate. The relationship of dopamine with reward prediction has been one of the most enduring theories and well-studied areas of neuroscience, with significant medical relevance to diseases such as addiction, Parkinson's, and schizophrenia (Dayan and Abbott, 2005). However, as study has deepened, the picture has grown more complicated rather than simpler. For example, dopamine is now widely thought to encode different kinds of information (not just reward prediction error) in different



parts of the brain (Montague et al., 2004; Watabe-Uchida and Uchida, 2018; Greenstreet et al., 2022), and there is evidence that populations of dopaminergic neurons encode the distribution over return rather than just its expectation (Dabney et al., 2020).

### 2.4.2 Habits

In 1911, while studying operant conditioning in cats, Edward Thorndike postulated the *Law of Effect*, which states that actions which have been rewarded in the past are likely to be repeated, and the *Law of Exercise*, which holds that actions that have been performed in the past are also likely to be repeated (Thorndike, 1911). These simple ideas have been tremendously influential in psychology and neuroscience, with the Law of Effect heavily associated with theories of goal-directed learning and the Law of Exercise underpinning ideas surrounding habitual behavior. The basic idea is that an action is likely to be repeated once it has been rewarded due to the Law of Effect, and its continued repetition is then encouraged simply by virtue of its selection in the past due to the Law of Exercise. (To be more precise, the Laws of Effect and Exercise describe these behavioral patterns, rather than “cause” them.) This second part is especially important in the study of habits, because once an action or sequence of actions becomes ingrained, even if the stimulus which originally triggered its selection loses its association with reward, the stimulus will continue to elicit the learned response. This perseveration of behavior independent of reward is a defining feature of habit acquisition (Graybiel, 2008). The slow adaptation of habitual behavior is one reason why, in RL-based theories, it has been traditionally linked with model-free control, while goal-directed control is generally associated with model-based learning (Daw et al., 2005; Dolan and Dayan, 2013).

However, these connections have recently been challenged by work which instead focuses on the reward-insensitive nature of habitual stimulus-response associations. For example, Miller et al. (2016b) posit that a more appropriate computational model of habit acquisition can be driven by *action prediction errors*, wherein the cached association between stimulus  $s$  and action  $a$ ,  $H(s, a)$ , is updated according to the rule:

$$H_{t+1}(s, a) = H_t(s, a) + \eta(\mathbb{1}(a_t = a) - H_t(s, a)), \quad (2.31)$$

so that  $H(s, a)$  encodes an exponentially moving average of the frequency with which  $a$  is selected in response to  $s$ . In addition to agreement with behavioral data found by [Miller et al. \(2016b\)](#), one recent study ([Greenstreet et al., 2022](#)) found evidence for dopamine-based coding of action prediction errors in the tail of the striatum within the basal ganglia of mice.

## **Part I**

### **Shared Behaviors**

This section of the thesis focuses on the use of shared behavioral structure across tasks to learn new policies more efficiently. Put simply, the results here show that if some pattern of behavior is useful to accomplish multiple goals, then identifying that pattern can accelerate learning to accomplish new goals. For example, if you run to a new spot in the park every day, memorizing the route to the park makes it easier to learn new routes within the park. This section begins with a close examination of how an agent can capture this structure and encode it in a reference “default policy.” Such a policy can act as a form of supervision via regularization, guiding the agent when learning new tasks. Chapter 3 analyzes the situations in which this is helpful, as well as limitations. Subsequent chapters develop this theory further, introducing a minimum description length-based approach to ensure the default policy does not overfit to spurious behavioral structure and then showing this method recapitulates a variety of behavioral results in neuroscience associated with dual process theories of cognition.

## Chapter 3

# Towards an Understanding of Default Policies in Multitask Policy Optimization

### 3.1 Introduction

Appropriate regularization has been a key factor in the widespread success of policy-based deep reinforcement learning (RL) (Levine, 2018a; Furuta et al., 2021). The key idea underlying such *regularized policy optimization* (RPO) methods is to train an agent to maximize reward while minimizing some cost which penalizes deviations from useful behavior, typically encoded as a *default policy*. In addition to being easily scalable and compatible with function approximation, these methods have been shown to ameliorate the high sample complexity of deep RL methods, making them an attractive choice for high-dimensional problems (Berner et al., 2019; Espeholt et al., 2018).

A natural question underlying this success is *why* these methods are so effective. Fortunately, there is a strong foundation for the formal understanding of regularizers in the single-task setting. These methods can be seen as approximating a form of natural gradient ascent (Kakade, 2002; Pacchiano et al., 2020; Moskovitz et al., 2021a), trust region or proximal point optimization (Schulman et al., 2015, 2017), or variational inference (Levine, 2018a; Marino et al., 2020; Abdolmaleki et al., 2018; Haarnoja et al., 2018), and thus are well-motivated by theory (Agarwal et al., 2020b).

However, as interest has grown in training general agents capable of providing real world utility, there has been a shift in emphasis towards *multitask* learning. Accordingly, there are a number of approaches to learning or constructing default policies for regularized policy optimization in multitask settings (Galashov et al., 2019a; Teh et al., 2017a; Goyal et al., 2019, 2020; Tirumala et al., 2020b). The basic idea is to obtain a default policy which is generally useful for some family of tasks, thus offering a form of supervision to the learning process. However, there is little theoretical understanding of how the choice of default policy affects optimization. Our goal in this chapter, adapted from Moskovitz et al. (2022a), is to take a first step towards bridging this gap, asking:

***Q1:** What properties does a default policy need to have in order to improve optimization on new tasks?*

This is a nuanced question. The choice of penalty, structural commonalities among the tasks encountered by the agent, and even the distribution space in which the regularization is applied have dramatic effects on the resulting algorithm and the agent’s performance characteristics.

In this work, we focus on methods using the Kullback-Leibler (KL) divergence with respect to the default policy, as they are the most common in the literature. We first consider this form of regularized policy optimization applied to a single task, with the goal of understanding how the relationship between the default and optimal policies for a given problem affect optimization. We then generalize these results to the multitask setting, where we not only quantify the advantages of this family of approaches, but also identify its limitations, both fundamental and algorithm-specific.

In the process of garnering new understanding of these algorithms, our results also imply a new framework through which to understand families of tasks. Because different algorithms are sensitive to different forms of structure, this leads to another guiding question, closely tied to the first:

***Q2:** What properties does a group of tasks need to share for a given algorithm to provide a measurable benefit?*

It’s clear that in order to be effective, any multitask learning algorithm must be ap-

plied to a task distribution with some form of structure identifiable by that algorithm: if tasks have nothing in common, no understanding gained from one task will be useful for accelerating learning on another. Algorithms may be designed to accommodate—or learn—a broader array of structures, but at increased computational costs. In high-dimensional problems, function approximation mandates new compromises. In this work, which we view as a first step towards understanding these trade-offs, we make the following contributions:

- We show the error bound and iteration complexity for optimization using an  $\alpha$ -optimal default policy, where sub-optimality is measured via the distance from the optimal policy for a given task.
- From these results, we derive a principled RPO algorithm for multitask learning, which we term *total variation policy optimization* (TVPO). We show that popular multitask KL-based algorithms can be seen as approximations to TVPO and demonstrate the strong performance of TVPO on simple tasks.
- We offer novel insights on the optimization characteristics—both limitations and advantages—of common multitask RPO frameworks in the literature.

## 3.2 Regularized Policy Optimization

**Reinforcement learning** In this chapter, we consider finite, discounted MDPs  $M = (\mathcal{S}, \mathcal{A}, P, r, \gamma, \rho)$ . We also assume access to a restart distribution for training  $\mu \in \Delta(\mathcal{S})$  such that  $\mu(s) > 0 \forall s \in \mathcal{S}$ , as is common in the literature (Kakade and Langford, 2002; Agarwal et al., 2020b). The agent takes actions using a stationary policy  $\pi : \mathcal{S} \rightarrow \Delta(\mathcal{A})$ , which, in conjunction with the transition dynamics, induces a distribution over trajectories  $\tau = (s_t, a_t)_{t=0}^\infty$ . We overload notation and define  $V^\pi(\rho) := \mathbb{E}_{s_0 \sim \rho} [V^\pi(s_0)]$  as the expected value for initial state distribution  $\rho$ . By  $d_{s_0}^\pi$ , we denote the discounted state visitation distribution of  $\pi$  with starting state distribution  $\mu$ , so that

$$d_{s_0}^\pi(s) = \mathbb{E}_{s_0 \sim \mu} \left[ (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \Pr^\pi(s_t = s | s_0) \right], \quad (3.1)$$

where  $d_\mu^\pi := \mathbb{E}_{s_0 \sim \mu} [d_{s_0}^\pi(s)]$ . The goal of the agent is to adapt its policy so as to maximize its value, i.e., optimize  $\max_\pi V^\pi(\rho)$ . We use  $\pi^* \in \operatorname{argmax}_\pi V^\pi(\rho)$  to denote the optimal policy and  $V^*$  and  $Q^*$  as shorthand for  $V^{\pi^*}$  and  $Q^{\pi^*}$ , respectively.

**Policy Parameterization and Objective** In this chapter, we primarily consider the tabular softmax policy class

$$\pi_\theta(a|s) = \frac{\exp(\theta_{s,a})}{\sum_{a' \in \mathcal{A}} \exp(\theta_{s,a'})}, \quad (3.2)$$

where  $\theta \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{A}|}$ . The general form of the *regularized policy optimization* (RPO) objective function is given by

$$\mathcal{J}_\lambda(\theta) := V^{\pi_\theta}(\mu) - \lambda \Omega(\theta), \quad (3.3)$$

where  $\Omega$  is some convex regularization functional. Gradient ascent updates proceed according to

$$\theta^{(t+1)} = \theta^{(t)} + \eta \nabla_\theta \mathcal{J}_\lambda(\theta^{(t)}). \quad (3.4)$$

For simplicity of notation, from this point forward, for iterative algorithms which obtain successive estimates of parameters  $\theta^{(t)}$ , we denote the associated policy and value functions as  $\pi^{(t)}$  and  $V^{(t)}$ , respectively. The choice of  $\Omega$  plays a significant role in algorithm design and practice, as we discuss below. It's also important to note that the error bounds and convergence rates we derive are based on the basic policy gradient framework in Algorithm 2, in which update Eq. (3.4) is applied across a batch after every  $B$  trajectories  $\{\tau_b\}_{b=1}^B$  are sampled from the environment. Therefore, the iteration complexities below are proportional to the associated sample complexities.



**Algorithm 2** Regularized policy gradient algorithm

- 
- 1: **Input** MDP  $M$ , policy class  $\Theta$ , regularization strength  $\lambda$ , default policy  $\pi_0$
  - 2: initialize  $\theta^{(0)} \in \Theta$
  - 3: **for** iteration  $k = 0, 1, 2, \dots, K$  **do**
  - 4:   sample  $B$  trajectories ( $b = 1, \dots, B$ ):

$$\tau_b = (s_0, a_0, s_1, \dots) \sim \Pr_{\mu}^{\pi_{\theta^{(k)}}}(\cdot) = \mu(s_0) \prod_{t=0}^{\infty} P(s_{t+1}|s_t, a_t) \pi_{\theta^{(k)}}(a_t|s_t)$$

- 5:   update parameters:

$$\theta^{(k+1)} = \theta^{(k)} + \eta \widehat{\nabla_{\theta} \mathcal{J}_{\lambda}}(\theta^{(k)})$$

where

$$\widehat{\nabla_{\theta} \mathcal{J}_{\lambda}}(\theta) = \widehat{\nabla_{\theta} V^{\pi_{\theta}}}(\mu) - \lambda \nabla_{\theta} \Omega(\pi_0, \pi_{\theta})$$

and  $\widehat{\nabla_{\theta} V^{\pi_{\theta}}}(\mu)$  is as in Algorithm 1.

- 6: **end for**
  - 7: **return**  $\theta^{(K)}$
- 

### 3.3 Related Work

**Single-task learning** The majority of the theoretical (Agarwal et al., 2020b; Grill et al., 2020) and empirical (Schulman et al., 2015, 2017; Abdolmaleki et al., 2018; Pacchiano et al., 2020) literature has focused on the use of RPO in a single-task setting, i.e., applied to a single MDP  $M$ . The majority of these methods place a soft or hard constraint on the Kullback-Leibler (KL) divergence between the updated policy at each time step and the

current policy, maximizing an objective of the form

$$\begin{aligned} \mathcal{J}_\lambda(\pi_p, \pi_q) = & \sum_{t=0}^{\infty} \mathbb{E}_{s_t \sim d_\mu^{\pi_q}} \mathbb{E}_{a_t \sim \pi_q(\cdot|s_t)} [\mathcal{G}(s_t, a_t) \\ & - \lambda \text{KL}[\pi_p(\cdot|s_t); \pi_q(\cdot|s_t)]], \end{aligned} \quad (3.5)$$

where  $\mathcal{G} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$  is typically the  $Q$ - or advantage function and  $\pi_q, \pi_p \in \{\pi_\theta, \pi_0\}$  (Furuta et al., 2021). At each update, then, the idea is to maximize reward while minimizing the regularization cost. From a theoretical perspective, such methods can often be framed as a form of approximate variational inference, with either learned (Abdolmaleki et al., 2018; Song et al., 2019; Peng et al., 2021; Nair et al., 2021; Peters et al., 2010) or fixed (Todorov, 2007; Toussaint and Storkey, 2006; Rawlik et al., 2013; Fox et al., 2016)  $\pi_0$ . When  $\pi_0 \approx \pi_\theta$ , we can also understand such approaches as approximating the natural policy gradient (Kakade, 2002), which is known to accelerate convergence (Agarwal et al., 2020b). Similarly, regularizing the objective using the Wasserstein distance (Pacchiano et al., 2020) rather than the KL divergence produces updates which approximate those of the Wasserstein natural policy gradient (Moskovitz et al., 2021a). Other approaches can be understood as trust region or proximal point methods (Schulman et al., 2015, 2017; Touati et al., 2020), or even model-based approaches (Grill et al., 2020). It’s also important to note the special case of entropy regularization, where  $\Omega(\theta) = -\mathbb{E}_{s \sim \text{Unif}_\mathcal{S}} \mathbb{H}[\pi_\theta(\cdot|s)] = \mathbb{E}_{s \sim \text{Unif}_\mathcal{S}} \text{KL}[\pi_\theta(\cdot|s); \text{Unif}_\mathcal{A}]$  (where  $\text{Unif}_\mathcal{X}$  denotes the uniform distribution over a space  $\mathcal{X}$ ) which is perhaps the most common form of RPO (Levine, 2018a; Mnih et al., 2016; Schulman et al., 2018; Williams and Peng, 1991; Haarnoja et al., 2018) and has been shown to aid optimization by encouraging exploration and smoothing the objective function landscape (Ahmed et al., 2019).

**Multitask learning** Less common in the literature are policy regularizers designed explicitly for multitask settings. In many multitask RL algorithms which apply RPO, shared task structure is leveraged in other forms (e.g., importance weighting), and the regularizer itself doesn’t reflect shared information (Espeholt et al., 2018; Riedmiller et al., 2018). However, in cases where the penalty is designed for multitask learning, the policy is penalized for deviating from a more general *task-agnostic* default policy meant to en-

code behavior which is generally useful for the *family* of tasks at hand. The use of such a behavioral default is intuitive: by distilling the common structure of the tasks the agent encounters into behaviors which have shown themselves to be useful, optimization on new tasks can be improved with the help of prior knowledge. For example, some approaches (Goyal et al., 2019, 2020) construct a default policy by marginalizing over goals  $g$  for a set of goal-conditioned policies  $\pi_0(a|s) = \sum_g P(g)\pi_\theta(a|s, g)$ . Such partitioning of the input into goal-dependent and goal-agnostic features can be used to create structured internal representations via an information bottleneck (Tishby et al., 2000), shown empirically to improve generalization. In other multitask RPO algorithms, the default policies are derived from a Bayesian framework which views  $\pi_0$  as a prior (Wilson et al., 2007; O’Donoghue et al., 2020). Still other methods learn  $\pi_0$  online through distillation (Hinton et al., 2015) by minimizing  $\text{KL}[\pi_0; \pi]$  with respect to  $\pi_0$  (Galashov et al., 2019a; Teh et al., 2017a). When  $\pi_0$  is preserved across tasks but  $\pi_\theta$  is re-initialized,  $\pi_0$  learns the average behavior across task-specific policies. However, to our knowledge, there has been no investigation of the formal optimization properties of explicitly multitask approaches, and basic questions remain unanswered.

### 3.4 A Basic Theory for Default Policies

At an intuitive level, the question we’d like to explore is: *What properties does a default policy need in order to improve optimization?* By “improve” we refer either to a reduction in the error at convergence with respect to the optimal value function or a reduction in the number of updates required to reach a given error threshold. To begin, we consider perhaps the simplest default: the uniform policy. The proofs for this section are provided in Appendix 3.A.

#### 3.4.1 Log-barrier regularization

For now, we’ll restrict ourselves to the direct softmax parameterization (Eq. (3.2)) with access to exact gradients. Our default is a uniform policy over actions, i.e.:  $\pi_0(a|s) =$

$\text{Unif}_{\mathcal{A}}$ , resulting in the objective

$$\begin{aligned}\mathcal{J}_\lambda(\theta) &:= V^{\pi_\theta}(\mu) - \lambda \mathbb{E}_{s \sim \text{Unif}_{\mathcal{S}}} [\text{KL}(\text{Unif}_{\mathcal{A}}, \pi_\theta(\cdot|s))] \\ &\equiv V^{\pi_\theta}(\mu) + \frac{\lambda}{|\mathcal{S}||\mathcal{A}|} \sum_{s,a} \log \pi_\theta(a|s),\end{aligned}\tag{3.6}$$

where we have dropped terms that are constant with respect to  $\theta$ . Importantly, it's known that even this default policy has beneficial effects on optimization by erecting a log-barrier against low values of  $\pi_\theta(a|s)$ . This barrier prevents gradients from quickly dropping to zero due to exponential scaling, facilitating a polynomial convergence rate<sup>†</sup>. We now briefly restate convergence error and iteration complexity results for this case, first presented by Agarwal et al. (2020b) (Theorem 5.1 and Corollary 5.1, respectively):

**Lemma 3.4.1** (Error bound for log-barrier regularization). *Suppose  $\theta$  is such that  $\|\nabla_\theta \mathcal{J}_\lambda(\theta)\|_2 \leq \epsilon_{\text{opt}}$ , with  $\epsilon_{\text{opt}} \leq \frac{\lambda}{2|\mathcal{S}||\mathcal{A}|}$ . Then we have for all starting state distributions  $\rho$ ,*

$$V^{\pi_\theta}(\rho) \geq V^*(\rho) - \frac{2\lambda}{1-\gamma} \left\| \frac{d_\rho^{\pi^*}}{\mu} \right\|_\infty.$$

We briefly comment on the term  $\left\| \frac{d_\rho^{\pi^*}}{\mu} \right\|_\infty$  (in which the division refers to component-wise division), known as the *distribution mismatch coefficient*, which roughly quantifies the difficulty of the exploration problem faced by the optimization algorithm. In particular,  $\left\| \frac{d_\rho^{\pi^*}}{\mu} \right\|_\infty$  is an upper bound on  $(1-\gamma) \left\| \frac{d_\rho^{\pi^*}}{d_\mu^{\pi_\theta}} \right\|_\infty$ , which directly measures the mismatch between the optimal occupancy measure under the target start state distribution and the current policy's occupancy measure under the training start state distribution. We emphasize that while  $\mu$  is the starting distribution used for training/optimization, the ultimate goal is to perform well on the target starting state distribution  $\rho$ . The iteration complexity is given below.

**Lemma 3.4.2** (Iteration complexity for log-barrier regularization). *Let  $\beta_\lambda := \frac{8\gamma}{(1-\gamma)^3} + \frac{2\lambda}{|\mathcal{S}|}$ . Starting from any initial  $\theta^{(0)}$ , consider the updates Eq. (3.4) with  $\lambda = \frac{\epsilon(1-\gamma)}{2\left\| \frac{d_\rho^{\pi^*}}{\mu} \right\|_\infty}$  and*

---

<sup>†</sup>It remains an open question whether entropy regularization, which is gentler in penalizing low probabilities, produces a polynomial convergence rate.

$\eta = 1/\beta_\lambda$ . Then for all starting state distribution  $\rho$ , we have

$$\min_{t \leq T} \{V^*(\rho) - V^{(t)}(\rho)\} \leq \epsilon$$

$$\text{whenever } T \geq \frac{320|\mathcal{S}|^2|\mathcal{A}|^2}{(1-\gamma)^6\epsilon^2} \left\| \frac{d_\rho^{\pi^*}}{\mu} \right\|_\infty^2.$$

These results will act as useful reference points for the following investigation. At a minimum, we'd like a default policy to provide guarantees that are at least as good as those of log-barrier regularization.

### 3.4.2 Regularization with an $\alpha$ -optimal policy

To understand what properties are required of the default policy, we place an upper-bound on the suboptimality of  $\pi_0$  via the TV distance. For each  $s \in \mathcal{S}$ , we have

$$d_{\text{TV}}(\pi^*(\cdot|s), \pi_0(\cdot|s)) \leq \alpha(s) \quad (3.7)$$

Our regularized objective is

$$\begin{aligned} \mathcal{J}_\lambda^\alpha(\theta) &= V^{\pi_\theta}(\mu) - \lambda \mathbb{E}_{s \sim \text{Unif}_\mathcal{S}} [\text{KL}(\pi_0(\cdot|s), \pi_\theta(\cdot|s))] \\ &\equiv V^{\pi_\theta}(\mu) + \frac{\lambda}{|\mathcal{S}|} \sum_{s,a} \pi_0(a|s) \log \pi_\theta(a|s) \end{aligned} \quad (3.8)$$

for starting state distribution  $\mu \in \Delta(\mathcal{S})$ . We then have

$$\begin{aligned} \frac{\partial \mathcal{J}_\lambda^\alpha(\theta)}{\partial \theta_{s,a}} &= \frac{1}{1-\gamma} d_\mu^{\pi_\theta}(s) \pi_\theta(a|s) A^{\pi_\theta}(s, a) \\ &\quad + \frac{\lambda}{|\mathcal{S}|} (\pi_0(a|s) - \pi_\theta(a|s)). \end{aligned} \quad (3.9)$$

Intuitively, we can see that the gradient of the regularization terms serves to decrease the likelihood of taking actions that have higher probability under the current policy than the default policy.

Our first result presents the error bound for first-order stationary points of the  $\pi_0$ -regularized objective.

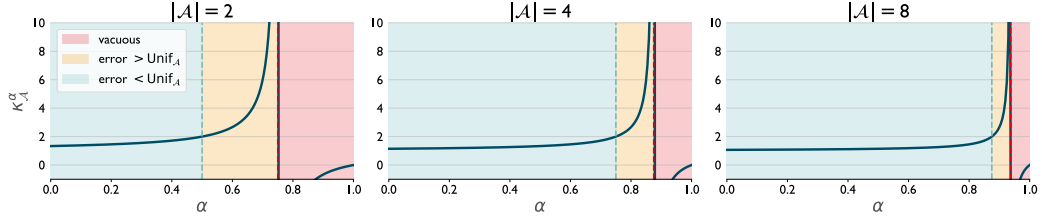
**Lemma 3.4.3** (Error bound for  $\alpha(s)$ -optimal  $\pi_0$ ). *Suppose  $\theta$  is such that  $\|\nabla \mathcal{J}_\lambda^\alpha(\theta)\|_2 \leq \epsilon_{\text{opt}}$ . Then we have that for all starting distributions  $\rho$ :*

$$V^{\pi_\theta}(\rho) \geq V^*(\rho) - \min \left\{ \frac{1}{1-\gamma} \times \right. \\ \left. \mathbb{E}_{s \sim \text{Unif}_\mathcal{S}} \left[ \frac{\epsilon_{\text{opt}} |\mathcal{S}|}{\max \left\{ 1 - \alpha(s) - \frac{\epsilon_{\text{opt}} |\mathcal{S}|}{\lambda}, 0 \right\}} + \lambda \alpha(s) \right] \left\| \frac{d_{\rho}^{\pi^*}}{\mu} \right\|_\infty, \right. \\ \left. \frac{|\mathcal{A}| - 1}{(1-\gamma)^2} \left( \mathbb{E}_{s \sim \mu} [\alpha(s)] \left\| \frac{d_{\rho}^{\pi_\theta}}{\mu} \right\|_\infty + \frac{\epsilon_{\text{opt}} |\mathcal{S}|}{\lambda} \right) \right\}$$

The  $\min\{\cdot\}$  operation above reflects the fact that the value of  $\lambda$  effectively determines whether reward-maximization or the regularization dominates the optimization of Eq. (3.8). Note that a similar effect also applies to log-barrier regularization, but the “high”  $\lambda$  setting is excluded in that instance because as  $\lambda \rightarrow \infty$ ,  $\pi_\theta(a|s) \rightarrow \text{Unif}_\mathcal{A}$ . In this case, however, as  $\alpha \rightarrow 0$ , a high value of  $\lambda$  might be preferable, as it would amount to doing supervised learning with respect to a (nearly) optimal policy. When the reward-maximization dominates, we can see that the error bound becomes vacuous as  $\alpha(s)$  approaches  $\alpha^- := 1 - \epsilon_{\text{opt}} |\mathcal{S}| / \lambda$  from below. In other words, as  $\alpha$  approaches this point, the error can grow arbitrarily high.

In the KL-minimizing case, we can see that as the policy error  $\alpha \rightarrow 0$ , the value gap is given by  $\frac{\epsilon_{\text{opt}} |\mathcal{S}| (|\mathcal{A}| - 1)}{\lambda (1-\gamma)^2}$ . Intuitively, then, as the default policy moves closer to  $\pi^*$ , we can drive the value error to zero as  $\lambda \rightarrow \infty$ . Interestingly, we can also see that as the distribution mismatch  $\left\| \frac{d_{\rho}^{\pi^*}}{\mu} \right\|_\infty \rightarrow 0$ , the influence of the policy distance  $\alpha$  diminishes and the error can again be driven to zero by increasing  $\lambda$ . We leave a more detailed discussion of the impact of the distribution mismatch coefficient to future work. Note that in most practical cases, neither  $\alpha$  nor  $\left\| \frac{d_{\rho}^{\pi^*}}{\mu} \right\|_\infty$  will be low enough to achieve a lower error via KL minimization alone. We will therefore focus on the reward-maximizing case ( $\lambda < 1$ ) for the majority of our further analysis.

Before considering iteration complexity however, it’s also helpful to note that Lemma 3.4.3 generalizes Lemma 3.4.1 given the same upper-bound on  $\epsilon_{\text{opt}}$  as Agarwal et al. (2020b).



**Figure 3.1:** As  $|\mathcal{A}|$  grows, regularizing using  $\pi_0$  with larger  $d_{TV}(\pi^*(\cdot|s), \pi_0(\cdot|s))$  will converge to a lower error than log-barrier regularization. In other words, there is a more forgiving margin of error for the default policy.

**Corollary 3.4.1.** *Suppose  $\theta$  is such that  $\|\nabla \mathcal{J}_\lambda^\alpha(\theta)\|_\infty \leq \epsilon_{\text{opt}}$ , with  $\epsilon_{\text{opt}} \leq \frac{\lambda}{2|\mathcal{S}||\mathcal{A}|}$  and  $\lambda < 1$ . Then we have that for all states  $s \in \mathcal{S}$ ,*

$$V^{\pi_\theta}(\rho) \geq V^*(\rho) - \frac{\mathbb{E}_{s \sim \text{Unif}_\mathcal{S}} [\kappa_\mathcal{A}^\alpha(s)] \lambda}{1 - \gamma} \left\| \frac{d_\rho^{\pi^*}}{\mu} \right\|_\infty$$

where  $\kappa_\mathcal{A}^\alpha(s) = \frac{2|\mathcal{A}|(1-\alpha(s))}{2|\mathcal{A}|(1-\alpha(s))-1}$ .

We can see that in this case, the coefficient  $\kappa_\mathcal{A}^\alpha(s)$  takes on key importance. In particular, we can see that the error-bound becomes vacuous as  $\alpha(s)$  approaches  $\alpha^- = 1 - 1/(2|\mathcal{A}|)$  from below. The error bound is improved with respect to log-barrier regularization when the coefficient  $\kappa_\mathcal{A}^\alpha(s) < 2$ , which occurs for  $\alpha(s) < 1 - 1/|\mathcal{A}|$ . Note that this is the TV distance between the uniform policy and a deterministic optimal policy. These relationships are visualized in Fig. 3.1. We can see that the range of values over which  $\alpha$ -optimal regularization will result in lower error than log-barrier regularization grows as the size of the action space increases. This may have implications for the use of a uniform default policy in continuous action spaces, which we leave to future work.

We can then combine this result with standard results for the convergence of gradient ascent to first order stationary points to obtain the iteration complexity for convergence. First, however, we require an upper bound on the smoothness of  $\mathcal{J}_\lambda^\alpha$  as defined in Eq. (3.8).

**Lemma 3.4.4** (Smoothness of  $\mathcal{J}_\lambda^\alpha$ ). *For the softmax parameterization, we have that*

$$\|\nabla_\theta \mathcal{J}_\lambda^\alpha(\theta) - \nabla_\theta \mathcal{J}_\lambda^\alpha(\theta')\|_2 \leq \beta_\lambda \|\theta - \theta'\|_2$$

where  $\beta_\lambda = \frac{8}{(1-\gamma)^3} + \frac{2\lambda}{|\mathcal{S}|}$ .

We can now bound the iteration complexity.

**Lemma 3.4.5** (Iteration complexity for  $\mathcal{J}_\lambda^\alpha$ ). *Let  $\rho$  be a starting state distribution. Following Lemma 3.4.4, let  $\beta_\lambda = \frac{8\gamma}{(1-\gamma)^3} + \frac{2\lambda}{|\mathcal{S}|}$ . From any initial  $\theta^{(0)}$  and following Eq. (3.4) with  $\eta = 1/\beta_\lambda$  and*

$$\lambda = \frac{\epsilon(1-\gamma)}{\mathbb{E}_{s \sim \text{Unif}_\mathcal{S}} [\kappa_\mathcal{A}^\alpha(s)] \left\| \frac{d_\rho^{\pi^*}}{\mu} \right\|_\infty} < 1,$$

we have

$$\begin{aligned} \min_{t \leq T} \{V^*(\rho) - V^{(t)}(\rho)\} &\leq \epsilon \\ \text{whenever } T &\geq \frac{80 \mathbb{E}_{s \sim \text{Unif}_\mathcal{S}} [\kappa_\mathcal{A}^\alpha(s)]^2 |\mathcal{S}|^2 |\mathcal{A}|^2}{(1-\gamma)^6 \epsilon^2} \left\| \frac{d_\rho^{\pi^*}}{\mu} \right\|_\infty^2. \end{aligned}$$

It is also natural to consider the case in which  $\pi_0$  is used as an initialization for  $\pi_\theta$ .

**Corollary 3.4.2.** *Given the same assumptions as Lemma 3.4.5, if the initial policy is chosen to be  $\pi_0$ , i.e.,  $\pi_{\theta^{(0)}} = \pi_0$  where  $\pi_0(\cdot|s)$  is  $\alpha(s)$ -optimal with respect to  $\pi^*(\cdot|s) \forall s$ , then*

$$\begin{aligned} \min_{t \leq T} \{V^*(\rho) - V^{(t)}(\rho)\} &\leq \epsilon \\ \text{whenever } T &\geq \frac{320 |\mathcal{A}|^2 |\mathcal{S}|^2}{\epsilon^2 (1-\gamma)^7} \left\| \frac{d_\rho^{\pi^*}}{\mu} \right\|_\infty^2 \left\| \frac{1}{\mu} \right\|_\infty \mathbb{E}_{s \sim \mu} [\alpha(s)]. \end{aligned}$$

In the case of random initialization, note that when  $\alpha(s) = \alpha = 1 - 1/|\mathcal{A}|$ ,  $\mathbb{E} \kappa_\mathcal{A}^\alpha(s) = 2$ , recovering the iteration complexity for log-barrier regularization, as expected. We also see that as the error  $\alpha$  moves higher or lower than  $1 - 1/|\mathcal{A}|$ , the iteration complexity grows or shrinks quadratically. Therefore, a default policy within this range will not only linearly reduce the error at convergence, but will also quadratically increase the rate at which that error is reached. When the initial policy is  $\pi_0$ , the iteration complexity depends on the factor  $\mathbb{E}_{s \sim \text{Unif}_\mathcal{S}} [\alpha(s)]$ . Hence, for good initialization,  $\alpha$  is small, resulting in fewer iterations. The natural question, then, is how to find such a default policy, with high probability, for some family of tasks.



### 3.5 Extension to Multitask Learning

The results above provide guidance for the construction of default policies in the multi-task setting. The key insight is that if the optimal policies for the tasks drawn from a given task distribution have commonalities, the agent can use the optimal policies it learns from previous tasks to construct a useful  $\pi_0$ . More precisely, consider a distribution  $\mathbb{P}_{\mathcal{M}}$  over a family of tasks  $\mathcal{M} := \{M_k\}$ . (The simplest example of such a distribution is a categorical distribution over a discrete set of tasks, although continuous distributions over MDPs are possible.) We assume only that the tasks have shared state and action spaces  $\mathcal{S}$  and  $\mathcal{A}$ , and we denote their optimal deterministic policies by  $\{\pi_k^*\}$ . We assume that the other task components (reward function, transition distribution, etc.) are independent. Then by Corollary 3.4.1 and Lemma 3.4.5, if the TV barycenter at a given state  $s$ , given by

$$\pi_0(\cdot|s) = \underset{\pi}{\operatorname{argmin}} \mathbb{E}_{M_k \sim \mathbb{P}_{\mathcal{M}}} [d_{\text{TV}}(\pi_k^*(\cdot|s), \pi(\cdot|s))] \quad (3.10)$$

is such that  $\mathbb{E}[d_{\text{TV}}(\pi_k^*(\cdot|s), \pi_0(\cdot|s))] < 1 - 1/|\mathcal{A}|$ , then regularizing with  $\pi_0$  will, in expectation, result in faster convergence and lower error than using a uniform distribution. Crucially, when there is a lack of shared structure, which in this particular approach is manifested as a lack of agreement among optimal policies,  $\pi_0(\cdot|s)$  collapses to  $\text{Unif}_{\mathcal{A}}$ . Therefore, in the worst case, regularizing with  $\pi_0(\cdot|s)$  can do no worse than log-barrier regularization, which already enjoys polynomial iteration complexity.

When the optimal policies  $\{\pi_k^*\}$  are deterministic, the following result gives a convenient expression for the TV-barycenter policy:

**Lemma 3.5.1** (TV barycenter). *Let  $\mathbb{P}_{\mathcal{M}}$  be a distribution over tasks  $\mathcal{M} = \{M_k\}$ , each with a deterministic optimal policy  $\pi_k^* : \mathcal{S} \rightarrow \mathcal{A}$ . Define the average optimal action as*

$$\xi(s, a) := \mathbb{E}_{M_k \sim \mathbb{P}_{\mathcal{M}}} [\mathbb{1}(\pi_k^*(s) = a)]. \quad (3.11)$$

*Then, the TV barycenter  $\pi_0(\cdot|s)$  defined in Eq. (3.10) is given by a greedy policy over  $\xi$ , i.e.,  $\pi_0(a|s) = \delta(a \in \operatorname{argmax}_{a' \in \mathcal{A}} \xi(s, a'))$ , where  $\delta(\cdot)$  is the Dirac delta distribution.*

The proof, along with the rest of the proofs for this section, is provided in Ap-

pendix 3.B. Interestingly, this result also holds for the KL barycenter, which we show in Appendix Lemma 3.B.4. Because the average optimal action  $\xi$  is closely related to a recently-proposed computational model of *habit formation* in cognitive psychology (Miller et al., 2016b), from now on we refer to it as the *habit function* for task family  $\mathcal{M}$ . When the agent has observed  $K$  tasks sampled from  $\mathbb{P}_{\mathcal{M}}$ ,  $\xi$  is approximated by the sample average  $\hat{\xi}(s, a) = \frac{1}{K} \sum_{k=1}^K \mathbb{1}(\pi_k^*(s) = a)$  provided that the optimal policies  $\pi_k^*$  are available. In practice, however, the agent only has access to an approximation  $\tilde{\pi}_k$  of  $\pi_k^*$  obtained, for instance, through the use of a learning algorithm  $A$ , such as Appendix Algorithm 2. Hence,  $\hat{\xi}(s, a)$  is instead given by  $\hat{\xi}(s, a) = \frac{1}{K} \sum_{k=1}^K \mathbb{1}(\tilde{\pi}_k(s) = a)$  which induces an approximate barycenter  $\hat{\pi}_0$  by taking the greedy policy over  $\hat{\xi}$ . The following result provides the iteration complexity for the multitask setting when using  $\hat{\pi}_0$  as the default policy.

**Lemma 3.5.2** (Multitask iteration complexity). *Let  $M_k \sim \mathbb{P}_{\mathcal{M}}$  and denote by  $\pi_k^* : \mathcal{S} \rightarrow \mathcal{A}$  its optimal policy. Denote by  $T_k$  the number of iterations to reach  $\epsilon$ -error for  $M_k$  in the sense that:*

$$\min_{t \leq T_k} \{V^{\pi_k^*}(\rho) - V^{(t)}(\rho)\} \leq \epsilon.$$

*Set  $\lambda, \beta_\lambda$ , and  $\eta$  as in Lemma 3.4.5. From any initial  $\theta^{(0)}$ , and following Eq. (3.4),  $\mathbb{E}_{M_k \sim \mathbb{P}_{\mathcal{M}}} [T_k]$  satisfies:*

$$\mathbb{E}_{M_k \sim \mathbb{P}_{\mathcal{M}}} [T_k] \geq \frac{80|\mathcal{A}|^2|\mathcal{S}|^2}{\epsilon^2(1-\gamma)^6} \mathbb{E}_{\substack{M_k \sim \mathbb{P}_{\mathcal{M}} \\ s \sim \text{Unif}_{\mathcal{S}}}} \left[ \kappa_{\mathcal{A}}^{\alpha_k}(s) \left\| \frac{d_{\rho}^{\pi_k^*}}{\mu} \right\|_{\infty}^2 \right],$$

*where  $\alpha_k(s) := d_{\text{TV}}(\pi_k^*(\cdot|s), \hat{\pi}_0(\cdot|s))$ . If  $\hat{\pi}_0$  is also used for initialization, then  $\mathbb{E}_{M_k \sim \mathbb{P}_{\mathcal{M}}} [T_k]$  satisfies:*

$$\mathbb{E}_{M_k \sim \mathbb{P}_{\mathcal{M}}} [T_k] \geq \frac{320|\mathcal{A}|^2|\mathcal{S}|^2}{\epsilon^2(1-\gamma)^7} \left\| \frac{1}{\mu} \right\|_{\infty}^3 \mathbb{E}_{\substack{M_k \sim \mathbb{P}_{\mathcal{M}} \\ s \sim \mu}} [\alpha_k(s)],$$

Lemma 3.5.2 characterizes the average iteration complexity over tasks when using  $\hat{\pi}_0$  as a default policy. In particular, when the learning algorithm is also initialized with

$\hat{\pi}_0$ , we obtain that the average number of iterations to reach  $\epsilon$  accuracy is proportional to the expected TV distance of  $\hat{\pi}_0$  to the optimal policies  $\pi_k^*$  for tasks  $\{M_k\} \sim \mathbb{P}_{\mathcal{M}}$ . We expect this distance to approach  $\mathbb{E}[d_{\text{TV}}(\pi_0(\cdot|s), \pi_k^*(\cdot|s))]$  as the number of tasks increases and  $\tilde{\pi}_k$  become more accurate. Note that even in this case, the regularization is *still* required to assure polynomial convergence. To provide a precise quantification, we let  $\tilde{\pi}_k(\cdot|s)$  be, on average,  $\zeta(s)$ -optimal in state  $s$  across tasks  $\{M_k\}$ , i.e.  $\mathbb{E}_{M_k \sim \mathcal{M}}[d_{\text{TV}}(\tilde{\pi}_k(\cdot|s), \pi_k^*(\cdot|s))] \leq \zeta(s)$  for some  $\zeta(s) \in [0, 1]$ . The following lemma quantifies how close  $\hat{\pi}_0$  grows to the TV barycenter of  $\{\pi_k^*\}_{k=1}^K$  as  $K \rightarrow \infty$ :

**Lemma 3.5.3** (Barycenter concentration). *Let  $\delta$  be  $0 < \delta < 1$ . Then with probability higher than  $1 - \delta$ , for all  $s \in \mathcal{S}$ , it holds that:*

$$\begin{aligned} & |\mathbb{E}_{M_k \sim \mathbb{P}_{\mathcal{M}}}[d_{\text{TV}}(\pi_k^*(\cdot|s), \hat{\pi}_0(\cdot|s)) - d_{\text{TV}}(\pi_k^*(\cdot|s), \pi_0(\cdot|s))]| \\ & \leq 2\zeta(s) + \sqrt{\frac{2 \log(\frac{2}{\delta})}{K}} + 2C\sqrt{\frac{|\mathcal{A}|}{K}}, \end{aligned}$$

for some constant  $C$  that depends only on  $|\mathcal{A}|$ .

In other words, in order to produce a default policy which improves over log-barrier regularization as  $K \rightarrow \infty$ , the margin of error for the trained policies is half that which is required for the default policy.

In practice, due to the epistemic uncertainty about the task family early in training (in other words, when only a few tasks have been sampled), regularizing using  $\hat{\pi}_0$  risks misleading  $\pi_\theta$  by placing all of the default policy's mass on a sub-optimal action. We can therefore define  $\hat{\pi}_0$  using a softmax  $\hat{\pi}_0(a|s) \propto \exp(\hat{\xi}(s, a)/\beta(k))$  with some temperature parameter  $\beta(k)$  which tends to zero as the number of observed tasks  $k$  approaches infinity. Therefore,  $\pi_0$  converges to the optimal default policy in the limit. This suggests the simple approach to multitask RPO presented in Algorithm 3 which we call *total variation policy optimization* (TVPO).

Note that if  $\mathbb{P}_{\mathcal{M}}$  is non-stationary, the moving average in Line 8 can be changed to an exponentially weighted moving average to place more emphasis on recent tasks.

**Algorithm 3** TV Policy Optimization (TVPO)

- 
- 1: **Input** Task set  $\mathcal{M}$ , policy class  $\Theta$ , fixed- $\pi_0$  RPO algorithm  $A(M, \Theta, \pi_0, \lambda)$ , as in Appendix Algorithm 2
  - 2: initialize  $\pi_0(\cdot|s) = \xi^{(0)}(s, \cdot) = \text{Unif}_{\mathcal{A}} \forall s \in \mathcal{S}$
  - 3: **for** iteration  $k = 1, 2, \dots$  **do**
  - 4:   Sample a task  $M^{(k)} \sim \mathbb{P}_{\mathcal{M}}$
  - 5:   Solve the task:  $\tilde{\theta}^{(k)} = A(M_k, \Theta, \pi_0^{(k-1)}, \lambda)$
  - 6:   Set  $\tilde{\pi}_k \leftarrow \pi_{\tilde{\theta}^{(k)}}$ .
  - 7:   Update habit moving average  $\forall (s, a) \in \mathcal{S} \times \mathcal{A}$ :

$$\xi^{(k)}(s, a) \leftarrow \frac{k-1}{k} \xi^{(k-1)}(s, a) + \frac{1}{k} \mathbb{1} \left( a \in \operatorname{argmax}_{a'} \tilde{\pi}_k(a'|s) \right)$$

- 8:   Update default policy  $\forall (s, a) \in \mathcal{S} \times \mathcal{A}$ :

$$\pi_0^{(k)}(a|s) \propto \exp(\xi^{(k)}(s, a)/\beta(k))$$

- 9: **end for**
- 

### 3.6 Understanding the Literature

As stated previously, many approaches to multitask RPO in the literature learn a default policy  $\pi_0(a|s; \phi)$  parameterized by  $\phi$  via gradient descent on the KL divergence (Galashov et al., 2019a; Teh et al., 2017a), e.g., via

$$\phi = \operatorname{argmin}_{\phi'} \mathbb{E}_{s \sim \text{Unif}_{\mathcal{S}}} [\text{KL}(\pi_{\theta}(\cdot|s), \pi_0(\cdot|s; \phi))]. \quad (3.12)$$

The idea is that by updating  $\phi$  across multiple tasks,  $\pi_0$  will acquire the average behaviors of the goal-directed policies  $\pi_{\theta}$ . This objective can be seen as an approximation of Eq. (3.10) in which we can view the use of the KL as a relaxation of the TV distance:

$$\begin{aligned} \pi_0(\cdot|s) &= \operatorname{argmin}_{\pi} \mathbb{E}_{M_k \sim \mathbb{P}_{\mathcal{M}}} [d_{\text{TV}}(\pi_k^*(\cdot|s), \pi(\cdot|s))] \\ &\leq \operatorname{argmin}_{\pi} \mathbb{E}_{M_k \sim \mathbb{P}_{\mathcal{M}}} [d_{\text{TV}}(\pi_k^*(\cdot|s), \pi(\cdot|s))^2] \\ &\leq \operatorname{argmin}_{\pi} \mathbb{E}_{M_k \sim \mathbb{P}_{\mathcal{M}}} [\text{KL}(\pi_k^*(\cdot|s), \pi(\cdot|s))], \end{aligned}$$

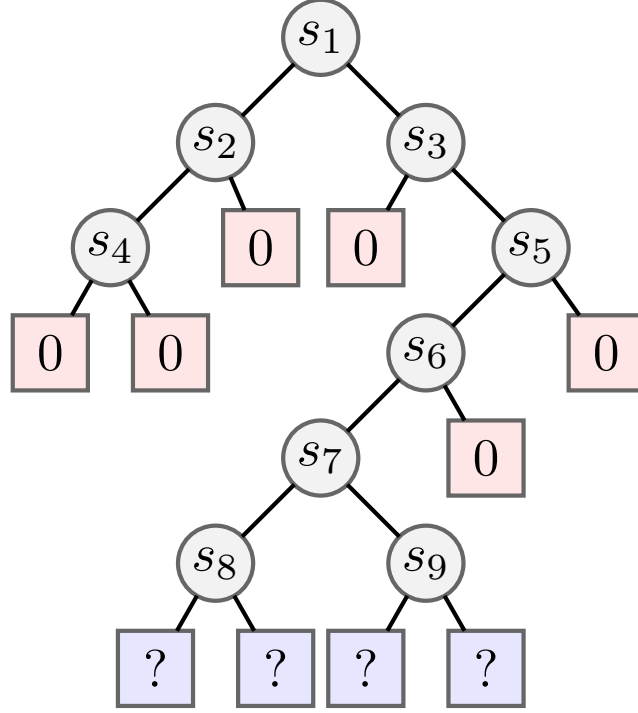
where the first inequality is due to Jensen’s inequality and the second is due to Pollard (2000) and where  $\pi_\theta(\cdot|s) \approx \pi^*(\cdot|s)$ . The use of the KL is natural due to its easy computation and differentiability, however the last approximation is crucial. By distilling  $\pi_0$  from  $\pi_\theta$  via Eq. (3.12) from the outset of each task, there is an implicit assumption that  $\pi_\theta \approx \pi^*$  even early in training. This is a source of suboptimality, as we discuss in Section 3.7.

### 3.7 Experiments

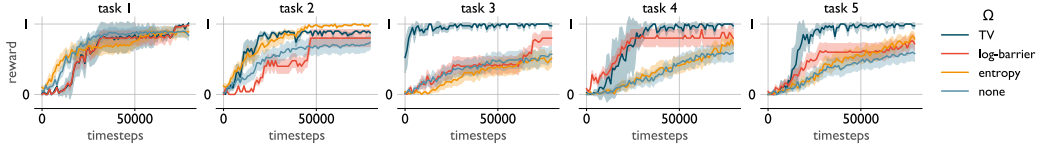
We now study the implications of these ideas in a simple empirical setting: a family of tasks whose state space follows the tree structure shown in Fig. 3.2. In these tasks, the agent starts at the root  $s_1$  and at each timestep chooses whether to proceed down its left subtree or right subtree ( $|\mathcal{A}| = 2$ ). The episode ends when the agent reaches a leaf node. In this setup, there is zero reward in all states other than the leaf nodes marked with a ‘?’, for which one or more are randomly assigned a reward of 1 for each draw from the task distribution. To encourage sparsity, the number of rewards is drawn from a geometric distribution with success parameter  $p = 0.5$ .

One training run consisted of five rounds of randomly sampling a task and solving it. Despite the simplicity of this environment, we found that it could prove surprisingly difficult for many algorithms to solve consistently. As can be seen in Fig. 3.2, the key structural consistency in this task is that every optimal policy makes the same choices in states  $\{s_1, s_3, s_5, s_6\}$ , with necessary exploration limited to the lower subtree rooted at  $s_7$ .

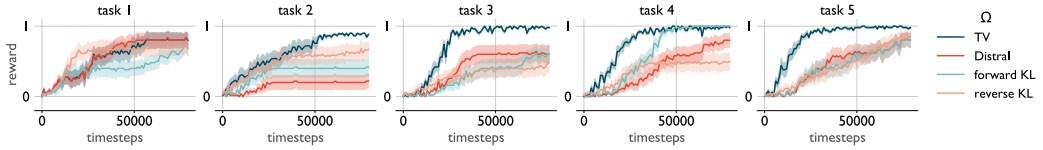
For comparison, we selected RPO approaches with both *fixed* default policies (LOG-BARRIER, ENTROPY, and NONE) and *learned* default policies: DISTRAL ( $-\text{KL}(\pi_\theta, \pi_0) + \text{H}[\pi_\theta]$ ; (Teh et al., 2017a)), FORWARD KL ( $-\text{KL}(\pi_0, \pi_\theta)$ ), and REVERSE KL ( $-\text{KL}(\pi_\theta, \pi_0)$ ). To make the problem more challenging for the learned default policies, the reward distribution was made sparser by setting  $p = 0.7$ . Each approach was applied over 20 random seeds, with results plotted in Fig. 3.3 (fixed  $\pi_0$ ) and Fig. 3.4 (learned  $\pi_0$ ). Hyperparameters were kept constant across methods (further experimental details can be found in Appendix 3.C). We see that TVPO most consistently solves



**Figure 3.2:** A tree environment. Each task in the family randomly distributes rewards among leaves marked with a ‘?’. All other states result in zero reward.



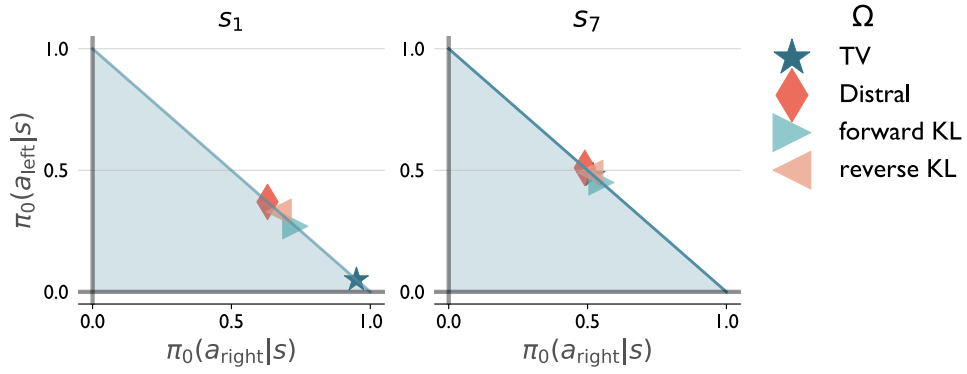
**Figure 3.3:** Fixed  $\pi_0$  baselines. Results are averaged over 20 seeds, with the shaded region denoting one standard deviation.



**Figure 3.4:** Learned  $\pi_0$  baselines. Results are averaged over 20 seeds, with the shaded region denoting one standard deviation.

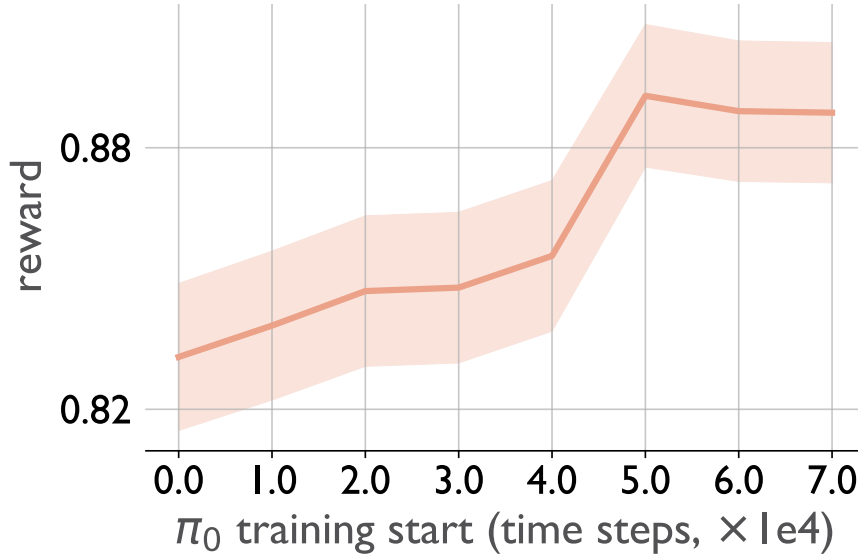
the tasks. This is not surprising, as  $\mathbb{E}_{M_k \sim \mathbb{P}_{\mathcal{M}}} [\alpha_k(s)] = 0$  for all states en route to the rewarded leaves until  $s_7$ . Thus,  $\hat{\pi}_0(\cdot|s) \rightarrow \pi_k^*(s)$  quickly for these states as the number of tasks grows. This dramatically reduces the size of the exploration problem for TVPO, confining it to the subtree rooted at  $s_7$ .

To gain a better understanding of the results and the learned default policies, we



**Figure 3.5:** Learned default policies in states  $s_1$  and  $s_7$  after five tasks. In the simplex for  $s_7$ , the marker for TVPO is behind the markers for the other methods.

plotted the average default policies for each method on the 2-simplex for states  $s_1$  and  $s_7$  in Fig. 3.5. For all tasks in the family, the optimal policy goes right in  $s_1$ , while, on average, reward could be located in either subtree rooted at  $s_7$ . This is reflected in the default policies, which prefer right in  $s_1$  and are close to uniform in  $s_7$ . There is a notable difference, however, in that the KL-, gradient-based methods are much less deterministic in  $s_1$ . The critical difference is that the KL-based methods are trained online via distillation from suboptimal  $\pi_\theta \not\approx \pi^*$ . Early in training,  $\pi_\theta$  is inconsistent across tasks and runs, resulting in a more uniform target for  $\pi_0$ . This delays its convergence across tasks to the shared TV/KL barycenter. To test this effect empirically, we repeated the same experiment with REVERSE KL but started training  $\pi_0$  progressively later within each task.



**Figure 3.6:** Delayed training of  $\pi_0$  improves performance.

Fig. 3.6 depicts the average final reward across tasks for different time steps at which the default policy began training. Note, however, that  $\pi_0$  is still used to regularize  $\pi_\theta$ , it just isn't updated based on  $\pi_\theta$  until  $\pi_\theta$  is a reasonable approximation of  $\pi^*$ . We can see that, as predicted, delaying training within each task improves performance. There is a slight drop in performance if  $\pi_0$  does not have a sufficient number of updates at the end of training.

### 3.8 Discussion

In this work, we introduce novel, more general bounds on the error and iteration complexity of KL-regularized policy optimization. We then show how these bounds apply to the multitask setting, showing the first formal results for a popular class of algorithms and deriving a novel multitask RPO algorithm with formal guarantees. We then demonstrate the implications of our findings in a simple experimental setting. Taken together, we believe our results provide preliminary answers to our guiding questions for KL-regularized RPO:

**A1:** *In order to provide benefit, a default policy must in expectation be at least as close to the optimal policy as the uniform policy.*

**A2:** *For KL-regularized RPO to provide a measurable benefit, a group of tasks must induce*



*optimal policies which agree over some portion of the state space.*

There are several important ramifications for future work. First, these results imply an algorithm-dependent definition of task families, such that a group of tasks can be considered a *family* for a given algorithm if that algorithm can leverage their shared properties to improve optimization. For RPO algorithms, then, the choice of divergence measure, default policy, and distribution space implicitly determines task groupings. As an example, the particular class of algorithm we investigate here is sensitive to state-dependent similarities in the space of optimal policies for a group of tasks. There are a multitude of other forms of shared structure which alternative approaches can leverage, however, such as consistent transition dynamics (Barreto et al., 2020; Moskovitz et al., 2022c) or even structure in an abstract *behavioral* space (Pacchiano et al., 2020; Moskovitz et al., 2021a; Agarwal et al., 2021). From Fig. 3.1, it is important to observe that the performance gains relative to log-barrier regularization are, ultimately, relatively small. It may be necessary to develop algorithms with stronger assumptions about the task family and/or sensitivity to a different form of structure in order to drive further improvement. Conducting an effective taxonomy of algorithms and associated task families will be crucial for the development of practical real-world agents.

We also believe this work provides a formal framework for settings where forward transfer is possible during lifelong learning scenarios with multiple interrelated tasks (Lopez-Paz and Ranzato, 2017). While we tested these ideas in a toy setting, the underlying theory has implications for state-of-the-art deep RL methods. When state and action spaces grow large, however,  $\pi_0$  is necessarily represented by a restricted policy class. Both TVPO and the learned  $\pi_0$  baseline methods can be scaled to this domain, with TVPO’s  $\pi_0$  being trained online to predict the next action taken by  $\pi_\theta$ . One useful lesson which equally applies to KL-based methods, however, is that it’s preferable from an optimization standpoint to distill  $\pi_0$  from  $\pi_\theta$  only late in training when  $\pi_\theta \approx \pi^*$ . Given the promise of this general class of methods, we hope that the insight garnered by these results will help propel the field towards more robust and general algorithms.

## Appendix

### Appendix 3.A: Single-task Analysis

We now consider the error bound for  $\pi_0$  such that  $d_{\text{TV}}(\pi^*(\cdot|s), \pi_0(\cdot|s)) \leq \alpha(s) \forall s \in \mathcal{S}$ .

**Lemma 3.4.3** (Error bound for  $\alpha(s)$ -optimal  $\pi_0$ ). *Suppose  $\theta$  is such that  $\|\nabla \mathcal{J}_\lambda^\alpha(\theta)\|_2 \leq \epsilon_{\text{opt}}$ . Then we have that for all starting distributions  $\rho$ :*

$$V^{\pi_\theta}(\rho) \geq V^*(\rho) - \min \left\{ \frac{1}{1-\gamma} \times \right. \\ \left. \mathbb{E}_{s \sim \text{Unif}_\mathcal{S}} \left[ \frac{\epsilon_{\text{opt}} |\mathcal{S}|}{\max \left\{ 1 - \alpha(s) - \frac{\epsilon_{\text{opt}} |\mathcal{S}|}{\lambda}, 0 \right\}} + \lambda \alpha(s) \right] \left\| \frac{d_\rho^{\pi^*}}{\mu} \right\|_\infty, \right. \\ \left. \frac{|\mathcal{A}| - 1}{(1-\gamma)^2} \left( \mathbb{E}_{s \sim \mu} [\alpha(s)] \left\| \frac{d_\rho^{\pi_\theta}}{\mu} \right\|_\infty + \frac{\epsilon_{\text{opt}} |\mathcal{S}|}{\lambda} \right) \right\}$$

*Proof.* Let's assume that  $\pi^*$  is a deterministic policy. By [Puterman \(1994\)](#) such an optimal policy always exists for an MDP. We'll use the notation  $a^*(s)$  to denote the optimal action at state  $s$ . This, combined with the assumption that  $d_{\text{TV}}(\pi^*(\cdot|s), \pi_0(\cdot|s)) \leq \alpha(s)$  for all  $s \in \mathcal{S}$ , tells us that  $\pi_0(a^*(s)|s) \geq \pi^*(a^*(s)|s) - \alpha(s) = 1 - \alpha(s)$ . Similarly, for  $a \neq a^*(s)$ ,  $\pi_0(a|s) \leq \alpha(s)$ . Using this, we can start by showing that whenever  $A^{\pi_\theta}(s, a^*(s)) \geq 0$  we can lower bound  $\pi_\theta(a^*(s)|s)$  for all  $s$ .

The gradient norm assumption  $\|\nabla \mathcal{J}_\lambda^\alpha(\theta)\|_\infty \leq \epsilon_{\text{opt}}$  implies that for all  $s, a$ :

$$\epsilon_{\text{opt}} \geq \frac{\partial \mathcal{J}_\lambda^\alpha(\theta)}{\partial \theta_{s,a}} = \frac{1}{1-\gamma} d_\mu^{\pi_\theta}(s) \pi_\theta(a|s) A^{\pi_\theta}(s, a) + \frac{\lambda}{|\mathcal{S}|} (\pi_0 - \pi_\theta(a|s))$$

In particular for all  $s$ ,

$$\begin{aligned}
\epsilon_{\text{opt}} &\geq \frac{\partial \mathcal{J}_\lambda^\alpha(\theta)}{\partial \theta_{s,a^*(s)}} \stackrel{(i)}{\geq} \frac{1}{1-\gamma} d_\mu^{\pi_\theta}(s) \pi_\theta(a^*(s)|s) A^{\pi_\theta}(s, a^*(s)) \\
&\quad + \frac{\lambda}{|\mathcal{S}|} (\pi^*(a^*(s)|s) - \alpha(s) - \pi_\theta(a^*(s)|s)) \\
&= \frac{1}{1-\gamma} d_\mu^{\pi_\theta}(s) \pi_\theta(a^*(s)|s) A^{\pi_\theta}(s, a^*(s)) + \frac{\lambda}{|\mathcal{S}|} (1 - \alpha(s) - \pi_\theta(a^*(s)|s))
\end{aligned} \tag{3.13}$$

And therefore if  $A^{\pi_\theta}(s, a^*(s)) \geq 0$ ,

$$\epsilon_{\text{opt}} \geq \frac{\lambda}{|\mathcal{S}|} (1 - \alpha(s) - \pi_\theta(a^*(s)|s))$$

Thus,

$$\pi_\theta(a^*(s)|s) \geq 1 - \alpha(s) - \frac{\epsilon_{\text{opt}} |\mathcal{S}|}{\lambda}. \tag{3.14}$$

We then have

$$\begin{aligned}
A^{\pi_\theta}(s, a^*(s)) &\leq \frac{1-\gamma}{d_\mu^{\pi_\theta}(s)} \left( \frac{1}{\pi_\theta(a^*(s)|s)} \frac{\partial \mathcal{J}_\lambda^\alpha(\theta)}{\partial \theta_{s,a}} \right. \\
&\quad \left. - \frac{\lambda}{|\mathcal{S}|} \frac{1}{\pi_\theta(a^*(s)|s)} (1 - \alpha(s) - \pi_\theta(a^*(s)|s)) \right) \\
&= \frac{1-\gamma}{d_\mu^{\pi_\theta}(s)} \left( \frac{1}{\pi_\theta(a^*(s)|s)} \frac{\partial \mathcal{J}_\lambda^\alpha(\theta)}{\partial \theta_{s,a}} + \frac{\lambda}{|\mathcal{S}|} \left( 1 - \frac{1-\alpha(s)}{\pi_\theta(a^*(s)|s)} \right) \right) \\
&\stackrel{(i)}{\leq} \frac{1}{\mu(s)} \left( \frac{1}{\max \left\{ 1 - \alpha(s) - \frac{\epsilon_{\text{opt}} |\mathcal{S}|}{\lambda}, 0 \right\}} \cdot \epsilon_{\text{opt}} \right. \\
&\quad \left. + \frac{\lambda}{|\mathcal{S}|} (1 - (1 - \alpha(s))) \right) \\
&\leq \frac{1}{\mu(s)} \left( \frac{1}{\max \left\{ (1 - \alpha(s)) - \frac{\epsilon_{\text{opt}} |\mathcal{S}|}{\lambda}, 0 \right\}} \cdot \epsilon_{\text{opt}} + \frac{\lambda}{|\mathcal{S}|} \alpha(s) \right)
\end{aligned}$$

where (i) follows because  $d_\mu^{\pi_\theta}(s) \geq (1-\gamma)\mu(s)$ ,  $\frac{\partial \mathcal{J}_\lambda^\alpha(\theta)}{\partial \theta_{s,a}} \leq \epsilon_{\text{opt}}$  and  $\max(1 - \alpha(s) - \frac{\epsilon_{\text{opt}} |\mathcal{S}|}{\lambda}, 0) \leq \pi_\theta(a^*(s)|s) \leq 1$ . Then applying the performance difference lemma

(Kakade and Langford, 2002) gives

$$\begin{aligned}
V^*(\rho) - V^{\pi_\theta}(\rho) &= \frac{1}{1-\gamma} \sum_{s,a} d_\rho^{\pi^*}(s) \pi^*(a|s) A^{\pi_\theta}(s, a) \\
&= \frac{1}{1-\gamma} \sum_s d_\rho^{\pi^*}(s) A^{\pi_\theta}(s, a^*(s)) \\
&\leq \frac{1}{1-\gamma} \sum_s d_\rho^{\pi^*}(s) A^{\pi_\theta}(s, a^*(s)) \mathbb{1}(A^{\pi_\theta}(s, a^*(s)) \geq 0) \\
&\leq \frac{1}{1-\gamma} \sum_s \frac{d_\rho^{\pi^*}(s)}{\mu(s)} \left( \frac{1}{\left\{1 - \alpha(s) - \frac{\epsilon_{\text{opt}}|\mathcal{S}|}{\lambda}, 0\right\}} \cdot \epsilon_{\text{opt}} + \frac{\lambda}{|\mathcal{S}|} \alpha(s) \right) \\
&\quad \cdot \mathbb{1}(A^{\pi_\theta}(s, a^*(s)) \geq 0) \\
&\leq \frac{1}{1-\gamma} \mathbb{E}_{s \sim \text{Unif}_{\mathcal{S}}} \left[ \frac{\epsilon_{\text{opt}}|\mathcal{S}|}{\left\{1 - \alpha(s) - \frac{\epsilon_{\text{opt}}|\mathcal{S}|}{\lambda}, 0\right\}} + \lambda \alpha(s) \right] \left\| \frac{d_\rho^{\pi^*}}{\mu} \right\|_\infty.
\end{aligned}$$

Now let's relate the values of  $\pi^*$  and  $\pi_\theta$ . We will again apply the performance difference lemma, this time in the other direction:

$$\begin{aligned}
V^{\pi_\theta}(\rho) - V^*(\rho) &= \frac{1}{1-\gamma} \sum_{s,a} d_\rho^{\pi_\theta}(s) \pi_\theta(a|s) A^{\pi^*}(s, a) \\
&\stackrel{(i)}{=} \frac{1}{1-\gamma} \sum_s \left( \sum_{a \neq a^*(s)} d_\rho^{\pi_\theta}(s) \pi_\theta(a|s) A^{\pi^*}(s, a) \right) \\
&\stackrel{(ii)}{\geq} \frac{-1}{1-\gamma} \sum_s d_\rho^{\pi_\theta}(s) \left( \alpha(s) + \frac{\epsilon_{\text{opt}}|\mathcal{S}|}{\lambda} \right) \frac{|\mathcal{A}| - 1}{1-\gamma} \\
&\stackrel{(iii)}{=} -\frac{|\mathcal{A}| - 1}{1-\gamma} \left( \sum_s \frac{d_\rho^{\pi_\theta}(s)}{1-\gamma} \alpha(s) + \frac{\epsilon_{\text{opt}}|\mathcal{S}|}{\lambda} \sum_s \frac{d_\rho^{\pi_\theta}(s)}{1-\gamma} \right) \\
&= -\frac{|\mathcal{A}| - 1}{1-\gamma} \left( \sum_s \frac{d_\rho^{\pi_\theta}(s)}{1-\gamma} \alpha(s) \right) - \frac{(|\mathcal{A}| - 1)\epsilon_{\text{opt}}|\mathcal{S}|}{\lambda(1-\gamma)^2} \\
&= -\frac{|\mathcal{A}| - 1}{(1-\gamma)^2} \left( \sum_s d_\rho^{\pi_\theta}(s) \alpha(s) \right) - \frac{(|\mathcal{A}| - 1)\epsilon_{\text{opt}}|\mathcal{S}|}{\lambda(1-\gamma)^2} \\
&= -\frac{|\mathcal{A}| - 1}{(1-\gamma)^2} \left( \sum_s \frac{d_\rho^{\pi_\theta}(s)}{\mu(s)} \mu(s) \alpha(s) \right) - \frac{(|\mathcal{A}| - 1)\epsilon_{\text{opt}}|\mathcal{S}|}{\lambda(1-\gamma)^2} \\
&\geq -\frac{|\mathcal{A}| - 1}{(1-\gamma)^2} \left\| \frac{d_\rho^{\pi_\theta}}{\mu} \right\|_\infty \left( \sum_s \mu(s) \alpha(s) \right) - \frac{(|\mathcal{A}| - 1)\epsilon_{\text{opt}}|\mathcal{S}|}{\lambda(1-\gamma)^2}
\end{aligned}$$

where (i) is due to the fact that  $A^{\pi^*}(s, a^*(s)) = 0$ , (ii) is due to the fact that  $A^{\pi^*}(s, a)$  for  $a \neq a^*$  is lower-bounded by  $-1/(1-\gamma)$  and Eq. (3.30), and (iii) is because  $\sum_s d_\rho^{\pi_\theta}(s) = 1$ . Therefore,

$$V^{\pi_\theta}(\rho) + \frac{|\mathcal{A}| - 1}{(1-\gamma)^2} \left( \mathbb{E}_{s \sim \mu} [\alpha(s)] \left\| \frac{d_\rho^{\pi^*}}{\mu} \right\|_\infty + \frac{\epsilon_{\text{opt}} |\mathcal{S}|}{\lambda} \right) \geq V^*(\rho).$$

This completes the proof.  $\square$

We now present a comparatively looser bound which applies the same upper bound on the norm of the gradient used by Agarwal et al. (2020b).

**Corollary 3.4.1.** *Suppose  $\theta$  is such that  $\|\nabla \mathcal{J}_\lambda^\alpha(\theta)\|_\infty \leq \epsilon_{\text{opt}}$ , with  $\epsilon_{\text{opt}} \leq \frac{\lambda}{2|\mathcal{S}||\mathcal{A}|}$  and  $\lambda < 1$ . Then we have that for all states  $s \in \mathcal{S}$ ,*

$$V^{\pi_\theta}(\rho) \geq V^*(\rho) - \frac{\mathbb{E}_{s \sim \text{Unif}_\mathcal{S}} [\kappa_\mathcal{A}^\alpha(s)] \lambda}{1-\gamma} \left\| \frac{d_\rho^{\pi^*}}{\mu} \right\|_\infty$$

where  $\kappa_\mathcal{A}^\alpha(s) = \frac{2|\mathcal{A}|(1-\alpha(s))}{2|\mathcal{A}|(1-\alpha(s))-1}$ .

*Proof.* The proof proceeds as in Lemma 3.4.3, except that we use the upper bound on  $\epsilon_{\text{opt}}$  in Eq. (3.30) to get

$$\pi_\theta(a^*(s)|s) \geq 1 - \alpha(s) - \frac{\epsilon_{\text{opt}} |\mathcal{S}|}{\lambda} \geq 1 - \alpha(s) - \frac{1}{2|\mathcal{A}|} = \frac{2|\mathcal{A}|(1-\alpha(s)) - 1}{2|\mathcal{A}|} \quad (3.15)$$

In this case we can upper bound  $A^{\pi_\theta}(s, a^*(s))$ . From Eq. (3.13) inequality (i), we have

$$\begin{aligned}
A^{\pi_\theta}(s, a^*(s)) &\leq \frac{1-\gamma}{d_\mu^{\pi_\theta}(s)} \left( \frac{1}{\pi_\theta(a^*(s)|s)} \frac{\partial \mathcal{J}_\lambda^\alpha(\theta)}{\partial \theta_{s,a}} \right. \\
&\quad \left. - \frac{\lambda}{|\mathcal{S}|} \frac{1}{\pi_\theta(a^*(s)|s)} (1 - \alpha(s) - \pi_\theta(a^*(s)|s)) \right) \\
&= \frac{1-\gamma}{d_\mu^{\pi_\theta}(s)} \left( \frac{1}{\pi_\theta(a^*(s)|s)} \frac{\partial \mathcal{J}_\lambda^\alpha(\theta)}{\partial \theta_{s,a}} + \frac{\lambda}{|\mathcal{S}|} \left( 1 - \frac{1-\alpha(s)}{\pi_\theta(a^*(s)|s)} \right) \right) \\
&\stackrel{(i)}{\leq} \frac{1}{\mu(s)} \left( \frac{1}{(1-\alpha(s)) - \frac{\epsilon_{\text{opt}}|\mathcal{S}|}{\lambda}} \cdot \epsilon_{\text{opt}} + \frac{\lambda}{|\mathcal{S}|} (1 - (1-\alpha(s))) \right) \\
&\leq \frac{1}{\mu(s)} \left( \frac{1}{(1-\alpha(s)) - \frac{\epsilon_{\text{opt}}|\mathcal{S}|}{\lambda}} \cdot \epsilon_{\text{opt}} + \frac{\lambda}{|\mathcal{S}|} \alpha(s) \right) \\
&\stackrel{(ii)}{\leq} \frac{1}{\mu(s)} \left( \frac{2|\mathcal{A}|}{(2|\mathcal{A}|(1-\alpha(s)) - 1)} \frac{\lambda}{2|\mathcal{S}||\mathcal{A}|} + \frac{\lambda}{|\mathcal{S}|} \alpha(s) \right) \\
&= \frac{\lambda}{|\mathcal{S}|\mu(s)} \left( \frac{1}{2|\mathcal{A}|(1-\alpha(s)) - 1} + \underbrace{\alpha(s)}_{\leq 1} \right) \\
&\leq \frac{\lambda}{|\mathcal{S}|\mu(s)} \left( \underbrace{\frac{2|\mathcal{A}|(1-\alpha(s))}{2|\mathcal{A}|(1-\alpha(s)) - 1}}_{:=\kappa_{\mathcal{A}}^\alpha(s)} \right)
\end{aligned}$$

Where (i) follows because  $d_\mu^{\pi_\theta}(s) \geq (1-\gamma)\mu(s)$ ,  $\frac{\partial \mathcal{J}_\lambda^\alpha(\theta)}{\partial \theta_{s,a}} \leq \epsilon_{\text{opt}}$  and  $\max(1 - \alpha(s) - \frac{\epsilon_{\text{opt}}|\mathcal{S}|}{\lambda}, 0) \leq \pi_\theta(a^*(s)|s) \leq 1$ . (ii) is obtained by plugging in the upper bound on  $\epsilon_{\text{opt}}$ .

We now make use of the performance difference lemma:

$$V^*(\rho) - V^{\pi_\theta}(\rho) = \frac{1}{1-\gamma} \sum_{s,a} d_\rho^{\pi^*}(s) \pi^*(a|s) A^{\pi_\theta}(s, a) \quad (3.16)$$

$$= \frac{1}{1-\gamma} \sum_s d_\rho^{\pi^*}(s) A^{\pi_\theta}(s, a^*(s)) \quad (3.17)$$

$$\leq \frac{1}{1-\gamma} \sum_s d_\rho^{\pi^*}(s) A^{\pi_\theta}(s, a^*(s)) \mathbb{1}(A^{\pi_\theta}(s, a^*(s)) \geq 0) \quad (3.18)$$

$$\leq \frac{\lambda}{(1-\gamma)|\mathcal{S}|} \sum_s \kappa_{\mathcal{A}}^\alpha(s) \frac{d_\rho^{\pi^*}(s)}{\mu(s)} \mathbb{1}(A^{\pi_\theta}(s, a^*(s)) \geq 0) \quad (3.19)$$

$$\leq \frac{\lambda}{(1-\gamma)} \mathbb{E}_{s \sim \text{Unif}_{\mathcal{S}}} [\kappa_{\mathcal{A}}^\alpha(s)] \left\| \frac{d_\rho^{\pi^*}}{\mu} \right\|_\infty \quad (3.20)$$

This completes the proof. □

We can bound the smoothness of the objective as follows.

**Lemma 3.4.4** (Smoothness of  $\mathcal{J}_\lambda^\alpha$ ). *For the softmax parameterization, we have that*

$$\|\nabla_\theta \mathcal{J}_\lambda^\alpha(\theta) - \nabla_\theta \mathcal{J}_\lambda^\alpha(\theta')\|_2 \leq \beta_\lambda \|\theta - \theta'\|_2$$

where  $\beta_\lambda = \frac{8}{(1-\gamma)^3} + \frac{2\lambda}{|\mathcal{S}|}$ .

*Proof.* We can first bound the smoothness of  $V^{\pi_\theta}(\mu)$  using Lemma D.4 from Agarwal et al. (2020b). We get

$$\|\nabla_\theta V^{\pi_\theta}(\mu) - \nabla_\theta V^{\pi_{\theta'}}(\mu)\|_2 \leq \beta \|\theta - \theta'\|_2$$

for

$$\beta = \frac{8}{(1-\gamma)^3}.$$

We now need to bound the smoothness of the regularizer  $\frac{\lambda}{|\mathcal{S}|}\Omega(\theta)$  where

$$\Omega(\theta) = \sum_{s,a} \pi_0(a|s) \log \pi_\theta(a|s).$$

Using that  $\frac{\partial}{\partial \theta_{s',a'}} \log \pi_\theta(a|s) = \mathbb{1}(s = s')[\mathbb{1}(a = a') - \pi_\theta(a'|s)]$  for the softmax parameterization, we get

$$\begin{aligned} \nabla_{\theta_s} \Omega(\theta) &= \pi_0(\cdot|s) - \pi_\theta(\cdot|s), \\ \nabla_{\theta_s}^2 \Omega(\theta) &= -\text{diag}(\pi_\theta(\cdot|s)) + \pi_\theta(\cdot)\pi_\theta(\cdot|s)^\top. \end{aligned}$$

The remainder of the proof follows directly from that of Lemma D.4 in Agarwal et al. (2020b), as the second-order gradients are identical. We then have that  $\Omega(\theta)$  is 2-smooth and therefore  $\frac{\lambda}{|\mathcal{S}|}\Omega(\theta)$  is  $\frac{2\lambda}{|\mathcal{S}|}$ -smooth, completing the proof. □

Note that the second value of  $\lambda$  will nearly always be greater than 1 for most values of  $\epsilon$ ,  $\epsilon_{\text{opt}}$ ,  $|\mathcal{S}|$ ,  $|\mathcal{A}|$ , as that's the case when  $\mathbb{E}_\mu [\alpha(s)] > \frac{(1-\gamma)^2\epsilon}{|\mathcal{A}|-1} - \epsilon_{\text{opt}}|\mathcal{S}|$ , which is usually negative, thus trivially satisfying the inequality for  $\alpha(s) \in [0, 1] \forall s \in \mathcal{S}$ .

**Lemma 3.4.5** (Iteration complexity for  $\mathcal{J}_\lambda^\alpha$ ). *Let  $\rho$  be a starting state distribution. Following Lemma 3.4.4 let  $\beta_\lambda = \frac{8\gamma}{(1-\gamma)^3} + \frac{2\lambda}{|\mathcal{S}|}$ . From any initial  $\theta^{(0)}$  and following Eq. (3.4) with  $\eta = 1/\beta_\lambda$  and*

$$\lambda = \frac{\epsilon(1-\gamma)}{\mathbb{E}_{s \sim \text{Unif}_\mathcal{S}} [\kappa_\mathcal{A}^\alpha(s)] \left\| \frac{d_\rho^{\pi^*}}{\mu} \right\|_\infty} < 1,$$

we have

$$\begin{aligned} \min_{t \leq T} \{V^*(\rho) - V^{(t)}(\rho)\} &\leq \epsilon \\ \text{whenever } T &\geq \frac{80 \mathbb{E}_{s \sim \text{Unif}_\mathcal{S}} [\kappa_\mathcal{A}^\alpha(s)]^2 |\mathcal{S}|^2 |\mathcal{A}|^2}{(1-\gamma)^6 \epsilon^2} \left\| \frac{d_\rho^{\pi^*}}{\mu} \right\|_\infty^2. \end{aligned}$$

*Proof.* The proof rests on bounding the iteration complexity of making the gradient sufficiently small. Because the optimization process is deterministic and unconstrained, we can use the standard result that after  $T$  updates with stepsize  $1/\beta_\lambda$ , we have

$$\min_{t \leq T} \|\nabla_\theta \mathcal{J}_\lambda^*(\theta^{(t)})\|_2^2 \leq \frac{2\beta_\lambda(\mathcal{J}_\lambda^*(\theta^*) - \mathcal{J}_\lambda^*(\theta^{(0)}))}{T} = \frac{2\beta_\lambda}{(1-\gamma)T}, \quad (3.21)$$

where  $\beta_\lambda$  upper-bounds the smoothness of  $\mathcal{J}_\lambda^*(\theta)$ . Using the above and Corollary 3.4.1, we want

$$\epsilon_{\text{opt}} \leq \sqrt{\frac{2\beta_\lambda}{(1-\gamma)T}} \leq \frac{\lambda}{2|\mathcal{S}||\mathcal{A}|}.$$

Solving the above inequality for  $T$  gives  $T \geq \frac{8|\mathcal{S}|^2|\mathcal{A}|^2\beta_\lambda}{\lambda^2(1-\gamma)}$ . From Lemma 3.4.4, we can set  $\beta_\lambda = \frac{8}{(1-\gamma)^3} + \frac{2\lambda}{|\mathcal{S}|}$ . Plugging this in gives

$$T \geq \frac{8|\mathcal{S}|^2|\mathcal{A}|^2\beta_\lambda}{(1-\gamma)\lambda^2} = \left( \frac{64|\mathcal{S}|^2|\mathcal{A}|^2}{(1-\gamma)^4\lambda^2} + \frac{16|\mathcal{S}||\mathcal{A}|^2}{(1-\gamma)\lambda} \right).$$



Corollary 3.4.1 gives us the possible values for  $\lambda$  for value error margin  $\epsilon$ . Then if

$$\lambda = \frac{\epsilon(1-\gamma)}{\mathbb{E}_\mu [\kappa_{\mathcal{A}}^\alpha(s)] \left\| \frac{d_\rho^{\pi^*}}{\mu} \right\|_\infty} < 1,$$

we can write

$$\begin{aligned} \frac{64|\mathcal{S}|^2|\mathcal{A}|^2}{(1-\gamma)^4\lambda^2} + \frac{16|\mathcal{S}||\mathcal{A}|^2}{(1-\gamma)\lambda} &\leq \frac{80|\mathcal{S}|^2|\mathcal{A}|^2}{(1-\gamma)^4\lambda^2} \\ &= \frac{80\mathbb{E}_\mu [\kappa_{\mathcal{A}}^\alpha(s)]^2 |\mathcal{S}|^2|\mathcal{A}|^2}{\epsilon^2(1-\gamma)^6} \left\| \frac{d_\rho^{\pi^*}}{\mu} \right\|_\infty^2. \end{aligned}$$

□

**Corollary 3.4.2.** *Given the same assumptions as Lemma 3.4.5, if the initial policy is chosen to be  $\pi_0$ , i.e.,  $\pi_{\theta(0)} = \pi_0$  where  $\pi_0(\cdot|s)$  is  $\alpha(s)$ -optimal with respect to  $\pi^*(\cdot|s) \forall s$ , then*

$$\begin{aligned} \min_{t \leq T} \{V^*(\rho) - V^{(t)}(\rho)\} &\leq \epsilon \\ \text{whenever } T &\geq \frac{320|\mathcal{A}|^2|\mathcal{S}|^2}{\epsilon^2(1-\gamma)^7} \left\| \frac{d_\rho^{\pi^*}}{\mu} \right\|_\infty^2 \left\| \frac{1}{\mu} \right\|_\infty \mathbb{E}_{s \sim \mu} [\alpha(s)]. \end{aligned}$$

*Proof.* The proof rests on bounding the iteration complexity of making the gradient sufficiently small. Because the optimization process is deterministic and unconstrained, we can use the standard result that after  $T$  updates with stepsize  $1/\beta_\lambda$ , we have

$$\min_{t \leq T} \|\nabla_\theta \mathcal{J}_\lambda^*(\theta^{(t)})\|_2^2 \leq \frac{2\beta_\lambda(\mathcal{J}_\lambda^*(\theta^*) - \mathcal{J}_\lambda^*(\theta^{(0)}))}{T} = \frac{2\beta_\lambda}{T} \Delta, \quad (3.22)$$

where  $\beta_\lambda$  upper-bounds the smoothness of  $\mathcal{J}_\lambda^*(\theta)$  and we define  $\Delta := \mathcal{J}_\lambda^*(\theta^*) - \mathcal{J}_\lambda^*(\theta^{(0)})$  for conciseness. Using the above and Corollary 3.4.1, we want

$$\epsilon_{\text{opt}} \leq \sqrt{\frac{2\beta_\lambda \Delta}{T}} \leq \frac{\lambda}{2|\mathcal{S}||\mathcal{A}|}.$$

Solving the above inequality for  $T$  gives  $T \geq \Delta \frac{8|\mathcal{S}|^2|\mathcal{A}|^2\beta_\lambda}{\lambda^2}$ . From Lemma 3.4.4, we can

set  $\beta_\lambda = \frac{8}{(1-\gamma)^3} + \frac{2\lambda}{|\mathcal{S}|}$ . Plugging this in gives

$$T \geq \Delta \frac{8|\mathcal{S}|^2|\mathcal{A}|^2\beta_\lambda}{\lambda^2} = \Delta \left( \frac{64|\mathcal{S}|^2|\mathcal{A}|^2}{(1-\gamma)^3\lambda^2} + \frac{16|\mathcal{S}||\mathcal{A}|^2}{\lambda} \right).$$

Corollary 3.4.1 ensures that  $\min_{t \leq T} V^*(\rho) - V^{(t)}(\rho) \leq \epsilon$  provided that  $\lambda$  is of the form:

$$\lambda = \frac{\epsilon(1-\gamma)}{\mathbb{E}_\mu [\kappa_{\mathcal{A}}^\alpha(s)] \left\| \frac{d_{\rho}^{\pi^*}}{\mu} \right\|_\infty} < 1,$$

we can therefore write:

$$\begin{aligned} \frac{64|\mathcal{S}|^2|\mathcal{A}|^2}{(1-\gamma)^3\lambda^2} + \frac{16|\mathcal{S}||\mathcal{A}|^2}{\lambda} &\leq \frac{80|\mathcal{S}|^2|\mathcal{A}|^2}{(1-\gamma)^3\lambda^2} \\ &= \frac{80\mathbb{E}_\mu [\kappa_{\mathcal{A}}^\alpha(s)]^2 |\mathcal{S}|^2|\mathcal{A}|^2}{\epsilon^2(1-\gamma)^5} \left\| \frac{d_{\rho}^{\pi^*}}{\mu} \right\|_\infty^2. \end{aligned}$$

This implies the following condition on  $T$ :

$$T \geq \frac{80\Delta|\mathcal{A}|^2|\mathcal{S}|^2}{\epsilon^2(1-\gamma)^5} \mathbb{E}_\mu [\kappa_{\mathcal{A}}^\alpha(s)]^2 \left\| \frac{d_{\rho}^{\pi^*}}{\mu} \right\|_\infty^2 \quad (3.23)$$

It remains to control the error  $\Delta$  due to initialization with policy  $\pi_0$ . Denote by  $\pi_\lambda^*$  the optimal policy maximizing  $\mathcal{J}_\lambda^*$ . We have the following:

$$\begin{aligned} \Delta &:= V^{\pi_\lambda^*}(\rho) - V^{\pi_0}(\rho) - \lambda \text{KL}(\pi_0, \pi_\lambda^*) \\ &\leq V^{\pi_\lambda^*}(\rho) - V^*(\rho) + V^*(\rho) - V^{\pi_0}(\rho) \\ &\leq V^*(\rho) - V^{\pi_0}(\rho) \\ &\leq \frac{1}{(1-\gamma)^2} \left\| \frac{d_{\rho}^{\pi_0}}{\mu'} \right\|_\infty \mathbb{E}_{s \sim \mu'} [\alpha(s)] \end{aligned} \quad (3.24)$$

where the first line is by definition of  $\Delta$ , the second line uses that the KL term is non-positive. The third line uses that  $V^{\pi_\lambda^*}(\rho) - V^*(\rho) \leq 0$  and the last line follows from

Lemma 3.A.2. Hence, it suffice to choose  $T$  satisfying:

$$T \geq \frac{80|\mathcal{A}|^2|\mathcal{S}|^2}{\epsilon^2(1-\gamma)^7} \left\| \frac{d_\rho^{\pi^*}}{\mu} \right\|_\infty^2 \left\| \frac{d_\rho^{\pi_0}}{\mu'} \right\|_\infty \mathbb{E}_\mu [\kappa_{\mathcal{A}}^\alpha(s)]^2 \mathbb{E}_{s \sim \mu'} [\alpha(s)] \quad (3.25)$$

As a final step, we simply observe that  $\mathbb{E}_\mu [\kappa_{\mathcal{A}}^\alpha(s)] \leq 2$  and  $d_\rho^{\pi_0} \leq 1$ .

□

**Lemma 3.A.1.** Following Lemma 3.4.4 let  $\beta_\lambda = \frac{8\gamma}{(1-\gamma)^3} + \frac{2\lambda}{|\mathcal{S}|}$ . From any initial  $\theta^{(0)}$  and following Eq. (3.4) with  $\eta = 1/\beta_\lambda$  and

$$\lambda = \frac{\epsilon_{\text{opt}}|\mathcal{S}|(|\mathcal{A}| - 1)}{(1-\gamma)^2\epsilon - (|\mathcal{A}| - 1)\mathbb{E}_\mu [\alpha(s)]}, \quad (3.26)$$

for all starting state distributions  $\rho$ , we have,

$$\begin{aligned} \min_{t \leq T} \{V^*(\rho) - V^{(t)}(\rho)\} &\leq \epsilon \\ \text{whenever } T &\geq \min \left\{ \frac{80\mathbb{E}_\mu [\kappa_{\mathcal{A}}^\alpha(s)]^2 |\mathcal{S}|^2 |\mathcal{A}|^2}{(1-\gamma)^6 \epsilon^2} \left\| \frac{d_\rho^{\pi^*}}{\mu} \right\|_\infty^2, \right. \\ &\quad \left. 80|\mathcal{S}||\mathcal{A}|^2 \left( \frac{\epsilon}{\epsilon_{\text{opt}}(1-\gamma)^2(|\mathcal{A}| - 1)} - \frac{\mathbb{E}_\mu [\alpha(s)]}{\epsilon_{\text{opt}}(1-\gamma)^4} \right) \right\}. \end{aligned} \quad (3.27)$$

*Proof.* The proof proceeds identically as above, except we set

$$\lambda = \frac{\epsilon_{\text{opt}}|\mathcal{S}|(|\mathcal{A}| - 1)}{(1-\gamma)^2\epsilon - (|\mathcal{A}| - 1)\mathbb{E}_\mu [\alpha(s)]} > 1,$$

we have

$$\begin{aligned} \frac{64|\mathcal{S}|^2|\mathcal{A}|^2}{(1-\gamma)^4\lambda^2} + \frac{16|\mathcal{S}||\mathcal{A}|^2}{(1-\gamma)\lambda} &\leq \frac{80|\mathcal{S}|^2|\mathcal{A}|^2}{(1-\gamma)^4\lambda} \\ &= \frac{80|\mathcal{S}||\mathcal{A}|^2((1-\gamma)^2\epsilon - (|\mathcal{A}| - 1)\mathbb{E}_\mu [\alpha(s)])}{(1-\gamma)^4\epsilon_{\text{opt}}(|\mathcal{A}| - 1)} \\ &= 80|\mathcal{S}||\mathcal{A}|^2 \left( \frac{\epsilon}{\epsilon_{\text{opt}}(1-\gamma)^2(|\mathcal{A}| - 1)} - \frac{\mathbb{E}_\mu [\alpha(s)]}{\epsilon_{\text{opt}}(1-\gamma)^4} \right) \end{aligned}$$

completing the proof. □

Note that this value of  $\lambda$  will nearly always be greater than 1 for most values of  $\epsilon, \epsilon_{\text{opt}}, |\mathcal{S}|, |\mathcal{A}|$ , as that's the case when  $\mathbb{E}_\mu [\alpha(s)] > \frac{(1-\gamma)^2 \epsilon}{|\mathcal{A}|-1} - \epsilon_{\text{opt}} |\mathcal{S}|$ , which is usually negative, thus trivially satisfying the inequality for  $\alpha(s) \in [0, 1] \forall s \in \mathcal{S}$ .

**Lemma 3.A.2.** *Assume that  $\pi$  is such that  $\pi(a^*(s)|s) \geq 1 - \beta(s)$  for some state dependent error  $s \mapsto \beta(s)$  and that  $\rho(s) > 0$  for all states  $s$ . Then the following inequality holds:*

$$V^\pi(\rho) - V^*(\rho) \geq -\frac{1}{(1-\gamma)^2} \left\| \frac{d_\rho^\pi}{\rho} \right\|_\infty \mathbb{E}_\rho [\beta(s)] \quad (3.28)$$

*Proof.*

$$\begin{aligned} V^\pi(\rho) - V^*(\rho) &= \frac{1}{1-\gamma} \sum_{s,a} d_\rho^\pi(s) \pi(a|s) A^{\pi^*}(s, a) \\ &= \frac{1}{1-\gamma} \sum_s \sum_{a \neq a^*(s)} (d_\rho^\pi(s) \pi(a|s) A^{\pi^*}(s, a)) \\ &\geq -\frac{1}{(1-\gamma)^2} \sum_s d_\rho^\pi(s) \sum_{a \neq a^*(s)} \pi(a|s) \\ &\geq -\frac{1}{(1-\gamma)^2} \sum_s (d_\rho^\pi(s) \beta(s)) \\ &\geq -\frac{1}{(1-\gamma)^2} \left\| \frac{d_\rho^\pi}{\mu} \right\|_\infty \mathbb{E}_{s \sim \mu} [\beta(s)] \end{aligned}$$

where the first line follows by application of the performance different lemma (Agarwal et al., 2020b, Lemma 3.2), the second line is due to the fact that  $A^{\pi^*}(s, a^*(s)) = 0$ , the third line from  $A^{\pi^*}(s, a) \geq -1/(1-\gamma)$  for  $a \neq a^*$ . The fourth line uses that  $\sum_{a \neq a^*(s)} \pi(a|s) = 1 - \pi(a^*(s)|s) \leq \beta(s)$  for  $a \neq a^*(s)$  since by assumption  $\pi(a^*(s)|s) \geq 1 - \beta(s)$ . Finally, the last line uses that  $d_\rho^\pi$  is a probability distribution over states  $s$  satisfying  $\sum_{s \in \mathcal{S}} d_\rho^\pi(s) = 1$ . □

### 3.A.1 State dependent $\lambda$ and $\epsilon$

We can further generalize these results by allowing the error  $\epsilon$  and regularization weight  $\lambda$  to be state-dependent. The gradient with state dependent regularized  $\lambda$  equals

$$\mathcal{J}^{\pi_0}(\theta) = V^{\pi_\theta}(\mu) + \sum_{s,a} \frac{\lambda(s)}{|\mathcal{S}|} \pi_0(a|s) \log \pi_\theta(a|s)$$

**Lemma 3.A.3.** *Suppose  $\theta$  is such that  $(\nabla \mathcal{J}_\lambda^\alpha(\theta))_{s,a} \leq \epsilon_{\text{opt}}(s, a)$ . Then we have that for all states  $s \in \mathcal{S}$ ,*

$$\begin{aligned} & V^*(\rho) - V^{\pi_\theta}(\rho) \\ & \leq \min \left\{ \frac{1}{1-\gamma} \mathbb{E}_{s \sim \text{Unif}_{\mathcal{S}}} \left[ \frac{\epsilon_{\text{opt}}(s, a^*(s)) |\mathcal{S}|}{\max \left( (1 - \alpha(s)) - \frac{\epsilon_{\text{opt}}(s, a^*(s)) |\mathcal{S}|}{\lambda(s)}, 0 \right)} + \lambda(s) \alpha(s) \right] \left\| \frac{d_{\rho}^{\pi^*}}{\mu} \right\|_{\infty}, \right. \\ & \quad \left. \frac{|\mathcal{A}|}{(1-\gamma)^2} \mathbb{E}_{s \sim \mu} [\alpha(s)] \left\| \frac{d_{\rho}^{\pi_\theta}}{\mu} \right\|_{\infty} + \frac{|\mathcal{S}|}{(1-\gamma)^2} \left\| \frac{\sum_a \epsilon_{\text{opt}}(s, a)}{\lambda(s)} \right\|_{\infty} \right\} \end{aligned}$$

*Proof.* Let's assume that  $\pi^*$  is a deterministic policy. By [Puterman \(1994\)](#) such an optimal policy always exists for an MDP. We'll use the notation  $a^*(s)$  to denote the optimal action at state  $s$ . This, combined with the assumption that  $d_{\text{TV}}(\pi^*(\cdot|s), \pi_0(\cdot|s)) \leq \alpha(s)$  for all  $s \in \mathcal{S}$ , tells us that  $\pi_0(a^*(s)|s) \geq \pi^*(a^*(s)|s) - \alpha(s) = 1 - \alpha(s)$ . Similarly, for  $a \neq a^*(s)$ ,  $\pi_0(a|s) \leq \alpha(s)$ . Using this, we can start by showing that whenever  $A^{\pi_\theta}(s, a^*(s)) \geq 0$  we can lower bound  $\pi_\theta(a^*(s)|s)$  for all  $s$ .

The gradient norm assumption  $(\nabla \mathcal{J}_\lambda^\alpha(\theta))_{s,a} \leq \epsilon_{\text{opt}}(s, a)$  implies that for all  $s, a$ :

$$\epsilon_{\text{opt}}(s, a) \geq \frac{\partial \mathcal{J}_\lambda^\alpha(\theta)}{\partial \theta_{s,a}} = \frac{1}{1-\gamma} d_{\mu}^{\pi_\theta}(s) \pi_\theta(a|s) A^{\pi_\theta}(s, a) + \frac{\lambda(s)}{|\mathcal{S}|} (\pi_0 - \pi_\theta(a|s))$$

In particular for all  $s$ ,

$$\begin{aligned}
\epsilon_{\text{opt}}(s, a^*(s)) &\geq \frac{\partial \mathcal{J}_\lambda^\alpha(\theta)}{\partial \theta_{s, a^*(s)}} \\
&\stackrel{(i)}{\geq} \frac{1}{1-\gamma} d_\mu^{\pi_\theta}(s) \pi_\theta(a^*(s)|s) A^{\pi_\theta}(s, a^*(s)) \\
&\quad + \frac{\lambda(s)}{|\mathcal{S}|} (\pi^*(a^*(s)|s) - \alpha(s) - \pi_\theta(a^*(s)|s)) \\
&= \frac{1}{1-\gamma} d_\mu^{\pi_\theta}(s) \pi_\theta(a^*(s)|s) A^{\pi_\theta}(s, a^*(s)) \\
&\quad + \frac{\lambda(s)}{|\mathcal{S}|} (1 - \alpha(s) - \pi_\theta(a^*(s)|s))
\end{aligned} \tag{3.29}$$

And therefore if  $A^{\pi_\theta}(s, a^*(s)) \geq 0$ ,

$$\epsilon_{\text{opt}}(s, a) \geq \frac{\lambda(s)}{|\mathcal{S}|} (1 - \alpha(s) - \pi_\theta(a^*(s)|s))$$

Thus,

$$\begin{aligned}
\pi_\theta(a^*(s)|s) &\geq \max \left( 1 - \alpha(s) - \frac{\epsilon_{\text{opt}}(s, a^*(s)) |\mathcal{S}|}{\lambda(s)}, 0 \right) \\
&\geq 1 - \alpha(s) - \frac{\epsilon_{\text{opt}}(s, a^*(s)) |\mathcal{S}|}{\lambda(s)}.
\end{aligned} \tag{3.30}$$

In this case we can upper bound  $A^{\pi_\theta}(s, a^*(s))$ . From Eq. (3.13) inequality (i), we have

$$\begin{aligned}
A^{\pi_\theta}(s, a^*(s)) &\leq \frac{1 - \gamma}{d_\mu^{\pi_\theta}(s)} \left( \frac{1}{\pi_\theta(a^*(s)|s)} \frac{\partial \mathcal{J}_\lambda^\alpha(\theta)}{\partial \theta_{s, a^*(s)}} \right. \\
&\quad \left. - \frac{\lambda(s)}{|\mathcal{S}|} \frac{1}{\pi_\theta(a^*(s)|s)} (1 - \alpha(s) - \pi_\theta(a^*(s)|s)) \right) \\
&= \frac{1 - \gamma}{d_\mu^{\pi_\theta}(s)} \left( \frac{1}{\pi_\theta(a^*(s)|s)} \frac{\partial \mathcal{J}_\lambda^\alpha(\theta)}{\partial \theta_{s, a^*(s)}} + \frac{\lambda(s)}{|\mathcal{S}|} \left( 1 - \frac{1 - \alpha(s)}{\pi_\theta(a^*(s)|s)} \right) \right) \\
&\stackrel{(i)}{\leq} \frac{1}{\mu(s)} \left( \frac{1}{\max \left( (1 - \alpha(s)) - \frac{\epsilon_{\text{opt}}(s, a^*(s))|\mathcal{S}|}{\lambda}, 0 \right)} \cdot \epsilon_{\text{opt}}(s, a^*(s)) \right. \\
&\quad \left. + \frac{\lambda(s)}{|\mathcal{S}|} (1 - (1 - \alpha(s))) \right) \\
&\leq \frac{1}{\mu(s)} \left( \frac{1}{\max \left( (1 - \alpha(s)) - \frac{\epsilon_{\text{opt}}(s, a^*(s))|\mathcal{S}|}{\lambda}, 0 \right)} \cdot \epsilon_{\text{opt}}(s, a^*(s)) \right. \\
&\quad \left. + \frac{\lambda(s)}{|\mathcal{S}|} \alpha(s) \right)
\end{aligned}$$

Where (i) follows because  $d_\mu^{\pi_\theta}(s) \geq (1 - \gamma)\mu(s)$ ,  $\frac{\partial \mathcal{J}_\lambda^\alpha(\theta)}{\partial \theta_{s, a}} \leq \epsilon_{\text{opt}}$  and  $\max \left( 1 - \alpha(s) - \frac{\epsilon_{\text{opt}}(s, a^*(s))|\mathcal{S}|}{\lambda}, 0 \right) \leq \pi_\theta(a^*(s)|s) \leq 1$ .

We now make use of the performance difference lemma:

$$\begin{aligned}
& V^*(\rho) - V^{\pi_\theta}(\rho) \\
&= \frac{1}{1-\gamma} \sum_{s,a} d_\rho^{\pi^*}(s) \pi^*(a|s) A^{\pi_\theta}(s, a) \\
&= \frac{1}{1-\gamma} \sum_s d_\rho^{\pi^*}(s) A^{\pi_\theta}(s, a^*(s)) \\
&\leq \frac{1}{1-\gamma} \sum_s d_\rho^{\pi^*}(s) A^{\pi_\theta}(s, a^*(s)) \mathbb{1}(A^{\pi_\theta}(s, a^*(s)) \geq 0) \\
&\leq \frac{1}{1-\gamma} \sum_s \frac{d_\rho^{\pi^*}(s)}{\mu(s)} \left( \frac{1}{\max\left((1-\alpha(s)) - \frac{\epsilon_{\text{opt}}(s,a)|\mathcal{S}|}{\lambda}, 0\right)} \cdot \epsilon_{\text{opt}}(s, a^*(s)) \right. \\
&\quad \left. + \frac{\lambda(s)}{|\mathcal{S}|} \alpha(s) \right) \mathbb{1}(A^{\pi_\theta}(s, a^*(s)) \geq 0) \\
&\leq \frac{1}{1-\gamma} \mathbb{E}_{s \sim \text{Unif}_{\mathcal{S}}} \left[ \frac{\epsilon_{\text{opt}}(s, a^*(s))|\mathcal{S}|}{\max\left((1-\alpha(s)) - \frac{\epsilon_{\text{opt}}(s,a^*(s))|\mathcal{S}|}{\lambda(s)}, 0\right)} + \lambda(s)\alpha(s) \right] \left\| \frac{d_\rho^{\pi^*}}{\mu} \right\|_\infty
\end{aligned}$$

Now let's relate the values of  $\pi^*$  and  $\pi_\theta$ . We will again apply the performance differ-



ence lemma, this time in the other direction:

$$\begin{aligned}
& V^{\pi_\theta}(\rho) - V^*(\rho) \\
&= \frac{1}{1-\gamma} \sum_{s,a} d_\rho^{\pi_\theta}(s) \pi_\theta(a|s) A^{\pi^*}(s, a) \\
&\stackrel{(1)}{=} \frac{1}{1-\gamma} \sum_s \left( \sum_{a \neq a^*(s)} d_\rho^{\pi_\theta}(s) \pi_\theta(a|s) A^{\pi^*}(s, a) \right) \\
&\stackrel{(2)}{\geq} \frac{-1}{1-\gamma} \sum_s \sum_a d_\rho^{\pi_\theta}(s) \left( \alpha(s)(|\mathcal{A}| - 1) + \frac{\sum_{a \neq a^*(s)} \epsilon_{\text{opt}}(s, a) |\mathcal{S}|}{\lambda(s)} \right) \frac{1}{1-\gamma} \\
&= -\frac{1}{1-\gamma} \left( \sum_s \frac{d_\rho^{\pi_\theta}(s)}{1-\gamma} \alpha(s)(|\mathcal{A}| - 1) + \sum_s |\mathcal{S}| d_\rho^{\pi_\theta}(s) \sum_{a \neq a^*(s)} \frac{\epsilon_{\text{opt}}(s, a)}{(1-\gamma)\lambda(s)} \right) \\
&\stackrel{(3)}{\geq} -\sum_s \frac{d_\rho^{\pi_\theta}(s)}{(1-\gamma)^2} \alpha(s)(|\mathcal{A}| - 1) - \frac{|\mathcal{S}|}{(1-\gamma)^2} \left\| \frac{\sum_{a \neq a^*(s)} \epsilon_{\text{opt}}(s, a)}{\lambda(s)} \right\|_\infty \\
&\geq -\frac{|\mathcal{A}|}{(1-\gamma)^2} \mathbb{E}_{s \sim d_\rho^{\pi_\theta}} [\alpha(s)] - \frac{|\mathcal{S}|}{(1-\gamma)^2} \left\| \frac{\sum_a \epsilon_{\text{opt}}(s, a)}{\lambda(s)} \right\|_\infty \\
&\geq -\frac{|\mathcal{A}|}{(1-\gamma)^2} \mathbb{E}_{s \sim \mu} [\alpha(s)] \left\| \frac{d_\rho^{\pi_\theta}}{\mu} \right\|_\infty - \frac{|\mathcal{S}|}{(1-\gamma)^2} \left\| \frac{\sum_a \epsilon_{\text{opt}}(s, a)}{\lambda(s)} \right\|_\infty
\end{aligned}$$

where (1) is due to the fact that  $A^{\pi^*}(s, a^*(s)) = 0$ , and (2) is due to the fact that  $A^{\pi^*}(s, a)$  for  $a \neq a^*$  is lower-bounded by  $-1/(1-\gamma)$  and Eq. (3.30). and (3) holds because of Holder and  $\sum_s d_\rho^{\pi_\theta}(s) = 1$ . Therefore,

$$V^{\pi_\theta}(\rho) + \frac{|\mathcal{A}|}{(1-\gamma)^2} \mathbb{E}_{s \sim \mu} [\alpha(s)] \left\| \frac{d_\rho^{\pi_\theta}}{\mu} \right\|_\infty + \frac{|\mathcal{S}|}{(1-\gamma)^2} \left\| \frac{\sum_a \epsilon_{\text{opt}}(s, a)}{\lambda(s)} \right\|_\infty \geq V^*(\rho).$$

□

A simple corollary of Lemma 3.A.3 is,

**Corollary 3.A.1.** If  $\epsilon_{\text{opt}}(s, a) \leq \frac{(1-\alpha(s))\lambda(s)}{|\mathcal{S}|}$  then

$$\begin{aligned} & V^*(\rho) - V^{\pi_\theta}(\rho) \\ & \leq \min \left\{ \frac{1}{1-\gamma} \mathbb{E}_{s \sim \text{Unif}_{\mathcal{S}}} \left[ \frac{2\epsilon_{\text{opt}}(s, a^*(s))|\mathcal{S}|}{1-\alpha(s)} + \lambda(s)\alpha(s) \right] \left\| \frac{d_\rho^{\pi^*}}{\mu} \right\|_\infty, \right. \\ & \quad \left. \frac{|\mathcal{A}|}{(1-\gamma)^2} \mathbb{E}_{s \sim \mu} [\alpha(s)] \left\| \frac{d_\rho^{\pi_\theta}}{\mu} \right\|_\infty + \frac{|\mathcal{S}|}{(1-\gamma)^2} \left\| \frac{\sum_a \epsilon_{\text{opt}}(s, a)}{\lambda(s)} \right\|_\infty \right\} \end{aligned}$$

*Proof.* If  $\epsilon(s, a) \leq \frac{(1-\alpha(s))\lambda(s)}{|\mathcal{S}|}$  then  $\max \left( (1-\alpha(s)) - \frac{\epsilon_{\text{opt}}(s, a)|\mathcal{S}|}{\lambda(s)}, 0 \right) \geq \frac{1-\alpha(s)}{2}$ . The result follows.  $\square$

Corollary 3.A.1 recovers the results of Agarwal et al. (2020b) by noting the TV distance between the optimal policy and the uniform one equals  $1 - \frac{1}{|\mathcal{A}|}$  and therefore  $1 - \alpha(s) = \frac{1}{|\mathcal{A}|}$ .

We now concern ourselves with the problem of finding a true  $\epsilon > 0$  optimal policy. This will require us to set the values of  $\lambda(s)$  appropriately. We restrict ourselves to the following version of the results of Corollary 3.A.1. If  $\epsilon(s, a) \leq \frac{(1-\alpha(s))\lambda(s)}{|\mathcal{S}|}$  then

$$V^{\pi_\theta}(\rho) \geq V^*(\rho) - \frac{1}{1-\gamma} \mathbb{E}_{s \sim \text{Unif}_{\mathcal{S}}} \left[ \frac{2\epsilon_{\text{opt}}(s, a^*(s))|\mathcal{S}|}{1-\alpha(s)} + \lambda(s)\alpha(s) \right] \left\| \frac{d_\rho^{\pi^*}}{\mu} \right\|_\infty$$

By setting

$$\begin{aligned} \lambda(s) &= \frac{\epsilon(1-\gamma)}{2\alpha(s) \left\| \frac{d_\rho^{\pi^*}}{\mu} \right\|_\infty} \\ \text{and } \epsilon_{\text{opt}}(s, a) &= \min \left( \frac{(1-\alpha(s))\epsilon(1-\gamma)}{4|\mathcal{S}| \left\| \frac{d_\rho^{\pi^*}}{\mu} \right\|_\infty}, \frac{(1-\alpha(s))\lambda(s)}{|\mathcal{S}|} \right) \end{aligned}$$

we get

$$V^{\pi_\theta}(\rho) \geq V^*(\rho) - \epsilon.$$

Observe that the level of regularization depends on the state's error. If the error is very low, the regularizer  $\lambda(s)$  should be set to a larger value.

## Appendix 3.B: Multitask Analysis

Assume we are given  $K$  i.i.d. tasks  $M_k$  sampled from  $\mathcal{P}_{\mathcal{M}}$ , denote by  $\pi_k^*(\cdot|s)$  their corresponding optimal policies and let  $\tilde{\pi}_k(\cdot|s)$  be  $\alpha(s)$  policies, i.e.  $d_{TV}(\tilde{\pi}_k(\cdot|s), \pi_k^*(\cdot|s)) \leq \alpha(s)$  for some  $\alpha(s) \leq 1$ . To simplify notation, we may also refer to  $\mathcal{P}$  directly as the distribution over these optimal policies. Let  $\hat{\pi}_0$  be the total variation barycenter of the policies  $\tilde{\pi}_k$ , i.e.:  $\hat{\pi}_0 = \arg \min_{\pi} \frac{1}{K} \sum_{k=1}^K d_{TV}(\pi, \tilde{\pi}_k)$ , while

$$\pi_0 = \arg \min_{\pi} \mathbb{E}_{M_k \sim \mathcal{P}_{\mathcal{M}}} [d_{TV}(\pi, \pi_k^*)].$$

**Lemma 3.5.1** (TV barycenter). *Let  $\mathbb{P}_{\mathcal{M}}$  be a distribution over tasks  $\mathcal{M} = \{M_k\}$ , each with a deterministic optimal policy  $\pi_k^* : \mathcal{S} \rightarrow \mathcal{A}$ . Define the average optimal action as*

$$\xi(s, a) := \mathbb{E}_{M_k \sim \mathbb{P}_{\mathcal{M}}} [\mathbb{1}(\pi_k^*(s) = a)]. \quad (3.11)$$

*Then, the TV barycenter  $\pi_0(\cdot|s)$  defined in Eq. (3.10) is given by a greedy policy over  $\xi$ , i.e.,  $\pi_0(a|s) = \delta(a \in \arg \max_{a' \in \mathcal{A}} \xi(s, a'))$ , where  $\delta(\cdot)$  is the Dirac delta distribution.*

*Proof.* Let's first express the barycenter loss in a more convenient form:

$$\mathbb{E}_{\pi' \sim P} [d_{TV}(\pi(\cdot|s), \pi'(\cdot|s))] = \mathbb{E}_{\pi' \sim P} \left[ \frac{1}{2} \sum_{a \neq a_{\pi'}(s)} \pi(a) + \frac{1}{2} (1 - \pi(a_{\pi'}(s), s)) \right] \quad (3.31)$$

$$= \mathbb{E}_{\pi' \sim P} [(1 - \pi(a_{\pi'}(s), s))] \quad (3.32)$$

$$= 1 - \sum_a \mathbb{P}(\pi'(a|s) = 1) \pi(a|s) \quad (3.33)$$

$$= 1 - \sum_a \pi_{soft}(a|s) \pi(a|s). \quad (3.34)$$

Therefore, the barycenter loss is minimized when  $\pi(a|s)$  puts all its mass on the maximum value of  $\pi_{soft}(a|s)$  over actions  $a \in \mathcal{A}$ .

□

The KL barycenter can be described as follows.

**Lemma 3.B.4** (KL barycenter). *Let  $\mathcal{P}_{\mathcal{M}}$  be a distribution over tasks such that for every  $M_k \in \mathcal{M}$ , there exists a unique optimal action  $a_k^*(s)$  for each state  $s$  such that  $\pi_k^*(s) = a_k^*$ . Then the KL barycenter for state  $s$  is:*

$$\operatorname{argmin}_{\pi} \mathbb{E}_{M_k \sim \mathcal{P}_{\mathcal{M}}} \text{KL}(\pi_k^*(\cdot|s), \pi(\cdot|s)) = \delta(a = \mathbb{E}_{M_k \sim \mathcal{P}_{\mathcal{M}}} \pi_k^*(s)) \quad (3.35)$$

where  $\delta(\cdot)$  is the Dirac delta distribution. This holds for both directions of the KL.

*Proof.* We have

$$\begin{aligned} & \mathbb{E}_{M_k \sim \mathcal{P}_{\mathcal{M}}} \text{KL}(\pi_k^*(\cdot|s), \pi(\cdot|s)) \\ &= \mathbb{E}_{M_k \sim \mathcal{P}_{\mathcal{M}}} \sum_a \pi_k^*(a|s) \log \frac{\pi_k^*(a|s)}{\pi(a|s)} \\ &= \mathbb{E}_{M_k \sim \mathcal{P}_{\mathcal{M}}} \left[ -\log \pi(a_k^*(s)|s) + \underbrace{\sum_{a \neq a_k^*(s)} 0 \cdot \log \frac{0}{\pi(a|s)}}_{=0} \right] \\ &= \mathbb{E}_{M_k \sim \mathcal{P}_{\mathcal{M}}} [-\log \pi(a_k^*(s)|s)] \end{aligned}$$

Therefore, the barycenter loss is minimized when  $\pi(a|s)$  puts all its mass on the expected  $a_k^*(s)$ . Note that we consider the underbrace term zero because  $\lim_{x \rightarrow 0} x \log x = 0$ . It is easy to verify that this result holds for the reverse KL.  $\square$

**Lemma 3.5.2** (Multitask iteration complexity). *Let  $M_k \sim \mathbb{P}_{\mathcal{M}}$  and denote by  $\pi_k^* : \mathcal{S} \rightarrow \mathcal{A}$  its optimal policy. Denote by  $T_k$  the number of iterations to reach  $\epsilon$ -error for  $M_k$  in the sense that:*

$$\min_{t \leq T_k} \{V^{\pi_k^*}(\rho) - V^{(t)}(\rho)\} \leq \epsilon.$$

Set  $\lambda, \beta_\lambda$ , and  $\eta$  as in Lemma 3.4.5. From any initial  $\theta^{(0)}$ , and following Eq. (3.4),  $\mathbb{E}_{M_k \sim \mathbb{P}_{\mathcal{M}}} [T_k]$  satisfies:

$$\mathbb{E}_{M_k \sim \mathbb{P}_{\mathcal{M}}} [T_k] \geq \frac{80|\mathcal{A}|^2|\mathcal{S}|^2}{\epsilon^2(1-\gamma)^6} \mathbb{E}_{M_k \sim \mathbb{P}_{\mathcal{M}}} \left[ \kappa_{\mathcal{A}}^{\alpha_k}(s) \left\| \frac{d_{\rho}^{\pi_k^*}}{\mu} \right\|_{\infty}^2 \right],$$

where  $\alpha_k(s) := d_{TV}(\pi_k^*(\cdot|s), \hat{\pi}_0(\cdot|s))$ . If  $\hat{\pi}_0$  is also used for initialization, then  $\mathbb{E}_{M_k \sim \mathbb{P}_{\mathcal{M}}} [T_k]$  satisfies:

$$\mathbb{E}_{M_k \sim \mathbb{P}_{\mathcal{M}}} [T_k] \geq \frac{320|\mathcal{A}|^2|\mathcal{S}|^2}{\epsilon^2(1-\gamma)^7} \left\| \frac{1}{\mu} \right\|_{\infty}^3 \mathbb{E}_{M_k \sim \mathbb{P}_{\mathcal{M}}} [\alpha_k(s)],$$

*Proof.* Let  $M_i$  be a random task sampled according to  $\mathcal{M}$  and denote by  $\pi_i^*$  its corresponding optimal policy. Set  $\alpha(s) = d_{TV}(\hat{\pi}_0, \pi_i^*)$  and choose  $\lambda = \frac{\epsilon(1-\gamma)}{2\left\|\frac{d_{\rho}^{\pi^*}}{\mu}\right\|}$ . By Lemma 3.4.5, we have that:

$$\begin{aligned} \min_{t \leq T} \{V^*(\rho) - V^{(t)}(\rho)\} &\leq \epsilon \\ \text{whenever } T &\geq \frac{160|\mathcal{A}|^2|\mathcal{S}|^2}{\epsilon^2(1-\gamma)^7} \left\| \frac{d_{\rho}^{\pi^*}}{\mu} \right\|_{\infty}^2 \left\| \frac{d_{\rho}^{\hat{\pi}_0}}{\mu'} \right\|_{\infty} \mathbb{E}_{s \sim \mu'} [\alpha(s)]. \end{aligned} \quad (3.36)$$

By choosing  $\mu'$  to be uniform and recalling that  $d_{\rho}^{\hat{\pi}_0} \leq 1$ , it suffice to have:

$$T \geq \frac{160|\mathcal{A}|^2|\mathcal{S}|^3}{\epsilon^2(1-\gamma)^7} \left\| \frac{d_{\rho}^{\pi^*}}{\mu} \right\|_{\infty}^2 \mathbb{E}_{s \sim \mu'} [d_{TV}(\hat{\pi}_0, \pi_i^*)]. \quad (3.37)$$

Taking the expectation over the tasks and treating  $T$  as a random variable depending on the task, we get that:

$$\mathbb{E}[T] \geq \frac{160|\mathcal{A}|^2|\mathcal{S}|^3}{\epsilon^2(1-\gamma)^7} \left\| \frac{d_{\rho}^{\pi^*}}{\mu} \right\|_{\infty}^2 \mathbb{E}_{s \sim \mu'} \mathbb{P}[d_{TV}(\hat{\pi}_0, \pi')]. \quad (3.38)$$

□

The following lemma quantifies how  $\hat{\pi}_0$  is close to be the TV barycenter of  $\{\pi_k^*\}_{1 \leq k \leq K}$  when  $K$  grows to infinity. We let  $\tilde{\pi}_k(\cdot|s)$  be, on average,  $\zeta(s)$ -optimal in state  $s$  across tasks  $M_k$ , i.e.  $\mathbb{E}_{M_k \sim \mathcal{M}} [d_{TV}(\tilde{\pi}_k(\cdot|s), \pi_k^*(\cdot|s))] \leq \zeta(s)$  for some  $\zeta(s) \in [0, 1]$ . For concision, we shorten  $\pi(\cdot|s)$  as  $\pi$ .

**Lemma 3.5.3** (Barycenter concentration). *Let  $\delta$  be  $0 < \delta < 1$ . Then with probability*

higher than  $1 - \delta$ , for all  $s \in \mathcal{S}$ , it holds that:

$$\begin{aligned} & |\mathbb{E}_{M_k \sim \mathbb{P}_{\mathcal{M}}} [d_{TV}(\pi_k^*(\cdot|s), \hat{\pi}_0(\cdot|s)) - d_{TV}(\pi_k^*(\cdot|s), \pi_0(\cdot|s))]| \\ & \leq 2\zeta(s) + \sqrt{\frac{2\log(\frac{2}{\delta})}{K}} + 2C\sqrt{\frac{|\mathcal{A}|}{K}}, \end{aligned}$$

for some constant  $C$  that depends only on  $|\mathcal{A}|$ .

*Proof.* To simplify the proof, we fix a state  $s$  and omit the dependence in  $s$ . We further introduce the following notations:

$$f(\pi) = \mathbb{E}_{M_i \sim \mathcal{P}_{\mathcal{M}}} [d_{TV}(\pi, \pi_i^*)] \quad (3.39)$$

$$\tilde{f}(\pi) = \frac{1}{K} \sum_{i=1}^K d_{TV}(\pi, \tilde{\pi}_i) \quad (3.40)$$

$$\hat{f}(\pi) = \frac{1}{K} \sum_{i=1}^K d_{TV}(\pi, \pi_i^*) \quad (3.41)$$

Let  $\pi_0 = \arg \min_{\pi} f(\pi)$  and  $\hat{\pi}_0 = \arg \min_{\pi} \tilde{f}(\pi)$ . It is easy to see that:

$$\begin{aligned} f(\hat{\pi}_0) & \leq \tilde{f}(\hat{\pi}_0) + |\hat{f}(\hat{\pi}_0) - f(\hat{\pi}_0)| + |\tilde{f}(\hat{\pi}_0) - \hat{f}(\hat{\pi}_0)| \\ & \leq \hat{f}(\pi_0) + |\hat{f}(\hat{\pi}_0) - f(\hat{\pi}_0)| + |\hat{f}(\hat{\pi}_0) - \hat{f}(\pi_0)| \\ & \leq f(\pi_0) + |\hat{f}(\hat{\pi}_0) - f(\hat{\pi}_0)| + |\tilde{f}(\hat{\pi}_0) - \hat{f}(\hat{\pi}_0)| \\ & \quad + |\hat{f}(\pi_0) - f(\pi_0)| + |\tilde{f}(\pi_0) - \hat{f}(\pi_0)| \\ & \leq f(\pi_0) + 2 \sup_{\pi} |\hat{f}(\pi) - f(\pi)| + 2 \sup_{\pi} |\hat{f}(\pi) - \tilde{f}(\pi)|. \end{aligned}$$

where the first line follows by a triangular inequality, the second line uses that  $\hat{f}(\hat{\pi}_0) \leq \hat{f}(\pi_0)$  since  $\hat{\pi}_0$  minimizes  $\hat{f}$ . The third line uses a triangular inequality again while the last line follows by definition of the supremum. Moreover, recall that  $f(\pi_0) \leq f(\hat{\pi}_0)$  as  $\pi_0$  minimizes  $f$  and that  $|\hat{f}(\pi) - \tilde{f}(\pi)| \leq \zeta$  since, by assumption, we have that  $d_{TV}(\pi_i^*, \tilde{\pi}_i) \leq \zeta$ . Therefore, it follows that:

$$|f(\hat{\pi}_0) - f(\pi_0)| \leq 2\zeta + 2 \sup_{\pi} |\hat{f}(\pi) - f(\pi)|. \quad (3.42)$$

By application of the bounded difference inequality (McDiarmid's inequality) (Sen, 2018, Theorem 13.8), we know that for any  $t > 0$ :

$$\mathbb{P} \left[ \left| \sup_{\pi} |\hat{f}(\pi) - f(\pi)| - \mathbb{E} \left[ \sup_{\pi} |\hat{f}(\pi) - f(\pi)| \right] \right| > t \right] \leq 2e^{-2t^2K} \quad (3.43)$$

This implies that for any  $0 < \eta < 1$ , we have with probability higher than  $1 - \eta$  that:

$$\sup_{\pi} |\hat{f}(\pi) - f(\pi)| \leq \sqrt{\frac{\log(\frac{2}{\delta})}{2K}} + \mathbb{E} \left[ \sup_{\pi} |\hat{f}(\pi) - f(\pi)| \right] \quad (3.44)$$

Combining Eq. (3.42) with Eq. (3.44) and using Lemma 3.B.5 to control  $\mathbb{E} \left[ \sup_{\pi} |\hat{f}(\pi) - f(\pi)| \right]$ , we have that for any  $0 < \delta < 1$ , with probability higher than  $1 - \delta$ , it holds that:

$$|f(\hat{\pi}_0) - f(\pi_0)| \leq 2\zeta + \sqrt{\frac{2\log(\frac{2}{\delta})}{K}} + 2C\sqrt{\frac{|\mathcal{A}|}{K}}, \quad (3.45)$$

for some constant  $C$  that depends only on  $|\mathcal{A}|$ .

□

**Lemma 3.B.5.**

$$\mathbb{E} \left[ \sup_{\pi} |\hat{f}(\pi) - f(\pi)| \right] \leq C\sqrt{\frac{|\mathcal{A}|}{N}}, \quad (3.46)$$

where  $C$  is a constant that depends only on  $|\mathcal{A}|$ .

*Proof.* To control the quantity  $\mathbb{E} \left[ \sup_{\pi} |\hat{f}(\pi) - f(\pi)| \right]$ , we will use a classical result from empirical process theory (Van der Vaart, 2000, Corollary 19.35). We begin by introducing some useful notions to state the result. Denote by  $\mathcal{F}$  the set of functions  $\pi' \mapsto d_{TV}(\pi, \pi')$  that are indexed by a fixed  $\pi$ . Given a random task  $M_i \sim \mathcal{M}$ , we call  $\pi_i^*$  its optimal policy and denote by  $P$  the probability distribution of  $\pi_i^*$  when the task  $M_i$  is drawn from  $\mathcal{M}$ . Note that we can express  $f(\pi)$  as an expectation w.r.t.  $P$ :  $f(\pi) = \mathbb{E}_{\pi' \sim P} [d_{TV}(\pi, \pi')]$ . Moreover,  $\hat{f}(\pi)$  is an empirical average over i.i.d. samples  $\pi_i^*$  drawn from  $P$ .

The *bracketing number*  $N_{[]}(\epsilon, \mathcal{F}, L_2(P))$  is the smallest number of functions  $f_j$  and  $g_j$  such that for any  $\pi$ , there exists  $j$  such that  $f_j(\pi') \leq d_{TV}(\pi, \pi') \leq g_j(\pi')$  and  $\|f_j - g_j\|_{L_2(P)} \leq \epsilon$ . The following result is a direct application of (Van der Vaart, 2000, Corollary 19.35) and provides a control on  $\mathbb{E} \left[ \sup_{\pi} |\hat{f}(\pi) - f(\pi)| \right]$  in terms of the bracketing number  $N_{[]}$ :

$$\sqrt{N} \mathbb{E} \left[ \sup_{\pi} |\hat{f}(\pi) - f(\pi)| \right] \leq \int_0^R \sqrt{\log N_{[]}(\epsilon, \mathcal{F}, L_2(P))}. \quad (3.47)$$

where  $R^2 = \mathbb{E}_{\pi' \sim P} [\sup_{\pi} d_{TV}(\pi, \pi')^2] \leq 1$ . It remains to control the bracketing number  $N_{[]}$ . To achieve this, note that the functions in  $\mathcal{F}$  are all 1-Lipschitz, meaning that:

$$|d_{TV}(\pi, \pi) - d_{TV}(\pi', \pi)| \leq d_{TV}(\pi, \pi') \leq 1. \quad (3.48)$$

Moreover, the family  $\mathcal{F}$  admits the constant function  $F(\pi') = 1$  as an envelope, which means, in other words, that the following upper-bound holds:

$$\sup_{\pi} d_{TV}(\pi, \pi') \leq 1. \quad (3.49)$$

Therefore, we can apply (Van der Vaart, 2000, Example 19.7) to the family  $\mathcal{F}$ , which directly implies the following upper-bound on  $N_{[]}$ :

$$N_{[]}(\epsilon, \mathcal{F}, L_2(P)) \leq K \left( \frac{1}{\epsilon} \right)^{|\mathcal{A}|} \quad (3.50)$$

where  $K$  is a constant that depends only on  $|\mathcal{A}|$ . Combining 3.48 and 3.50 and recalling that  $R \leq 1$ , it follows that:

$$\mathbb{E} \left[ \sup_{\pi} |\hat{f}(\pi) - f(\pi)| \right] \leq C \sqrt{\frac{|\mathcal{A}|}{N}}. \quad (3.51)$$

where  $C$  is a constant that depends only on  $|\mathcal{A}|$ . □



## Appendix 3.C: Experimental Details

The policy model for all algorithms was given by the tabular softmax with single parameter vector  $\theta \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$  such that

$$\pi_{\theta}(a|s) = \frac{\exp(\theta_{s,a})}{\sum_{a' \in \mathcal{A}} \exp(\theta_{s,a'})}.$$

All agents were trained for 80,000 time steps per task using standard stochastic gradient ascent with learning rate  $\eta = 0.02$ . For methods with learned regularizers, the learning for the regularizer was halved, with  $\eta_{\text{reg}} = 0.01$ . Each episode terminated when the agent reached a leaf node. For those using regularization, the regularization weight was  $\lambda = 0.2$ . For DISTRAL, this weight was applied equally to both the KL term and the entropy term. Each task was randomly sampled with  $r(s) = 0$  for all nodes other than the leaf nodes of the subtree rooted at  $s_7$  (Fig. 3.2). For those nodes,  $r(s) \sim \text{Geom}(p)$  with  $p = 0.5$  for experiments with fixed default policies and  $p = 0.7$  for those with learned default policies. The sparsity of the reward distribution made learning challenging, and so limiting the size of the effective search space (via an effective default policy) was crucial to consistent success. A single run consisted of 5 draws from the task distribution, with each method trained for 20 runs with different random seeds. For TVPO, the softmax temperature decayed as  $\beta(k) = \exp(-k/10)$ , with  $k$  being the number of tasks. The plotted default policies in Fig. 3.5 were the average default policy probabilities in the selected states across these runs.

## Chapter 4

# Minimum Description Length Control

The previous chapter provided a theoretical analysis as well as small-scale experimental results studying regularized policy optimization for MTRL. It demonstrated that similarity among the optimal policies for a group of tasks constitutes a form of behavioral structure which can be distilled into a default policy and leveraged into faster learning of new policies. This chapter, adapted from [Moskovitz et al. \(2023a\)](#), draws inspiration from the dual process theory of biological cognition to extend these results, providing a principled approach to mitigating overfitting of the default policy to previously observed tasks, thereby ensuring that the control policy can flexibly adapt to new demands.

### 4.1 Introduction

In order to learn efficiently in a complex world with multiple rapidly changing objectives, both animals and machines must leverage past experience. This is a challenging task, as processing and storing all relevant information is computationally infeasible. How can an intelligent agent address this problem? We hypothesize that one route may lie in the *dual process theory* of cognition, a longstanding framework in cognitive psychology introduced by William James ([James et al., 1890](#)) which lies at the heart of many dichotomies in both cognitive science and machine learning. Examples include goal-directed versus habitual behavior ([Graybiel, 2008](#)), model-based versus model-free reinforcement learning ([Daw et al., 2011](#); [Sutton and Barto, 2018a](#)), and “System 1” versus “System 2” thinking

(Kahneman, 2011). In each of these paradigms, a complex, “control” process trades off with a simple, “default” process to guide actions. Why has this been such a successful and enduring conceptual motif? Our hypothesis is that default processes often serve to distill common structure from the tasks consistently faced by animals and agents, facilitating generalization and rapid learning on new objectives. For example, drivers can automatically traverse commonly traveled roads en route to new destinations, and chefs quickly learn new dishes on the back of well-honed fundamental techniques. Importantly, even intricate tasks can become automatic, if repeated often enough (e.g., the combination of fine motor commands required to swing a tennis racket): the default process must be sufficiently expressive to learn common behaviors, regardless of their complexity. In reality, most processes likely lie on a continuum between simplicity and complexity.

In reinforcement learning (RL; Sutton and Barto, 2018a), improving sample efficiency on new tasks is crucial to the development of general agents which can learn effectively in the real world (Botvinick et al., 2015; Kirk et al., 2021). Intriguingly, one family of approaches which have shown promise in this regard are *regularized policy optimization* algorithms, in which a goal-specific control *policy* is paired with a simple yet general default *policy* to facilitate learning across multiple tasks (Teh et al., 2017a; Galashov et al., 2019a; Goyal et al., 2020, 2019; Moskovitz et al., 2022a). One difficulty in algorithm design, however, is how much or how little to constrain the default policy, and in what way. An overly simple default policy will fail to identify and exploit commonalities among tasks, while a complex model may overfit to a single task and fail to generalize. Most approaches manually specify an asymmetry between the control and default policies, such as hiding input information (Galashov et al., 2019a) or constraining the model class (Lai and Gershman, 2021). Ideally, we’d like an adaptive approach that learns the appropriate degree of complexity via experience.

The *minimum description length principle* (MDL; Rissanen, 1978), which in general holds that one should prefer the simplest model that accurately fits the data, offers a guiding framework for algorithm design that does just that, enabling the default policy to optimally trade off between adapting to information from new tasks and maintaining simplicity. Inspired by dual process theory and the MDL principle, we propose *MDL-*

*control* (MDL-C, pronounced “middle-see”), a principled RPO framework for multitask RL. In Section 4.2, we formally introduce multitask RL and describe RPO approaches within this setting. In Section 4.3, we describe MDL and the variational coding framework, from which we extract MDL-C and derive its formal performance characteristics. In Section 4.5, we demonstrate its empirical effectiveness in both discrete and continuous control settings. Finally, we discuss related ideas from the literature (Section 4.6) and conclude (Section 4.7).

## 4.2 Setting

While the previous chapter focused solely on sequential MTRL, here we consider both the sequential parallel settings. The objective in each case is to maximize *average reward across tasks*, equivalent to minimizing cumulative regret over the agent’s ‘lifetime.’ More specifically, we assume a (possibly infinite) set of tasks (MDPs)  $\mathcal{M} = \{M\}$  presented to the agent by sampling from some task distribution  $\mathbb{P}_{\mathcal{M}} \in \mathcal{P}(\mathcal{M})$ . In the *sequential* task setting (Moskovitz et al., 2022a; Pacchiano et al., 2022), tasks (MDPs) are sampled one at a time from  $\mathbb{P}_{\mathcal{M}}$ , with the agent training on each until convergence. In the parallel task training (Yu et al., 2019), a new MDP is sampled from  $\mathbb{P}_{\mathcal{M}}$  at the start of every episode and is associated with a particular input feature  $g \in \mathcal{G}$  that indicates to the agent which task has been sampled. We continue the previous chapter’s focus on RPO-based approaches, here denoting the default policy by  $\pi_w$ , where  $w$  are the policy parameters.

## 4.3 The Minimum Description Length Principle

**General principle** Storing all environment interactions across multiple tasks is computationally infeasible, so multitask RPO algorithms offer a compressed representation in the form of a default policy. However, the type of information that is compressed (and that which is lost) is often hard-coded *a priori*. Preferably, we’d like an approach which can distill structural regularities among tasks without needing to know what they are beforehand. The *minimum description length* (MDL) framework offers a principled approach to this problem. So-called “ideal” MDL seeks to find the shortest so-

lution written in a general-purpose programming language<sup>†</sup> which accurately reproduces the data—an idea rooted in the concept of Kolmogorov complexity (Li and Vitányi, 2008). Given the known impossibility of computing Kolmogorov complexity for all but the simplest cases, a more practical MDL approach instead prescribes selecting the hypothesis  $H^*$  from some hypothesis class  $\mathcal{H}$  which minimizes the two-part code  $H^* = \operatorname{argmin}_{H \in \mathcal{H}} L(\mathcal{D}|H) + L(H)$ , where  $L(\mathcal{D}|H)$  is the number of bits required to encode the data given the hypothesis and  $L(H)$  is the number of bits needed to encode the hypothesis itself. There are a variety of so-called *universal* coding schemes which can be used to model these quantities.

**Variational code** One popular encoding scheme is the variational code (Blier and Olivier, 2018; Hinton and Van Camp, 1993; Honkela and Valpola, 2004):

$$L_{\nu}^{\text{var}}(\mathcal{D}) = \underbrace{\mathbb{E}_{\theta \sim \nu} [-\log p_{\theta}(\mathcal{D})]}_{L^{\text{var}}(\mathcal{D}|H)} + \underbrace{\text{KL}[\nu(\cdot); p(\cdot)]}_{L^{\text{var}}(H)} \quad (4.1)$$

where the hypothesis class is of a set of parametric models  $\mathcal{H} = \{p_{\theta}(\mathcal{D}) : \theta \in \Theta\}$ . The model parameters are random variables with prior distribution  $p(\theta)$  and  $\nu(\theta)$  is any distribution over  $\Theta$ . Minimizing  $L_{\nu}^{\text{var}}(\mathcal{D})$  with respect to  $\nu$  is equivalent to performing variational inference, maximizing a lower-bound to the data log-likelihood  $\log p(\mathcal{D}) = \log \int p(\theta) p_{\theta}(\mathcal{D}) d\theta \geq -L_{\nu}^{\text{var}}(\mathcal{D})$ . Roughly speaking, MDL encourages the choice of “simple” models when limited data are available (Grunwald, 2004). In the variational coding scheme, simplicity is enforced via the choice of prior.

**Sparsity-inducing priors and variational dropout** Sparsity-inducing priors can be used to improve the compression rate within the variational coding scheme, as they encourage the model to prune out parameters that do not contribute to reducing  $L^{\text{var}}(\mathcal{D}|\theta)$ . Many sparsity-inducing priors belong to the family of scale mixtures of normal distributions (?):  $z \sim p(z)$ ,  $\theta \sim p(\theta|z) = \mathcal{N}(w; 0, z^2)$  where  $p(z)$  defines a distribution over the variance  $z^2$ . Common choices of  $p(z)$  include the Jeffreys prior  $p(z) \propto |z|^{-1}$  (Jeffreys, 1946), the inverse-Gamma distribution, and the half-Cauchy distribution (Polson and Scott, 2012; Gelman, 2006). Such priors have deep connections to MDL theory.

<sup>†</sup>The *invariance theorem* (Kolmogorov, 1965) ensures that, given a sufficiently long sequence, Kolmogorov complexity is invariant to the choice of general-purpose language.

For example, the Jeffreys prior in conjunction with an exponential family likelihood is asymptotically identical to the *normalized maximum likelihood* estimator, perhaps the most fundamental ‘MDL’ estimator (Grünwald and Roos, 2019). *Variational dropout* (VDO) is an effective algorithm for minimizing Eq. (4.1) for these sparsity-inducing priors (Louizos et al., 2017; Kingma et al., 2015; Molchanov et al., 2017). Briefly, this involves choosing an approximate posterior distribution with the form

$$p(w, z|\mathcal{D}) \approx \nu(w, z) = \mathcal{N}(z; \mu_z, \alpha\sigma_z^2)\mathcal{N}(w; z\mu, z^2\sigma^2 I_d) \quad (4.2)$$

and optimizing Eq. (4.1) via stochastic gradient descent on the variational parameters given by  $\{\alpha, \mu_z, \sigma_z^2, \mu, \sigma^2\}$ . As its name suggests—and importantly for its ease of application to large models—VDO can be implemented as a form of dropout (Srivastava et al., 2014) by reparameterizing the noise on the weights as activation noise (Kingma et al., 2015). Application of VDO to Bayesian neural networks has achieved impressive compression rates, sparsifying deep neural networks while maintaining prediction performance on supervised learning problems (Molchanov et al., 2017; Louizos et al., 2017). Equipped with a powerful approach for MDL-grounded posterior inference, we can now integrate these ideas with multitask RPO.

## 4.4 Minimum Description Length Control

As part of its underlying philosophy, the MDL principle holds that 1) *learning* is the process of discovering regularity in data, and 2) any regularity in the data can be used to *compress* it (Grünwald, 2004). Applying this perspective to RL is non-obvious—from the agent’s perspective, what ‘data’ is it trying to compress? Our hypothesis, which forms the basis for the framework we propose in this chapter, is that an agent faced with a set of tasks in the world should seek to elucidate structural regularity from the environment interactions generated by the optimal policies for the tasks. This makes intuitive sense: the agent ought to compress information which indicates how to correctly perform the tasks with which it is faced. That is, we propose that the *data* in multitask RL are the state-action interactions generated by the optimal policies for a set of tasks:  $\mathcal{D} = \{\mathcal{D}_M\}_{M \in \mathcal{M}} = \{(s, a) : \forall s \in \mathcal{S}, a \sim \pi_M^*(\cdot|s)\}_{M \in \mathcal{M}}$ . This interpretation is in

line with work suggesting that a useful operational definition of ‘task’ can be derived directly from the set of optimal (or near-optimal) policies it induces (Abel et al., 2021). It also suggests a natural mapping to the multitask RPO framework. In this view, the control policy is responsible for learning and the default policy for compression: by converging to the optimal policy for a given task, the control policy “discovers” regularity which is then distilled into a low-complexity representation by the default policy. In our approach, the default policy is encouraged to learn a compressed representation not by artificially constraining the network architecture or via hand-designed information asymmetry, but rather through a prior distribution  $p(w)$  over its parameters which biases a variational posterior  $\nu(w)$  towards simplicity. The default policy is therefore trained to minimize the variational code:

$$\begin{aligned} \operatorname{argmin}_{\nu \in \mathbf{N}} \mathbb{E}_{s, a \sim \mathcal{D}} \mathbb{E}_{w \sim \nu} & -\log \pi_w(a|s) + \text{KL}[\nu(\cdot); p(\cdot)] \\ & = \operatorname{argmin}_{\nu \in \mathbf{N}} \mathbb{E}_{M \sim \mathbb{P}_{\mathcal{M}}} \mathbb{E}_{\substack{s \sim d^{\pi_M^*} \\ w \sim \nu_\phi}} \text{KL}[\pi_M^*(\cdot|s); \pi_w(\cdot|s)] + \text{KL}[\nu(\cdot); p(\cdot)], \end{aligned} \quad (4.3)$$

where  $\mathbf{N}$  is the distribution family for the posterior. This suggests the approach presented in Algorithm 4 in which for each task  $M_k$ , the control policy  $\pi_\theta$  is trained to approximate the optimal policy  $\pi_k^*$  via RPO, and the result is compressed into a new default policy distribution  $\nu_{k+1}$ . We now further motivate sparsity-inducing priors for the default policy in multitask settings, derive formal performance guarantees for MDL-C, and demonstrate its empirical effectiveness.

#### 4.4.1 Motivating the choice of sparsity-inducing priors

In Section 4.3, compression (via pruning extraneous parameters) is the primary motivation for using sparsity-inducing priors that belong to the family of scaled-mixtures of normal distributions. Intuitively, placing a distribution over the default parameters reflects the agent’s *epistemic uncertainty* about the task distribution—when few tasks have been sampled, a sparse prior prevents the default policy from overfitting to spurious correlations in the limited data that the agent has collected. Here, we make this motivation more precise, describing an example generative model of optimal policy parameters which

**Algorithm 4** MDL-C for Sequential Multitask Learning with Persistent Replay

- 
- 1: Require: task distribution  $\mathbb{P}_{\mathcal{M}}$ , policy class  $\Theta$ , non-increasing coefficients  $\{\eta_k\}_{k=1}^K$
  - 2: Initialize: default policy distribution  $\nu_1 \in \mathcal{N} \subseteq \mathcal{P}(\Theta)$ , default policy dataset  $\mathcal{D}_0 \leftarrow \emptyset$
  - 3: **for** tasks  $k = 1, 2, \dots, K$  **do**
  - 4:   Sample a task  $M_k = (\mathcal{S}, \mathcal{A}, \mathbf{P}_k, r_k, \gamma_k, \rho_k) \sim \mathcal{P}_{\mathcal{M}}(\cdot)$
  - 5:   Optimize control policy:

$$\theta_k \leftarrow \operatorname{argmax}_{\theta \in \Theta} V_{M_k}^{\pi} - \alpha \mathbb{E}_{s \sim d_{\rho_k}^{\pi}} \mathbb{E}_{w \sim \nu_k} \text{KL}[\pi_w(\cdot|s); \pi_{\theta}(\cdot|s)] \quad (4.4)$$

- 6:   Add data to default policy replay ( $J = |\mathcal{S}|$  for finite/small state spaces):

$$\mathcal{D}_k \leftarrow \mathcal{D}_{k-1} \cup \{(s_j, \hat{\pi}_{\theta_k}(s_j))\}_{j=1}^J. \quad (4.5)$$

- 7:   Update default policy distribution:

$$\nu_{k+1} \leftarrow \operatorname{argmin}_{\nu \in \mathcal{N}} \frac{1}{\eta_{k-1}} \text{KL}[\nu(\cdot); p(\cdot)] + \sum_{i=1}^k \sum_{j=1}^J \mathbb{E}_{w \sim \nu} \text{KL}[\hat{\pi}_{\theta_i}^*(\cdot|s_j); \pi_w(\cdot|s_j)] \quad (4.6)$$

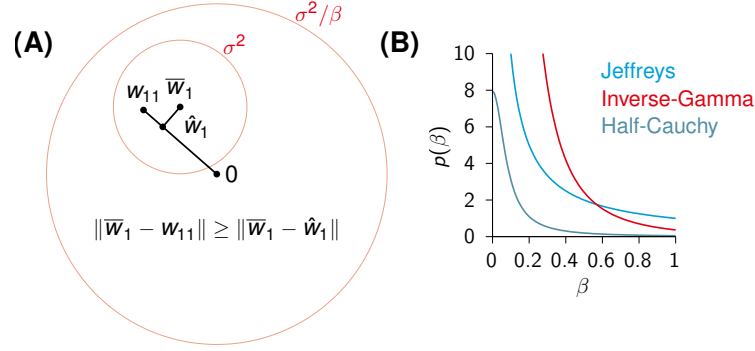
- 8: **end for**
- 

provides a principled interpretation for prior choice  $p(z)$  in multitask RL.

**Generative model of optimal policy parameters** Consider a set of tasks  $\mathcal{M} = \{M_{ik}\}_{i=1, k=1}^{I, K_i}$  that are clustered into  $I$  groups, such that the MDPs in each group are more similar to one another than to members of other groups. As an example, the overall family  $\mathcal{M}$  could be all sports, while clusters  $\mathcal{M}_i \subseteq \mathcal{M}$  could consist of, say, ball sports or endurance competitions. To make this precise, we assume that the optimal policies of every MDP belong to a parametric family  $\Pi = \{\pi_w(\cdot|s) : w \in \mathbb{R}^d, \forall s \in \mathcal{S}\}$  (e.g., softmax policies with parameters  $w$ ), and that the optimal policies for each group are randomly distributed within parameter space. In particular, we assume that the parameters of the optimal policies of  $\mathcal{M}$  have the following generative model:

$$\bar{w}_i | \beta, \sigma^2 \sim \mathcal{N}(\bar{w}_m; 0, (1 - \beta)\beta^{-1}\sigma^2 I_d), \quad w_{ik} | \bar{w}_i, \sigma^2 \sim \mathcal{N}(w_{ik}; \bar{w}_i, \sigma^2 I_d).$$





**Figure 4.1:** (A) Illustration of a generative model of optimal policy parameters.  $\hat{w}_1 = (1 - \beta)w_{11}$  shrinks towards the origin, growing closer to  $\bar{w}_1$  than  $w_{11}$ . (B) Sparsity-inducing priors over  $\beta$ .

where  $I_d$  is the  $d$ -dimensional identity matrix. If we marginalize out  $\bar{w}_i$ , we get the marginal distribution  $p(w_{ik}|\beta, \sigma^2) = \mathcal{N}(w_{ik}; 0, \sigma^2\beta^{-1}I_d)$ . We can then visualize the parameter distribution of the optimal policies for  $\mathcal{M}$  as a  $d$ -dimensional Gaussian within which lie clusters of optimal policies for related tasks which are themselves normally distributed (see Fig. 4.1A for  $d = 2$ ).

**Interpretation of  $\beta$**  The parameter  $\beta \in (0, 1]$  can be interpreted as encoding the squared distance between optimal policy parameters within a group divided by the squared distance between optimal policies in  $\mathcal{M}$ . Intuitively,  $\beta$  determines how much information one gains about the optimal parameters of a task in a group, given knowledge about the optimal parameters of another task in the same group. To see this, we compute our posterior belief about the value  $\bar{w}_i$  given observation of  $w_{ik}$ :  $p(\bar{w}_i|w_{ik}, \beta, \sigma^2) = \mathcal{N}(\bar{w}_i; (1 - \beta)w_{ik}, (1 - \beta)\sigma^2I_d)$ . When  $\beta = 1$  (inner circle in Fig. 4.1A has the same radius as the outer circle), our posterior mean estimate of  $\bar{w}_i$  is simply 0, suggesting we have learned nothing new about the mean of the optimal parameters in group  $i$ , by observing  $w_{ik}$ . In the other extreme when  $\beta \rightarrow 0$ , the posterior mean approaches the maximum-likelihood estimator  $w_{ik}$ , suggesting that observation of  $w_{ik}$  provides maximal information about the optimal parameters in group  $i$ . Any  $\beta$  in between the two extremes results in an estimator that “shrinks”  $w_{ik}$  towards 0. The value of  $\beta$  thus has important implications for multitask learning. Suppose an RL agent learns the optimal parameters  $w_{11}$  (task 1, group 1), and proceeds to learn task 2 in group 1. The value of  $\beta$  determines whether  $w_{11}$  can be used to inform the agent’s learning of  $w_{21}$ . In this way,

$\beta$  determines the effective degree of epistemic uncertainty the agent has about the task distribution.

**Choice of  $p(\beta)$  and connection to  $p(z)$**  Given its importance, it's natural to ask what value  $\beta$  should take. Instead of treating  $\beta$  as a parameter, we can choose a prior  $p(\beta)$  and perform Bayesian inference. Ideally,  $p(\beta)$  should (i) encode our prior belief about the extent to which the optimal parameters cluster into groups and (ii) result in a posterior mean estimator  $\hat{w}^{(p(\beta))}(x) = 1 - \mathbb{E}[\beta|x]x$  that is close to  $\bar{w}$  for  $x|\bar{w} \sim \mathcal{N}(x; \bar{w}, \sigma^2)$ . This condition encourages the expected default policy (under the posterior  $\nu$ ; Eq. (4.3)) to be close to optimal policies in the same MDP group (centered at  $\bar{w}$ ). One prior choice that satisfies both conditions is  $p(\beta) \propto \beta^{-1}$ . It places high probability for small  $\beta$  and low probability for high  $\beta$ , thus encoding the prior belief that the optimal task parameters are clustered (see Fig. 4.1B; blue). It is instructive to compare  $p(\beta) \propto \beta^{-1}$  with two extreme choices of  $p(\beta)$ . When  $p(\beta) = \delta(\beta - 1)$ ,  $p(z) = \delta(\sigma)$  and the marginal  $p(w)$  is the often-used Gaussian prior over the parameters  $w$  with fixed variance  $\sigma^2$ . This corresponds to the prior belief that knowing  $w_{i1}$  provides no information about  $w_{i2}$ . On the other hand,  $p(\beta) = \delta(\beta)$  recovers a uniform prior over the parameters  $w$  and reflects the prior belief that the MDP groups are infinitely far apart. In relation to (ii), one can show the  $\hat{w}^{(p(\beta))}$  strictly dominates the maximum-likelihood estimator  $\hat{w}^{(\text{ML})}(x) = x$  (Efron and Morris, 1973; Appendix 4.D), for  $p(\beta) \propto \beta^{-1}$ . This means  $\text{MSE}(\bar{w}, \hat{w}^{(p(\beta))}) \leq \text{MSE}(\bar{w}, \hat{w}^{(\text{ML})})$  for all  $\bar{w}$ , where  $\text{MSE}(\bar{w}, \hat{w}) = \mathbb{E}_{x \sim \mathcal{N}(x; \bar{w}, \sigma^2)} \|\bar{w} - \hat{w}(x)\|^2$ .

**Connection to  $p(z)$  and application of VDO** Defining  $z^2 = \sigma^2 \beta^{-1}$  and applying the change-of-variable formula to  $p(\beta) \propto \beta^{-1}$  gives  $p(z) \propto |z|^{-1}$  and thus the Normal-Jeffreys prior in Section 4.3. VDO (see Section 4.3) can then be applied to obtain an approximate posterior  $\nu(w, z)$  which minimizes the variational code Eq. (4.3). Similar correspondences may also be derived for the inverse-Gamma distribution and the half-Cauchy distribution (Fig. 4.1B; Appendix 4.D).

#### 4.4.2 Performance Analysis

At a fundamental level, we'd like assurance (i) that MDL-C's default policy will be able to effectively distill the optimal policies for previously observed tasks, and (ii) that regularization using this default policy gives strong performance guarantees for the control

policy on future tasks.

**Default policy performance** One way we can verify (i) is to obtain an upper bound on the average KL between default policies sampled from the default policy distribution and an optimal policy for a task sampled from the task distribution. This enables us to perform analysis using *online convex optimization* (OCO). In OCO, the learner observes a series of convex loss functions  $\ell_k : \mathbf{N} \rightarrow \mathbb{R}, k = 1, \dots, K$ , where  $\mathbf{N} \subseteq \mathbb{R}^d$  is a convex set. After each round, the learner produces an output  $x_k \in \mathbf{N}$  for which it will then incur a loss  $\ell_k(x_k)$  (Orabona, 2019). At round  $k$ , the learner is usually assumed to have knowledge of  $\ell_1, \dots, \ell_{k-1}$ , but no other assumptions are made about the sequence of loss functions. The learner's goal is to minimize its average regret. For further background, see Appendix 4.F. Crucially, the MDL-C learning procedure for the default policy distribution is equivalent to *follow the regularized leader* (FTRL), an OCO algorithm which enjoys sublinear regret. In each round of FTRL, the learner selects the solution  $x \in \mathbf{N}$  according to the following general objective:  $x_{k+1} = \operatorname{argmin}_{x \in \mathbf{N}} \psi_k(x) + \sum_{i=1}^{k-1} \ell_i(x)$ , where  $\psi : \mathbf{N} \rightarrow \mathbb{R}$  is a convex regularization function. Using standard results, this connection allows us to bound MDL-C's regret in learning the default policy distribution. All proofs are provided in Appendix 4.G.

**Proposition 4.4.1** (Persistent Replay FTRL Regret). *Let tasks  $M_k$  be independently drawn from  $\mathbb{P}_{\mathcal{M}}$  at every round, and let them each be associated with a deterministic optimal policy  $\pi_k^* : \mathcal{S} \rightarrow \mathcal{A}$ . We make the following mild assumptions: i)  $\pi_w(a^*|s) \geq \epsilon > 0 \forall s \in \mathcal{S}$ , where  $a^* = \pi_k^*(s)$  and  $\epsilon$  is a constant. ii)  $\min_{\nu} \operatorname{KL}[\nu(\cdot); p(\cdot)] = 0$  asymptotically as  $\operatorname{Var}[\nu] \rightarrow \infty$ . Then with  $\eta_{k-1} = \log(1/\epsilon)\sqrt{k}$ , Algorithm 4 guarantees*

$$\frac{1}{K} \sum_{k=1}^K \ell_k(\nu_k) - \frac{1}{K} \sum_{k=1}^K \ell_k(\bar{\nu}_K) \leq (\operatorname{KL}[\bar{\nu}_K; p] + 1) \frac{\log(1/\epsilon)}{\sqrt{K}}, \quad (4.7)$$

where  $\bar{\nu}_K = \operatorname{argmin}_{\nu \in \mathbf{N}} \sum_{k=1}^K \ell_k(\nu)$ .

In other words, the average regret decreases at rate  $\mathcal{O}(1/\sqrt{K})$  with respect to the number of observed tasks.

**Control policy performance** Intuitively, this result shows that the average regret is upper-bounded by factors which depend on the divergence of the barycenter distribu-

tion from the prior and the “worst-case” prediction of the default policy. Importantly, the KL between the default policy distribution and the barycenter distribution goes to zero as  $K \rightarrow \infty$ . We can also now be assured of point (ii) above, in that this result can be used to obtain a sample-complexity bound for the *control* policy. Specifically, we can use Proposition 4.G.2 to place an upper-bound on the total variation distance between default policies sampled from  $\nu$  and the KL between the maximum likelihood solution and a sparsity-inducing prior  $p$ . This is useful, as it allows to translate low regret for the default policy into a sample complexity result for the control policy using Moskowitz et al. (2022a), Lemma 5.2.

**Proposition 4.4.2** (Control Policy Sample Complexity). *Under the setting described in Proposition 4.G.2 denote by  $T_k$  the number of iterations to reach  $\epsilon$ -error for  $M_k$  in the sense that  $\min_{t \leq T_k} \{V^{\pi_k^*} - V^{(t)}\} \leq \epsilon$ . whenever  $t > T_k$ . Further, denote the upper-bound in Eq. (4.49) by  $G(K)$ . In a finite MDP, from any initial  $\theta^{(0)}$ , and following gradient ascent,  $\mathbb{E}_{M_k \sim \mathbb{P}_{\mathcal{M}}} [T_k]$  satisfies:*

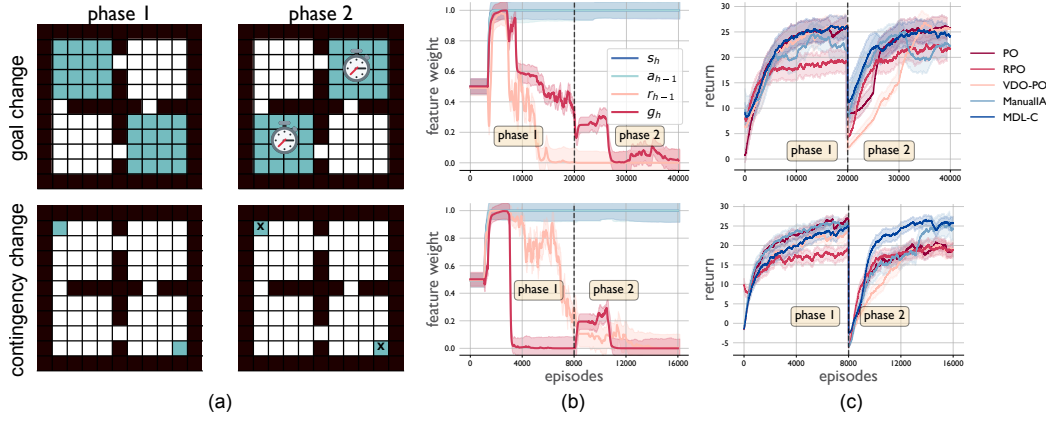
$$\mathbb{E}_{M_k \sim \mathbb{P}_{\mathcal{M}_i}} [T_k] \geq \frac{80|\mathcal{A}|^2|\mathcal{S}|^2}{\epsilon^2(1-\gamma)^6} \mathbb{E}_{M_k \sim \mathbb{P}_{\mathcal{M}_i} \atop s \sim \text{Unif}_{\mathcal{S}}} \left[ \kappa_{\mathcal{A}}^{\alpha_k}(s) \left\| \frac{d_{\rho}^{\pi_k^*}}{\mu} \right\|_{\infty}^2 \right],$$

where  $\alpha_k(s) := d_{\text{TV}}(\pi_k^*(\cdot|s), \hat{\pi}_0(\cdot|s)) \leq \sqrt{G(K)}$ ,  $\kappa_{\mathcal{A}}^{\alpha_k}(s) = \frac{2|\mathcal{A}|(1-\alpha(s))}{2|\mathcal{A}|(1-\alpha(s))-1}$ , and  $\mu$  is a measure over  $\mathcal{S}$  such that  $\mu(s) > 0 \forall s \in \mathcal{S}$ .

Intuitively, this means that when the average number of samples is sufficiently large, the control policy is guaranteed to have reached  $\epsilon$  error. Therefore, as the agent is trained on more tasks, the default policy distribution regret, upper-bounded by  $G(K)$ , decreases asymptotically to zero, and as the default policy regret decreases, the control policy will learn more rapidly, as  $\text{poly}(G(K))$ .

## 4.5 Experiments

We tested MDL-C applied to discrete and continuous control in both the sequential and parallel task settings. To quantify performance, in addition to measuring per-task reward, we also report the cumulative regret for each method in each experimental setting in Section 4.I.8.



**Figure 4.2:** MDL-C rapidly adapts to new goal locations (top row) and rule changes (bottom row). All curves represent averages taken over 10 random seeds, with the shading indicating standard error.

#### 4.5.1 2D Navigation

We first test MDL-C in the classic FourRooms environment (Fig. 4.2a, (Sutton et al., 1999)). The baselines in this case are entropy-regularized policy optimization (PO), regularized policy optimization with no constraint on the default policy (RPO), an agent with VDO applied to the control policy and no default policy (VDO-PO), and MANUALIA (Galashov et al., 2019a) in which the goal feature is manually withheld from the default policy. Observe that RPO is an approximate version of TVPO, introduced in the previous chapter. Details for all methods can be found in Appendix 4.H.

**Generalization Across Goals** In the first setting, we test MDL-C’s ability to facilitate rapid learning on previously unseen goals. In the first phase of training, a single goal location is randomly sampled at the start of each episode, and may be placed anywhere in two of the four rooms in the environment (Fig. 4.2a, top left). In the second phase, the goal location is again randomly sampled at the start of each episode, but in this case, only in the rooms which were held out in the first phase. Additionally, the agent is limited to 25 rather than 100 steps per episode. Importantly, VDO induces the MDL-C default policy to ignore input features which are, on average, less predictive of the control policy’s behavior. In this case, the default policy learns to ignore the goal feature and the reward obtained on the previous timestep. This is because, when averaging across goal locations, the agent’s current position ( $s_h$ ) and its previous direction ( $a_{h-1}$ ) are more informative of its next action—typically, heading towards the nearest door. In contrast, the un-regularized

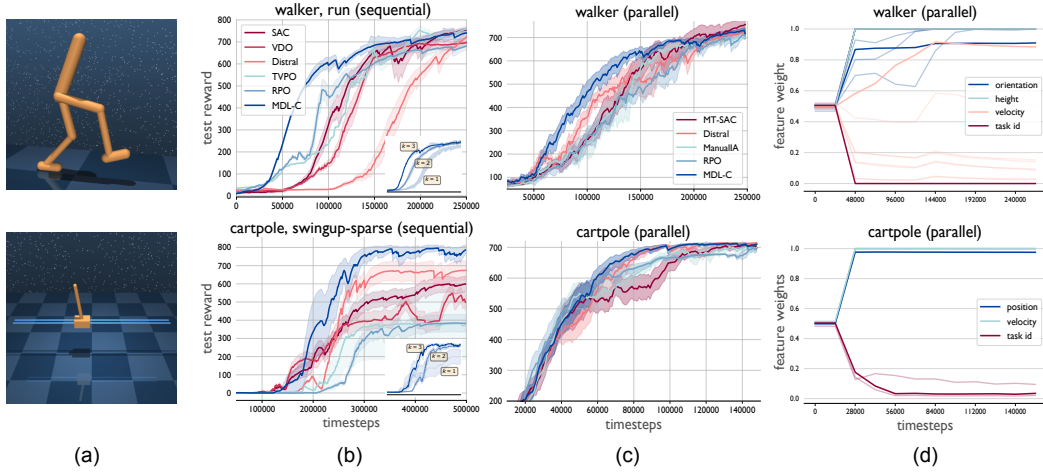
default policy of the RPO agent does not drop these features (Appendix 4.I for a visualization and Appendix 4.H for more details). By learning to ignore the goals present in phase 1 and encoding useful behavior regardless of goal location, MDL-C’s develops more effective regularization in phase 2, enabling it to adapt more quickly than other methods (Fig. 4.2c, top), particularly RPO, which overfits to phase 1’s goals. MANUALIA also adapts quickly, as its default policy is hard-coded to ignore the goal feature.

**Robustness to Rule Changes** In this setting, there are only two possible goal locations, one at the top left of the environment, and the other at the bottom right. In training phase 1, the agent receives a goal feature as input which indicates the state index of the rewarded location for that episode. In phase 2, the goal feature switches from marking the reward location to marking the unrewarded location. That is, if the reward is in the top left, the goal feature will point to the bottom right. Here, the danger for the agent isn’t overfitting to a particular goal or goals, but rather “overfitting” to the reward-based rules associated with a given feature. As we saw in Fig. 4.2c (top), an un-regularized default policy, will copy the control policy and overfit to a particular setting. Fortunately, the MDL-C default policy learns to ignore features which are, on average, less useful for predicting the control policy’s behavior—the goal and previous reward features. This renders the agent more robust to contingency switches like the one described, as we can see in Fig. 4.2c (bottom). These examples illustrate that MDL-C enables agents to effectively learn the consistent structure of a group of tasks, regardless of its semantics, and “compress out” information which is less informative on average.

### 4.5.2 Continuous Control

A more challenging application area is that of high-dimensional continuous control. In this setting, we presented agents with multitask learning problems using environments from the DeepMind Control Suite (DMC; (Tassa et al., 2018)). We used soft actor critic (SAC; (Haarnoja et al., 2018)) as the base agent. We tested MDL-C in both the sequential and parallel settings on two domains from DMC: Walker and Cartpole (Fig. 4.3a). Additional details can be found in Appendix 4.H.

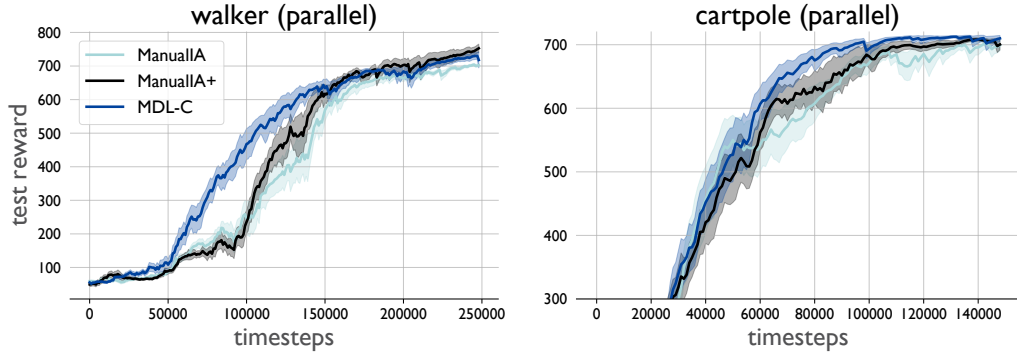
**Sequential Tasks** In the sequential setting, tasks are sampled one at a time uniformly without replacement from the available tasks within each domain, with the default pol-



**Figure 4.3:** MDL-C improves both sequential and parallel learning in continuous control tasks. All curves represent averages taken over 8 random seeds, with the shading indicating standard error. In (b), insets show the improvement of MDL-C as  $k$  increases, and in (d), solid curves represent averages over each feature within a category.

icy distribution conserved across tasks. For walker, these tasks are `stand`, `walk`, and `run`. In `stand`, the agent is rewarded for increasing the height of its center of mass, and in the latter two tasks, an additional reward is given for forward velocity. For cartpole, there are four tasks: `balance`, `balance-sparse`, `swingup`, and `swingup-sparse`. In the `balance` tasks, the agent must keep a rotating pole upright, and in the `swingup` tasks, it must additionally learn to swing the pole upwards from an initial downward orientation. Performance results for the hardest task within each domain (`run` in walker and `swingup-sparse` in cartpole) for each method are plotted in Fig. 4.3b, where  $k$  indicates the task round at which the task was sampled. We can see that as  $k$  increases (as more tasks have been seen previously), MDL-C’s performance improves. Importantly, the RPO agent’s default policy, which is un-regularized, overfits to the previous task, essentially copying the optimal policy’s behavior. This can severely hinder the agent’s performance when the subsequent task requires different behavior. For example, on `swingup-sparse`, if the previous task is `swingup`, the RPO agent performs well, as the goal is identical. However, if the previous task is `balance` or `balance-sparse`, the agent never learns to swing the pole upwards, significantly reducing its average performance. Another important point to note is that because the agent is not given an explicit goal feature in this setting, methods like `MANUALIA` which rely on prior knowledge





**Figure 4.4:** To test the effect of information asymmetry on its on performance, we trained a variant of MANUALIA in which we withheld the input features that MDL-C learned to gate out (Fig. 4.3) in addition to the task ID feature. We call this modified method MANUALIA+. Average performance is plotted above over 10 seeds, with the shading representing one unit of standard error. We can see that while MANUALIA+ narrowly outperforms MANUALIA, the performance gains of MDL-C can’t solely be ascribed to effective information asymmetry.

about the agent’s inputs cannot be applied.

**Parallel Tasks** We also tested parallel-task versions of SAC, MANUALIA, and MDL-C based on the model of Yu et al. (2019). In this framework, a task within each domain is randomly sampled at the start of each episode and the agent learns a single control policy for all tasks. Performance is plotted in Fig. 4.3c, where we can again see that MDL-C accelerates convergence relative to the baseline methods. This marks a difference compared to the easier FourRooms environment, in which MDL-C and MANUALIA performed roughly the same. As before, one clue to the difference can be found in the input features that the MDL-C default policy chooses to ignore (Fig. 4.3d). For walker, inputs are 24-dimensional, with 14 features related to the joint orientations, 1 feature indicating the height of the agent’s center of mass, and 9 features indicating velocity components. For cartpole, there are 5 input dimensions, with 3 pertaining to position and 2 to velocity. In the walker domain, where the performance difference is greatest, the MDL-C agent not only ignores the added task ID feature, but also the several features related to velocity. In contrast, in the cartpole domain, MDL-C only ignores the task ID feature, just as MANUALIA does, and the performance gap is smaller. This illustrates that MDL-C learns to compress out spurious information even in settings for which it is difficult to identify *a priori*. In order to test the effect of the learned asymmetry on performance more directly,



we implemented a variant of MANUALIA in which all of the features which MDL-C learned to ignore were manually hidden from the default policy (Fig. 4.4). Interestingly, while this method improved over standard MANUALIA, it didn’t completely close the gap with MDL-C, indicating there are downstream effects within the network beyond input processing which are important for the default policy’s effectiveness. We hope to explore these effects in more detail in future work.

## 4.6 Related Work

MDL-C can be viewed as an extension of recent approaches to learning default policies (“behavioral priors”) from the optimal policies of related tasks (Teh et al., 2017a; Tirumala et al., 2020a). For a default policy to be useful for transfer learning, it is crucial to balance the ability of the default policy to “copy” the control policies with its expressiveness. If the default policy is too expressive, it is likely to overfit on past tasks and fail to generalize to unseen tasks. Whereas prior work primarily hand-crafts structural constraints into the default policies to avoid overfitting (e.g., by hiding certain state information from the default policy; Galashov et al., 2019a), MDL-C learns such a balance from data with sparsity-inducing priors via variational inference. MDL-C may also be derived from the RL-as-inference framework (Levine, 2018a; Appendix 4.A). MDL-C thus has close connections with algorithms such as MPO (Abdolmaleki et al., 2018) and VIREL (Fellows et al., 2020), discussed in Appendix 4.A. As a general framework, MDL-C is also connected to the long and well-established literature on choosing appropriate Bayesian priors (Jeffreys, 1946; Bernardo, 2005; Casella, 1985), and more recent work that focuses on learning such priors for large-scale machine learning models (Nalisnick and Smyth, 2017; Nalisnick et al., 2021; Atanov et al., 2018). For a further discussion of related work, particularly concerning the application of MDL to the RL setting, see Appendix 4.C.

## 4.7 Conclusion

Inspired by dual process theories and the MDL principle, we propose a regularized policy optimization framework for multitask RL which aims to learn a simple default policy encoding a low-complexity distillation of the optimal behavior for some family of tasks. By encouraging the default policy to maintain a low effective description length, MDL-

C ensures that it does not overfit to spurious correlations among the (approximately) optimal policies learned by the agent. We described MDL-C’s formal properties and demonstrated its empirical effectiveness in discrete and continuous control tasks. There are of course limitations of MDL-C, which we believe represent opportunities for future work (see Appendix 4.E). Promising research directions include integrating MDL-C with multitask RL approaches which balance a larger set of policies (Barreto et al., 2020; Moskovitz et al., 2022c; Thakoor et al., 2022) as well considering nonstationary environments (Parker-Holder et al., 2022). We hope MDL-C inspires further advances in multitask RL.

## Appendix

### Appendix 4.A: Reinforcement Learning as Inference

The control as inference framework (Levine, 2018a) associates every time step  $h$  with a binary “optimality” random variable  $\mathcal{O}_h \in \{0, 1\}$  that indicates whether  $a_h$  is optimal at state  $s_h$  ( $\mathcal{O}_h = 1$  for optimal, and  $\mathcal{O}_h = 0$  for not). The optimality variable has the conditional distribution  $P(\mathcal{O}_h = 1 | s_h, a_h) = \exp(r(s_h, a_h))$ , which scales exponentially with the reward received taking action  $a_h$  in state  $s_h$ .

Denote  $\mathcal{O}_H$  as the event that  $\mathcal{O}_s = 1$  for  $s = 0, \dots, H-1$ . Then the log-likelihood that a policy  $\pi_w(a|s)$  is optimal over a horizon  $H$  is given by:

$$\mathbb{P}(\mathcal{O}_H) = \int \mathbb{P}(\mathcal{O}_H | \tau) \mathbb{P}^{\pi_w}(\tau | w) p(w) d\tau dw.$$

By performing variational inference, we can lower-bound the log-likelihood with the ELBO:

$$\begin{aligned} \log \mathbb{P}(\mathcal{O}_H) \geq \mathbb{E}_{\nu_{\pi}(\tau)} \sum_{h=0}^{H-1} (r(s_h, a_h) - \mathbb{E}_{\nu_{\theta}(w)} \text{KL}[\pi_{\theta}(a_h | s_h), \pi_w(a_h | s_h)]) \\ - \text{KL}[\nu_{\phi}(w), p(w)], \end{aligned} \quad (4.8)$$

where  $\nu_{\theta, \phi}(\tau, w) = \nu_{\theta}(\tau) \nu_{\phi}(w)$  is the variational posterior,

$$\nu_{\theta}(\tau) = \rho(s_0) \prod_{h=0}^{H-1} \mathbb{P}(s_{h+1} | s_h, a_h) \pi_{\theta}(a_h, s_h)$$

and  $\{\theta, \phi\}$  are the variational parameters. We can maximize this objective iteratively by performing coordinate ascent on  $\{\theta, \phi\}$ :

$$\theta \leftarrow \theta + \eta \nabla_{\theta} \left( \mathbb{E}_{\nu_{\theta}(\tau)} \sum_{h=0}^{H-1} (r(s_h, a_h) - \mathbb{E}_{\nu_{\theta}(w)} \text{KL}[\pi_{\theta}(a_h | s_h), \pi_w(a_h | s_h)]) \right), \quad (4.9)$$

$$\phi \leftarrow \phi - \eta \nabla_{\phi} \left( \mathbb{E}_{\nu_{\theta}(\tau)} \sum_{h=0}^{H-1} \mathbb{E}_{\nu_{\theta}(w)} \text{KL}[\pi_{\theta}(a_h | s_h), \pi_w(a_h | s_h)] + \text{KL}[\nu_{\phi}(w), p(w)] \right) \quad (4.10)$$

where  $\eta$  is a learning rate parameter. Note that Eq. (4.10) is equivalent to Eq. (4.3) and Eq. (4.56), and Eq. (4.9) is equivalent to Eq. (4.55) with the KL reversed.

**Connection to Maximum a Posteriori Policy Optimization (MPO)** MDL-C is closely related to MPO (Abdolmaleki et al., 2018), with three key differences. First, MDL-C performs variational inference on the parameters of the default policy with an approximate posterior  $\nu_\phi(w)$ , whereas MPO performs MAP inference. Second, MPO places a normal prior on  $w$ , which in effect penalizes the L2 norm of  $w$ . In contrast, MDL-C uses sparsity-inducing priors such as the normal-Jeffreys prior. Third, MDL-C uses a parametric  $\pi_\theta$ , whereas MPO uses a non-parametric one<sup>2</sup>. While there is also a parametric variant of MPO, this variant does not maintain  $\theta$  and  $\phi$  separately. Instead, this variant directly sets  $\theta$  to  $\phi$  in Eq. (4.9). This illustrates the key conceptual difference between MDL-C and MPO. MDL-C makes a clear distinction between the control policy  $\pi_\theta$  and the default policy  $\pi_w$ , with the two policies serving two distinct purposes: the control policy for performing on the current task, the default policy for distilling optimal policies across tasks and generalizing to new ones. MPO, on the other hand, treats  $\pi_\theta$  and  $\pi_w$  as fundamentally the same object.

Like MPO, VIREL (Fellows et al., 2020) can be derived from the control as inference framework. In fact, Fellows et al. showed that a parametric variant of MPO can be derived from VIREL (Fellows et al., 2020). The key novelty that sets VIREL apart from both MPO and MDL-C is an adaptive temperature parameter that dynamically updates the influence of the KL term in Eq. (4.9).

## Appendix 4.B: Multitask RL Frameworks

We believe the objective which best captures naturalistic settings is the average reward obtained over the agent’s “lifetime”:  $\lim_{T \rightarrow \infty} \frac{1}{T} \mathbb{E} \sum_{t=1}^T r(s_t, a_t)$ . Typical objectives include finding either a single policy or a set of policies which maximize worst- or average-case value:  $\max_\pi \min_{M \in \mathcal{M}} V_M^\pi$  (Zahavy et al., 2021) or  $\max_\pi \mathbb{E}_{\mathbb{P}_{\mathcal{M}}} V_M^\pi$  (Moskovitz et al., 2022a). When the emphasis is on decreasing the required sample complexity of learning new tasks, a useful metric is *cumulative regret*: the agent’s total shortfall across

---

<sup>2</sup>In practice, MPO parametrizes  $\pi_\theta$  implicitly with a parameterized action-value function and the default policy.

training compared to an optimal agent. In practice, it’s often simplest to consider the task distribution  $\mathbb{P}_{\mathcal{M}}$  to be a categorical distribution defined over a discrete set of tasks  $\mathcal{M} := \{M_k\}_{k=1}^K$ , though continuous densities over MDPs are also possible. Two multitask settings which we consider here are *parallel task* RL and *sequential task* RL. In typical parallel task training (Yu et al., 2019), a new MDP is sampled from  $\mathbb{P}_{\mathcal{M}}$  at the start of every episode and is associated with a particular input feature  $g \in \mathcal{G}$  that indicates to the agent which task has been sampled. The agent’s performance is evaluated on all tasks  $M \in \mathcal{M}$  together. In the *sequential* task setting (Moskovitz et al., 2022a; Pacchiano et al., 2022), tasks (MDPs) are sampled one at a time from  $\mathbb{P}_{\mathcal{M}}$ , with the agent training on each until convergence. In contrast to continual learning (Kessler et al., 2021), the agent’s goal is simply to learn a new policy for each task more quickly as more are sampled, rather than learning a single policy which maintains its performance across tasks. Another important setting is *meta-RL*, which we do not consider here. In the meta-RL setting, the agent trains on each sampled task for only a few episodes each with the goal of improving few-shot performance and is meta-tested on a set of held-out tasks (Yu et al., 2019; Finn et al., 2017).

Another strain of work in multitask RL assumes some form of shared structure in the transition dynamics (Pacchiano et al., 2022; Agarwal et al., 2022; Cheng et al., 2022). Specifically, the core assumption made by these works is that the transition dynamics are linearly decodable from a set of features which is shared across tasks or in which the transition matrix admits a low-rank decomposition. This is very different from our own structural assumption—that is, in its simplest form, that the optimal policies of the tasks with which our agents are faced take similar actions in at least some part of the state space. Beyond this, the MDPs in  $\mathcal{M}$  need only share the same state and action space, with no direct assumptions about transitions or rewards. This is important, because the assumed structures in the transition distribution made by Pacchiano et al. (2022); Agarwal et al. (2022); Cheng et al. (2022) act as the starting points for algorithm development. MDL-C/RPO/TVPO however, can leverage similarity among optimal policies when it exists, but are not dependent on it as a prerequisite. (E.g., TVPO (and RPO/MDL-C) is guaranteed to perform no worse than log-barrier regularization, which has a polynomial sample

complexity guarantee.) Ideally, we'd like a generalist method which can identify on its own and exploit different types of structure in the environment.

## Appendix 4.C: Additional Related Work

Previous work has also applied the MDL principle in an RL context, though primarily in the context of unsupervised skill learning (Zhang et al., 2021; Thrun and Schwartz, 1994). For example, Thrun and Schwartz (1994) are concerned with a set of “skills” which are policies defined only over a subset of the state space that are reused across tasks. They consider tabular methods, measuring a pseudo-description length as

$$DL = \sum_{s \in \mathcal{S}} \sum_{M \in \mathcal{M}} P_M^*(s) + \sum_{n \in \mathcal{N}} |S_n|, \quad (4.11)$$

where  $P_M^*(s)$  is the probability that no skill selects an action in state  $s$  for task  $M$  and the agent must compute the optimal  $Q$ -values in state  $s$  for  $M$ ,  $N$  is the number of skills, and  $|S_n|$  is the number states for which skill  $n$  is defined. They then trade off this description length term with performance across a series of tabular environments.

One other related method is DISTRAL (Teh et al., 2017a), which uses the following objective in the parallel task setting:

$$\mathcal{J}^{\text{Distral}}(\theta, \phi) = V^{\pi_\theta} - \mathbb{E}_{s \sim d^{\pi_\theta}} [\alpha \text{KL}[\pi_\theta(\cdot|s), \pi_\phi(\cdot|s)] + \beta \text{H}[\pi_\theta(\cdot|s)]] . \quad (4.12)$$

That is, like the un-regularized RPO method, DISTRAL can be seen as performing maximum-likelihood estimation to learn the (unconstrained) default policy, while adding an entropy bonus to the control policy.

Another important method in the sequential setting is TVPO (Moskovitz et al., 2022a), in which (in the tabular case) the default policy is defined as a softmax over the average action frequencies of the optimal policies for the tasks that the agent has seen so far. That is, if the average optimal action in a state  $s$  is given by

$$\hat{\xi}_k(s, a) = \frac{1}{k} \sum_{i=1}^k \mathbb{1}(\pi_i^*(s) = a),$$

then the TVPO default policy is

$$\pi_w(a|s) = \text{softmax} \left( \hat{\xi}_k(s, a) / \beta(k) \right),$$

where  $\beta(k)$  is a temperature which decays as  $k \rightarrow \infty$ . In high-dimensional state and action spaces, this tabular solution can be approximated by training a default policy to predict the converged control policy’s actions in each task. Importantly, this is equivalent to using KL distillation in that the default policies will converge to the same barycenter policy (Moskovitz et al., 2022a), as long as the distillation is only performed once the control policy has converged in each task. Using KL distillation in this way is exactly the RPO baseline that we use in this paper. Crucially, the use of the softmax with decaying temperature was introduced by Moskovitz et al. (2022a) as a useful ‘hack’ to prevent the default policy from overfitting to early tasks, as the optimal default policy is the barycenter policy (approximated as the number of draws from the task distribution grows). Thus, MDL-C can itself be seen as a scalable advancement of TVPO which models the agent’s epistemic uncertainty about the task distribution by placing a sparse prior over the default policy parameters (and uses a distillation loss rather than action prediction). In other words, MDL-C represents a principled approach to reducing the risk of default policy overfitting in the low-data regime.

Finally, Brunskill and Li (2013) consider a similar training and task structure to our own, but use a model-based approach to learn the underlying MDPs.

## Appendix 4.D: Motivating the choice of sparsity-inducing priors

As a reminder, the generative model of optimal parameters in Section 4.4.1 is given by:

$$\bar{w}_i | \beta, \sigma^2 \sim \mathcal{N}(0, \frac{1 - \beta}{\beta} \sigma^2 I_d), \quad (4.13)$$

$$w_{ik} | \bar{w}_i, \sigma^2, \beta \sim \mathcal{N}(\bar{w}_i, \sigma^2 I_d) \quad (4.14)$$

with marginal and posterior densities

$$p(w_{ik}|\sigma^2, \beta) = \mathcal{N}(0, \sigma^2 \beta^{-1} I_d), \quad (4.15)$$

$$p(\bar{w}_i|w_{ik}, \sigma^2, \beta) = \mathcal{N}((1 - \beta)w_{ik}, (1 - \beta)\sigma^2 I_d). \quad (4.16)$$

In the rest of this section, we set  $\sigma^2 = 1$  for simplicity and drop the indices on  $w$  and  $\bar{w}$  to remove clutter.

#### 4.D.1 Correspondence between $p(z)$ and $p(\beta)$

In Section 4.4.1, we draw a connection between  $p(\beta) \propto \beta^{-1}$  and the normal-Jeffreys prior, which is commonly used for compressing deep neural networks (Louizos et al., 2017). In Table 4.1, we expand on this connection and list  $p(\beta)$  for two other commonly-used priors for scale mixture of normal distributions: Jeffreys, Inverse-gamma, and Inverse-beta. Note that the half-Cauchy distribution  $p(z) \propto (1 + z^2)^{-1}$  is a special case of the inverse-beta distribution for  $s = t = 1/2$ . Half-cauchy prior is another commonly used prior for compressing Bayesian neural networks (Louizos et al., 2017).

#### 4.D.2 MSE risk

In this section, we prove that the Bayes estimators for the Jeffreys, inverse-gamma, and the inverse-beta (by extension the half-Cauchy) distributions dominate the maximum-likelihood estimator with respect to the mean-squared error.

Define the mean-squared error of an estimator  $\hat{w}(x)$  of  $\bar{w}$  as

$$\text{MSE}(\bar{w}, \hat{w}) = \mathbb{E}_x \|\hat{w}(x) - \bar{w}\|^2, \quad (4.17)$$

where the expectation is taken over  $\mathcal{N}(x; \bar{w}, \alpha^2)$ . Immediately, we have  $R(\bar{w}, \hat{w}^{(\text{ML})}) =$

Prior name	$p(z^2)$	$p(\beta)$
Jeffreys	$p(z^2) \propto z^{-2}$	$p(\beta) \propto \beta^{-1}$
Inverse-gamma	$p(z^2) \propto z^{-2(s+1)} e^{-t/(2z^2)}$	$p(\beta) \propto \beta^{s-1} e^{-t\beta/2}$
Inverse-beta	$p(z^2) \propto (z^2)^{t-1} (1 + z^2)^{-(s+t)}$	$p(\beta) \propto \beta^{-(s+2t+1)} (1 + \beta)^{-(s+t)}$

**Table 4.1:** Correspondence between  $p(z^2)$  and  $p(\beta)$ .



$d$ , where  $\hat{w}^{(\text{ML})}(x) = x$  is the maximum-likelihood estimator. An estimator  $\hat{w}^{(a)}(x)$  is said to dominate another estimator  $\hat{w}^{(b)}(x)$  if  $\text{MSE}(\bar{w}, \hat{w}_a) \leq \text{MSE}(\bar{w}, \hat{w}_b)$  for all  $\bar{w}$  and the inequality is strict for a set of positive Lebesgue measure. It is well-known that the maximum-likelihood estimator is minimax (George et al., 2006), and thus any estimator that dominates the maximum-likelihood estimator is also minimax.

To compute the mean-squared error risk for an estimator  $\hat{w}(x)$ , observe that

$$\|\hat{w}(x) - \bar{w}\|^2 = \|x - \hat{w}(x)\|^2 - \|x - \bar{w}\|^2 + 2(\hat{w}(x) - \bar{w})^\top (x - \bar{w}). \quad (4.18)$$

Taking expectations on both sides gives

$$\text{MSE}(\bar{w}, \hat{w}) = \mathbb{E}_x \|x - \hat{w}(x)\|^2 - d + 2 \sum_{i=1}^d \text{Cov}(\hat{w}_i(x), x_i) \quad (4.19)$$

$$= \mathbb{E}_x \|x - \hat{w}(x)\|^2 - d + 2\mathbb{E}_x \nabla \cdot \hat{w}(x) \quad (4.20)$$

where  $\nabla = (\partial/\partial x_1, \dots, \partial/\partial x_d)$  and we apply Stein's lemma  $\text{cov}(\hat{w}_i(x), x_i) = \mathbb{E}_x \partial \hat{w}_i / \partial x_i$  in the last line. If the estimator takes the form  $\hat{w}(x) = x + \gamma(x)$ , the expression simplifies as:

$$\text{MSE}(\bar{w}, \hat{w}) = d + \mathbb{E}_x \|\gamma(x)\|^2 + 2\mathbb{E}_x \nabla \cdot \gamma(x). \quad (4.21)$$

Therefore, an estimator  $\hat{w}(x) = x + \gamma(x)$  dominates  $\hat{w}^{(\text{ML})}(x)$  if

$$\text{MSE}(\bar{w}, \hat{w}) - \text{MSE}(\bar{w}, \hat{w}^{(\text{ML})}) = \mathbb{E}_x [\|\gamma(x)\|^2 + 2\nabla \cdot \gamma(x)] \leq 0 \quad (4.22)$$

for all  $\bar{w}$  and the inequality is strict on a set of positive Lebesgue measure.

#### 4.D.2.1 James-Stein estimator

The famous James-Stein estimator is defined as

$$\hat{w}^{(\text{JS})}(x) = x + \gamma^{(\text{JS})}(x), \quad \gamma^{(\text{JS})}(x) = -(d-2)x/\|x\|^2, \quad (4.23)$$

with

$$\nabla \cdot \gamma^{(\text{JS})}(x) = \sum_{i=1}^d \left[ -\frac{d-2}{\|x\|^2} + 2\frac{d-2}{(\|x\|^2)^2} x_i^2 \right] = -\frac{(d-2)^2}{\|x\|^2}, \quad (4.24)$$

$$\|\gamma^{(\text{JS})}(x)\|^2 = \frac{(d-2)^2}{\|x\|^2}. \quad (4.25)$$

Substituting  $\nabla \cdot \gamma^{(\text{JS})}(x)$  and  $\|\gamma^{(\text{JS})}(x)\|^2$  into Eq. (4.22), we have

$$\text{MSE}(\bar{w}, \hat{w}^{(\text{JS})}) - \text{MSE}(\bar{w}, \hat{w}^{(\text{ML})}) = \mathbb{E}_x \frac{(d-2)^2}{\|x\|^2}. \quad (4.26)$$

Thus, the James-Stein estimator dominates the maximum-likelihood estimator for  $d > 2$ .

#### 4.D.2.2 Bayes estimators

The Bayes estimator for a prior choice  $p(\beta)$  is given by (Brown, 1971):

$$\hat{w}^{(p(\beta))}(x) = x + \gamma^{(p(\beta))}(x), \quad \gamma^{(p(\beta))}(x) = \nabla \log m(x), \quad (4.27)$$

where

$$m(x) = \int \mathcal{N}(x; 0, \beta^{-1} I_d) p(\beta) d\beta \quad (4.28)$$

$$= \int (2\pi)^{-\frac{1}{2}} \beta^{d/2} \exp(-\beta x^2/2) p(\beta) d\beta. \quad (4.29)$$

Substituting  $\gamma^{(p(\beta))}(x)$  into Eq. (4.22), we find that the condition for the Bayes estimator to be minimax is given by (George et al., 2006):

$$\text{MSE}(\bar{w}, \hat{w}^{(\text{B})}) - \text{MSE}(\bar{w}, \hat{w}^{(\text{ML})}) = \mathbb{E}_x \left[ -\|\nabla \log m(x)\|^2 + 2\frac{\nabla^2 m(x)}{m(x)} \right] \quad (4.30)$$

$$= \mathbb{E}_x \left[ 4\frac{\nabla^2 \sqrt{m(x)}}{\sqrt{m(x)}} \right] \leq 0, \quad (4.31)$$

where  $\nabla^2 = \sum_i \partial^2 / \partial x_i^2$  is the Laplace operator. This condition holds when  $\sqrt{m(x)}$  is superharmonic (i.e.,  $\sqrt{m(x)} \leq 0, \forall x \in \mathbb{R}^d$ ), suggesting a recipe for constructing Bayes estimators that dominate the maximum likelihood estimator, summarized in the

following proposition.

**Proposition 4.D.1** (Extension of Theorem 1 in Fourdrinier et al., 1998). *Let  $p(\beta)$  be a positive function such that  $f(\beta) = \beta p'(\beta)/p(\beta)$  can be decomposed as  $f_1(\beta) + f_2(\beta)$  where  $f_1$  is non-decreasing,  $f_1 \leq A$ ,  $0 < f_2 \leq B$ , and  $A/2 + B \leq (d-6)/4$ . Assume also that  $\lim_{\beta \rightarrow 0} \beta^{d/2+2} p(\beta) = 0$ . Then,  $\nabla^2 \sqrt{m(x)} \leq 0$  and the Bayes estimator is minimax. If  $A/2 + B < (d-6)/4$ , then the Bayes estimator dominates  $\hat{w}^{(ML)}(x)$ .*

*Proof.* This proof largely follows the proof of Theorem 1 in (Fourdrinier et al., 1998).

Note that Eq. (4.30) holds if

$$\nabla^2 \sqrt{m(x)} = \frac{1}{2\sqrt{m(x)}} \left( \nabla^2 m(x) - \frac{1}{2} \frac{\|\nabla m(x)\|^2}{m(x)} \right) \leq 0 \quad \forall x \in \mathbb{R}^d, \quad (4.32)$$

or equivalently

$$\frac{\nabla^2 m(x)}{\|\nabla m(x)\|} - \frac{1}{2} \frac{\|\nabla m(x)\|}{m(x)} \leq 0 \quad \forall x \in \mathbb{R}^d. \quad (4.33)$$

Computing the derivatives, we get the condition

$$\frac{\int_0^1 (\beta \|x\|^2 - d) \beta^{d/2+1} e^{-\beta \|x\|^2/2} p(\beta) d\beta}{\|x\| \int_0^1 \beta^{d/2+1} e^{-\beta \|x\|^2/2} p(\beta) d\beta} - \frac{1}{2} \frac{\|x\| \int_0^1 \beta^{d/2+1} e^{-\beta \|x\|^2/2} p(\beta) d\beta}{\int_0^1 \beta^{d/2} e^{-\beta \|x\|^2/2} p(\beta) d\beta} \leq 0. \quad (4.34)$$

Divide both sides by  $\|x\|$  and rearrange to get

$$\frac{\int_0^1 \beta^{d/2+2} e^{-\beta \|x\|^2/2} p(\beta) d\beta}{\int_0^1 \beta^{d/2+1} e^{-\beta \|x\|^2/2} p(\beta) d\beta} - \frac{1}{2} \frac{\int_0^1 \beta^{d/2+1} e^{-\beta \|x\|^2/2} p(\beta) d\beta}{\int_0^1 \beta^{d/2} e^{-\beta \|x\|^2/2} p(\beta) d\beta} \leq \frac{d}{\|x\|^2}. \quad (4.35)$$

Next, we integrate by parts the numerator of the first term on the left-hand side to get:

$$\begin{aligned} \int_0^1 \beta^{d/2+2} e^{-\beta \|x\|^2/2} p(\beta) d\beta &= -\frac{2}{\|x\|^2} \left[ \beta^{d/2+2} e^{-\beta \|x\|^2/2} p(\beta) \right]_0^1 \\ &\quad + \frac{d+4}{\|x\|^2} \int_0^1 \beta^{d/2+1} e^{-\beta \|x\|^2/2} p(\beta) d\beta \\ &\quad + \frac{2}{\|x\|^2} \int_0^1 \beta^{d/2+2} e^{-\beta \|x\|^2/2} p'(\beta) d\beta, \end{aligned} \quad (4.36)$$

where the middle term is the same as the denominator of the first term in Eq. (4.35). Integrating by parts the second term gives the same expression as that of the first term, but with  $d - 2$  in place of  $d$  everywhere. Substituting these expressions back into Eq. (4.35), collecting like terms, and dividing both sides by  $2/\|x\|^2$ , gives:

$$\begin{aligned} & \frac{\int_0^1 \beta^{d/2+2} e^{-\beta\|x\|^2/2} p'(\beta) d\beta}{\int_0^1 \beta^{d/2+1} e^{-\beta\|x\|^2/2} p(\beta) d\beta} - \frac{1}{2} \frac{\int_0^1 \beta^{d/2+1} e^{-\beta\|x\|^2/2} p'(\beta) d\beta}{\int_0^1 \beta^{d/2} e^{-\beta\|x\|^2/2} p(\beta) d\beta} + \kappa_0 + \kappa_1 \\ & \leq \frac{d}{2} - \frac{d+4}{2} + \frac{1}{2} \frac{d+2}{2} = \frac{d-6}{4}, \end{aligned} \quad (4.37)$$

where

$$\kappa_1 = -\frac{\lim_{\beta \rightarrow 1} \beta^{d/2+2} e^{-\beta\|x\|^2/2} p(\beta)}{\int_0^1 \beta^{d/2+1} e^{-\beta\|x\|^2/2} p(\beta) d\beta} + \frac{1}{2} \frac{\lim_{\beta \rightarrow 1} \beta^{d/2+1} e^{-\beta\|x\|^2/2} p(\beta)}{\int_0^1 \beta^{d/2} e^{-\beta\|x\|^2/2} p(\beta) d\beta}, \quad (4.38)$$

$$\kappa_0 = \frac{\lim_{\beta \rightarrow 0} \beta^{d/2+2} e^{-\beta\|x\|^2/2} p(\beta)}{\int_0^1 \beta^{d/2+1} e^{-\beta\|x\|^2/2} p(\beta) d\beta} - \frac{1}{2} \frac{\lim_{\beta \rightarrow 0} \beta^{d/2+1} e^{-\beta\|x\|^2/2} p(\beta)}{\int_0^1 \beta^{d/2} e^{-\beta\|x\|^2/2} p(\beta) d\beta}. \quad (4.39)$$

Here, both  $\kappa_0$  and  $\kappa_1$  are nonpositive: (i)  $\kappa_0$  is nonpositive because the first term vanishes due to the boundary conditions and the second term is nonpositive, and (ii)  $\kappa_1$  is nonpositive because the limits of the numerators of the two terms are equal while the denominator of the second term is larger than that of the first. We can thus drop  $\kappa_0$  and  $\kappa_1$  to get the sufficient condition:

$$\mathbb{E}_d(f) - \frac{1}{2} \mathbb{E}_{d-2}(f) \leq \frac{d-6}{4}, \quad (4.40)$$

where  $\mathbb{E}_d$  denotes expectation with respect to the density

$$g_d(\beta) = \frac{\beta^{d/2+1} e^{-\beta\|x\|^2/2} p(\beta)}{\int_0^1 \beta^{d/2+1} e^{-\beta\|x\|^2/2} p(\beta) d\beta} \quad (4.41)$$

and where  $f(\beta) = \beta p'(\beta)/p(\beta)$ .

Because  $g_d(\beta)$  is a family of monotone increasing likelihood ratio in  $d$  and  $f_1$  is nonincreasing and bounded by  $A$ , we have  $\mathbb{E}_d(f_1) - \mathbb{E}_{d-2}(f_1)/2 \leq A/2$ . We have

$\mathbb{E}_d(f_2) - \mathbb{E}_{d-2}(f_2)/2 \leq B$  because  $0 < f_2 \leq B$ . Taken together, we have

$$\mathbb{E}_d(f) - \mathbb{E}_{d-2}(f)/2 \leq A/2 + B \leq (k-6)/4. \quad (4.42)$$

When the inequality is strict (i.e.,  $A/2 + B < (k-6)/4$ ), then  $\nabla^2 \sqrt{m(x)} < 0$  and the Bayes estimator dominates the maximum-likelihood estimator.  $\square$

Checking whether a given  $p(\beta)$  satisfy the conditions in Proposition 4.D.1 may be tedious. The following corollary is useful for construction  $p(\beta)$  that satisfies the conditions in Proposition 4.D.1.

**Corollary 4.D.1** (Extension of Corollary 1 in Fourdrinier et al., 1998). *Let  $\psi$  be a continuous function that can be decomposed as  $\psi_1 + \psi_2$ , with  $\psi_1 \leq C$ ,  $\psi_1$  non-decreasing,  $0 < \psi_2 \leq D$ , and  $C/2 + D \leq 0$ . Let*

$$p(\beta) = \exp \left( \frac{1}{2} \int_{\beta_0}^{\beta} \frac{2\psi(u) + d - 6}{u} du \right) \quad \forall \beta_0 \geq 0, \quad (4.43)$$

*such that  $\lim_{\beta \rightarrow 0} \beta^{d/2+2} p(\beta) = 0$  and  $\beta_0 \in (0, 1)$  is a constant. Then,  $p(\beta)$  results in a minimax Bayes estimator, which dominates the maximum likelihood estimator when  $C/2 + D < 0$ .*

*Proof.* The proof is the same as that of Corollary 1 in Fourdrinier et al. (1998), with Proposition 4.D.1 in place of Theorem 1 in Fourdrinier et al. (1998).  $\square$

Using Corollary 4.D.1, we now check that the three priors listed in Table 4.1 and referenced in Section 4.4.1 lead to Bayes estimators that dominate the maximum-likelihood estimator.

**Jeffreys prior** Let  $\psi_1(u) = a$  for  $a \leq 0$  and  $\psi_2(u) = 0$ . We have

$$p(\beta) = \exp \left( \frac{1}{2} \int_{\beta_0}^{\beta} \frac{2a + d - 6}{u} du \right) \propto \beta^{a+(d-6)/2}. \quad (4.44)$$

To satisfy  $\lim_{\beta \rightarrow 0} \beta^{d/2+2} p(\beta) = 0$ , we require  $1 - d < a \leq 0$ . We recover the improper normal-Jeffreys prior  $p(\beta) \propto \beta^{-1}$ , for  $a = 2 - d/2$ . The corresponding Bayes estimator dominates the maximum likelihood estimator when  $d > 4$ .

**Inverse-gamma prior** Let  $\psi_1(u) = a$  and  $\psi_2(u) = b(1 - u)/2$  for  $a \leq 0$  and  $b \geq 0$ . We have

$$p(\beta) = \exp \left( \int_{\beta_0}^{\beta} \frac{a + b(1 - u)/2 + (d - 6)/2}{u} du \right) \propto \beta^{a+(b+d-6)/2} e^{-b\beta/2}. \quad (4.45)$$

Setting  $C = a$  and  $D = b/2$ , we get the followings conditions:  $a + b \leq 0$  and  $1 - d \leq a + b/2$ . Note that when these conditions are met with  $s = a + (b + d - 4)/2$  and  $t = b$ , we recover the inverse-gamma prior in Table 4.1.

**Inverse-beta (half-Cauchy) prior** Let  $\psi_1(u) = a$  and  $\psi_2(u) = b/(u + 1)$  for  $a \leq 0$  and  $b \geq 0$ . We have

$$p(\beta) = \exp \left( \int_{\beta_0}^{\beta} \frac{a + b/(1 + u) + (d - 6)/2}{u} du \right) \propto \beta^{a+b+(d-6)/2} (1 + \beta)^{-b}. \quad (4.46)$$

Setting  $C = a$  and  $D = b$ , we get the condition  $a/2 + b \leq 0$ . To satisfy  $\lim_{\beta \rightarrow 0} \beta^{d/2+2} p(\beta) = 0$ , we require  $1 - d < a + b \leq 0$ . Note that this corresponds to the inverse-beta prior in Table 4.1 with  $t = a + (d - 8)/2$  and  $s = b - t$ .

To recover the half-Cauchy prior, we set  $b = 1$  and  $a = (5 - d)/2$ . All conditions in Corollary 4.D.1 are satisfied when  $d > 9$ .

## Appendix 4.E: Limitations

One weakness of the current theoretical analysis regarding the choice of sparsity-inducing priors is the assumption of Gaussian (and in particular, isotropic Gaussian) structure in the parameter space of optimal policies for clusters of tasks. In reality, there is likely a nontrivial degree of covariance among task parameterizations. Extending our analysis to more realistic forms of task structure is an important direction for future work. In a similar vein, the assumption that tasks are drawn iid from a fixed distribution is also unrealistic in naturalistic settings. It would be interesting to introduce some form of sequential structure (e.g., tasks are drawn from a Markov process). Another direction for future work is expanding beyond the “one control policy, one default policy” setup—having, for example, one default policy per task cluster and the ability to reuse and select (for exam-

ple, using successor feature-like representations (Barreto et al., 2020; Barth-Maron et al., 2018; Moskovitz et al., 2022c)) among an actively-maintained set of control policies across tasks and task clusters would be useful.

## Appendix 4.F: OCO Background

In *online convex optimization* (OCO), the learner observes a series of convex loss functions  $\ell_k : \mathbf{N} \rightarrow \mathbb{R}, k = 1, \dots, K$ , where  $\mathbf{N} \subseteq \mathbb{R}^d$  is a convex set. After each round, the learner produces an output  $x_k \in \mathbf{N}$  for which it will then incur a loss  $\ell_k(x_k)$  (Orabona, 2019). At round  $k$ , the learner is usually assumed to have knowledge of  $\ell_1, \dots, \ell_{k-1}$ , but no other assumptions are made about the sequence of loss functions. The learner's goal is to minimize its average regret:

$$\bar{\mathcal{R}}_K := \frac{1}{K} \sum_{k=1}^K \ell_k(x_k) - \min_{x \in \mathbf{N}} \frac{1}{K} \sum_{k=1}^K \ell_k(x). \quad (4.47)$$

One OCO algorithm which enjoys sublinear regret is *follow the regularized leader* (FTRL). In each round of FTRL, the learner selects the solution  $x \in \mathbf{N}$  according to the following objective:

$$x_{k+1} = \operatorname{argmin}_{x \in \mathbf{N}} \psi_k(x) + \sum_{i=1}^{k-1} \ell_i(x), \quad (4.48)$$

where  $\psi_k : \mathbf{N} \rightarrow \mathbb{R}$  is a convex regularization function.

## Appendix 4.G: Proofs of Performance Bounds and Additional Theoretical Results

The following result is useful.

**Lemma 4.G.I.** *The function  $\ell(\nu) = \mathbb{E}_{w \sim \nu} f(w)$  is  $L$ -Lipschitz as long as  $f : \mathcal{W} \rightarrow \mathbb{R}$  lies within  $[0, L] \forall w \in \mathcal{W}$ , where  $\mathcal{W} \subseteq \mathbb{R}^d$  is a Hilbert space and  $L < \infty$ .*

*Proof.* We have

$$\begin{aligned}
|\ell(\nu_1) - \ell(\nu_2)| &= |\mathbb{E}_{w \sim \nu_1} f(w) - \mathbb{E}_{w \sim \nu_2} f(w)| \\
&= \left| \int_{\mathcal{W}} (\nu_1(w) - \nu_2(w)) f(w) dw \right| \\
&= |\langle f, \nu_1 - \nu_2 \rangle_{\mathcal{W}}| \\
&\leq \|f\|_{\mathcal{W}} \|\nu_1 - \nu_2\|_{\mathcal{W}} \\
&\leq L \|\nu_1 - \nu_2\|_{\mathcal{W}},
\end{aligned}$$

where the first inequality is due to Cauchy-Schwarz and the second is by assumption on  $f$ .  $\square$

**Proposition 4.G.2** (Default Policy Distribution Regret). *Let tasks  $M_k$  be independently drawn from  $\mathbb{P}_{\mathcal{M}}$  at every round, and let them each be associated with a deterministic optimal policy  $\pi_k^* : \mathcal{S} \rightarrow \mathcal{A}$ . We make the following mild assumptions: i)  $\pi_w(a^*|s) \geq \epsilon > 0 \forall s \in \mathcal{S}$ , where  $a^* = \pi_k^*(s)$  and  $\epsilon$  is a constant. ii)  $\min_{\nu} \text{KL}[\nu(\cdot), p(\cdot)] \rightarrow 0$  as  $\text{Var}[\nu] \rightarrow \infty$  for an appropriate choice of sparsity-inducing prior  $p$ . Then Algorithm 4.4 guarantees*

$$\mathbb{E}_{\mathbb{P}_{\mathcal{M}}}[\ell_K(\nu_K) - \ell_K(\bar{\nu}_K)] \leq (\mathbb{E}_{\mathbb{P}_{\mathcal{M}}} \text{KL}[\bar{\nu}_K, p] + 1) \frac{\log(1/\epsilon)}{\sqrt{K}}. \quad (4.49)$$

where  $\bar{\nu}_K = \arg\min_{\nu \in \mathcal{N}} \sum_{k=1}^K \ell_k(\nu)$ .

*Proof.* The first part of the proof sets up an application of Orabona (2019), Corollary 7.9.

To establish grounds for its application, we first note the standard result that the regularization functional  $\psi(\nu) = \text{KL}[\nu(w), p(w)]$  for probability measures  $\nu, p \in \mathcal{P}(\mathcal{W})$  is 1-strongly convex in  $\nu$  (Melbourne, 2020).

Finally, assumption (i) implies that the KL between the default policy and the optimal policy is upper-bounded:  $\text{KL}[\pi_k^*, \pi_w] \leq \log 1/\epsilon$ . Then by Lemma 4.G.1,  $\ell_k(\nu)$  is  $L$ -Lipschitz wrt the TV distance, where  $L = \log 1/\epsilon$ .

Note also that under a Gaussian parameterization for  $\nu$ , the distribution space  $\mathcal{N}$  is the Gaussian parameter space  $\mathcal{N} = \{(\mu, \Sigma) : \mu \in \mathbb{R}^d, \Sigma \in \mathbb{R}^{d \times d}, \Sigma \succeq 0\}$ , which is



convex (Boyd and Vandenberghe, 2004).

Then Orabona (2019), Corollary 7.9 gives

$$\frac{1}{K} \sum_{k=1}^K \ell_k(\nu_k) - \frac{1}{K} \sum_{k=1}^K \ell_k(\bar{\nu}_K) \leq \left( \frac{1}{\alpha} \text{KL}[\bar{\nu}_K, p] + \alpha \right) \frac{L}{\sqrt{K}}, \quad (4.50)$$

where  $\bar{\nu}_K = \text{argmin}_{\nu} \sum_{k=1}^K \ell_k(\nu)$ . The constant  $\alpha \in \mathbb{R}^+$  is a hyperparameter, so we are free to set it to 1 (Orabona, 2019). Finally, we observe that  $\mathbb{E}_{\mathbb{P}_{\mathcal{M}_i}} \frac{1}{K} \sum_{k=1}^K \ell(\nu_k) = \mathbb{E}_{\mathbb{P}_{\mathcal{M}_i}} \ell_K(\nu_K)$  and take the expectation with respect to  $\mathbb{P}_{\mathcal{M}_i}$  of both sides of Eq. (4.50) to get the desired result:

$$\mathbb{E}_{\mathbb{P}_{\mathcal{M}_i}} [\ell_K(\nu_K) - \ell_K(\bar{\nu}_K)] \leq \left( \mathbb{E}_{\mathbb{P}_{\mathcal{M}_i}} \text{KL}[\bar{\nu}_K, p] + 1 \right) \frac{L}{\sqrt{K}}. \quad (4.51)$$

□

**Proposition 4.4.2** (Control Policy Sample Complexity). *Under the setting described in Proposition 4.G.2 denote by  $T_k$  the number of iterations to reach  $\epsilon$ -error for  $M_k$  in the sense that  $\min_{t \leq T_k} \{V^{\pi_k^*} - V^{(t)}\} \leq \epsilon$ . whenever  $t > T_k$ . Further, denote the upper-bound in Eq. (4.49) by  $G(K)$ . In a finite MDP, from any initial  $\theta^{(0)}$ , and following gradient ascent,  $\mathbb{E}_{M_k \sim \mathbb{P}_{\mathcal{M}}} [T_k]$  satisfies:*

$$\mathbb{E}_{M_k \sim \mathbb{P}_{\mathcal{M}_i}} [T_k] \geq \frac{80|\mathcal{A}|^2|\mathcal{S}|^2}{\epsilon^2(1-\gamma)^6} \mathbb{E}_{\substack{M_k \sim \mathbb{P}_{\mathcal{M}_i} \\ s \sim \text{Unif}_{\mathcal{S}}}} \left[ \kappa_{\mathcal{A}}^{\alpha_k}(s) \left\| \frac{d_{\rho}^{\pi_k^*}}{\mu} \right\|_{\infty}^2 \right],$$

where  $\alpha_k(s) := d_{\text{TV}}(\pi_k^*(\cdot|s), \hat{\pi}_0(\cdot|s)) \leq \sqrt{G(K)}$ ,  $\kappa_{\mathcal{A}}^{\alpha_k}(s) = \frac{2|\mathcal{A}|(1-\alpha(s))}{2|\mathcal{A}|(1-\alpha(s))-1}$ , and  $\mu$  is a measure over  $\mathcal{S}$  such that  $\mu(s) > 0 \forall s \in \mathcal{S}$ .

Note: In the above, there is a small error—it should be

$$\alpha_k(s) := \mathbb{E}_{w \sim \nu} d_{\text{TV}}(\pi_k^*(\cdot|s), \pi_w(\cdot|s)) \leq \sqrt{\frac{1}{2}G(K)}.$$

$d_{\rho}^{\pi}$  refers to the discounted state-occupancy distribution under  $\pi$  with initial state distri-

bution  $\rho$ :

$$d_\rho^\pi(s) = \mathbb{E}_{s_0 \sim \rho} (1 - \gamma) \sum_{h \geq 0} \gamma^h \mathbb{P}^\pi(s_h = s | s_0). \quad (4.52)$$

Division between probability mass functions is assumed to be element-wise.

*Proof.* Without loss of generality, we prove the bound for a fixed state  $s \in \mathcal{S}$ , noting that the bound applies independently of our choice of  $s$ . We use the shorthand  $\text{KL}[\pi(\cdot|s), \pi_w(\cdot|s)] \rightarrow \text{KL}[\pi, \pi_w]$  for brevity. We start by multiplying both sides of the bound from Proposition 4.G.2 by 1/2 and rearranging:

$$\begin{aligned} & \frac{1}{2} \left( \mathbb{E}_{\mathbb{P}_{\mathcal{M}_i}} \ell_K(\bar{\nu}_K) + \frac{L}{\sqrt{K}} \left( \mathbb{E}_{\mathbb{P}_{\mathcal{M}_i}} \text{KL}[\bar{\nu}_K, p] + 1 \right) \right) \\ & \geq \mathbb{E}_{\mathbb{P}_{\mathcal{M}_i}} \frac{1}{2} \ell_K(\nu_K) \\ & = \mathbb{E}_{\mathbb{P}_{\mathcal{M}_i}} \mathbb{E}_{\nu_K} \frac{1}{2} \text{KL}[\pi_K^*, \pi_w] \\ & \stackrel{(i)}{=} \mathbb{E}_{\mathbb{P}_{\mathcal{M}_i}} \left[ \text{Var}_{\nu_K} \left[ \sqrt{\frac{1}{2} \text{KL}[\pi_K^*, \pi_w]} \right] + \mathbb{E}_{\nu_K} \left[ \sqrt{\frac{1}{2} \text{KL}[\pi_K^*, \pi_w]} \right]^2 \right] \\ & \stackrel{(ii)}{\geq} \mathbb{E}_{\mathbb{P}_{\mathcal{M}_i}} \left[ \mathbb{E}_{\nu_K} \left[ \sqrt{\frac{1}{2} \text{KL}[\pi_K^*, \pi_w]} \right]^2 \right] \end{aligned} \quad (4.53)$$

where (i) follows from the definition of the variance, and (ii) follows from its non-negativity. We can rearrange to get

$$\begin{aligned} \frac{L}{2\sqrt{K}} \left( \mathbb{E}_{\mathbb{P}_{\mathcal{M}_i}} \text{KL}[\bar{\nu}_K, p] + 1 \right) & \geq \mathbb{E}_{\mathbb{P}_{\mathcal{M}_i}} \mathbb{E}_{\nu_K} \left[ \sqrt{\frac{1}{2} \text{KL}[\pi_K^*, \pi_w]} \right]^2 \\ & \stackrel{(ii)}{\geq} \mathbb{E}_{\mathbb{P}_{\mathcal{M}_i}} \mathbb{E}_{\nu_K} [d_{\text{TV}}(\pi_K^*, \pi_w)]^2 \end{aligned} \quad (4.54)$$

where (ii) follows from Pinsker's inequality. Letting  $\alpha_K(s) = \sqrt{\frac{1}{2} G(K)}$  and applying Moskovitz et al. (2022a), Lemma 5.2 gives the desired result.  $\square$

This upper-bound is significant, as it shows that, all else being equal, a high complexity barycenter default policy distribution  $\bar{\nu}_K$  (where complexity is measured by  $\text{KL}[\bar{\nu}_K, p]$ ) leads to a slower convergence rate in the control policy.

**Algorithm 5** Idealized MDL-C for Multitask Learning

- 
- 1: require: task distribution  $\mathbb{P}_{\mathcal{M}}$ , policy class  $\Pi$ , coefficients  $\{\eta_k\}$
  - 2: initialize: default policy distribution  $\nu_1 \in \mathcal{N}$
  - 3: **for** tasks  $k = 1, 2, \dots, K$  **do**
  - 4:   Sample a task  $M_k \sim \mathcal{P}_{\mathcal{M}}(\cdot)$
  - 5:   Optimize control policy:
- 

$$\hat{\pi}_k^* = \operatorname{argmax}_{\pi \in \Pi} V_{M_k}^\pi - \lambda \mathbb{E}_{s \sim d^\pi} \mathbb{E}_{w \sim \nu_k} \text{KL}[\pi_w(a|s), \pi(a|s)] \quad (4.55)$$

- 6:   Update default policy distribution:

$$\nu_{k+1} = \operatorname{argmin}_{\nu \in \mathcal{N}} \text{KL}[\nu, p] + \mathbb{E}_{w \sim \nu} \text{KL}[\hat{\pi}_k^*, \pi_w] \quad (4.56)$$

- 7: **end for**
- 

**4.G.3 MDL-C with Persistent Replay**

Rather than rely on iid task draws to yield a bound on the expected regret under the task distribution, a more general formulation of MDL-C for sequential task learning is described in Algorithm 4. In this setting, the dataset of optimal agent-environment interactions is explicitly constructed by way of a replay buffer which persists across tasks and is used to train the default policy distribution. This is much more directly in line with standard FTRL, and we can obtain the standard FTRL bound.

**Proposition 4.G.3** (Persistent Replay FTRL Regret; (Orabona, 2019), Corollary 7.9).

Let tasks  $M_k$  be independently drawn from  $\mathbb{P}_{\mathcal{M}}$  at every round, and let them each be associated with a deterministic optimal policy  $\pi_k^* : \mathcal{S} \rightarrow \mathcal{A}$ . We make the following mild assumptions: i)  $\pi_w(a^*|s) \geq \epsilon > 0 \forall s \in \mathcal{S}$ , where  $a^* = \pi_k^*(s)$  and  $\epsilon$  is a constant. ii)  $\min_{\nu} \text{KL}[\nu(\cdot), p(\cdot)] = 0$  asymptotically as  $\text{Var}[\nu] \rightarrow \infty$ . Then with  $\eta_{k-1} = L\sqrt{k}$ , Algorithm 4 guarantees

$$\frac{1}{K} \sum_{k=1}^K \ell_k(\nu_k) - \frac{1}{K} \sum_{k=1}^K \ell_k(\bar{\nu}_K) \leq (\text{KL}[\bar{\nu}_K, p] + 1) \frac{L}{\sqrt{K}}, \quad (4.57)$$

where  $\bar{\nu}_K = \operatorname{argmin}_{\nu \in \mathcal{N}} \sum_{k=1}^K \ell_k(\nu)$ .

*Proof.* This follows directly from the arguments made in the proof of Proposition 4.G.2.

□

As before, this result can be used to obtain a performance bound for the control policy.

**Proposition 4.G.4** (Control Policy Sample Complexity for MDL-C with Persistent Replay). *Under the setting described in Proposition 4.4.1 denote by  $T_k$  the number of iterations to reach  $\epsilon$ -error for  $M_k$  in the sense that  $\min_{t \leq T_k} \{V^{\pi_k^*} - V^{(t)}\} \leq \epsilon$ . In a finite MDP, from any initial  $\theta^{(0)}$ , and following gradient ascent,  $\mathbb{E}_{M_k \sim \mathcal{P}_{\mathcal{M}}} [T_k]$  satisfies:*

$$\mathbb{E}_{M_k \sim \mathcal{P}_{\mathcal{M}_i}} [T_k] \geq \frac{80|\mathcal{A}|^2|\mathcal{S}|^2}{\epsilon^2(1-\gamma)^6} \mathbb{E}_{M_k \sim \mathcal{P}_{\mathcal{M}_i} s \sim \text{Unif}_{\mathcal{S}}} \left[ \kappa_{\mathcal{A}}^{\alpha_k}(s) \left\| \frac{d_{\rho}^{\pi_k^*}}{\mu} \right\|_{\infty}^2 \right],$$

where  $\alpha_k(s) := \mathbb{E}_{w \sim \nu} d_{\text{TV}}(\pi_k^*(\cdot|s), \pi_w(\cdot|s)) \leq \sqrt{\frac{1}{2}G(K)}$ ,

$$G(K) := \ell_K(\bar{\nu}_K) + \sum_{k=1}^{K-1} (\ell_k(\bar{\nu}_K) - \ell_k(\nu_k)) + (\text{KL}[\bar{\nu}, p] + 1) L\sqrt{K},$$

$\kappa_{\mathcal{A}}^{\alpha_k}(s) = \frac{2|\mathcal{A}|(1-\alpha(s))}{2|\mathcal{A}|(1-\alpha(s))-1}$ , and  $\mu$  is a probability measure over  $\mathcal{S}$  such that  $\mu(s) > 0 \forall s \in \mathcal{S}$ .

*Proof.* Without loss of generality, we select a single state  $s \in \mathcal{S}$ , observing that the same analysis applies  $\forall s \in \mathcal{S}$ . For simplicity, we denote  $\pi(\cdot|s)$  by  $\pi$ . We start by multiplying each side of Eq. (4.50) by  $K$  and rearranging:

$$\begin{aligned} \sum_{k=1}^K \ell_k(\nu_k) - \sum_{k=1}^K \ell_k(\bar{\nu}_K) &\leq (\text{KL}[\bar{\nu}, p] + 1) L\sqrt{K} \\ \Rightarrow \ell_K(\nu_K) &\leq \sum_{k=1}^K \ell_k(\bar{\nu}_K) - \sum_{k=1}^{K-1} \ell_k(\nu_k) + (\text{KL}[\bar{\nu}, p] + 1) L\sqrt{K} \\ &= \underbrace{\ell_K(\bar{\nu}_K) + \sum_{k=1}^{K-1} (\ell_k(\bar{\nu}_K) - \ell_k(\nu_k))}_{:=G(K)} + (\text{KL}[\bar{\nu}, p] + 1) L\sqrt{K} \end{aligned} \tag{4.58}$$

We can multiply both sides by  $1/2$  and expand  $\ell_K(\nu_K)$ :

$$\begin{aligned}
\frac{1}{2}G(K) &\geq \mathbb{E}_{w \sim \nu_K} \frac{1}{2} \text{KL}[\pi_K^*, \pi_w] \\
&\stackrel{(i)}{=} \text{Var}_{\nu_K} \left[ \sqrt{\frac{1}{2} \text{KL}[\pi_K^*, \pi_w]} \right] + \mathbb{E}_{\nu_K} \left[ \sqrt{\frac{1}{2} \text{KL}[\pi_K^*, \pi_w]} \right]^2 \\
&\stackrel{(ii)}{\geq} \left( \mathbb{E}_{\nu_K} \left[ \sqrt{\frac{1}{2} \text{KL}[\pi_K^*, \pi_w]} \right] \right)^2 \\
&\stackrel{(iii)}{\geq} (\mathbb{E}_{\nu_K} d_{\text{TV}}(\pi_K^*, \pi_w))^2
\end{aligned} \tag{4.59}$$

where (i) follows from the definition of the variance, (ii) follows from its non-negativity, and (iii) follows from Pinsker's inequality. We then have

$$\mathbb{E}_{\nu_K} d_{\text{TV}}(\pi_K^*, \pi_w) \leq \sqrt{\frac{1}{2}G(K)}. \tag{4.60}$$

Letting  $\alpha_K(s) = \sqrt{\frac{1}{2}G(K)}$  and applying Moskowitz et al. (2022a), Lemma 5.2 gives the desired result.  $\square$

#### 4.G.4 Comment on Improvement Across Tasks

To gain intuition for these bounds, there are several important values of  $\alpha(s)$  that we can consider. First, as  $\alpha(s) \rightarrow 1 - 1/|\mathcal{A}|$ , which is the TV distance between a uniform default policy and a deterministic optimal policy,  $\kappa_{\mathcal{A}}^\alpha(s) \rightarrow 2$ . This is an important value because it's the coefficient obtained for log-barrier regularization—that is, when the default policy is uniform and encodes no information about the task distribution. Next, as  $\alpha(s) \rightarrow 0$  (that is, as the TV distance between the default policy and the optimal policy decreases),  $\kappa_{\mathcal{A}}^\alpha(s) \rightarrow 2|\mathcal{A}|/(2|\mathcal{A}| - 1) < 2$  for  $|\mathcal{A}| > 1$ . So, we get faster as the distance between the default policy and the optimal policy decreases, as we would hope. Another crucial point to note is that as  $|\mathcal{A}| \rightarrow \infty$  in this case,  $\kappa_{\mathcal{A}}^\alpha(s) \rightarrow 1$ . Finally, and importantly for MDL-C, as  $\alpha(s) \rightarrow 1 - 1/2|\mathcal{A}|$  from below,  $\kappa_{\mathcal{A}}^\alpha(s) \rightarrow \infty$ . In other words, a sufficiently bad default policy can preclude convergence entirely if it puts too much mass on a suboptimal action. For an illustration of this phenomenon, see Moskowitz et al. (2022a) Figure 4.1.

**Algorithm 6** Off-Policy MDL-C for Parallel Multitask Learning

- 
- 1: require: task distribution  $\mathbb{P}_{\mathcal{M}}$ , policy class  $\Pi$
  - 2: initialize: default policy distribution  $\nu_1 \in \mathcal{N}$ , control replay  $\mathcal{D}_0 \leftarrow \emptyset$ , default replay  $\mathcal{D}_0^\phi \leftarrow \emptyset$
  - 3: initialize control policy parameters  $\theta$  and default policy distribution parameters  $\phi$ .
  - 4: **while** not done **do**
  - 5:   **for** episodes  $k = 1, 2, \dots, K$  **do**
  - 6:     Sample a task  $M_k \sim \mathcal{P}_{\mathcal{M}}(\cdot)$  with goal ID feature  $g_k$
  - 7:     Collect trajectory  $\tau = (\tilde{s}_0, a_0, r_0, \dots, \tilde{s}_{H-1}, a_{H-1}, r_{H-1}) \sim \mathbb{P}^{\pi_\theta}(\cdot)$ , store experience

$$\mathcal{D}_k \leftarrow \mathcal{D}_{k-1} \cup \{(\tilde{s}_h, a_h, r_h, \tilde{s}_{h+1})\}_{h=0}^{H-1} \quad (4.61)$$

where  $\tilde{s}_h := (s_h, g_k)$ .

- 8:   **if**  $R(\tau) \geq R^*$  (i.e.,  $\pi_\theta \approx \pi_k^*$ ) **then**
- 9:     Add to default policy replay:

$$\mathcal{D}_k^\phi \leftarrow \mathcal{D}_{k-1}^\phi \cup \{(\tilde{s}_h, \pi_\theta(\cdot|\tilde{s}_h))\}_{h=0}^{H-1} \quad (4.62)$$

Note that, e.g., when  $\pi_\theta(a|\tilde{s}) = \mathcal{N}(a; \mu(\tilde{s}, g_k), \Sigma(\tilde{s}, g_k))$  is a Gaussian policy,  $\mu(\tilde{s}_h, g_k), \Sigma(\tilde{s}_h, g_k)$  are added to the replay with  $\tilde{s}_h$ .

- 10:   **end if**
- 11:   **end for**
- 12:   Update  $Q$ -function(s) as in Haarnoja et al. (2018).
- 13:   Update control policy:

$$\theta \leftarrow \underset{\theta'}{\operatorname{argmin}} \mathbb{E}_{\text{Unif}_{\mathcal{D}_k}} [V^{\pi_{\theta'}} - \alpha \mathbb{E}_{w \sim \nu_\phi} \text{KL}[\pi_{\theta'}(\cdot|\tilde{s}_h), \pi_w(\cdot|\tilde{s}_h)]] \quad (4.63)$$

- 14:   Update default policy distribution:

$$\phi \leftarrow \underset{\phi'}{\operatorname{argmin}} \text{KL}[\nu_{\phi'}(\cdot), p(\cdot)] + \mathbb{E}_{\text{Unif}_{\mathcal{D}_k^\phi}} \mathbb{E}_{w \sim \nu} \text{KL}[\pi_\theta(\cdot|\tilde{s}_h), \pi_w(\cdot|\tilde{s}_h)] \quad (4.64)$$

- 15: **end while**
- 

Indeed, this is why our Proposition 4.1 is so useful—by effectively placing an upper bound on  $\alpha(s)$  which shrinks as the number of tasks  $K$  increases, MDL-C’s default policy is guaranteed to a) avoid putting too much mass on a suboptimal action and thereby preclude or delay convergence for the control policy, and b) improve the rate as the default policy regret drops.

### 4.G.5 Parallel Task Setting

An overview of MDL-C as applied in the parallel task setting is presented in Algorithm 6. One important feature to note is the return threshold  $R^*$ . As a proxy for the control policy converging to  $\pi_k^*$ , data are only added to the default policy replay buffer when a trajectory return is above this threshold performance (on DM control suite tasks,  $R^*$  corresponded to a test reward of at least 700). We leave more in-depth theoretical analysis of this setting to future work, but note that as the task experience is interleaved,  $\bar{\pi}_w = \mathbb{E}_\nu \pi_w$  will converge to the prior-weighted KL barycenter. If, in expectation, this distribution is a TV distance of less than  $1 - 1/|\mathcal{A}|$  from  $\pi_k^*$ , then the control policy will converge faster than for log-barrier regularization (Moskovitz et al., 2022a).

## Appendix 4.H: Additional Experimental Details

Below, we describe experimental details for the two environment domains in the paper.

### 4.H.6 FourRooms

As input, the agent receives a 16-dimensional vector containing the index of the current state, a flattened  $3 \times 3$  local view of its surrounding environment, its previous action taken encoded as a 4-dimensional one-hot vector, the reward on the previous timestep, and a feature indicating the goal state index. The base learning algorithm in all cases is advantage actor critic (A2C; (Mnih et al., 2016)).

**Environment** The FOURROOMS experiments are set in an  $11 \times 11$  gridworld. The actions available to the agent are the four cardinal directions, up, down, left, and right, and transitions are deterministic. In both FOURROOMS experiments, the agent can begin an episode anywhere in the environment (sampled uniformly at random), and a single location with reward  $r = 50$  is sampled at the beginning of each episode from a set of possible goal states which varies depending on the experiment and the current phase. A reward of  $r = -1$  is given if the agent contacts the walls. All other states give a reward of zero. Episodes end when either a time (number of timesteps) limit is reached or the agent reaches the goal state. Observations were 16-dimensional vectors consisting of the current state index (1d), flattened  $3 \times 3$  local window surrounding the agent (includes walls, but not goals), a one-hot encoding of the action on the previous timestep (4d), the reward on the previous timestep (1d), and the state index of the current goal (1d). In the

“goal generalization” experiment, goals may be sampled anywhere in either the top left or bottom right rooms in the first phase and either the top right or bottom left rooms in the second phase. Each phase comprises 20,000 episodes, and in each phase, the agent may start each episode anywhere in the environment. In the first phase, the agent was allowed 100 steps per episode, and in the second phase 25 steps. In the “contingency change” experiment, the possible reward states in each phase were the top left state and bottom right state. In the second phase of training, however, the semantics of the goal feature change from indicating the location of the reward to the location where it is absent. Each phase consisted of 8,000 episodes with maximum length 100 timesteps. Results are averaged over 10 random seeds.

**Agents** All agents were trained on-policy with advantage actor-critic (Mnih et al., 2016). The architecture was a single-layer LSTM (Hochreiter and Schmidhuber, 1997a) with 128 hidden units. To produce the feature sensitivity plots in Fig. 4.2c, a gating function was added to the input layer of the network:

$$x_h = \sigma(b\kappa) \odot o_h, \quad (4.65)$$

where  $o_h$  is the current observation,  $\sigma(\cdot)$  was the sigmoid function,  $b \in \mathbb{R}$  is a constant (set to  $b = 150$  in all experiments),  $x_h \in \mathbb{R}^d$  is the filter layer output, and  $\kappa \in \mathbb{R}^d$  is a parameter trained using backpropagation. In this way, as  $\kappa_d \rightarrow \infty$ ,  $\sigma(b\kappa_d) \rightarrow 1$ , allowing input feature  $o_h, d$  through the gate. As  $\kappa_d \rightarrow -\infty$ , the gate is shut. The plots in Fig. 4.2c track  $\sigma(b\kappa_d)$  over the course of training. The baseline agent objective functions are as follows:

$$\begin{aligned} \mathcal{J}^{\text{PO}}(\theta) &= V^{\pi_\theta} + \alpha \mathbb{E}_{s \sim d^{\pi_\theta}} \text{H}[\pi_\theta(\cdot|s)] \\ \mathcal{J}^{\text{RPO}}(\theta, \phi) &= V^{\pi_\theta} - \alpha \mathbb{E}_{s \sim d^{\pi_\theta}} \text{KL}[\pi_\theta(\cdot|s), \pi_\phi(\cdot|s)] \\ \mathcal{J}^{\text{VDO-PO}}(\theta) &= \mathbb{E}_{w \sim \nu_\theta} V^{\pi_w} - \beta \text{KL}[\nu_\theta(\cdot), p(\cdot)] \\ \mathcal{J}^{\text{ManualIA}}(\theta, \phi) &= V^{\pi_\theta} - \alpha \mathbb{E}_{s \sim d^{\pi_\theta}} \text{KL}[\pi_\theta(\cdot|s), \pi_\phi(\cdot|s_d)]; \quad s_d = s \setminus g. \end{aligned} \quad (4.66)$$

In all cases  $\alpha = 0.1$ ,  $\beta = 1.0$ , and learning rates for all agents were set to 0.0007. Agents were optimized with Adam (Kingma and Ba, 2014). Agent control policies were reset



after phase 1.

#### 4.H.7 DeepMind Control Suite

**Environments/Task Settings** We use the walker and cartpole environments from the DeepMind Control Suite (Tassa et al., 2018). We consider two multitask settings: sequential tasks and parallel tasks. All results are averaged over 10 random seeds, and agents are trained for 500k timesteps. In the sequential task setting, tasks are sampled one at a time without replacement and solved by the agent. The control policy is reset after each task, but the default policy is preserved. For methods which have a default policy which can be preserved, performance on task  $k$  is averaged over runs with all possible previous tasks in all possible orders. For example, when `walker-run` is the third task, performance is averaged over previous tasks being `stand` then `walk` and `walk` then `stand`. In the parallel task setting, a different task is sampled randomly at the start of each episode, and a one-hot task ID vector is appended to the state observation. Learning was done directly from states, not from pixels.

**Agents** The base agent in all cases was SAC with automatic temperature tuning, following Haarnoja et al. (2018). Standard SAC seeks to optimize the maximum-entropy RL objective:

$$\mathcal{J}^{\text{max-ent}}(\pi) = V^\pi + \alpha \mathbb{E}_{s \sim d^\pi} \mathbf{H}[\pi(\cdot|s)] = V^\pi + \alpha \mathbb{E}_{s \sim d^\pi} \text{KL}[\pi(\cdot|s), \text{Unif}_{\mathcal{A}}] \quad (4.67)$$

Effectively, then, SAC uses a uniform default policy. The RPO algorithms with learned default policies replace  $\text{KL}[\pi(\cdot|s), \text{Unif}_{\mathcal{A}}]$  with  $\text{KL}[\pi(\cdot|s), \pi_w(\cdot|s)]$  (or  $\text{KL}[\pi_w(\cdot|s), \pi(\cdot|s)]$ ). As MDL-C, RPO, and TVPO require that the control policy approximate the optimal policy before being used to generate a learning signal for the default policy, in the sequential setting, the default policy is updated only after halfway through training. Because variational dropout can cause the network to over-sparsify (and not learn the task adequately) if turned on too early in training, we follow the strategy of Molchanov et al. (2017), linearly ramping up a coefficient  $\beta$  on the variational dropout KL from 0 to 1 starting from 70% through training to 80% through training. Note that MANUALIA is not applicable to the sequential task setting, as there is no explicit goal feature. In the

sequential task setting, we took inspiration from Haarnoja et al. (2018) and Abdolmaleki et al. (2018) and reframed the soft KL penalty for methods with learned default policies as a constraint, i.e.,

$$\max_{\pi} V^{\pi} - \alpha \mathbb{E} \text{KL}[\pi_w, \pi] \quad \longrightarrow \quad \max_{\pi} V^{\pi} \text{ s.t. } \mathbb{E} \text{KL}[\pi_w, \pi] \leq \varepsilon,$$

where  $\varepsilon > 0$  was a target KL divergence. Under this formulation,  $\alpha$  is treated as a dual variable via Lagrangian relaxation and optimized with the following objective:

$$\max_{\alpha \geq 0} J(\alpha) := \mathbb{E} \alpha \text{KL}[\pi_w, \pi] - \alpha \varepsilon.$$

In the parallel task setting, we convert the base SAC agent into the “multitask” variant used by Yu et al. (2019), in which the agent learns a vector of temperature parameters  $[\alpha_1, \dots, \alpha_K]$ , one for each task. In this setting, we found it more effective to set  $\alpha$  to a constant value. Test performance was computed by averaging performance across all  $K$  tasks presented to the agent. The baseline agent objectives are as in Eq. (4.66), and the Distal objective is given by

$$\mathcal{J}^{\text{Distal}}(\theta, \phi) = V^{\pi_{\theta}} - \alpha \mathbb{E}_{s \sim d^{\pi_{\theta}}} \text{KL}[\pi_{\theta}(\cdot|s), \pi_{\phi}(\cdot|s)] + \lambda \mathbb{E}_{s \sim d^{\pi_{\theta}}} \text{H}[\pi_{\theta}(\cdot|s)].$$

TVPO is trained in the same way as RPO, with the difference being that the default policy objective is to predict the control policy action, rather than a distillation objective. Hyperparameters shared by all agents can be viewed in Table 4.2.

As a note on performance, Distal performs very strongly in the parallel task setting, with overall performance slightly worse than MDL-C on Walker and virtually the same on Cartpole. However, the gap is significantly greater in the sequential setting, particularly on Walker. We hypothesize that this is due to the fact that by regularizing the control policy to be close to the default policy, but also encouraging the control policy to have high entropy (rather than regularizing the default policy as MDL-C does), Distal can in effect provide a conflicting objective to the control policy when strong structure is present. In particular, on Walker, the optimal policies for each task have significant overlap, and so

by encouraging high entropy in the control policy even on the third task, Distral negates the effect of an informative default policy. As evidence, both RPO and TVPO, which only regularize the control policy to be close to the default policy, perform significantly more strongly on Walker in the sequential setting.

Hyperparameter	Value
Collection Steps	1000
Random Action Steps	10000
Network Hidden Layers	256:256
Learning Rate	$3 \times 10^{-4}$
Optimizer	Adam
Replay Buffer Size	$1 \times 10^6$
Action Limit	$[-1, 1]$
Exponential Moving Avg. Parameters	$5 \times 10^{-3}$
(Critic Update:Environment Step) Ratio	1
(Policy Update:Environment Step) Ratio	1
Expected KL/Entropy Target	$0.2 / -\dim(\mathcal{A})^*$
Policy Log-Variance Limits	$[-20, 2]$

**Table 4.2:** DM control suite hyperparameters, used for all experiments. \*In the parallel setting,  $\alpha$  was simply set to 0.1 for methods with learned default policies.

## Appendix 4.I: Additional Experimental Results

### 4.I.8 FourRooms

Method	Goal Change	Contingency Change
PO	$1.25e5 \pm 1.76e4$	$8.80e4 \pm 1.64e4$
RPO	$1.77e5 \pm 1.11e4$	$1.04e5 \pm 2.20e4$
VDO-PO	$1.48e5 \pm 1.91e4$	$8.23e4 \pm 1.98e4$
ManualIA	$1.23e5 \pm 2.51e4$	$7.69e4 \pm 2.89e4$
MDL-C	$1.08e5 \pm 2.44e4$	$5.11e4 \pm 1.70e4$

**Table 4.3:** FourRooms: Average cumulative regret across 8 random seeds in phase 2 of the goal change and contingency change experiments for each method.  $\pm$  values are standard error.

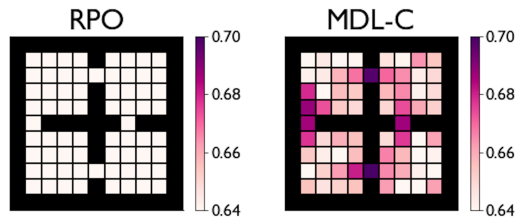
### 4.I.9 DeepMind Control Suite

Method	Cartpole	Walker
SAC	$1.25e5 \pm 1.76e$	$3.42e5 \pm 6.10e4$
RPO-SAC ( $k = 3$ )	$1.77e5 \pm 1.11e4$	$1.04e5 \pm 2.20e4$
VDO-SAC	$1.48e5 \pm 1.91e4$	$8.23e4 \pm 1.98e4$
MDL-C ( $k = 1$ )	$1.23e5 \pm 2.51e4$	$7.69e4 \pm 2.89e4$
MDL-C ( $k = 2$ )	$1.08e5 \pm 2.44e4$	$5.11e4 \pm 1.70e4$
MDL-C ( $k = 3$ )	$1.08e5 \pm 2.44e4$	$5.11e4 \pm 1.70e4$

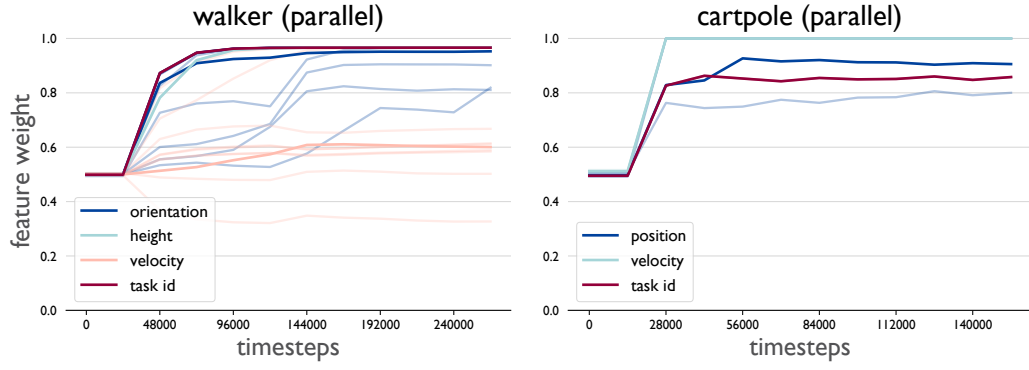
**Table 4.4:** DM Control Suite, Sequential: Average cumulative regret across 8 random seeds in the sequential setting.  $\pm$  values are standard error.

Method	Cartpole	Walker
SAC	$1.01e5 \pm 2.01e3$	$1.46e5 \pm 5.11e3$
ManualIA	$9.90e4 \pm 1.87e3$	$1.50e5 \pm 3.86e3$
MDL-C	$9.47e4 \pm 8.36e2$	$1.31e5 \pm 1.35e3$

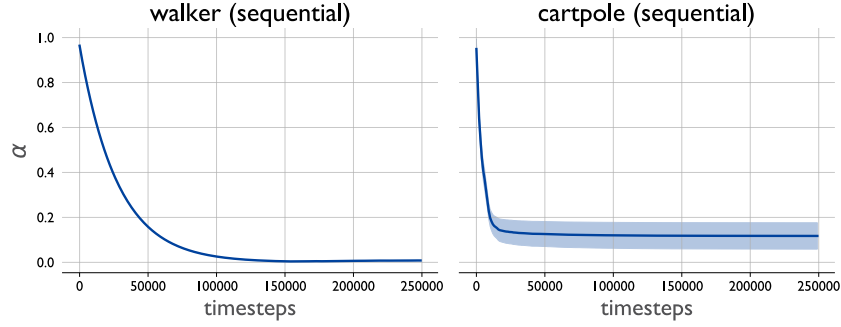
**Table 4.5:** DM Control Suite, Parallel: Average cumulative regret across 8 random seeds in the parallel task setting.  $\pm$  values are standard error.



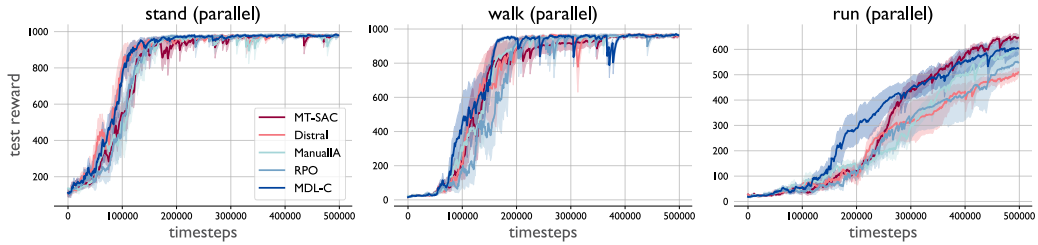
**Figure 4.5:** Heatmaps of  $\text{KL}[\pi_\theta(\cdot|s), \pi_w(\cdot|s)] \forall s \in \mathcal{S}$  for RPO and  $\text{KL}[\pi_\theta(\cdot|s), \pi_{\bar{w}}(\cdot|s)] \forall s \in \mathcal{S}$ , where  $\bar{w} = \mathbb{E}_\nu w$  for MDL-C, averaged over all possible goal states. The RPO default policy nearly perfectly matches the control policy, while the MDL-C default policy diverges most strongly from the control policy at the doorways. This is because the direction chosen by the policy in the doorways is highly goal-dependent. Because the MDL-C default policy learns to ignore the goal feature, it's roughly uniform in the doorways, whereas the control policy is highly deterministic, having access to the goal feature.



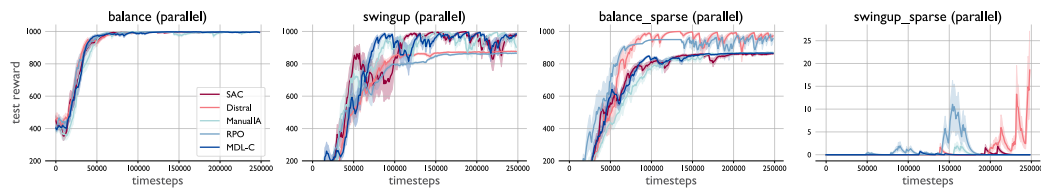
**Figure 4.6:** Without a sparse prior, RPO does not learn to ignore spurious input features.



**Figure 4.7:** MDL-C’s learned  $\alpha$ s in the DMC sequential setting. Because  $\alpha$  tends to decay, the control policy is able to specialize to the current task later in training. Results averaged over eight random seeds; error shading denotes standard error.



**Figure 4.8:** Test reward on each individual task in the walker domain over the course of parallel task training. Average performance is plotted above over 10 seeds, with the shading representing one unit of standard error. We can see the biggest performance difference on walker, run, the most challenging task.



**Figure 4.9:** Test reward on each individual task in the `cartpole` domain over the course of parallel task training. Average performance is plotted above over 10 seeds, with the shading representing one unit of standard error. Interestingly, unlike in the sequential learning setting, joint training seems to impede performance on `swingup_sparse`, with no method succeeding.

## Chapter 5

# A Unified Theory of Dual-Process Control

The previous chapter introduced MDL-C, a regularized policy optimization approach for learning (near-)optimal policies more quickly. The underlying idea is to capture consistent behavioral structure required to solve previously observed tasks in a default policy which provides partial supervision to the control policy as it learns new tasks. In order to prevent the default policy from overfitting to spurious structure mistakenly inferred due to limited data, MDL-C also limits the effective capacity of the default policy by regularizing its complexity using variational dropout. This chapter, adapted from [Moskovitz et al. \(2022b\)](#), explores MDL-C further as a model of behavioral phenomena associated with dual process cognition in the brain.

### 5.1 Introduction

As introduced in the previous chapter, the idea that cognition is split into a complex process, manifested in behavior via goal-directed, deliberative action, and a simple process, manifested through ingrained habits, is known as *dual process theory*. Dual process theories of cognition have had a long and influential history across multiple sub-fields of the cognitive sciences, such as cognitive control ([Diamond, 2013](#); [Botvinick and Cohen, 2014](#)), reward-based decision-making ([Dolan and Dayan, 2013](#); [Perez and Dickinson, 2020](#)), and judgement and decision-making ([Evans, 2008](#); [Kahneman, 2011](#)). These sub-fields not only attempt to answer different questions (and thus test hypotheses through

different tasks), they also use different modeling approaches and use different language to describe their experiments. This has made it challenging to derive a unified theoretical framework for these results despite their shared heritage in dual process ideas.

While the reduction of action selection to dual processes is undoubtedly a simplification, across these three domains, dual-process models have accumulated considerable empirical support, and each domain has developed explicit computational models of how dual processes might operate and interact (Lieder and Griffiths, 2017; Botvinick and Cohen, 2014; Rougier et al., 2005; Daw et al., 2005; Shenhav et al., 2013; Keramati et al., 2011; Boureau et al., 2015; Perez and Dickinson, 2020; Miller et al., 2019). These computational models, however, are typically domain-specific, reproducing behavioral phenomena that are within the scope of their domain. It remains unknown whether dual-process phenomena in different domains result from different sets of computational mechanisms, or whether they can be understood as different manifestations of a single, shared set. That common mechanisms might be at play is suggested by a wealth of neuroscientific data. Specifically, studies have linked controlled behavior, model-based action selection, and System-2 decision making with common circuits centering on the prefrontal cortex (Diamond, 2013; Dolan and Dayan, 2013; Mevel et al., 2019; De Neys and Goel, 2011; Miller and Cohen, 2001; Jeon and Friederici, 2015), while automatic behavior, habitual action selection, and heuristic decision making appear to engage shared circuits lying more posterior and running through the dorsolateral striatum (Lieberman, 2007; O'Reilly et al., 2020; Jeon and Friederici, 2015; Smith and Graybiel, 2022). While further study into these neuroanatomical relationships is required, these results do beg the question of whether a single computational model could account for these patterns of decision-making.

In this work, we seek a *normative explanation* for these phenomena. That is, we seek a theory that can reproduce behavioral findings associated with dual process cognition, but which is derived instead from an optimization principle, allowing dual process cognition to be understood as the solution to a fundamental behavioral or computational problem. To identify such a principle, we begin by considering a fundamental problem confronting both biological and machine intelligence: generalization. We discuss a fundamental computational theory of generalization, link it to behavior, and demonstrate



that a recently-proposed behavioral model from machine learning based on this principle can successfully reproduce canonical dual-process phenomena from executive control, reward-based learning, and JDM.

We propose that MDL-C may offer a useful normative model for dual-process behavioral phenomena. As in dual-process theory, MDL-C contains two distinct decision-making mechanisms. One of these ( $RNN_{\pi_0}$ ) distills as much target behavior as possible in an algorithmically simple form, reminiscent of the habit system or System 1 in dual-process theory. Meanwhile, the other ( $RNN_{\pi}$ ) enjoys greater computational capacity and intervenes when the simpler mechanism fails to select the correct action, reminiscent of executive control or System 2 in dual-process theory. MDL-C furnishes a normative explanation for this bipartite organization by establishing a connection with the problem of behavioral generalization. To test whether MDL-C can serve as such a model, we conducted a series of simulation studies spanning the three behavioral domains where dual-process theory has been principally applied: executive control in Simulation 1, reward-based decision making in Simulation 2, and JDM in Simulation 3.

## 5.2 General methods: Selection of target phenomena and approach to modeling

A detailed description of simulation methods is presented in Appendix 5.A. Briefly, for each target dual-process domain, we focused on a set of empirical phenomena that the relevant specialty literature treats as fundamental or canonical. We do not, of course, address all behavioral and neural phenomena that might be considered relevant to constrain theory in each domain, and we dedicate a later section to the question of whether any empirical findings that we do not directly model might present challenges for our theory. Nevertheless, the core phenomena in each field are fairly well recognized, and we expect our selections will be uncontroversial. Indeed, each target phenomenon has been the focus of previous computational work, and we dedicate a later section to comparisons between our modeling approach and previous proposals. While such comparisons are of course important, one point that we continue to stress throughout is that no previous model has addressed the entire set of target phenomena, bridging between the three

domains we address.

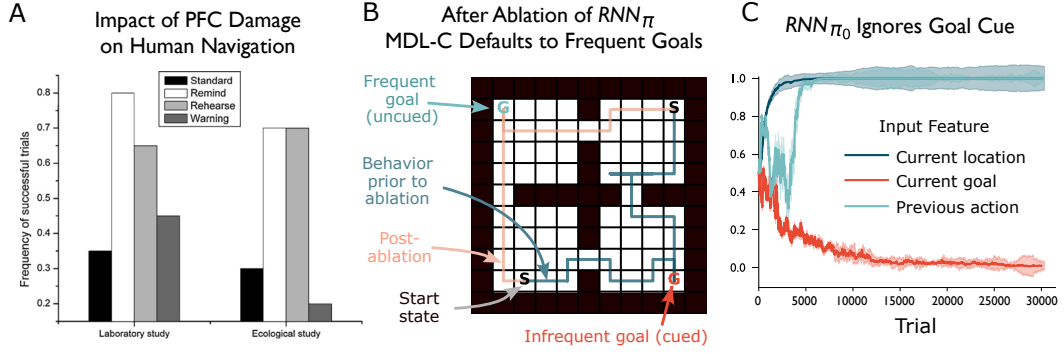
For each target phenomenon, we pursue the same approach to simulation: We begin with a generic MDL-C agent model, configured and initialized in the same way across simulations (with the exception of input and output unit labels tailored to the task context). The model is then trained on an appropriate target task and its behavior or internal computations queried for comparison with target phenomena. Importantly, the model is in no case directly optimized to capture target phenomena, only to solve the task at hand. In the rare case where target effects depend sensitively on experimenter-chosen hyperparameters of MDL-C, this dependency is described alongside other results.

While our simulations focus on target phenomena that have been documented across many experimental studies, in presenting each simulation we focus on observations from one specific (though representative) empirical study, to provide a concrete point of reference. It should be noted that the target phenomena we address, in almost all cases, take the form of qualitative rather than quantitative patterns. Our statistical tests, described in Appendix 5.A, thus take the form of qualitative hypothesis tests rather than quantitative fits to data, paralleling the reference experimental research.

## 5.3 Results

### 5.3.1 Simulation 1: Executive control

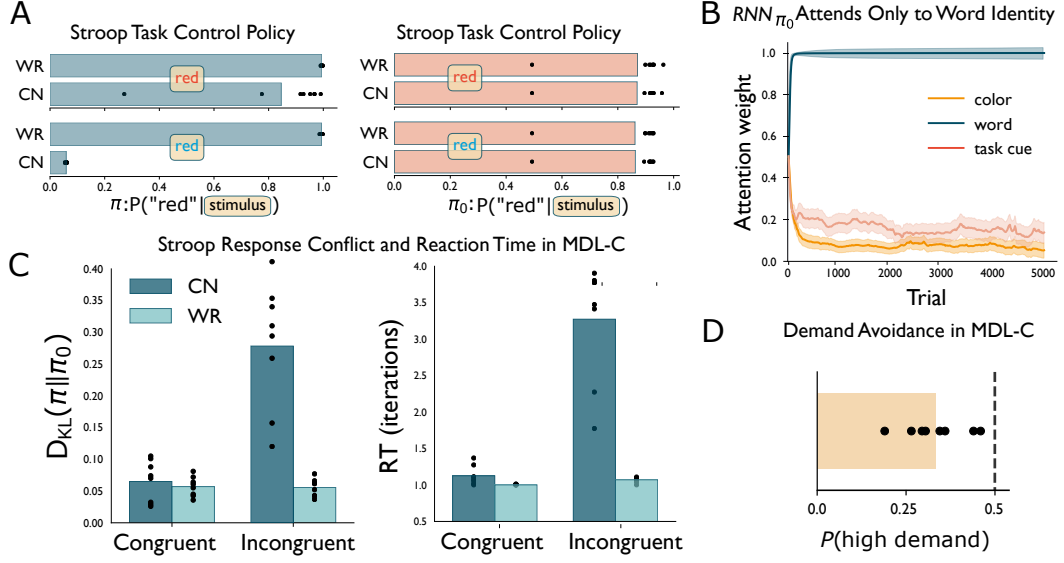
As introduced above, longstanding theories of executive function center on a contrast between two kinds of action. Habitual or automatic responses are default, reactive actions, shaped by frequency or practice. Controlled responses, in contrast, take fuller account of the task context, overriding automatic responses when they are inappropriate (Diamond, 2013; Botvinick and Cohen, 2014; Miller and Cohen, 2001). Some of the strongest support for this distinction comes from studies of prefrontal cortex. Prefrontal neural activity has been shown to play a special role in encoding goals, task instructions, and other aspects of task context (Miller and Cohen, 2001; Diamond, 2013). The importance of these representations for context-appropriate behavior is evident in the effects of prefrontal damage, where behavior tends to default to frequently performed actions, neglecting verbal instructions or context-appropriate goals.



**Figure 5.1:** **A.** Ciaramelli (2008) reported that damage to another (orbitofrontal) region of PFC impaired navigation to novel goals, both in the laboratory and an ecological study. In unsuccessful trials patients frequently navigated to familiar goal locations. Performance improved when patients were given frequent reminders of the goal or were asked to verbally rehearse the goal, but not when the goal reminder was replaced by an uninformative stimulus (*Warning*). **B.** In a modified navigation task only two goals were cued, one (blue G) occurring more frequently during training than the other (red G). When the infrequent goal is cued at test, the intact MDL-C agent navigates successfully to it from any start state (see blue example trajectories). When  $RNN_{\pi}$  is ablated, the agent ignores the instruction cue and navigates to the more frequent goal (pink trajectories). See Methods for simulation details. **C.** By inserting a gating layer over input features within  $RNN_{\pi_0}$  (see Methods), we can directly read out which information is processed by that pathway. The plot shows attention weights for the three input features in the navigation task referenced in Figure 1. Over the course of the initial training block,  $RNN_{\pi_0}$  learns to ignore the current goal cue.

One domain in which these effects can be observed in a particularly straightforward form is spatial navigation. Prefrontal damage impairs the ability to navigate to instructed goal locations, with behaviour defaulting to more familiar paths and destinations (Ciaramelli, 2008) (Fig. 5.1).

Strikingly similar effects arise when MDL-C is applied to spatial navigation. In our first simulation, a MDL-C agent was trained on a navigation task involving two cued goal locations, with one goal presented more frequently than the other (see Appendix 5.B). After training,  $RNN_{\pi}$  was able to successfully navigate to either goal when cued. However, when  $RNN_{\pi}$  was removed from the agent and it was forced to act using  $RNN_{\pi_0}$ —in a rough approximation of the PFC damage suffered by the patients studied by Ciaramelli (2008)—agents only ever navigated to the goal location that had been more frequently cued during training (Fig. 5.1B). To gain a mechanistic understanding of why this occurs, we inserted a gating layer over inputs in  $RNN_{\pi_0}$  to monitor which information is trans-



**Figure 5.2:** **A.** Policies for  $RNN_{\pi}$  (top) and  $RNN_{\pi_0}$  (bottom) for the stimuli shown, in word-reading ( $WR$ ) and color-naming ( $CN$ ) contexts. Response probabilities are shown for the response *red*, complementary to (unshown) probabilities for the alternative *blue* response. **B.** When the MDL-C agent is trained on the Stroop task (see Methods),  $RNN_{\pi_0}$  learns to ignore both the task cue and the stimulus color, attending only to word identity. **C.** Left: KL divergence between  $\pi$  and  $\pi_0$  for the four trial types shown in panel A. Right: Corresponding reaction times (see Methods). **D.** When trained on the Stroop task and then given a choice between blocks of color-naming trials that involve either high or low proportions of incongruent stimuli (see Methods), the MDL-C agent displays a preference for less frequent incongruence, paralleling the demand-avoidance effect seen in human decision making.

mitted to the policy. We found that, despite the fact that both  $RNN_{\pi}$  and  $RNN_{\pi_0}$  receive the same inputs, VDO induced  $RNN_{\pi_0}$  to ignore the goal cue during training, as due to the difference in goal presentation frequencies, it was less predictive of  $RNN_{\pi}$ 's behavior than other features.

To evaluate the generality of these effects, we applied MDL-C to another classic executive control problem, the Stroop task (Stroop, 1935) (see Appendix 5.B and Fig. 5.2). Here, words that name colors are presented in hues that are either incongruent (e.g. *RED* presented in blue) or congruent (*RED* in red). An instruction cue indicates whether the current task is to read the word, the highly practiced, automatic response, or to name the color, requiring cognitive control. Consistent with the navigation results, while the control policy correctly learned to respond to both word-reading and color-naming trials (the former being presented more frequently in training), the default policy learned

a simpler stimulus-response mapping based only on the written word (Fig. 5.2A). These habit-like responses are overridden (by policy  $\pi$ ) only when the task context requires it. When examining feature sensitivity,  $RNN_{\pi_0}$ , as in navigation, ignores the task context and is biased toward the behaviors executed most frequently during learning, consistent with the classical definition of automatic processing (Fig. 5.2B).

Perhaps the defining behavioral phenomenon associated with the Stroop task is delayed reaction times on incongruent color-naming trials (as people are more used to reading words than naming colors) (Botvinick and Cohen, 2014; Herd et al., 2006), another finding replicated by MDL-C. MDL-C provides a simple way to reason about this pattern: because the control policy is regularized towards the default policy—which disagrees with the control policy on these inputs—its output distribution is less concentrated over the correct output, requiring more recurrent cycles to reach the response threshold. The KL divergence between the control and default policies was therefore highest for color-naming conflict trials, as it was in these trials alone for which simply matching the written word resulted in the incorrect response (Fig. 5.2C). In this way, MDL-C provides a direct relationship between reaction time and the cost of control.

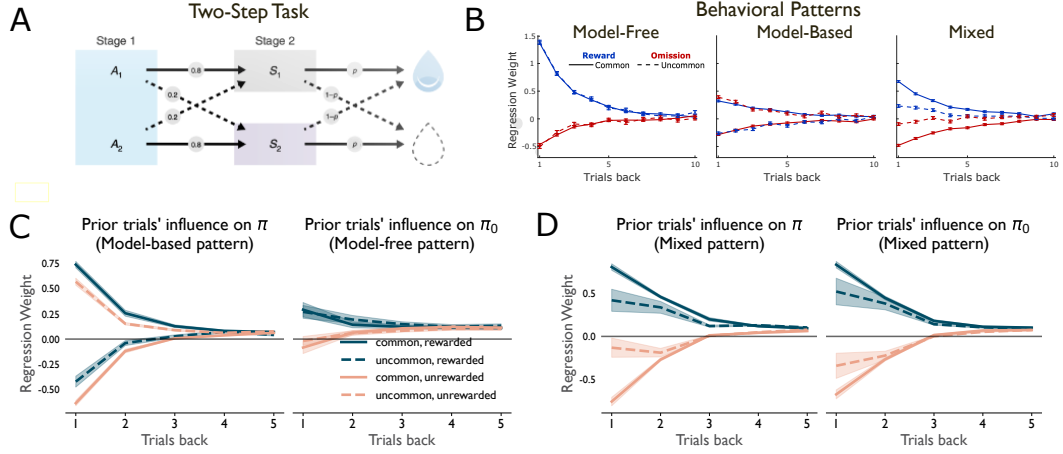
Another core phenomenon in the cognitive control literature is *demand avoidance*, the tendency for decision makers to avoid tasks that require intensive cognitive control (Kool and Botvinick, 2018). For example, when human participants are asked to select between two versions of the Stroop task, one involving more frequent incongruent trials than the other, they show a clear tendency to avoid the former task and the demands on cognitive control it involves (Schouppe et al., 2014). When MDL-C is trained in the same task context (see Appendix 5.B), the same choice bias arises (Fig. 5.2E). The explanation for this result is tied to the KL cost in the MDL-C objective function which penalizes conflict between policies  $\pi$  and  $\pi_0$  (compare Zenon et al. (2019); Piray and Daw (2021)). By avoiding control-demanding tasks, the agent can minimize this term, helping it to minimize the description length of its overall behavioral policy.

The relation of the above simulation results to those from previous models, and a consideration of a wider range of empirical phenomena in the domain of executive control, are discussed below under *Comparison with previous models*.

### 5.3.2 Simulation 2: Reward-based learning

According to prevailing theories, reward-based learning centers on two distinct neural systems. One, operating within parts of prefrontal cortex and associated basal ganglia circuits, implements a ‘goal-directed’ or ‘model-based’ algorithm, which takes task structure into account. The other system, more posterior or lateral, operates in a ‘habitual’ manner, based on simpler stimulus-response associations (Dolan and Dayan, 2013; Daw et al., 2005; Beierholm et al., 2011; Gläscher et al., 2010; Averbeck and O’Doherty, 2022; Drummond and Niv, 2020; Miller et al., 2019; Dickinson, 1985). Although the anatomical substrates proposed for these systems can resemble those associated with controlled and automatic processing, different behaviors have been used to study them. In research with humans, the most prominent of these is the so-called ‘two-step task’ (Daw et al., 2011), illustrated in Fig. 5.3A.

The two-step task was designed to probe the operation of model-based and habitual systems, under the hypothesis that these operate in parallel and that the habitual system implements model-free reinforcement learning (Averbeck and O’Doherty, 2022; Drummond and Niv, 2020) (see *Comparison with previous models* and Supplementary Discussion). In this task, subjects must choose between two options that will probabilistically transition them to one of two second stage states which themselves stochastically either produce reward or nothing (Fig. 5.3A). According to the logic of the task, if the agent is able to learn a model of this transition structure, its policy update on the first step will be sensitive both to second-step reward as well as to whether the second-step state was the “common” or “uncommon” one given first-step action. This ability is reflected in behavioral patterns classically thought-of as diagnostic for model-based and model-free behavior on this task (Fig. 5.3B), which shows the results of logistic regression from previous trial results to predict whether subjects repeated their most recent stage 1 choice. Synthetic behavioral data from a model-free (TD(1)) agent is associated with positive regression weights for trials which resulted in reward after both common and uncommon transitions, indicating a lack of understanding of the task structure. In contrast, synthetic behavioral data from a model-based agent is associated with positive regression weights for common, rewarded trials and uncommon, unrewarded trials. We trained MDL-C



**Figure 5.3:** **A.** Structure of the two-step task as introduced by ?. Choice occurs at Stage 1. The value of  $p$  varies over time, and so must be inferred by the participant. Following subsequent research, the version employed in our experiments additionally included explicitly cued reversals in the structure of transitions from Stage 1 to Stage 2. See Methods for full details. **B.** Classical behavioral signatures of model-free (left) and model-based (center) performance in the two-step task. Adapted from [Miller et al. (2016a)], the plots show logistic regression weights quantifying the influence of two factors on the probability of repeating on the index trial the same first-stage action selected on the previous trial: (1) whether reward was received or omitted on the previous trial, and (2) whether the previous trial featured a transition from stage 1 to 2 that was high-probability (*common*) or low (*uncommon*). The right panel shows a hybrid pattern, similar to that reported in the classic study by (?). **C.** Left: Two-step behavior of MDL-C, reflecting policy  $\pi$ . Right: Influence of the past on policy  $\pi_0$ . **D.** Same as Panel D but with different weighting of terms in the MDL-C objective (see Methods and compare panel C, right).

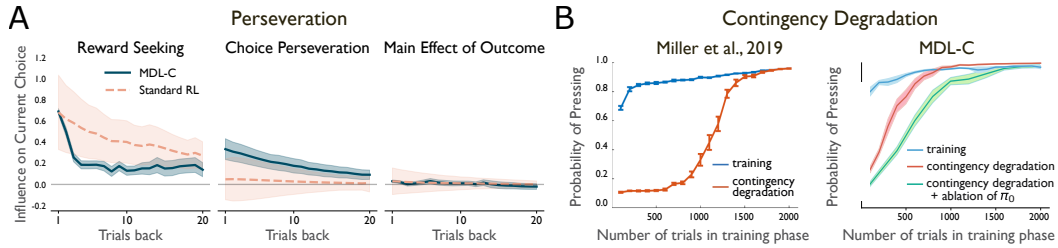
on a modified version of the task, in which the first stage transition probabilities also occasionally switch ([Akam et al., 2015] (see Appendix 5.C for details), which increases the difference in computational complexity needed to exhibit the canonical model-based vs model-free behavioral patterns. We find that, under certain carefully-chosen parameterizations, the classic patterns arising side by side, with policy  $\pi$  displaying the model-based profile, and  $\pi_0$  the model-free pattern (Fig. 5.3C). Because  $\pi$  dictates the overt behavior of the agent, the latter displays a model-based pattern, as also seen in human performance in some studies ([Feher da Silva and Hare, 2020]). When  $RNN_\pi$  is ablated, behavior then shifts away from the model-based pattern, in line with the observation that disruption of prefrontal function decreases model-based control in the two-step task ([Smittenaar et al., 2013; Otto et al., 2013]).

This differentiation of function arises, as in the previous simulations, from the MDL-C optimization objective. As has been noted in the literature on model-based versus model-free learning, the latter is less algorithmically complex (Daw et al., 2005). The simplicity bias in MDL-C, imposed on  $\pi_0$ , therefore tilts that policy toward the actions that would be chosen by a model-free agent. Policy  $\pi$ , meanwhile, can reap a bit more reward by implementing a policy that takes task structure more fully into account.

Work with the two-step task has consistently found that both humans and animals show a variety of "mixed" patterns (Akam et al., 2021; Miller et al., 2017; Daw et al., 2011) distinct from either of the classic patterns. It has also cast doubt on the idea that these patterns, quantified from behavior, map 1:1 onto other measures of goal-directed or habitual control (Feher da Silva and Hare, 2020; Collins and Cockburn, 2020; Gillan et al., 2015). When we train MDL-C over a broader range of hyperparameters (see Appendix 5.C), we observe similar mixed patterns across large portions of the parameter space (Fig. 5.3D and Supplementary Discussion), and that either primarily "model-based", "model-free" or "perseverative" behavior can appear in either  $\pi$  or  $\pi_0$ . Thus, while a clean separation between model-based and model-free learning can arise within MDL-C, such a division is not hardwired into the framework. Depending on the precise setting, minimizing the description length of behavior can also lead to graded intermediate patterns, providing leverage on some otherwise problematic experimental observations (Collins and Cockburn, 2020).

While the two-step task has been an important driver of dual-process theory in the domain of reward-based learning, important insights have also come from studies of instrumental learning. One key feature of animal behavior within this domain is *perseveration*: the tendency to repeat previous actions independent of their association with reward. Miller et al. (2018) administered a two-arm bandit task to rats, where the probability of one of two ports delivering a juice reward drifted randomly across trials. Performing logistic regression on different features of the last 20 trials showed that past choices contingent on reward and the repetition of previous actions had a strong influence on behavior on the current trial. We simulated this experiment, and found that agents trained for simple reward maximization were influenced by previous rewards contingent on choices, but





**Figure 5.4:** **A.** Logistic regression weights showing the influence on the current action of reward contingent on choice (reward seeking), previous choices (perseveration), and reward independent of choice (main effect of outcome) of MDL-C and a standard RL agent on the drifting two-armed bandit task from Miller et al. (2018). MDL-C displays a stronger tendency towards perseveration, reminiscent of rats trained on the same task. **B.** Left: Simulation of contingency degradation from Miller et al. (2019). The longer the training phase (x axis), the longer lever-pressing persists after reward is discontinued (red). Right: Corresponding behavior from MDL-C, also showing the effect of ablating  $\pi_0$ .

did not display perseverative tendencies, while MDL-C agents exhibited both (Fig. 5.4A, details in Appendix 5.C).

Another important experimental manipulation within this literature is known as *contingency degradation*. Here, rewards are at first delivered only in response to a particular action, but then later are delivered in a non-contingent manner, independent of whether the action was selected. Unsurprisingly, this change typically triggers a shift away from the action in question. Critically, however, this adjustment is reduced or slowed if the initial training with reward was extensive (Daw et al., 2005; Miller et al., 2019; Dickinson, 1998) (Fig. 5.4B). Prevailing explanations for this effect share a dual-process perspective, according to which insensitivity to contingency degradation reflects a transfer of control from one learning process that is relatively flexible to another which adjusts less quickly (Daw et al., 2005; Miller et al., 2019). Consistent with this account, lesions to dorsolateral striatum — a structure proposed to be involved in that latter system — partially protects against training-induced inflexibility (Yin et al., 2006).

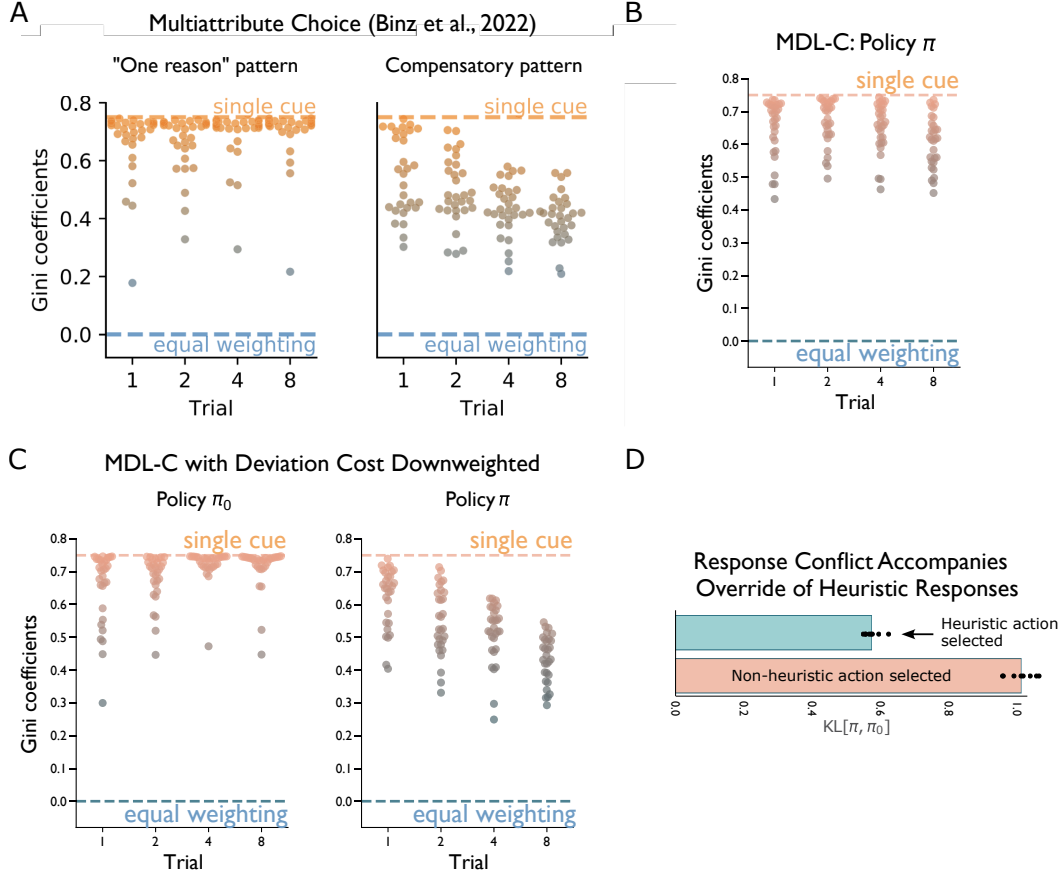
MDL-C captures the empirically observed effects of contingency degradation, but also offers a novel computational perspective. As shown in Fig. 5.4B, the speed with which the MDL-C agent reduces its response rate after contingency degradation depends on how long the agent was previously trained with reward (see Appendix 5.C for simulation

details). As in the experimental data, behavior becomes less flexible as the duration of training increases. This shift is a result of the MDL-C optimization objective. Policy  $\pi$  is initially able to adjust rapidly, responding to reward by emitting the rewarded action frequently. If contingency degradation occurs immediately,  $\pi$  is able to adapt flexibly. However, if reward continues for a longer period, the rewarded policy gradually comes to be mirrored in  $\pi_0$ , driven by the third term in Equation 2. Once  $\pi_0$  becomes strongly biased toward the rewarded action, it is difficult for policy  $\pi$  to diverge from this pattern, again due to the third term in Equation 2 (an effect that is attenuated if  $\pi_0$  is ablated, analogous to lesioning dorsolateral striatum; see Fig. 5.4B). This computational mechanism is related to others that have been proposed in models devised specifically to account for contingency degradation effects, based on uncertainty or habit strength (Daw et al., 2005; Miller et al., 2019) (see Supplementary Discussion). However, MDL-C ties the relevant learning dynamics to a higher-level computational objective, namely, minimizing the description length of behavior (compare Pezzulo et al. (2018); Lai and Gershman (2021)).

### 5.3.3 Simulation 3: Judgment and decision making

As noted earlier, dual-process models in JDM research distinguish between System-1 and System-2 strategies, the former implementing imprecise heuristic procedures, and the latter sounder but more computationally expensive analysis (Evans, 2008; Kahneman, 2011). As in the other dual-process domains we have considered, there appears to be a neuroanatomical dissociation in this case as well, with System-2 responses depending on prefrontal computations (Mével et al., 2019; De Neys and Goel, 2011).

Recent research on heuristics has increasingly focused on the hypothesis that they represent resource-rational approximations to rational choice (Lieder and Griffiths, 2020). In one especially relevant study, Binz et al. (2022) proposed that heuristic decision making arises from a process that “controls for how many bits are required to implement the emerging decision-making algorithm” (p. 8). This obviously comes close to the motivations behind MDL-C. Indeed, Binz et al. (2022) implement their theory in the form of a recurrent neural network, employing the same regularization that we apply to our  $RNN_{\pi_0}$ . They then proceed to show how the resulting model can account for heuristic use across several decision-making contexts. One heuristic they focus on,



**Figure 5.5:** **A.** Heuristic one-reason decision making (left) and richer compensatory decision making (right) in a multi-attribute choice task, from Binz et al. (2022). Gini coefficients, on the y axis, capture the degree to which decisions depend on one feature (higher values, with asymptotic maximum of one) versus all features evenly (zero), with references for one-reason decision making (*single cue*) and a fully compensatory strategy (*equal weighting*) indicated. Data points for each trial correspond to observations from separate simulation runs. Human participants in the study displayed both patterns of behavior, depending on the task conditions. **B.** Behavior of MDL-C in the task from Binz et al. (2022), under conditions where human participants displayed one-reason decision making. **C.** Behavior of  $\pi_0$  (left) and  $\pi$  (right) when the KL penalty for divergence between the two policies is reduced (see Methods). **D.** In the simulation from panel C, the divergence between policies is increased when the agent emits a non-heuristic decision.

called *one-reason decision making*, involves focusing on a single choice attribute to the exclusion of others (Newell and Shanks, 2003). As shown in Fig. 5.5A, reproduced from Binz et al. (2022), a description-length regularized network, trained under conditions where one-reason decision making is adaptive (see Binz et al. (2022) and Appendix 5.D), shows use of this heuristic in its behavior, as also seen in human participants performing

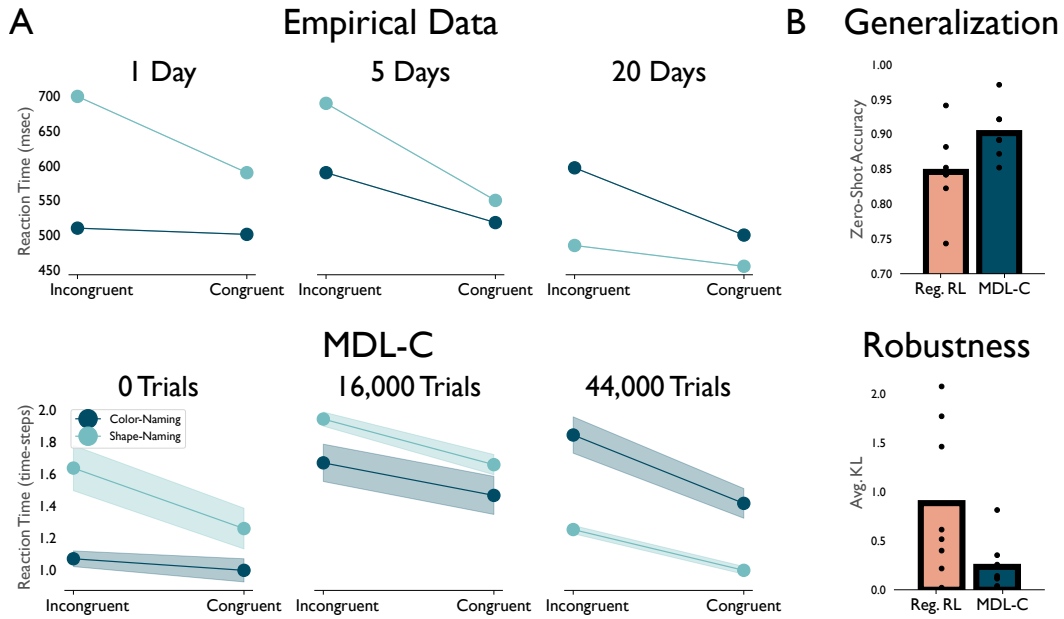
the same task. In contrast, an unregularized version of the same network implements a more accurate but also more expensive ‘compensatory’ strategy, weighing choice features more evenly.

As illustrated in Fig. 5.5B, when MDL-C is trained on the same task as the one used by Binz et al. (2022) (see Appendix 5.D), it displays precisely the same heuristic behavior those authors observed in their human experimental participants.

Digging deeper, MDL-C provides an explanation for some additional empirical phenomena that are not addressed by Binz et al. (2022) or, to the best of our knowledge, any other previous computational model. In an experimental study of one-reason decision making, Newell and Shanks (2003) observed that application of the heuristic varied depending on the available payoffs. Specifically, heuristic use declined with the relative cost of applying a compensatory strategy, taking more feature values into account. MDL-C shows the same effect. When the weighting of the deviation term  $D_{KL}(\pi||\pi_0)$  is reduced relative to the value-maximization term in the MDL-C objective (see Appendix 5.D), the policy  $\pi$  and thus the agent’s behavior take on a non-heuristic compensatory form (Fig. 5.5D). Critically, in this case MDL-C instantiates the non-heuristic policy side-by-side with the heuristic policy, which continues to appear at the level of  $\pi_0$ . This aligns with work suggesting that System-1 decision making can occur covertly even in cases where overt responding reflects a System-2 strategy. In particular, Mevel et al. (2019) observed activation in prefrontal areas associated with conflict detection in circumstances where a tempting heuristic response was successfully overridden by fuller reasoning (see also De Neys and Goel (2011)). A parallel effect is seen in our MDL-C agent in the degree of conflict (KL divergence) between policies  $\pi$  and  $\pi_0$  (Fig. 5.5D).

#### 5.3.4 Comparison with Previous Models

To our knowledge, no previous computational model has simultaneously captured the core dual-process phenomena we’ve considered, thereby bridging the domains of executive function, reward-based decision making and JDM. However, a range of previous models have addressed the relevant phenomena in a fashion limited to one of those domains. Having stressed the unifying, cross-disciplinary character of the present work, it is also befitting to consider the relationships between MDL-C and these domain-specific



**Figure 5.6: A.** Top: Behavioral data from the modified Stroop task studied by MacLeod and Dunbar (1988). Early in training, shape-naming responses were both slower than color-naming responses and more affected by stimulus congruence, consistent with shape-naming being the relatively ‘controlled’ response and color-naming relatively ‘automatic.’ With extensive training, the pattern flipped, with shape-naming becoming faster than color-naming and less affected by stimulus congruence. Bottom: Under training conditions mimicking the experimental study, MDL-C displayed a similar pattern of behavior, with a significant main effect of task and a significant interaction between task and trial-type ( $p < 0.05$ ) at both 0 trials and 44,000 trials. **B.** Zero-shot Stroop performance in MDL-C and an unregularized baseline model (see Methods). Top: Color-naming accuracy on incongruent Stroop stimuli, after training only with neutral stimuli (see main text and Methods). Bottom: KL divergence between action probability distributions under two conditions, (1) presentation of incongruent Stroop stimuli, and (2) presentation of Stroop stimuli with the word identity input masked out. MDL-C shows significantly lower divergence, indicating that the control policy attends less to the task-irrelevant factor — i.e., MDL-C is more robust to distractors — despite never having been trained on incongruent stimuli.

models. Particularly important is the question of whether such domain-specific models capture any empirical phenomena that MDL-C might have difficulty addressing.

In the area of executive control, our model bears strong connections with the classic connectionist model proposed by Miller and Cohen (2001). In particular, both characterize the distinction between controlled and automatic processing as arising from learning. To illustrate this point, Cohen and colleagues (1990) modeled results from a behavioral study by MacLeod and Dunbar (1988) (Fig. 5.6A). Here, participants were presented with

colored shapes, and asked either to name their color or to announce a color name that had been arbitrarily assigned to the relevant shape (e.g., a particular irregular pentagon might be given the name *blue*, independent of its display color). Interference between the two tasks was quantified by comparing response time on incongruent trials, where color- and shape-name conflicted, against congruent trials, where they matched. Early in training, interference was larger for the shape-naming task than the color-naming task, suggesting that color-naming was relatively ‘automatic’ and shape-naming relatively ‘controlled.’ However, after extensive training on the shape-naming task the pattern flipped, consistent with the idea that within-task learning had rendered shape-naming relatively ‘automatic.’ This effect was well captured by the neural network model of ?, and it also arises in our MDL-C model (see Fig. 5.6A Appendix 5.B).

As this example illustrates, gradual learning processes, operating over the course of extensive practice on relevant tasks, are important to the theoretical account we are proposing with MDL-C. On the face of it, this may seem to stand in tension with how learning occurs in most human behavioral experiments, where participants dive in on novel tasks given little more than some verbal instructions and few practice trials. For example, in the classic Stroop task, it seems reasonable to assume that participants have rarely if ever been asked to name the color of a word that itself names a color, but they do this ‘zero-shot,’ and immediately display Stroop interference. To show that our MDL-C implementation accommodates this kind of zero-shot learning, we trained our agent on color-naming and on word-reading, only ever presenting ‘neutral’ stimuli, omitting the word feature during color-naming and omitting the color feature during word-reading (see Appendix 5.B). At test, incongruent feature sets were presented. The model responded correctly on the vast majority of trials given the task-cue input — performing significantly better than an ablated network lacking MDL regularization — but also showed Stroop interference (see Fig. 5.6b). In recent work, Riveland and Pouget (2022) have shown how neural networks can follow verbal instructions zero-shot in a wider range of tasks. It would be exciting to expand our MDL-C implementation to incorporate greater behavioral breadth and flexibility in the same way.

Elaborations of the Miller and Cohen (2001) model have offered a mechanistic ex-

planation for the special role played by prefrontal cortex in representing aspects of context, attributing to prefrontal circuits a special set of gating-based memory mechanisms (O'Reilly et al., 2010). MDL-C offers a complementary account, instead addressing why it makes sense in normative terms for the brain to support both control and habit systems (see Musslick and Cohen (2021) for a related but domain-specific analysis). It is important to emphasize, however, that we are not attempting to claim that MDL-C's  $RNN_{\pi}$  and  $RNN_{\pi_0}$  map directly onto specific brain regions, but rather only that the split architecture of our MDL-C agents reflects evidence supporting neuroanatomical divisions between areas of controlled and automatic processing. As it turns out, however, MDL-C does in fact give rise to a solution that gates different information into different parts of the information-processing architecture, broadly consistent with gating-based models of cognitive control (O'Reilly et al., 2010). From the point of view of our theory, such gating mechanisms might be viewed as solutions to the MDL-C problem discovered by evolution rather than online learning. It is worth noting that some of the most recent work to apply the notion of gating to PFC function has postulated a multilevel hierarchy, deeper than the one we consider in our simulations. There is no practical impediment to extending the MDL-C architecture to include multiple hierarchical levels; a natural approach would be to regularize each pair of adjacent layers with respect to one another, varying the weight of the complexity cost monotonically across layers. We have not, however, implemented this idea and it therefore stands as an appealing opportunity for next-step research. Another elaboration of the Miller and Cohen (2001) model adds a 'cost of control,' a negative utility attached to the overriding of default response-selection processes (Shenhav et al., 2013; Zenon et al., 2019; Lieder et al., 2018; Piray and Daw, 2021). As noted in our simulation of demand avoidance, the deviation term in the MDL-C objective effectively imposes a cost of control, showing how this cost fits into a broader optimization process. While philosophically aligned, MDL-C differs from these models in important ways, most significantly in that its default policy is *learned*. That is, while the control policy may be learned using a similar objective (e.g., Piray and Daw (2021) also use KL-regularized policy optimization with respect to a default policy), MDL-C directly models the acquisition of automatic/habit-like behavior as the minimization of an MDL-based



objective, whereas most previous sequential decision-making approaches modeling a cost of control do so with respect to a fixed default policy.

The classic [Miller and Cohen \(2001\)](#) model has been elaborated in subsequent work to address another canonical phenomenon in the executive function literature, which we have not previously touched upon: task-switching costs (see, e.g., [Herd et al. \(2014\)](#), [Reynolds et al. \(2006\)](#), [Gilbert and Shallice \(2002\)](#)). Importantly, in order to capture switch-cost effects, including such phenomena as residual and asymmetric switch costs, the relevant computational models have had to build in temporally and mechanistically fine-grained accounts of working memory function, modeling attractor dynamics and hysteresis effects that fall well below the level of abstraction our MDL-C implementation occupies. It would be informative to implement MDL-C with an increased level of temporal granularity (as for example in [Herd et al. \(2014\)](#)) and to evaluate task-switching effects in this setting.

We turn now from executive function to reward-based decision making. As shown in Simulation 2, when MDL-C operates within an appropriate task context, it can yield side-by-side decision mechanisms with profiles matching model-based and model-free control. This links MDL-C with a wide range of recent models of reward-based decision making, which center on this side-by-side configuration ([Dolan and Dayan, 2013](#); [Daw et al., 2005](#); [Gläscher et al., 2010](#); [Beierholm et al., 2011](#)). As discussed under Results, the empirical data motivating those dual-system models is complex. In particular, neural activity aligning with model-free computations is not always ‘pure’ of model-based characteristics (see, e.g., [Daw et al. \(2011\)](#)). Such computational purity is not enforced in MDL-C, either, and under some parameterizations MDL-C displays the same intermediate patterns that have been observed in some experimental studies. (Indeed, such mixed patterns were seen across most of the parameter space we explored; see Figs. [5.7](#) to [5.9](#)). The interpretation of ostensibly model-based behavior in the two-step task is also nuanced ([Akam et al., 2015](#); [Miller et al., 2016a](#)). However, we have demonstrated elsewhere ([Wang et al., 2018](#)) that genuinely model-based computations can arise within recurrent neural networks under conditions comparable to those employed in the present work.

Beyond model-based and model-free RL, the dynamics of habit acquisition in



MDL-C also link it with recent models that replace model-free RL with a reward-independent, practice-based learning mechanism (Ashby et al., 2007; Miller et al., 2019; Bogacz, 2020). The learning mechanism of MDL-C’s default policy is closely related to these, with two important differences. The first is that the practice-based learning mechanisms adopt as the target of learning the discrete actions actually taken by the agent, while MDL-C’s default policy adopts as its target the full probabilistic control policy from which those actions are sampled. The second is that the addition of VDO effectively regulates the complexity of the habits that can be learned and the rate at which habit formation occurs. The results presented in Fig. 5.4A support this connection. Of particular interest, a recent study provided evidence that dopamine dynamics in a posterior sector of the striatum encode not a reward-prediction error, but instead an *action*-prediction error, which drives situation-action associations (Greenstreet et al., 2022). This aligns quite closely with how learning operates in  $RNN_{\pi_0}$  in our MDL-C implementation, where weight updates are driven by a mismatch between the actions predicted by  $\pi_0$  and those dictated by  $\pi$ .

Practice-based accounts of habits have been proposed (Miller et al., 2019) to explain not only classic assays of habits, but also trial-by-trial perseveration, an effect in which subjects tend to repeat in the future choices that have been made in the past, regardless of the associated stimuli and outcomes (Cho et al., 2002; Akaishi et al., 2014; Balcarras et al., 2016; Miller et al., 2018). To test whether MDL-C would show such effects, we ran it on a drifting two-armed bandit task, in which rats show robust perseveration (Miller et al., 2018). We find that MDL-C shows similar perseveration, while an ablation model lacking the default policy does not (Fig. 5.4A).

Despite all of these connections, MDL-C differs from most previous models in that it does not involve a direct competition between control systems (Daw et al., 2005; Lee et al., 2014). In MDL-C, the policy  $\pi$  always has the last word on action selection, which may be to either endorse or override default policy  $\pi_0$  (as discussed above). Interestingly, this relationship between systems resembles one proposal for the interplay between System 1 and System 2 in the JDM literature, according to which “System 1 quickly proposes intuitive answers to judgment problems as they arise, and System 2 monitors the quality

of these proposals, which it may endorse, correct or override” (Kahneman and Frederick, 2002).

Within the JDM literature, among computational models of heuristic judgment, our account aligns closely with the one recently proposed by Binz et al. (2022), adding to it in the ways noted earlier. Like Binz et al. (2022), we have only applied MDL-C to a small set of heuristics from among the many considered in the JDM literature. An important challenge, both for MDL-C and for the Binz et al. (2022) account, will be to test applicability to a wider range of the relevant behavioral phenomena. Needless to say, a still wider range of decision effects addressed by the JDM literature, from risk attitudes to self-control conflicts, remain untouched by the present introductory work, and the compatibility of the our theory with such effects will necessarily await further research.

Some readers will have remarked that the our account of dual-process control shares important characteristics with a range of research on ‘resource-rational’ cognition (Lieder and Griffiths, 2020), where limitations on computational capacity are understood to constrain strategies for adaptive information processing. In the context of goal pursuit, this perspective has given rise to the notion of a value-complexity tradeoff, where reward maximization balances against the cost of encoding or computing behavioral policies (Amir et al., 2020; Lai and Gershman, 2021; Binz et al., 2022; Tavoni et al., 2022). While our computational account resonates strongly with this set of ideas, two qualifying points call for consideration. First, a great deal depends on the exact nature of the computational bottleneck hypothesized. At the center of our account is a measure related to algorithmic complexity (Grünwald, 2007; Hinton and Van Camp, 1993; Binz et al., 2022), a measure that differs from the mutual information constraint that has provided the usual focus for value-complexity tradeoff theories (Lai and Gershman, 2021; Lerch and Sims, 2018) (see Appendix 5.A). Second and still more important, the MDL-C framework does not anchor on the assumption of fixed and insuperable resource restrictions. The relevant limitations on complexity are regarded not as inherent to neural computation, but rather as advantageous for representation learning and generalization (Chater and Vitányi, 2003). Indeed, while reward-complexity tradeoff models typically involve a single bottlenecked processing pathway (Binz et al., 2022; Lai and Gershman, 2021), MDL-C includes a sec-

ond pathway that allows the agent to work around constraints on computational capacity. This allows for the formation of expressive, task-specific representations alongside more compressed representations that capture shared structure across tasks (Musslick and Cohen, 2021).

## 5.4 Discussion

Dual-process structure appears ubiquitously across multiple domains of human decision making. Though this is almost certainly a simplification and action selection lies along a spectrum from controlled to automatic, this tradeoff has been a useful axis for studying behavior. While this has long been recognized by psychological and neuroscientific models, only recently has the normative question been raised: Can dual-process control be understood as solving some fundamental computational problem? MDL-C, an approach for efficient multitask RL from the machine learning literature, can be derived directly from the demands of generalization and sequential decision-making, without reference to neuroscientific data. Despite this independent theoretical lineage, MDL-C turns out to provide a compelling explanation for dual-process structure.

The account we have presented is also distinctive for its unifying character. Although sophisticated dual-process models have been proposed within each of the behavioral domains we have considered in the present work — executive control (e.g., Lieder et al. (2018)), reward-based decision making (e.g., Daw et al. (2005)), and JDM (e.g., Binz et al. (2022)) — to our knowledge MDL-C is the first computational proposal to account for empirical phenomena across all three of these fields. However, our treatment of the neuroscientific issues has, of necessity, been quite broad; important next steps for developing the theory would, for example, be to provide a more detailed account of MDL-C's relationship with specific neuroanatomical structures, particularly regional distinctions and hierarchical organization within prefrontal cortex (Badre and Nee, 2018). While we view MDL-C as a promising step in the direction of providing unified account of dual process phenomena across fields, deep questions remain and further work needs to be done.

Beyond psychology and neuroscience, MDL-C, with its origin in machine learn-

ing (Moskovitz et al., 2023a), bears a number of important links with existing work in that field. In particular, it belongs to a broad class of RL systems that employ regularized policy optimization, where the agent policy is regularized toward some reference or default (see (Tirumala et al., 2020a)). Most relevant are approaches where the default policy is itself learned from experience (Galashov et al., 2019b; Teh et al., 2017b; Goyal et al., 2020; Moskovitz et al., 2022a). In previous work involving such learning, it has been deemed necessary to stipulate an ‘information asymmetry,’ imposing some hand-crafted difference between the observations available to the control and default policies (Galashov et al., 2019b; Teh et al., 2017b; Piray and Daw, 2021; Goyal et al., 2020). MDL-C allows this information asymmetry itself to be learned, as our simulations have demonstrated. Given this point and others, we are hopeful that further insights gained into MDL-C’s relationship with biological cognition could spur modifications that provide benefits in a machine learning context as well.

## Appendix

### Appendix 5.A: Architecture and Learning Algorithm

All experiments employed a common architecture and learning algorithm with minor implementational variations across simulations. Our implementation of MDL-C consists of two recurrent neural networks (RNNs) which we call the *control policy network*  $RNN_\pi$  and the *default policy network*  $RNN_{\pi_0}$ . These RNNs have identical sizes and architectures. They are also provided with identical inputs for each time step of experience, consisting of a one-hot encoding of the previous action, a scalar indicating the previous reward, and a vector of task-specific information (the “observation”) which will be described separately for each task.

The control policy network  $RNN_\pi$  produces as output a vector of *policy* logits  $\pi$  which determine the probability of taking each available action, as well as a scalar *value* estimate of its expected future reward from the current state. It is trained using the *advantage actor-critic* (A2C; (Mnih et al., 2016)) algorithm, which encourages it to produce actions which maximize expected long-term reward  $\mathbb{E}_\pi[R]$ . The control policy network is also regularized using a term which encourages its action probabilities to match those produced by the default policy network. This is equivalent to encouraging the conditional description length  $L(\pi|\pi_0)$  to be low.

The default policy network also produces as output a vector of policy logits  $\pi_0$ . In the intact system, these are overwritten by the control policy network (see Supplementary Discussion), and so serve primarily to regularize the control policy. The default policy network is trained by *policy distillation* (Rusu et al., 2015; Hinton et al., 2015) to match the output of the control policy network  $\pi$ . It is regularized using *variational dropout* (VDO; (Kingma et al., 2015)), which encourages its absolute description length  $L(\pi_0)$  to be low.

The overall MDL-C objective for  $\pi$  and  $\pi_0$  can be written

$$\mathcal{L}_{\text{MDL-C}} = \mathbb{E}_\pi[R] - [\alpha D_{KL}(\pi(a|x_t; \theta) || \pi_0(a|x_t; w)) + \beta \bar{D}_{KL}(q(w; \phi) || p(w))],$$

corresponding to the reward maximization, complexity, and goodness-of-fit terms intro-

duced in Equation (2). Note that this expression introduces an additional weighting parameter relative to Equation (2), the rationale for which is presented in our Supplementary Discussion. Also, as will become clear in what follows, the overall objective above can be decomposed and sub-parts used to train different sectors of our agent, since only certain terms affect different pathways. Below, we describe each term in the objective in detail.

**Control Policy Network** Unless otherwise noted, the control policy  $RNN_\pi$  was trained via a modification of Advantage Actor-Critic (A2C), which is described in detail in Wang et al. (2018); Mnih et al. (2016). Briefly, A2C is an on-policy actor-critic algorithm which weights gradients by a Monte-Carlo estimate of the advantage at each time step. In order to prevent premature convergence to suboptimal local maxima, Mnih et al. (2016) add an entropy bonus to the objective to prevent the policy from becoming overly deterministic early in training. In MDL-C, entropy regularization is replaced with a Kullback-Leibler (KL) divergence penalty with respect to the default policy distribution  $\pi_0$ :

$$\begin{aligned}\nabla \mathcal{L}_\pi &= \nabla \mathcal{L}_{A2C} + \alpha \nabla \mathcal{L}_{KL}, \quad \text{where} \\ \mathcal{L}_{A2C} &= -\delta_t(x_t; \theta_v) \log \pi(a_t|x_t; \theta) + \frac{\alpha_v}{2} \delta_t(x_t; \theta_v)^2, \\ \mathcal{L}_{KL} &= D_{KL}(\pi(a|x_t; \theta) || \pi_0(a|x_t; w)), \\ \delta_t(x_t; \theta_v) &= R_t - v(x_t; \theta_v), \\ R_t &= \sum_{i=1}^{k-1} \gamma^i r_{t+i} + \gamma^k v(s_{t+k}; \theta_v),\end{aligned}$$

where  $x_t = [s_t, a_{t-1}, r_{t-1}]^\top$  is the observation vector at time  $t$  consisting of the state  $s_t$ , previous action  $a_{t-1}$ , and previous reward  $r_{t-1}$ ,  $\theta$  are the control policy parameters,  $\alpha_v$  is a hyperparameter controlling the weight on the value-learning loss,  $\theta_v$  are the value function parameters,  $D_{KL}(q||p) = \sum_a q(a) \log \frac{q(a)}{p(a)}$  is the KL divergence between distributions  $q$  and  $p$ ,  $w$  are the sampled parameters of the default policy network (details below), and  $\gamma$  is a scalar discount factor. This KL-regularized RL framework has a rich theoretical and experimental history in both machine learning and neuroscience (Piray

and Daw, 2021; Todorov, 2007), and can be derived (depending on the direction of the KL cost) through an interpretation of RL as Bayesian inference (Levine, 2018b). Moreover, it has been shown that this approach guarantees accelerated convergence compared to non-regularized methods under the condition that the tasks faced by the agent induce optimal policies which behave similarly (Moskovitz et al., 2022a). Intuitively, the control policy network is trained to maximize reward while simultaneously being encouraged to remain close to the behavior encoded by the default policy  $\pi_0$ . Early in training, obtaining reward may be challenging, and so the control policy network is primarily taught via learning signals generated by the default policy network. If the default policy network encodes behavior that is useful for the task, then learning from it may enable the control policy network to obtain reward earlier, accelerating training. In multitask settings where  $\pi_0$  is conserved across tasks, it is therefore important that  $\pi_0$  encodes behavior which is generally useful for the tasks faced by the agent. For further detail on KL-regularized multitask policy optimization, we refer readers to (Moskovitz et al., 2022a, 2023a).

**Default Policy Network** The default policy was trained off-policy via distillation (Hinton et al., 2015) from the control policy network (in other words, the default policy aims to match the control policy distribution) offset by a regularization penalty on the effective bit length encoding of the network parameters:

$$\begin{aligned}\nabla \mathcal{L}_{\pi_0} &= \nabla \mathcal{L}_{\text{distill}} + \nabla \mathcal{L}_{\text{VDO}} \\ &= \sum_{k=1}^M \nabla_{\phi} D_{KL}(\pi(a|x_k; \theta) || \pi_0(a|x_k; w = f(\phi; \epsilon))) + \beta \nabla_{\phi} \bar{D}_{KL}(q(w; \phi) || p(w)); \\ f(\phi^{(i)}; \epsilon) &= \phi_0^{(i)} (1 + \sqrt{\phi_{\alpha}^{(i)}} \epsilon^{(i)}); \quad \epsilon^{(i)} \sim \mathcal{N}(0, 1),\end{aligned}$$

where  $M$  is the minibatch size of data sampled from an experience replay buffer (Mnih et al., 2015),  $w$  are default policy parameters sampled using the reparameterization trick (Kingma et al., 2015) to allow for automatic differentiation,  $\beta$  is a scalar hyperparameter weighting the regularization, and  $\phi = \{\phi_0, \phi_{\alpha}\}$  are learned parameters defining the distribution over default policy parameters:  $q(w; \phi) = \prod_i \mathcal{N}(w^{(i)}; \phi_0^{(i)}, \phi_{\alpha}^{(i)} (\phi_0^{(i)})^2)$ , where the superscript  $(i)$  denotes the  $i$ th parameter, and  $p(w)$  is the log-uniform prior

$p(|w^{(i)}|) \propto 1/|w^{(i)}|$ . The noise  $\epsilon$ —and therefore, effectively, the default policy weights  $w$ —are re-sampled after every episode of training. It’s this particular form of noise which limits the effective capacity of  $RNN_{\pi_0}$ . We use the average KL,

$$\bar{D}_{KL}(q(w; \phi) || p(w)) = \frac{1}{N} \sum_{i=1}^N D_{KL}(q(w^{(i)}; \phi^{(i)}) || p(w^{(i)})).$$

The regularization loss is computed and minimized using *variational dropout* (VDO; (Molchanov et al., 2017; Kingma et al., 2015)), which uses a local reparameterization trick to implement this KL regularization as a particular form of multiplicative noise placed on the network weights. Regularizing with respect to this choice of prior has the effect of limiting the effective bit-length of the parameters of  $RNN_{\pi_0}$ , reducing its effective complexity (Kingma et al., 2015). Note that the distillation loss  $D_{KL}(\pi(a|x_k; \theta) || \pi_0(a|x_k; w = f(\phi; \epsilon)))$  is a direct measure of goodness-of-fit  $L(\pi, \pi_0)$  (Equation 2)—the degree to which  $\pi_0$  is able to match the behavior of  $\pi$ . To see this, note that minimizing this KL is equivalent to performing maximum likelihood estimation for  $\pi_0$  with  $\pi$  defining the ‘true’ underlying data distribution. Also observe that  $RNN_{\pi}$  is trained on-policy and  $RNN_{\pi_0}$  is trained off-policy from a buffer of experience collected by the control policy. We can view this as the control policy actively learning via trial and error in the world, while intuitively, within a multitask context, the default policy is trained to capture the behavior of the control policy on each task. The default policy thereby learns an ‘average’ of the behaviors required to perform well on each task. However, when only a few tasks have been observed, the default policy can ‘overfit’ to the behaviors learned on those initial tasks. This can be problematic, as if future tasks differ, the default policy could misguide the learning process for the control policy (see above). The VDO complexity regularization forces the default policy network to simplify, preventing overfitting and facilitating generalization.



**Architecture Details** The LSTM dynamics were governed by the following standard equations:

$$\begin{aligned}
 i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \\
 f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \\
 o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \\
 c_t &= f_t \odot c_{t-1} + i_t \odot \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
 h_t &= o_t \odot \tanh(c_t),
 \end{aligned}$$

where  $i_t, f_t, o_t, c_t, h_t$  are the input gate, forget gate, output gate, cell state, and hidden state at time  $t$ , respectively,  $\sigma(x) = 1/(1 + \exp(-x))$  is the sigmoid function, and  $\odot$  denotes element-wise multiplication. In order to assess the degree to which the default policy learned to ignore certain input features, an element-wise gating layer  $\ell(x)$  was applied to the input:

$$\ell(x_t) = \sigma(\tau\omega) \odot x_t,$$

where  $\tau$  was a hyperparameter fixed across simulations and  $\omega$  was a learned vector of parameters with dimension equal to that of the input. As  $\omega_d \rightarrow \infty$ , the  $d$ th input feature is passed on to the layers above, while if  $\omega_d \rightarrow -\infty$ , the  $d$ th input feature is gated out. Importantly, gradients from the VDO loss (see above) did not flow into  $\omega$ , only those from the distillation loss, so  $\omega$  learned to gate features in or out that were already either being used or ignored by the network, rather than simply being ablated directly as the VDO penalty increased. We found that adding this gating layer did not affect the performance of the agent.

**Training Details** In all simulations, the agent was updated using a learning rate of  $\eta = 0.0007$ , a value function loss weight of  $\alpha_v = 0.05$ , a policy KL weight of  $\alpha = 0.1$ , a discount factor of  $\gamma = 0.9$ , a gating layer coefficient of  $\tau = 150$ , and a VDO KL weight of  $\beta = 1.0$  unless otherwise noted. All gradient updates were performed using the Adam optimizer (Kingma and Ba, 2014). Further simulation-specific details can be found below,

and a summary of hyperparameter values for each task is provided in Table 5.1. All results were obtained by averaging over 8 random seeds, with shading on line plots denoting one unit of standard error.

Task	$(\eta, \alpha_v, \alpha, \beta, \gamma, M, \tau, \# \text{ hidden}, \pi_0 \text{ buffer size})$
Navigation	$(7\text{e-}4, 0.05, 0.1, 1.0, 0.9, 32, 150, 48, 1\text{e}5)$
Stroop	$(7\text{e-}4, 0.05, 0.1, 1.0, 0.9, 1, 150, 48, 1\text{e}5)$
Demand Stroop	$(7\text{e-}4, 0.05, 0.1, 1.0, 0.9, 1, 150, 48, 1\text{e}5)$
Two-Step	$(7\text{e-}4, 0.05, 0.1, 3.0/100.0^*, 0.9, \text{N/A}, 150, 48, \text{N/A})$
O & D	$(7\text{e-}4, 0.05, 0.1, 1.0, \text{N/A}, 1, 150, 5, 1\text{e}3)$
Continuous Control	$(3\text{e-}4, 0.05, 0.1, 1.0, 0.99, 128, 150, 256, 1\text{e}6)$
Heuristics	$(3\text{e-}4, \text{N/A}, 0.1, 1.0, \text{N/A}, 32, 150, 128, 1\text{e}6)$

**Table 5.1:** Hyperparameters for each task. \*The Two-Step task settings are described in greater detail below.

## Appendix 5.B: Simulation 1: Executive control

**Stroop Task** In the Stroop task, the agent must perform either “word-reading” (WR) or “color-naming” (CN) tasks across two different colors and two different words, totalling eight different possible stimuli: (red[WR], blue[WR], blue[WR], red[WR], red[CN], blue[CN], blue[CN], red[CN]), each presented to the agent as a three-dimensional vector  $x_t = [\text{color}, \text{word}, \text{task}]$  with the following encodings: blue  $\rightarrow -1$ , red  $\rightarrow +1$ , CN  $\rightarrow -1$ , WR  $\rightarrow +1$ . The presentation frequencies were 20% for all WR stimuli and 5% for all CN stimuli. There were two possible actions, corresponding to  $-1$  and  $+1$ . The agent received  $+1$  reward when its action matched the appropriate value of the stimulus feature (e.g., if the task feature is  $-1$  and the color feature is  $+1$ , the desired action is  $+1$ ) and zero otherwise. In order to simulate reaction times (RTs), the input stimulus for a given trial was re-presented up to a maximum of 5 times to the agent until the entropy of the control policy  $H[\pi] = -\sum_a \pi(a|x_t; \theta) \log \pi(a|x_t; \theta)$  dropped below a threshold  $b = 0.5$ , similar to the approach to modeling RTs used by Cohen et al. (1990); Song et al. (2016). The RT for each trial was the number of presentations of the stimulus until a response was produced. After a response was generated, the trial ended. The agent was trained for 15,000 trials and each LSTM had 48 hidden units.

**Demand Avoidance Stroop Task** In this task, the agent was presented with word-reading (WR) and color-naming (CN) trials, encoded as in the Stroop task described above, with each WR stimulus having a 20% chance of presentation on any given trial. CN trials were presented 20% of the time, but in this task CN trials consisted of two time steps. On the first time step, the agent was presented with the stimulus  $[0, 0, -1]$  to indicate a CN trial. Its action at this stage served to select between two categories, referred to as *high-demand* and *low-demand*. If the agent selected the high-demand category, then in the second time step of the trial, a conflict stimulus was presented with a 90% chance and a congruent stimulus with a 10% chance. If the agent selected the low-demand category, these probabilities were reversed. The agent shared the same settings as in the main Stroop task, and was also trained for 15,000 trials.

**Interference Stroop Task** In this case, the agent was first pre-trained on color-naming-only trials for 30,000 trials. It was then trained for 45,000 trials on word-reading. During this training phase, the agent’s reaction time was evaluated on both CN and WR trials (in evaluation trials, the agent’s weights were not updated). The agent architecture and environment set-up were the same as in the “standard” Stroop task above. One important characteristic of the Stroop task is that there are interaction effects between task and congruity—that is, at the outset of training, color interferes with shape more than shape interferes with color. This means that shape-naming conflict trials result in disproportionately higher RTs compared to shape-naming congruent trials compared to the difference between color-naming conflict trials and color-naming congruent trials. This relationship is then reversed at the end of training, with a greater relative increase for color-naming trials. We verified that this property held for MDL-C by running a one-way ANOVA test on the differences in RT between each trial type (e.g., color-naming conflict RT - color-naming congruent RT vs. shape-naming conflict RT - shape-naming congruent RT), with the interaction effect at 0 trials yielding  $F = 11.3$  and  $p = 0.005$  and the interaction effect at 44,000 trials yielding  $F = 13.7$  and  $p = 0.002$ .

**Zero-Shot Stroop Task** The agent was trained 8,000 trials in which the unneeded feature was zeroed out from the stimulus (i.e., the agent gets 3d inputs [color, word, task id], where -1 = blue, +1 = red in the first two dims, -1 = color-naming, +1 = word-naming

for task id, and o = NULL in any location). So, the agent would see [-1, o, -1] for a "blue" color-naming task. The stimulus distribution was uniform. During training, both MDL-C and 'Regular RL' (just a control policy  $RNN_\pi$  with otherwise identical hyperparameters) get to 100% accuracy. The agent is then evaluated on 100 trials with fixed weights in which the unneeded feature is included in the stimuli. The evaluation performance is the percent correct over 100 evaluation trials with fixed weights. To test the hypothesis that the improved performance of MDL-C is rooted in robustness to changes in inputs, we also measured the average KL between the policy distributions for each approach on masked inputs (like the ones on which they were trained) and on inputs with the missing feature included. Regular RL had a greater difference, indicating that responses were more effected by the out-of-distribution inputs.

## Appendix 5.C: Simulation 2: Reward-based learning

**Two-Step Task** We use a variant of the two-step task based on the one used by Wang et al. (2018) in which transition contingencies—in addition to reward contingencies—may switch. The task was changed in this way following the finding by Akam et al. (2015) that when transition contingencies are fixed, a habit-like strategy in which second stage states which have recently yielded reward are directly mapped to actions in the choice stage can develop which closely matches the pattern of behavior expected of agents using planning. Additionally, the agent is provided with an input feature which indicates which transition contingency setting is currently active (an ingredient added to the task from Akam et al. (2015) in order to restore the property that model-based and -free strategies yield the classical patterns shown in Fig. 5.3C). To use this feature to inform its actions, the agent must compute a higher-complexity policy than if this feature is ignored, analogous to the difference between classifying inputs according to XOR versus OR logic. To be more precise, with two second stage states  $A$  and  $B$  and two actions  $a_L$  and  $a_R$ , we can have either

$$\text{Setting } o = \begin{cases} p(A|a_L) = 0.8, & p(B|a_L) = 0.2 \\ p(A|a_R) = 0.2, & p(B|a_R) = 0.8 \end{cases}$$

$$\text{Setting 1} = \begin{cases} p(A|a_L) = 0.2, & p(B|a_L) = 0.8 \\ p(A|a_R) = 0.8, & p(B|a_R) = 0.2 \end{cases}$$

In other words, in one setting  $a_L$  is likely to lead to  $A$  and  $a_R$  is likely to lead to  $B$ , and in the other, the reverse is true. The agent is shown a binary feature which indicates which transition setting the environment is in (however, it has to learn what this feature means through experience). More precisely, the state observation at each time step  $s_t$  is a 5-dimensional vector, with the first four dimensions comprising a one-hot encoding of the current position of the agent within the task (either `fixation stage`, `choice stage`,  $A$ , or  $B$ ), with the final dimension a binary encoding of the current transition setting. The agent has three possible actions:  $a_L$ ,  $a_R$ , and  $a_{fixate}$ , which the agent is required to produce in order to progress from the fixation stage to the choice stage. There are also two possible settings for the reward contingencies, with either  $A$  or  $B$  having a 90% chance of leading to reward, with the other state in either contingency having a 10% chance. The agent is trained for 16,000 episodes, where each episode consists of 100 trials. At the end of each episode, the agent networks' hidden states are reset and an update is performed via backprop. On any given trial, there is a 2.5% chance that the reward contingency switches and a 5% chance that the transition contingency changes. During training, we found it helpful to start with a 0% chance of reward contingency switches and linearly increase the probability to 2.5% over the first 2,000 episodes, as this helped the agent reliably learn the meaning of the transition setting feature. All other task settings and analysis details for stay probabilities and logistic regression are the same as in Wang et al. (2018). Importantly, in this task the default policy was trained online (but still off-policy) via full trajectories collected by the control policy, rather than via a buffer of  $(s, a, r, s')$  tuples. This is because the full episode history is required to effectively meta-learn, as demonstrated by Wang et al. (2018, 2017). The hyperparameter settings used to generate the plots in Fig. 5.3(D-F) were identified after an initial grid search with eight random seeds per  $(\alpha, \beta, \text{RewardScale})$  tuple with  $\alpha \in \{0.05, 0.1, 0.2\}$ ,  $\beta \in \{0.1, 1.0, 3.0, 5.0, 10.0, 100.0\}$ ,  $\text{RewardScale} \in \{0.5, 0.75, 1.0\}$  and further confirmed by an additional eight random seeds, for a total of 16. The 'classic' MB-MF

patterns were obtained with (0.1, 100.0, 1.0) and the mixed patterns were observed with (0.2, 3.0, 0.75). To further support the mixed MB-MF-ness of the response pattern in Fig. 5.3F, we performed Wilcoxon signed-rank tests between the average of the rewarded, common and unrewarded, uncommon responses and the average of the rewarded, uncommon and unrewarded, common responses as a measure of model based-ness, and between rewarded, common and rewarded, uncommon responses and unrewarded, common and unrewarded, uncommon responses as a measure of model free-ness. The response patterns for both the control and default policies were statistically significant ( $p = 0.012$ ) for both model-based and model-free behavior. The agent’s LSTMs had 48 hidden units each.

**Perseveration** In this experiment, the agent was trained on the drifting two-armed bandit task from Miller et al. (2018). In this task, trials consist of a single time-step in which the agent has two possible actions, with the probability of reward for each arm evolving with a Gaussian random walk. Specifically, if the probability of being rewarded by choosing a given action on trial  $t$  is  $P_t$ , then the probability of being rewarded for choosing that arm on the next trial is sampled from the distribution  $P_{t+1} \sim \mathcal{N}(P_t, 0.15^2)$ . The agent either receives a reward of 1 or 0, and is trained for 3,000 trials. In this case, each RNN had 5 hidden units. After training, logistic regression is performed to predict the agent’s behavior on a given trial, with the regressors being the choice made at each time-step ( $\pm 1$ , the whether a reward was given at each time-step ( $\pm 1$ ), and their product. A high regression weight for previous choices indicates a tendency to persevere, a high weight for the outcome/reward indicates that the agent is influenced by whether it was rewarded at each step independent of its previous choices, and a high regression weight for their product indicates that the agent is influenced by choices that led to rewards (reward-seeking behavior).

**Omission and Contingency Degradation** We use the same task set-up as Miller et al. (2019). As in Miller et al. (2019), in order to model the effect of overtraining on the agent’s sensitivity to omission of reward, the agent was first trained on a two-armed bandit task in which action 1 (“lever press”) led to a reward of 1 with 50% probability and action 2 (“leisure”) resulted in a reward of 0.1 100% of the time. It was then trained for 750 trials

on a modification of the task in which reward was never delivered for lever pressing and in which leisure resulted in a reward of 0.1 half the time and a reward of 1.1 half the time. The agent’s lever-pressing probability  $P(\text{lever press})$  was then measured at the end of the second training phase. This probability was plotted against the number of trials  $T$  for which the agent was trained on the first phase, where  $T \in [100, 200, 300, \dots, 2000]$ . The contingency degradation variant of this task was exactly the same, except that the leisure action always resulted in a reward of 0.1 in the second phase.

## Appendix 5.D: Simulation 3: Judgment and decision-making

**Heuristics** We use the same experimental setting as [Binz et al. \(2022\)](#). Briefly, the agent is meta-trained on a series of randomly generated paired comparison tasks with continuous input features  $x$  in which it must predict which of two presented feature vectors  $x_t = (x_t^A, x_t^B)$  is associated with a higher value of an unobserved scalar criterion  $y_t = (y_t^A, y_t^B)$ . More precisely, for each task  $i$ , there is an underlying linear relationship between features and the unobserved criterion:

$$\begin{aligned} y_{t,A} &= w_i^\top x_{t,A} + \epsilon_{t,A}; \\ y_{t,B} &= w_i^\top x_{t,B} + \epsilon_{t,B}, \end{aligned}$$

where  $\epsilon_{t,A}, \epsilon_{t,B} \sim \mathcal{N}(0, \sigma^2)$ , with  $\sigma^2$  a fixed variance and  $w_i \in \mathbb{R}^4$ . An ideal observer model then expresses the probability that  $y_A > y_B$  as

$$p(y_A > y_B | x, w_i) = p(C = 1 | x, w_i) = \Phi\left(\frac{w_i^\top x}{\sqrt{2}\sigma}\right), \quad (5.1)$$

where  $\Phi(\cdot)$  is the cumulative distribution function of a standard Gaussian distribution and  $C \in \{0, 1\}$  is a binary random variable which evaluates to 1 when  $y_A > y_B$  and 0 otherwise. Task feature weights  $w_i$  are randomly generated from a standard normal distribution, and the agent is meta-trained to estimate a posterior distribution over  $w$  with minibatches of 32 tasks and each task being presented to the agent for 10 trials. The reward for a given trial can be modeled as the log likelihood:  $p(C_t | x_t, \phi_t)$ . For a more

detailed description of the training process, see [Binz et al. \(2022\)](#). The control network  $RNN_\pi$  produces the parameters (mean  $\mu_t$  and variance  $\Psi_t$ ) of an approximate Gaussian posterior over  $w_i$ , which is then integrated into a predictive distribution for classification:

$$p(C_{t+1}|x_{t+1}, \phi_t, \Theta) = \int p(C_{t+1}|x_{t+1}, w)q(w; \phi_t) dw$$

where and  $\phi_t = \{\mu_t, \Psi_t\}$  are the parameters of the approximate Gaussian posterior  $q$  over parameters  $w$ . The conditional distribution is as above

$$p(C_{t+1} = 1|x_{t+1}, w) = \Phi\left(\frac{w^\top x_t}{\sqrt{2\sigma}}\right).$$

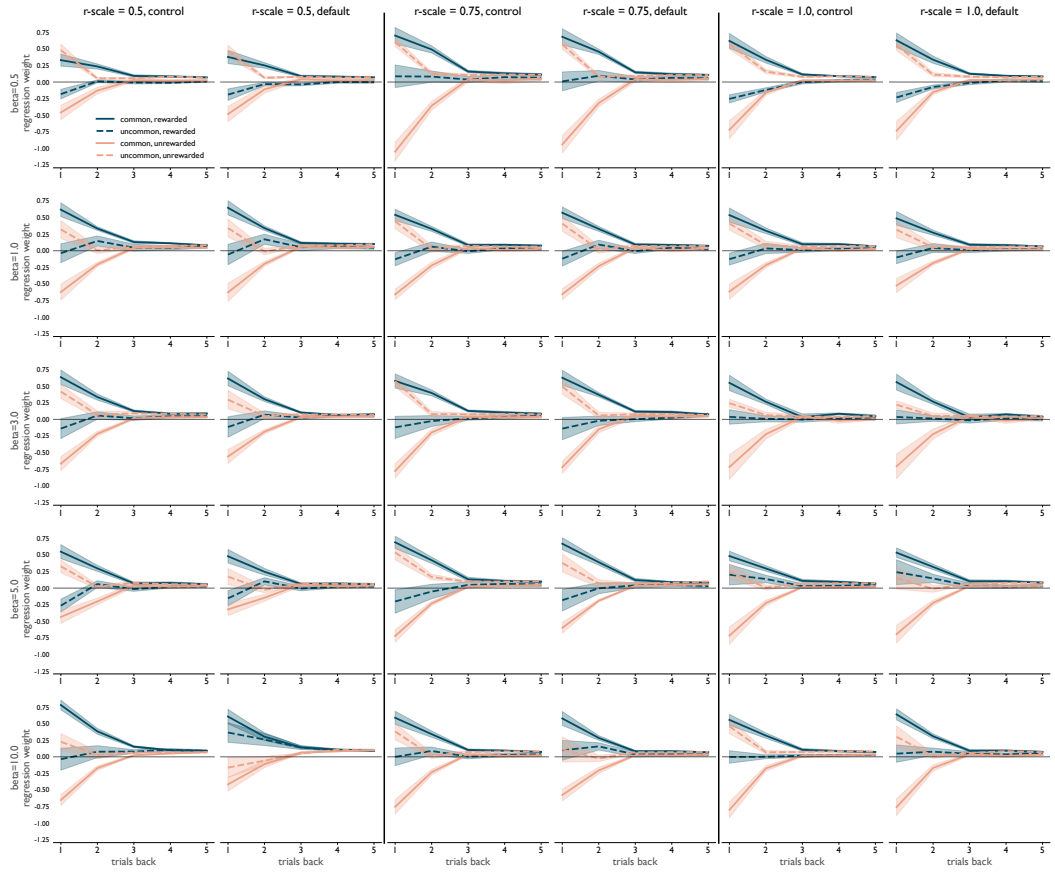
The default network  $RNN_{\pi_0}$  also produces parameters  $\phi_t^0$  of a Gaussian  $q_0$  and is trained to minimize  $D_{KL}(q(w, \phi_t)||q_0(w, \phi_t^0))$  in addition to the VDO complexity KL weighted by  $\beta$  (see “Default Policy” details above).

To test the emergence of heuristics, we use the task variant from [Binz et al. \(2022\)](#) in which there is a known ranking of input features, which classically induces a form of one-reason decision-making termed “take the best” (TTB), wherein subjects make decisions based on the top-ranked feature which differs between two inputs. To measure the emergence of such a heuristic in artificial agents, [Binz et al. \(2022\)](#) use the *Gini coefficient*  $G$  ([Atkinson, 2008](#)) measured over the feature weights  $w$ , defined below:

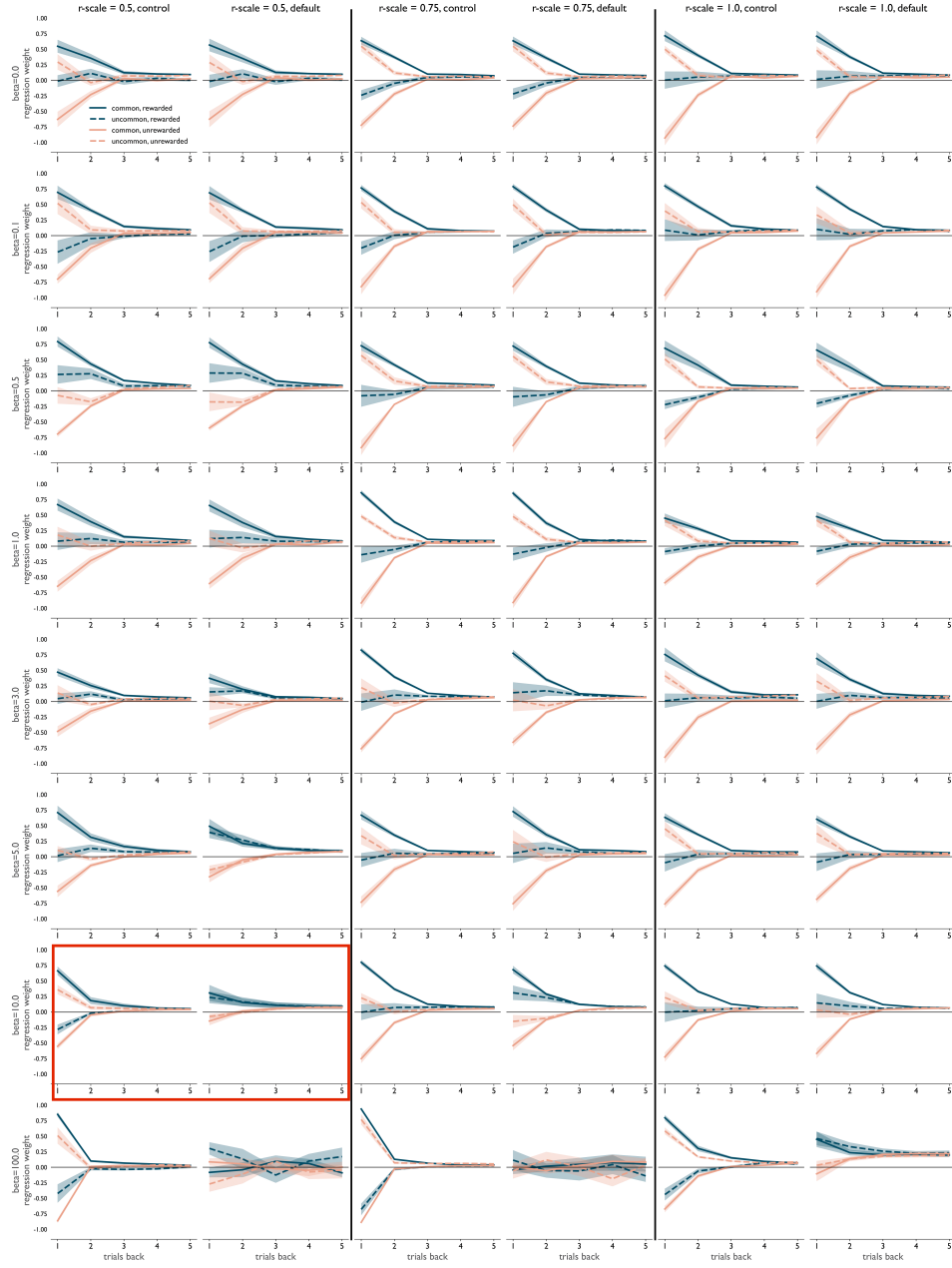
$$G(w) = \frac{\sum_{i=1}^d \sum_{j=1}^d |w_i - w_j|}{2d \sum_{i=1}^d w_i}.$$

The Gini coefficient can be thought of as a measure of inequality among feature weightings, so that it tends to 1 when one feature grows in importance compared to the others, and tends to 0 as all feature weights  $w_i$  converge to the same value. As a means of probing the effect of reducing the relative cost of employing a compensatory strategy (Fig. [5.5D](#)), we reduced the weighting on the KL between the default and control policies, setting  $\alpha = 0.01$ . This effectively lowers the penalty for deviation in behavior from the capacity-limited policy.

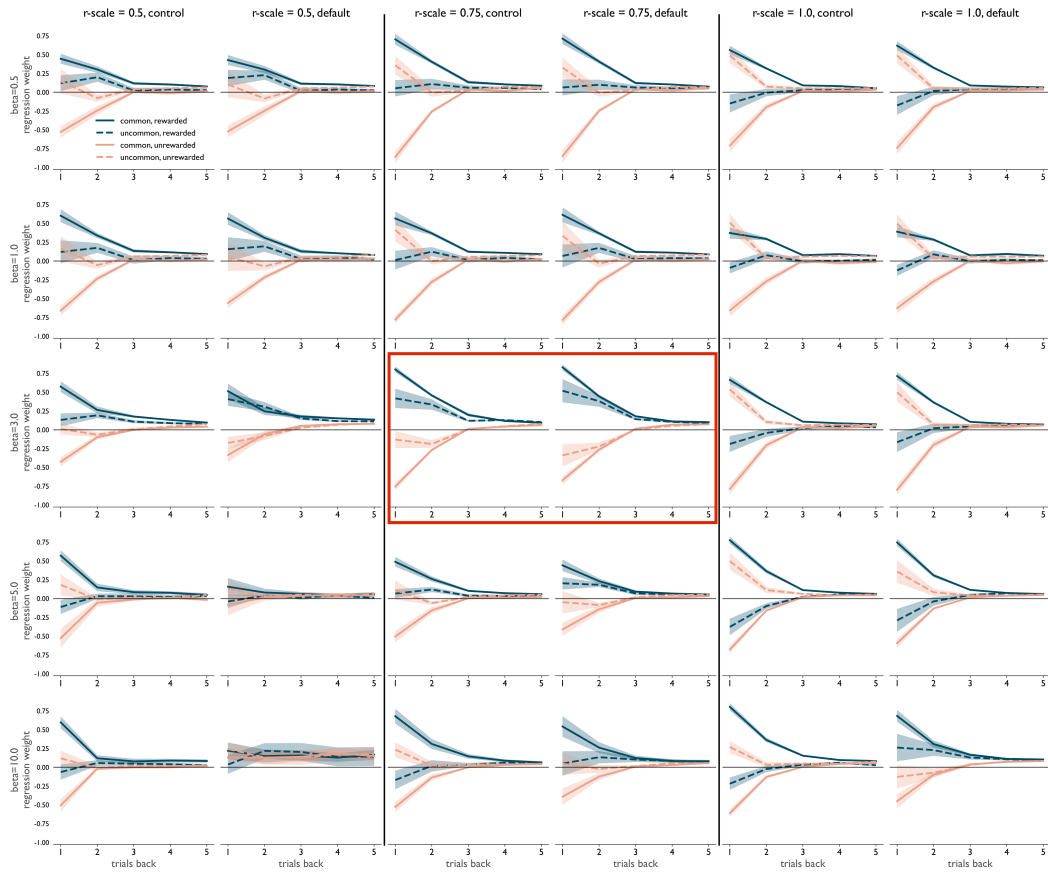




**Figure 5.7:** Two-step results from full hyperparameter sweep described in Methods, with  $\alpha = 0.05$ . Format as in Fig. 5.3 in the main text.



**Figure 5.8:** Two-step results from full hyperparameter sweep described in Methods, with  $\alpha = 0.1$ . The boxed plot appears in Fig. 5.3 in the main text. Format as in Fig. 5.3 in the main text.



**Figure 5.9:** Two-step results from full hyperparameter sweep described in Methods, with  $\alpha = 0.2$ . The boxed plot appears in Fig. 5.3E in the main text. Format as in Fig. 5.3 in the main text.

## **Part II**

# **Policy Composition Through Shared Dynamics**

This section focuses on a different variety of assumed structure: consistent transition dynamics across tasks. That is, given an agent's action, the world changes in the same way (in expectation) regardless of the particular reward the agent is pursuing. Previous work has shown that under this assumption, an agent can learn a representation which allows it to evaluate and compose previously learned policies in standard MDPs. The next two chapters extend these results to settings where reward availability obeys particular forms of nature-inspired non-stationarity and non-Markovianity.

## Chapter 6

# A First-Occupancy Representation for Reinforcement Learning

### 6.1 Introduction

In order to maximize reward, both animals and machines must quickly make decisions in uncertain environments with rapidly changing reward structure. Often, the strategies these agents employ are categorized as either model-free (MF) or model-based (MB) (Sutton and Barto, 2018a). In the former, the optimal action in each state is identified through trial and error, with propagation of learnt value from state to state. By contrast, the latter depends on the acquisition of a map-like representation of the environment's transition structure, from which an optimal course of action may be derived.

This dichotomy has motivated a search for intermediate models which cache information about environmental structure, and so enable efficient but flexible planning. One such approach, based on the *successor representation* (SR) (Dayan, 1993), has been the subject of recent interest in the context of both biological (Stachenfeld et al., 2017; Moennejad et al., 2017; Vértés and Sahani, 2019; Behrens et al., 2018; Gershman, 2020) and machine (Kulkarni et al., 2016; Barreto et al., 2017; Machado et al., 2020; Ma et al., 2020; Madarasz and Behrens, 2019; Barreto et al., 2020) learning. The SR associates with each state and policy of action a measure of the expected rate of future occupancy of all states if that policy were to be followed indefinitely. This cached representation can be acquired through experience in much the same way as MF methods and provides some of the flex-

ibility of MB behaviour at reduced computational cost. Importantly, the SR makes it possible to rapidly evaluate the expected return of each available policy in an otherwise unchanging environment, provided that the transition distribution remains consistent.

However, these requirements limit the applicability of the SR. In the real world, rewards are frequently non-Markovian. They may be depleted by consumption, frequently only being available on the first entry to a state. Internal goals for control—say, to pick up a particular object—need to be achieved as rapidly as possible, but only once at a time.

Furthermore, while a collection of SRs for different policies makes it possible to select the best amongst them, or to improve upon them all by considering the best immediate policy-dependent state values (Barreto et al., 2020), this capacity still falls far short of the power of planning within complete models of the environment.

Here, we propose a different form of representation in which the information cached is appropriate for achieving ephemeral rewards and for planning complex combinations of policies. Both features arise from considering the expected time at which other states will be *first* accessed by following the available policies. We refer to this as a *first-occupancy representation* (FR). The shift from expected rate of future occupancy (SR) to delay to first occupancy makes it possible to handle settings where the underlying environment remains stationary, but reward availability is not Markovian. This chapter, adapted from Moskovitz et al. (2022c), formally introduces the FR and to highlight the breadth of settings in which it offers a compelling alternative to the SR, including, but not limited to: exploration, unsupervised RL, planning, and modeling animal behavior.

## 6.2 Reinforcement Learning Preliminaries

**Policy evaluation and improvement** We begin by modeling a task  $T$  as a finite MDP,  $T = (\mathcal{S}, \mathcal{A}, P, r, \gamma, \rho)$ , where  $\mathcal{S}$  is a finite state space,  $\mathcal{A}$  is a finite action space,  $P : \mathcal{S} \times \mathcal{A} \rightarrow \Delta(\mathcal{S})$  is the transition distribution (where  $\Delta(\mathcal{S})$  is the probability simplex over  $\mathcal{S}$ ),  $r : \mathcal{S} \rightarrow \mathbb{R}$  is the reward function,  $\gamma \in [0, 1)$  is a discount factor, and  $\rho \in \Delta(\mathcal{S})$  is the distribution over initial states.

The goal of the agent is to maximize its expected *return*, or discounted cumulative reward  $\sum_t \gamma^t r(s_t)$ . To simplify notation, we will frequently write  $r(s_t) \triangleq r_t$  and  $\mathbf{r} \in$

$\mathbb{R}^{|\mathcal{S}|}$  as the vector of rewards for each state. The agent acts according to a stationary policy  $\pi : \mathcal{S} \rightarrow \Delta(\mathcal{A})$ . For finite MDPs, we can describe the expected transition probabilities under  $\pi$  using a  $|\mathcal{S}| \times |\mathcal{S}|$  matrix  $P^\pi$  such that  $P_{s,s'}^\pi = p^\pi(s'|s) \triangleq \sum_a P(s'|s, a)\pi(a|s)$ . Given  $\pi$  and a reward function  $r$ , the expected return is

$$Q_r^\pi(s, a) = \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k} \middle| s_t = s, a_t = a \right] = \mathbb{E}_{\substack{s' \sim p^\pi(\cdot|s) \\ a' \sim \pi(\cdot|s')}} [r_t + \gamma Q_r^\pi(s', a')]. \quad (6.1)$$

$Q_r^\pi$  are called the state-action values or simply the  $Q$ -values of  $\pi$ . The expectation  $\mathbb{E}_\pi[\cdot]$  is taken with respect to the randomness of both the policy and the transition dynamics. For simplicity of notation, from here onwards we will write expectations of the form  $\mathbb{E}_\pi[\cdot | s_t = s, a_t = a]$  as  $\mathbb{E}_\pi[\cdot | s_t, a_t]$ .

This recursive form is called the *Bellman equation*, and it makes the process of estimating  $Q_r^\pi$ —termed *policy evaluation*—tractable via dynamic programming (DP; [Bellman, 1957](#)). In particular, successive applications of the *Bellman operator*  $\mathcal{T}^\pi Q \triangleq r + \gamma P^\pi Q$  are guaranteed to converge to the true value function  $Q^\pi$  for any initial real-valued  $|\mathcal{S}| \times |\mathcal{A}|$  matrix  $Q$ .

**The successor representation** The *successor representation* (SR; [Dayan, 1993](#)) is motivated by the idea that a state representation for policy evaluation should be dependent on the similarity of different paths under the current policy. The SR is a policy's expected cumulative discounted state occupancy, and for discrete state spaces can be stored in an  $|\mathcal{S}| \times |\mathcal{S}|$  matrix  $M^\pi$ , where

$$M^\pi(s, s') \triangleq \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k \mathbb{1}(s_{t+k} = s') \middle| s_t = s \right] = \mathbb{E}_\pi \left[ \mathbb{1}(s_t = s') + \gamma M^\pi(s_{t+1}, s') \middle| s_t = s \right], \quad (6.2)$$

where  $\mathbb{1}(\cdot)$  is the indicator function. The SR can also be conditioned on actions, i.e.,  $M^\pi(s, a, s') \triangleq \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k \mathbb{1}(s_{t+k} = s') \middle| s_t = s, a_t = a \right]$ , and expressed in a vectorized format, we can write  $M^\pi(s) \triangleq M^\pi(s, \cdot)$  or  $M^\pi(s, a) \triangleq M^\pi(s, a, \cdot)$ . The recursion in Eq. [\(6.2\)](#)



admits a TD error:

$$\delta_t^M \triangleq \mathbf{1}(s_t) + \gamma M^\pi(s_{t+1}, \pi(s_{t+1})) - M^\pi(s_t, a_t), \quad (6.3)$$

where  $\mathbf{1}(s_t)$  is a one-hot state representation of length  $|\mathcal{S}|$ . One useful property of the SR is that, once converged, it facilitates rapid policy evaluation for any reward function in a given environment:

$$\mathbf{r}^\top M^\pi(s, a) = \mathbf{r}^\top \mathbb{E}_\pi \left[ \sum_k \gamma^k \mathbb{1}(s_{t+k}) \middle| s_t, a_t \right] = \mathbb{E}_\pi \left[ \sum_k \gamma^k r_{t+k} \middle| s_t, a_t \right] = Q_r^\pi(s, a). \quad (6.4)$$

**Fast transfer for multiple tasks** In the real world, we often have to perform multiple tasks within a single environment. A simplified framework for this scenario is to consider a set of MDPs  $\mathcal{M}$  that share every property (i.e.,  $\mathcal{S}, \mathcal{A}, P, \gamma, \rho$ ) except reward functions, where each task within this family is determined by a reward function  $r$  belonging to a set  $\mathcal{R}$ . Extending the notions of policy evaluation and improvement to this multitask setting, we can define *generalized policy evaluation* (GPE) as the computation of the value function of a policy  $\pi$  on a set of tasks  $\mathcal{R}$ . Similarly, *generalized policy improvement* (GPI) for a set of “base” policies  $\Pi$  is the definition of a policy  $\pi'$  such that

$$Q_r^{\pi'}(s, a) \geq \sup_{\pi \in \Pi} Q_r^\pi(s, a) \quad \forall (s, a) \in \mathcal{S} \times \mathcal{A} \quad (6.5)$$

for some  $r \in \mathcal{R}$ . As hinted above, the SR offers a way to take advantage of this shared structure by decoupling the agent’s evaluation of its expected transition dynamics under a given policy from a single reward function. Rather than needing to directly estimate  $Q^\pi \forall \pi \in \Pi$ ,  $M^\pi$  only needs to be computed once, and given a new reward vector  $\mathbf{r}$ , the agent can quickly perform GPE via Eq. (6.4). As shown by Barreto et al. (2017), GPE and GPI can be combined to define a new policy  $\pi'$  via

$$\pi'(s) \in \operatorname{argmax}_{a \in \mathcal{A}} \max_{\pi \in \Pi} \mathbf{r}^\top M^\pi(s, a). \quad (6.6)$$

For brevity, we will refer to this combined procedure of GPE and GPI simply as “GPI”, unless otherwise noted. The resulting policy  $\pi'$  is guaranteed to perform at least as well as any individual  $\pi \in \Pi$  (Barreto et al., 2020) and is part of a larger class of policies termed *set-improving policies* which perform at least as well as any single policy in a given set (Zahavy et al., 2021).

### 6.3 The First-Occupancy Representation

While the SR encodes states via total occupancy, this may not always be ideal. If a task lacks a time limit but terminates once the agent reaches a pre-defined goal, or if reward in a given state is consumed or made otherwise unavailable once encountered, a more useful representation would instead measure the duration until a policy is expected to reach states the *first* time. Such natural problems emphasize the importance of the first occupancy and motivate the *first-occupancy representation* (FR).

**Definition 6.3.1.** For an MDP with finite  $\mathcal{S}$ , the first-occupancy representation (FR) for a policy  $\pi$   $F^\pi \in [0, 1]^{|\mathcal{S}| \times |\mathcal{S}|}$  is given by

$$F^\pi(s, s') \triangleq \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k \mathbb{1}(s_{t+k} = s', s' \notin \{s_{t:t+k}\}) \middle| s_t \right], \quad (6.7)$$

where  $\{s_{t:t+k}\} = \{s_t, s_{t+1}, \dots, s_{t+k-1}\}$ , with the convention that  $\{s_{t:t+0}\} = \emptyset$ .

That is, as the indicator equals 1 iff  $s_{t+k} = s'$  and time  $t+k$  is the first occasion on which the agent has occupied  $s'$  since time  $t$ ,  $F^\pi(s, s')$  gives the expected discount at the time the policy first reaches  $s'$  starting from  $s$ . The idea of learning policy-dependent distances to target states has a long history in RL (Kaelbling, 1993; Pong et al., 2018; Hartikainen et al., 2020). However, previous methods don’t learn these distances as state *representations* and measure distance in the space of time steps, rather than discount factors. A more thorough discussion can be found in Appendix 6.H. We can write a recursive relationship for the FR (derivation in Appendix 6.A):

$$F^\pi(s, s') = \mathbb{E}_{s_{t+1} \sim p^\pi(\cdot|s)} \left[ \mathbb{1}(s_t = s') + \gamma(1 - \mathbb{1}(s_t = s'))F^\pi(s_{t+1}, s') \middle| s_t \right] \quad (6.8)$$

This recursion implies the following Bellman operator, analogous to the one used for policy evaluation:

**Definition 6.3.2** (FR Operator). *Let  $F \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$  be an arbitrary real-valued matrix. Then let  $\mathcal{G}^\pi$  denote the Bellman operator for the FR, such that*

$$\mathcal{G}^\pi F = I_{|\mathcal{S}|} + \gamma(\mathbf{1}\mathbf{1}^\top - I_{|\mathcal{S}|})P^\pi F, \quad (6.9)$$

where  $\mathbf{1}$  is the length- $|\mathcal{S}|$  vector of all ones. In particular, for a stationary policy  $\pi$ ,  $\mathcal{G}^\pi F^\pi = F^\pi$ .

The following result establishes  $\mathcal{G}^\pi$  as a contraction, with the proof provided in Appendix 6.D.

**Proposition 6.3.1** (Contraction). *Let  $\mathcal{G}^\pi$  be the operator as defined in Definition 6.3.2 for some stationary policy  $\pi$ . Then for any two matrices  $F, F' \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$ ,*

$$|\mathcal{G}^\pi F(s, s') - \mathcal{G}^\pi F'(s, s')| \leq \gamma |F(s, s') - F'(s, s')|, \quad (6.10)$$

with the difference equal to zero for  $s = s'$ .

This implies the following convergence property of  $\mathcal{G}^\pi$ .

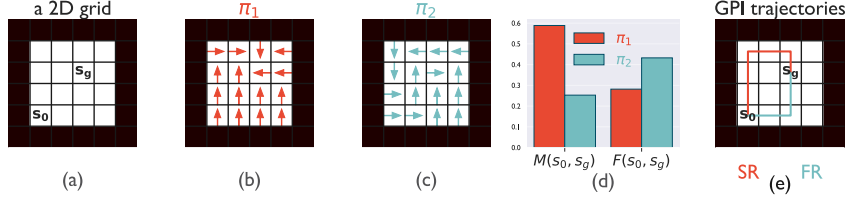
**Proposition 6.3.2** (Convergence). *Under the conditions assumed above, set  $F^{(0)} = I_{|\mathcal{S}|}$ . For  $k = 0, 1, \dots$ , suppose  $F^{(k+1)} = \mathcal{G}^\pi F^{(k)}$ . Then*

$$|F^{(k)}(s, s') - F^\pi(s, s')| < \gamma^k \quad (6.11)$$

for  $s \neq s'$  with the difference for  $s = s'$  equal to zero  $\forall k$ .

Therefore, repeated applications of the FR Bellman operator  $\mathcal{G}^k F \rightarrow F^\pi$  as  $k \rightarrow \infty$ . When the transition matrix  $P^\pi$  is unknown, the FR can instead be updated through the following TD error:

$$\delta_t^F = \mathbb{1}(s_t = s') + \gamma(1 - \mathbb{1}(s_t = s'))F^\pi(s_{t+1}, s') - F^\pi(s_t, s'). \quad (6.12)$$



**Figure 6.1: The FR is higher for shorter paths.** (a-c) A 2D gridworld and fixed policies. (d) The FR from  $s_0$  to  $s_g$  is higher for  $\pi_2$ , but the SR is lower. (e) SR-GPI with the SR picks  $\pi_1$ , while FR-GPI selects  $\pi_2$ .

In all following experiments, the FR is learned via TD updates, rather than via dynamic programming. To gain intuition for the FR, we can imagine a 2D environment with start state  $s_0$ , a rewarded state  $s_g$ , and deterministic transitions (Fig. 6.1a). One policy,  $\pi_1$ , reaches  $s_g$  slowly, but after first encountering it, re-enters  $s_g$  infinitely often (Fig. 6.1b). A second policy,  $\pi_2$ , reaches  $s_g$  quickly but never occupies it again (Fig. 6.1c). In this setting, because  $\pi_1$  re-enters  $s_g$  multiple times, despite arriving there more slowly than  $\pi_2$ ,  $M^{\pi_1}(s_0, s_g) > M^{\pi_2}(s_0, s_g)$ , but because the FR only counts the first occupancy of a given state,  $F^{\pi_1}(s_0, s_g) < F^{\pi_2}(s_0, s_g)$ . The FR thus reflects a policy’s path length between states.

**Policy evaluation and improvement with the FR** As with the SR, we can quickly perform policy evaluation with the FR. Crucially, however, the FR induces the following value function:

$$\mathbf{r}^\top F^\pi(s, a) = \mathbb{E}_\pi \left[ \sum_k \gamma^k r_{t+k}^F \mid s_t, a_t \right] \triangleq Q_{r^F}^\pi(s, a), \quad (6.13)$$

where  $r^F : \mathcal{S} \rightarrow \mathbb{R}$  is a reward function such that  $r^F(s_{t+k}) = r(s_{t+k})$  if  $s_{t+k} \notin \{s_{t:t+k}\}$  and 0 otherwise. In other words, multiplying any reward vector by the FR results in the value function for a corresponding task with *non-Markovian* reward structure in which the agent obtains rewards from states only once. Policy improvement can then be performed with respect to  $Q_{r^F}^\pi$  as normal. This structure is a very common feature of real-world tasks, such as foraging for food or reaching to a target. Accordingly, there is a rich history of studying tasks with this kind of structure, termed *non-Markovian reward decision processes* (NMRDPs; Littman et al., 2017; Gaon and Brafman, 2020; Ringstrom and Schrater, 2019; Peshkin et al., 2001). Helpfully, all NMRDPs can be converted into

an equivalent MDP with an appropriate transformation of the state space (Bacchus et al., 1996).

Most approaches in this family attempt to be generalists, *learning* an appropriate state transformation and encoding it using some form of logical calculus or finite state automaton (Bacchus et al., 1996; Littman et al., 2017; Gaon and Brafman, 2020). While it would technically be possible to learn or construct the transformation required to account for the non-Markovian nature of  $r^F$ , it would be exponentially expensive in  $|\mathcal{S}|$ , as every path would need to account for the first occupancy of each state along the path. That is,  $|\mathcal{S}|$  bits would need to be added to each successive state in the trajectory. Crucially, the FR has the added advantage of being task-agnostic, in that for *any* reward function  $r$  in a given environment, the FR can immediately perform policy evaluation for the corresponding  $r^F$ .

**Infinite state spaces** A natural question when extending the FR to real-world scenarios is how it can be generalized to settings where  $|\mathcal{S}|$  is either impractically large or infinite<sup>1</sup>. In these cases, the SR is reframed as *successor features*  $\psi^\pi$  (SFs; Kulkarni et al., 2016; Barreto et al., 2017), where the  $d$ th SF is defined as  $\psi_d^\pi(s) \triangleq \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k \phi_d(s_{t+k}) \mid s_t = s \right]$ , where  $d = 1, \dots, D$  and  $\phi : \mathcal{S} \rightarrow \mathbb{R}^D$  is a *base feature* function. The base features  $\phi(\cdot)$  are typically defined so that a linear combination predicts immediate reward (i.e.,  $\mathbf{w}^\top \phi(s) = r(s)$  for some  $\mathbf{w} \in \mathbb{R}^D$ ), and there are a number of approaches to learning them (Kulkarni et al., 2016; Ma et al., 2020). A natural extension to continuous  $\mathcal{S}$  for the FR would be to define a *first-occupancy feature* (FF) representation  $\varphi^\pi$ , where the  $d$ th FF is given by

$$\begin{aligned} \varphi_d^\pi(s) &\triangleq \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k \mathbb{1}(\phi_d(s_{t+k}) \geq \theta_d, \{\phi_d(s_{t'})\}_{t'=t:t+k} < \theta_d) \mid s_t = s \right] \\ &= \mathbb{1}(\phi_d(s_t) \geq \theta_d) + \gamma(1 - \mathbb{1}(\phi_d(s_t) \geq \theta_d)) \mathbb{E}_{s_{t+1} \sim p^\pi} [\varphi_d^\pi(s_{t+1})] \end{aligned} \quad (6.14)$$

where  $\theta_d$  is a threshold value for the  $d$ th feature. The indicator equals 1 only if  $s_{t+k}$  is the first state whose feature embedding exceeds the threshold. Note that this representation recovers the FR when the feature function is a one-hot state encoding and the thresholds

---

<sup>1</sup>Recent work (Blier et al., 2021; Touati and Ollivier, 2021a) has shown ways of extending the SR to continuous  $\mathcal{S}$  with the need for base features. We leave this as an interesting avenue for future work.

$\{\theta_d\}_{d=1}^D$  are all 1. One challenge with this particular parameterization is that it's challenging to set the thresholds as  $D$  grows. We consider alternative formulations of the FF representation in the next chapter.

## 6.4 Experiments

We now demonstrate the broad applicability of the FR, and highlight ways its properties differ from those of the SR. We focus on 4 areas: exploration, unsupervised RL, planning, and animal behavior.

### 6.4.1 The FR as an Exploration Bonus

Intuitively, representations which encode state visitation should be useful measures of exploration. [Machado et al. \(2020\)](#) proposed the SR as a way to encourage exploration in tasks with sparse or deceptive rewards. Specifically, the SR is used as a bonus in on-policy learning with Sarsa ([Rummery and Niranjan, 1994](#)):

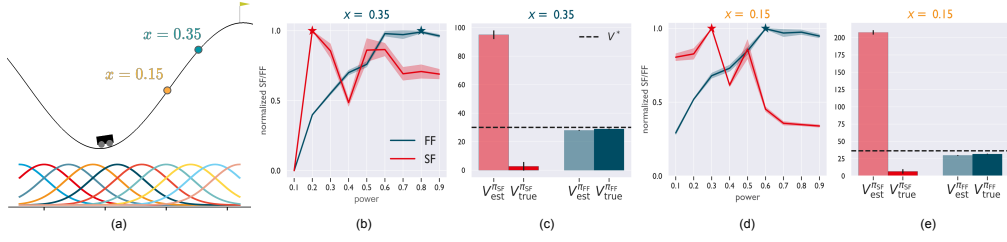
$$\delta_t = r_t + \frac{\beta}{\|M^\pi(s_t)\|_1} + \gamma Q^\pi(s_{t+1}, a_{t+1}) - Q^\pi(s_t, a_t), \quad a_{t+1} \sim \pi(\cdot | s_{t+1}), \quad (6.15)$$

where  $\beta \in \mathbb{R}$  controls the size of the exploration bonus. [Machado et al. \(2020\)](#) show that during learning,  $\|M^\pi(s)\|_1$  can act as a proxy for the state-visit count  $n(s)$ , with  $\|M^\pi(s)\|_1^{-1}$  awarding a progressively lower bonus for every consecutive visit of  $s$ . In the limit as  $t \rightarrow \infty$ , however,  $\|M^\pi(s)\|_1^{-1} \rightarrow 1 - \gamma \forall s$  as  $\pi$  stabilizes, regardless of whether  $\pi$  has effectively explored. To encourage exploration, we'd like for a bonus to maintain its effectiveness over time. In contrast to  $\|M^\pi(s)\|_1$ ,  $1 \leq \|F^\pi(s)\|_1 \leq \kappa_{|\mathcal{S}|} \triangleq \frac{1-\gamma^{|\mathcal{S}|+1}}{1-\gamma}$ , where  $\kappa_{|\mathcal{S}|} > 1$  for  $|\mathcal{S}| \geq 1$ . Note that  $\|F^\pi(s)\|_1 = \kappa_{|\mathcal{S}|}$  only if  $\pi$  reaches all states in as many steps. Because  $\|F^\pi\|_1$  only grows when new states or shorter paths are discovered, we can instead augment the reward as follows:

$$r_t \leftarrow r_t + \beta \|F^\pi(s_t)\|_1. \quad (6.16)$$

method	RIVERSWIM	SIXARMS
SARSA + FR	1,547,243 $\pm$ 34,050	1,191,490 $\pm$ 42,942
SARSA + SR	1,197,075 $\pm$ 36,999	1,025,750 $\pm$ 49,095
SARSA	25,075 $\pm$ 1,224	376,655 $\pm$ 8,449

**Table 6.1:** Exploration results.  $\pm$  values denote 1 SE across 100 trials.

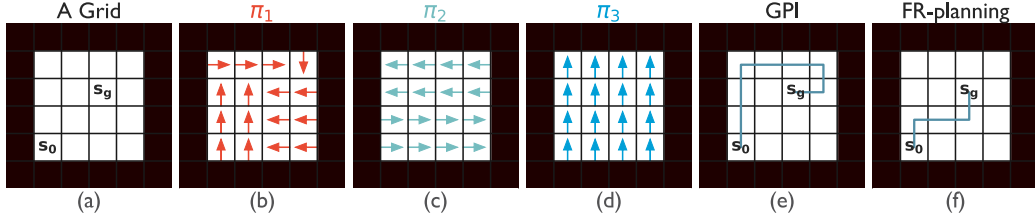


**Figure 6.2: The FF facilitates accurate policy evaluation and selection.** Shading denotes 1 SE over 20 seeds.

We tested our approach on the RIVERSWIM and SIXARMS problems (Strehl and Littman, 2008), two hard-exploration tasks from the PAC-MDP literature. In both tasks, visualized in Appendix Fig. 6.8, the transition dynamics push the agent towards small rewards in easy to reach states, with greater reward available in harder to reach states. In both cases, we ran Sarsa, Sarsa with an SR bonus (SARSA + SR) and Sarsa with an FR bonus (SARSA + FR) for 5,000 time steps with an  $\epsilon$ -greedy policy. The results are listed in Table 6.1, where we can see that the FR results in an added benefit over the SR. It’s also important to note that the maximum bonus  $\kappa_{|\mathcal{S}|}$  has another useful property, in that it scales exponentially in  $|\mathcal{S}|$ . This is desirable, because as the number of states grows, exploration frequently becomes more difficult. To measure this factor empirically, we tested the same approaches with the same settings on a modified RIVERSWIM, RIVERSWIM-N, with  $N = \{6, 12, 24\}$  states, finding that SARSA + FR was more robust to the increased exploration difficulty (see Appendix Table 6.2 and Appendix 6.C for results and more details). We also tested whether these results extend to the function approximation setting, comparing a DQN-style model (Mnih et al., 2015) using the SF and FF as exploration bonuses on the challenging DEEPSA task (Osband et al., 2020), finding that the advantage of the FF is conserved. See Appendix 6.C for details. Developing further understanding of the relationship between the FR and exploration represents an interesting topic for future work.

### 6.4.2 Unsupervised RL with the FF

We demonstrate the usefulness of the FF in the *unsupervised pre-training RL* (URL) setting, a paradigm which has gained popularity recently as a possible solution to the high sample complexity of deep RL algorithms (Liu and Abbeel, 2021; Gregor et al., 2016; Ey-



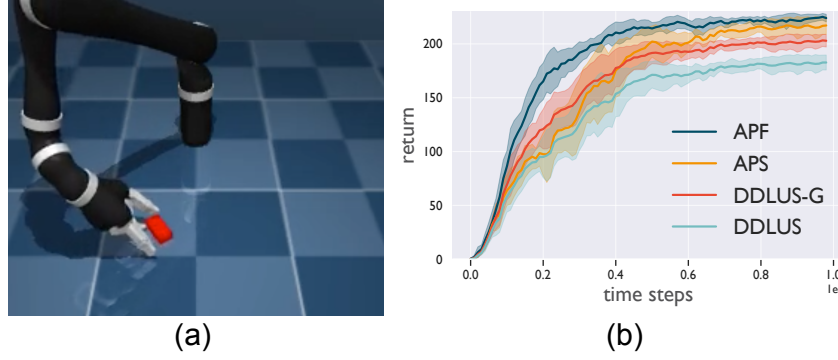
**Figure 6.3: The FR enables efficient planning.** (a-d) A 2D gridworld with start ( $s_0$ ) and goal ( $s_g$ ) states, along with three fixed policies. (e) GPI follows  $\pi_1$ . (f) Planning with the FR enables the construction of a shorter path.

senbach et al., 2018; Sharma et al., 2020). In URL, the agent first explores an environment without extrinsic reward with the objective of learning a useful representation which then enables rapid fine-tuning to a test task.

**Continuous MountainCar** We first demonstrate that if the test task is non-Markovian, the SR can produce misleading value estimates. To do so, we use a modified version of the continuous MountainCar task (Brockman et al., 2016) (Fig. 6.2(a)). The agent pre-trains for 20,000 time steps in a rewardless environment, during which it learns FFs or SFs for a set of policies  $\Pi$  which swing back and forth with a fixed acceleration or “power.” (details in Appendix 6.C). During fine-tuning, the agent must identify the policy  $\pi \in \Pi$  which reaches a randomly sampled goal location the fastest. We use radial basis functions as the base features  $\phi_d(\cdot)$  with fixed FF thresholds  $\theta_d = 0.7$ .

Fig. 6.2(b,d) plots the FF and SF values versus policy power from the start state for two different goal locations. The low-power policies require more time to gain momentum up the hill, but the policies which maximize the SF values slow down around the goal locations, dramatically increasing their total “occupancy” of that area. In contrast, high-powered policies reach the goal locations for the first time much sooner, and so the policies with the highest FF values have higher powers. In the test phase, the agent fits the reward vector  $\mathbf{w} \in \mathbb{R}^D$  by minimizing  $\sum_t \|r_t - \mathbf{w}^\top \phi(s_t)\|^2$ , as is standard in the SF literature (Barreto et al., 2017, 2020; Zahavy et al., 2021). The agent then follows the set-max policy (SMP; (Zahavy et al., 2021)), which selects the policy in  $\Pi$  which has the highest expected value across starting states:  $\pi^{\text{SMP}} \in \arg\max_{\pi \in \Pi} \mathbb{E}_{s_0 \sim \mu} [V^\pi(s_0)]$ , where  $V^\pi(s_0) = \mathbf{w}^\top \varphi^\pi(s_0)$  (with  $\varphi^\pi$  replaced by  $\psi^\pi$  for SF-based value estimates). Fig. 6.2(c,e) shows both the estimated ( $V_{\text{est}}$ ) and true ( $V_{\text{true}}$ ) values of the SMPs selected using both





**Figure 6.4: APF accelerates convergence in robotic reaching.** Shading denotes 1 SE over 10 seeds.

the SF and FF, along with the value of the optimal policy  $V^*$ . We can see that the accumulation of the SFs results in a significant overestimation in value, as well as a suboptimal policy. The FF estimates are nearly matched to the true values of the selected policies for each goal location and achieve nearly optimal performance.

**Robotic reaching** To test whether these results translate to high-dimensional problems, we applied the FF to the 6-DoF JACO robotic arm environment from Laskin et al. (2021). In this domain, the arm must quickly reach to different locations and perform simple object manipulations (Fig. 6.4(a)). We modify the *Active Pre-training with Successor features* (APS; Liu and Abbeel, 2021) URL algorithm, which leverages a nonparametric entropy maximization objective in conjunction with SFs during pre-training () in order to learn useful and adaptable behaviors. Our modification is to replace the SFs with FFs, resulting in *Active Pre-training with First-occupancy features* (APF), using the same intuition motivating the MountainCar experiments: cumulative features are misleading when downstream tasks benefit from quickly reaching a desired goal, in this case, the object. Here, the agent is first trained for  $10^6$  time steps using the aforementioned intrinsic reward objective before being applied to a specified reaching task. As additional baselines, we compare against two variants of *dynamic distance learning - unsupervised* (DDLUS and DDLUS-G; (Hartikainen et al., 2020)). Details can be found in Appendix 6.C. We found that the FF accelerated convergence (Fig. 6.4(b)).

### 6.4.3 Planning with the FR

While SRs effectively encode task-agnostic, pre-compiled environment models, they cannot be directly used for multi-task model-based planning. GPI is only able to select actions based on a one-step lookahead, which may result in suboptimal behavior. One simple situation that highlights such a scenario is depicted in Fig. 6.3. As before, there are start and goal states in a simple room (Fig. 6.3(a)), but here there are three policies comprising  $\Pi = \{\pi_1, \pi_2, \pi_3\}$  (Fig. 6.3(b-d)). GPI selects  $\pi_1$  because it is the only policy that reaches the goal  $s_g$  within one step of the start  $s_0$ :  $\pi_1 = \max_{\pi \in \Pi} \mathbf{r}^\top M^\pi(s_0, s_g)$  (Fig. 6.3(e)). (Note that using GPI with the FR instead would also lead to this choice.) To gain further intuition for the FR versus the SR, we plot the representations for the policies in Appendix Fig. 6.17. However, the optimal strategy using the policies in  $\Pi$  is instead to move right using  $\pi_2$  and up using  $\pi_3$ . How can the FR be used to find such a sequence?

Intuitively, starting in a state  $s$ , this strategy is grounded in following one policy until a certain state  $s'$ , which we refer to as a *subgoal*, and then *switching* to a different policy in the base set. Because the FR effectively encodes the shortest path between each pair of states  $(s, s')$  for a given policy, the agent can elect to follow the policy  $\pi \in \Pi$  with the greatest value of  $F^\pi(s, s')$ , then switch to another policy and repeat the process until reaching a desired state. The resulting approach is related to the hierarchical *options* framework (Sutton et al., 1999; Sutton and Barto, 2018a), where the planning process effectively converts the base policies into options with—as we show—optimal termination conditions. For a more detailed discussion, see Appendix 6.G.

More formally, we can construct a DP algorithm to solve for the optimal sequence of planning policies  $\pi^F$  and subgoals  $s^F$ . Denoting by  $\Gamma_k(s)$  the total discount of the full trajectory from  $s$  to  $s_g$  at step  $k$  of the procedure, we jointly optimize over policies  $\pi$  and subgoals  $s'$  for each state  $s$ :

$$\begin{aligned} \Gamma_{k+1}(s) &= \max_{\pi \in \Pi, s' \in \mathcal{S}} F^\pi(s, s') \Gamma_k(s'), \\ \text{with } \pi_{k+1}^F(s), s_{k+1}^F(s) &= \operatorname{argmax}_{\pi \in \Pi, s' \in \mathcal{S}} F^\pi(s, s') \Gamma_k(s'). \end{aligned}$$

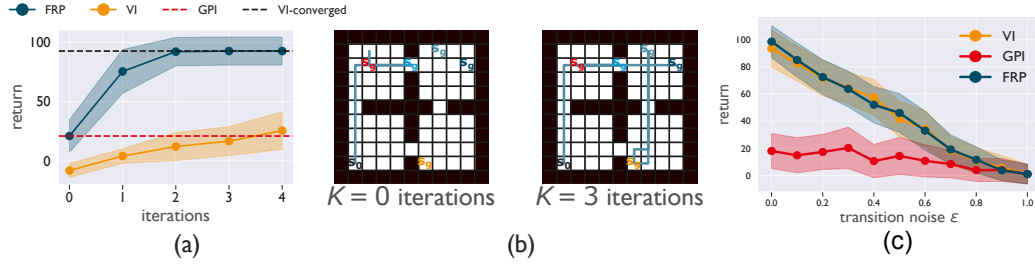
Intuitively, the product  $F^\pi(s, s') \Gamma_k(s')$  can be interpreted as the expected discount of

the plan consisting of first following  $\pi$  from  $s$  to  $s'$ , then the current best (shortest-path) plan from  $s'$  to  $s_g$ . Note that it is this property of the FR which allows such planning: multiplying total occupancies, as would be done with the SR, is not well-defined. The full procedure, which we refer to as FR-planning (FRP), is given by Alg. 7 in Appendix 6.B. Appendix Fig. 6.13 depicts the resulting policies  $\pi^F$  and subgoals  $s^F$  obtained from running FRP on the example in Fig. 6.3. The following result shows that under certain assumptions, FRP finds the shortest path to a given goal (proof in Appendix 6.D).

**Proposition 6.4.1** (Planning optimality). *Consider a deterministic, finite MDP with a single goal state  $s_g$ , and a base policy set  $\Pi$  composed of policies  $\pi : \mathcal{S} \rightarrow \mathcal{A}$ . We make the following coverage assumption: there exists a sequence of policies that reaches  $s_g$  from a given start state  $s_0$ . Then Alg. 7 converges so that  $\Gamma(s_0) = \gamma^{L_\pi^*}$ , where  $L_\pi^*$  is the shortest path length from  $s_0$  to  $s_g$  using  $\pi \in \Pi$ .*

**Performance and computational cost** We can see that each iteration of the planning algorithm adds at most one new subgoal to the planned trajectory from each state to the goal, with convergence when no policy switches can be made that reduce the number of steps required. If there are  $K$  iterations, the overall computational complexity of FRP is  $\mathcal{O}(K|\Pi||\mathcal{S}|^2)$ . The worst-case complexity occurs when the policy must switch at every state en route to the target— $K$  is upper-bounded by the the number of states along the shortest path to the goal. In contrast, running value iteration (VI; (Sutton and Barto, 2018a)) for  $N$  iterations is  $\mathcal{O}(N|\mathcal{A}||\mathcal{S}|^2)$ . Given the true transition matrix  $P$  and reward vector  $\mathbf{r}$ , VI will also converge to the shortest path to a specified goal state, but FRP converges more quickly than VI whenever  $K|\Pi| < N|\mathcal{A}|$ . To achieve an error  $\epsilon$  between the estimated value function and the value function of the optimal policy, VI requires  $N \geq \frac{1}{(1-\gamma)} \log \frac{2}{(1-\gamma)^2 \epsilon}$  (Puterman, 1994), which for  $\gamma = 0.95$ ,  $\epsilon = 0.1$ , e.g., gives  $N \geq 180$  iterations<sup>2</sup>. To test convergence rates in practice, we applied FRP, VI, and GPI using the FR to the classic FOURROOMS environment (Sutton et al., 1999) on a modified task in which agents start in the bottom left corner and move to a randomly located goal state. Once the goal is reached, a new goal is randomly sampled in a different loca-

<sup>2</sup>Other DP methods like policy iteration (PI), which is strongly polynomial, converge more quickly than VI, but in the example above, PI still needs  $N \geq \log(1/((1-\gamma)\epsilon))/(1-\gamma) = 106$  (Ye, 2011), for instance.

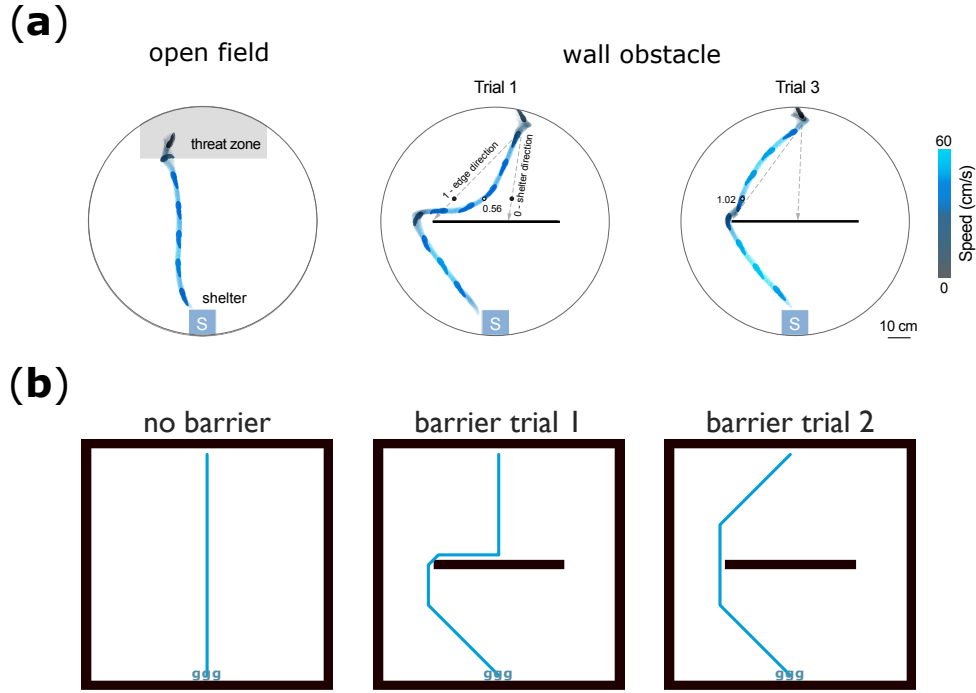


**Figure 6.5: FRP interpolates between GPI and model-based DP.** Shading represents 1 SE.

tion until the episode is terminated after 75 time steps. For GPI and FRP, we use four base policies which each only take one action:  $\{\text{up, down, left, right}\}$ , with their FRs learned by TD learning. We ran each algorithm for 100 episodes, with the results plotted in Fig. 6.5(a). Note that here GPI is equivalent to FRP with  $K = 0$  iterations. To see this, observe that when there is a single goal  $s_g$  such that only  $r(s_g) > 0$ , the policy selected by GPI is

$$\pi^{\text{GPI}}(s) \in \operatorname{argmax}_{\pi \in \Pi} \mathbf{r}^\top F^\pi(s) = \operatorname{argmax}_{\pi \in \Pi} r(s_g) F^\pi(s, s_g) = \operatorname{argmax}_{\pi \in \Pi} F^\pi(s, s_g). \quad (6.17)$$

When there are  $n$  goal states with equal reward, finding the ordering of the goals that results in the shortest expected path is in general  $\mathcal{O}(n!)$  (see Appendix 6.F). Due to the nature of the base policies used above, the number of subgoals on any path is equal to the number of turns the agent must take from its current state to the goal, which for this environment is three. We can then see that FRP reaches the optimal performance obtained by the converged VI after  $K = 3$  iterations (Fig. 6.5(a)). In contrast, for the same number of iterations, VI performs far worse. This planning process must be repeated each time a new goal is sampled, so that the computational benefits of FRP versus traditional DP methods compound for each new reward vector. Example FRP trajectories between goals for  $K = 0$  and  $K = 3$  iterations are plotted in Fig. 6.5(b). Finally, to test FRP’s robustness to stochasticity, we added transition noise  $\epsilon$  to the FOURROOMS task. That is, the agent moves to a random adjacent state with probability  $\epsilon$  regardless of action. We compared FRP to converged VI for increasing  $\epsilon$ , with the results plotted in Fig. 6.5(c), where we can see that FRP matches the performance of VI across noise levels. It’s important to



**Figure 6.6: FRP induces realistic escape behavior.**

note that this ability to adaptively interpolate between MF and MB behavior, based on the value of  $K$ , is a unique capability of the FR compared to the SR<sup>3</sup>. The same FRs can be combined using DP to plan for one task or for GPI on the next.

#### 6.4.4 Escape behavior

In prey species such as mice, escaping from threats using efficient paths to shelter is critical for survival (Lima and Dill, 1990). Recent work studying the strategies employed by mice when fleeing threatening stimuli in an arena containing a barrier has indicated that, rather than use an explicit cognitive map, mice instead appear to memorize a sequence of subgoals to plan efficient routes to shelter (Shamash et al., 2021a). When first threatened, most animals ran along a direct path and into the barrier. Over subsequent identical trials spanning 20 minutes of exploration, threat-stimulus presentation, and escape, mice learned to navigate directly to the edge of the wall before switching direction towards the shelter (Fig. 6.6). Follow-up control experiments suggest that mice acquire persis-

<sup>3</sup>We'd like to stress that this claim of uniqueness is only with respect to the SR. Previous work also explores the use of MF methods to support MB learning (e.g., Pong et al. (2018))

tent spatial memories of subgoal locations for efficient escapes. We model this task and demonstrate that FRP induces behavior consistent with these results.

We model the initial escape trial by an agent with a partially learned FR, leading to a suboptimal escape plan leading directly to the barrier. Upon hitting the barrier, sensory input prompts rapid re-planning to navigate around the obstacle. The FR is then updated and the escape plan is recomputed, simulating subsequent periods of exploration during which the mouse presumably memorizes subgoals. We find a similar pattern of behavior to that of mice (Fig. 6.6(b)). See Appendix 6.C for experimental details.

We do not claim that this is the exact process by which mice are able to efficiently learn escape behavior. Rather, we demonstrate that the FR facilitates behavior that is consistent with our understanding of animal learning in tasks which demand efficient planning. Given the recent evidence in support of SR-like representations in the brain (Stachenfeld et al., 2017; Momennejad et al., 2017), we are optimistic about the possibility of neural encodings of FR-like representations as well. We also re-emphasize that this type of rapid shortest-path planning is not possible with the SR.

## 6.5 Conclusion

In this work, we have introduced the FR, an alternative to the SR which encodes the expected path length between states for a given policy. We explored its basic formal properties, its use as an exploration bonus, and its usefulness for unsupervised representation learning in environments with an ethologically important type of non-Markovian reward structure. We then demonstrated that, unlike the SR, the FR supports a form of efficient planning which induces similar behaviors to those observed in mice escaping from perceived threats. As with any new approach, there are limitations. However, we believe that these limitations represent opportunities for future work. From a theoretical perspective, it will be important to more precisely understand FRP in stochastic environments. For the FF, we have limited understanding of the effect of feature choice on performance, especially in high dimensions. FRP is naturally restricted to discrete state spaces, and it could be interesting to explore approximations or its use in *partially-observable* MDPs with real-valued observations and discrete latents (e.g., Vértés and Sahani, 2019; Du et al.,

2019). Further exploration of FRP's connections to hierarchical methods like options would be valuable. Finally, it would be informative to test hypotheses of FR-like representations in the brain. We hope this research direction will inspire advancements on representations that can support efficient behavior in realistic settings.

## Appendix

### Appendix 6.A: FR recursion

For clarity, we provide the derivation of the recursive form of the FR below:

$$\begin{aligned}
F^\pi(s, s') &= \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k \mathbb{1}(s_{t+k} = s', s' \notin \{s_{t:t+k}\}) \middle| s_t \right] \\
&= \mathbb{E}_\pi \left[ \mathbb{1}(s_t = s', s' \notin \emptyset) + \sum_{k=1}^{\infty} \gamma^k \mathbb{1}(s_{t+k} = s', s' \notin \{s_{t:t+k}\}) \middle| s_t \right] \\
&= \mathbb{E}_\pi \left[ \mathbb{1}(s_t = s') + \sum_{k=1}^{\infty} \gamma^k \mathbb{1}(s_{t+k} = s', s_t \neq s', s' \notin \{s_{t+1:t+k}\}) \middle| s_t \right] \\
&= \mathbb{E}_{s_{t+1} \sim p^\pi(\cdot|s)} \left[ \mathbb{1}(s_t = s') + \gamma \mathbb{1}(s_t \neq s') F^\pi(s_{t+1}, s') \middle| s_t \right] \\
&= \mathbb{E}_{s_{t+1} \sim p^\pi(\cdot|s)} \left[ \mathbb{1}(s_t = s') + \gamma(1 - \mathbb{1}(s_t = s')) F^\pi(s_{t+1}, s') \middle| s_t \right]
\end{aligned} \tag{6.18}$$

### Appendix 6.B: FRP Algorithm

We present the full algorithm for FR planning (FRP) below.

---

#### Algorithm 7 FR Planning (FRP)

---

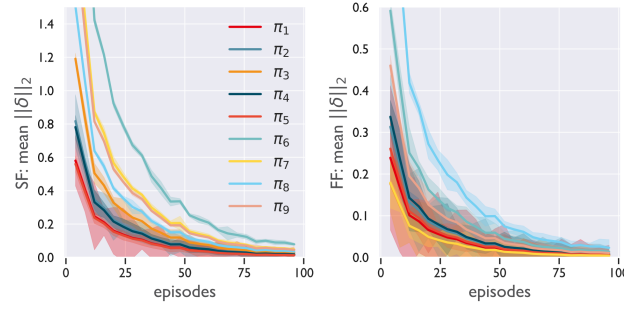
```

1: input: goal state  $s_g$ , base policies  $\Pi = \{\pi_1, \dots, \pi_n\}$  and FRs  $\{F^{\pi_1}, \dots, F^{\pi_n}\}$ .
2: // initialize discounts-to-goal  $\Gamma$ 
3:  $\Gamma_0(s) \leftarrow -\infty \forall s \in \mathcal{S}$ 
4: for  $s \in \mathcal{S}$  do
5:    $\Gamma_1(s) \leftarrow \max_{\pi \in \Pi} F^\pi(s, s_g)$ 
6:    $\pi_1^F(s), s_1^F(s) \leftarrow \operatorname{argmax}_{\pi \in \Pi} F^\pi(s, s_g), s_g$ 
7: end for
8: // iteratively refine  $\Gamma$ 
9:  $k \leftarrow 1$ 
10: while  $\exists s \in \mathcal{S}$  such that  $\Gamma_k(s) > \Gamma_{k-1}(s)$  do
11:   for  $s \in \mathcal{S}$  do
12:      $\Gamma_{k+1}(s) \leftarrow \max_{\pi \in \Pi, s' \in \mathcal{S}} F^\pi(s, s') \Gamma_k(s')$ 
13:      $\pi_{k+1}^F(s), s_{k+1}^F(s) \leftarrow \operatorname{argmax}_{\pi \in \Pi, s' \in \mathcal{S}} F^\pi(s, s') \Gamma_k(s')$ 
14:   end for
15:    $k \leftarrow k + 1$ 
16: end while
17: return  $\pi^F, s^F$ 

```

---

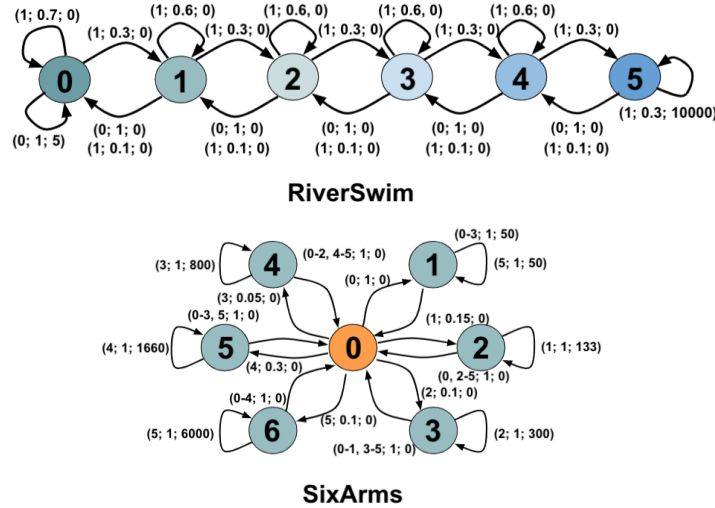




**Figure 6.7: FF and SF learning curves for continuous MountainCar.** Results averaged over 20 runs. Shading represents one standard deviation.

## Appendix 6.C: Additional Experimental Details

All experiments except for the robotic reaching experiment were performed on a single 8-core CPU. The robotic reaching experiment was performed using four Nvidia Quadro RTX 5000 GPUs.



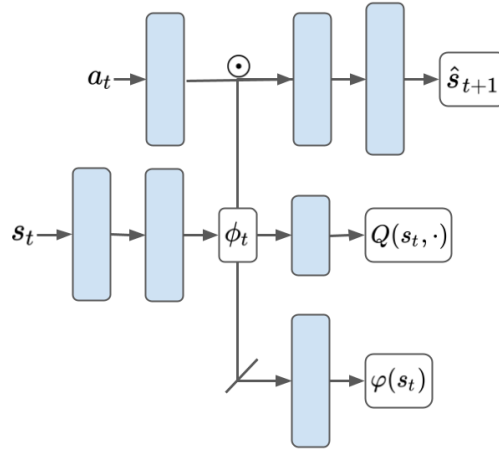
**Figure 6.8: Tabular environments for exploration.** The tuples marking each transition denote (action id(s); probability; reward). In RiverSwim, the agent starts in either state 1 or state 2 with equal probability, while for SixArms the agent always starts in state 0.

**Tabular exploration** We reuse all hyperparameter settings from [Machado et al. \(2020\)](#) in both the RIVERSWIM and SIXARMS environments, with the only difference being a lower value for  $\beta$ , the exploration bonus coefficient, for the FR, as the bonuses given by the FR are generally larger. The hyperparameters are  $\{\alpha, \eta, \gamma_{\text{SR/FR}}, \beta, \epsilon, \eta\}$ , which are the Sarsa learning rate, the SR/FR learning rate, the SR/FR discount factor, the explo-

ration bonus coefficient, and the probability of taking a random action in the  $\epsilon$ -greedy policy. For RIVERSWIM, these values were  $\{0.25, 0.01, 0.95, 100/50, 0.1\}$ , respectively, and for SIXARMS they were  $\{0.1, 0.01, 0.99, 100/50, 0.01\}$ . For the RIVERSWIM-N task, we chose  $N = \{6, 12, 24\}$  as default RIVERSWIM has  $N = 6$  states, and we chose to successively double the problem size. As  $N$  increased, the number of unrewarded central states was multiplied (with the same transition structure), while the endpoints remained the same. It's also worth noting that  $\beta$  could be manually adjusted upwards to compensate for the SR bonus' invariance to problem size, though this would require a longer hyperparameter search generally, which we believe is less preferable to a bonus which naturally scales.

RIVERSWIM-N	SARSA + FR	SARSA + SR	SARSA
$N = 6$	$1,547,243 \pm 34,050$	$1,197,075 \pm 36,999$	$25,075 \pm 1,224$
$N = 12$	$1,497,937 \pm 29,291$	$714,797 \pm 34,574$	$14,590 \pm 3,145$
$N = 24$	$962,376 \pm 33,325$	$519,511 \pm 20,580$	$11,950 \pm 2,643$

**Table 6.2:** RIVERSWIM-N results.  $\pm$  values denote 1 SE across 100 trials.



**Figure 6.9:** Network architecture for DQN + FF and DQN + SF

**Exploration with function approximation** In order to test the usefulness of the FR/FF in a function approximation setting, we use a similar approach to Machado et al. (2020). That is, we train a modified DQN agent (Mnih et al., 2015) using an architecture inspired by Machado et al. (2020) and Oh et al. (2015) (see Fig. 6.9), such that the base feature representation  $\phi(s)$  is an intermediate layer of the network. Like the standard DQN,

the architecture outputs an  $|\mathcal{A}|$ -length vector of predicted  $Q$ -values for the current state, trained off-policy using minibatches of transition tuples  $\{(s_t^i, a_t^i, r_t^i, s_{t+1}^i)\}_{i=1}^B$  (where  $B$  is the minibatch size) to minimize the squared Bellman error

$$\mathcal{L}_Q = \sum_{i=1}^B \|r_t^i + \gamma \max_a Q_{-}(s_{t+1}^i, a) - Q(s_t^i, a_t^i)\|_2^2 \quad (6.19)$$

via gradient descent (the subscript  $-$  on the target  $Q$ -values indicates that gradients do not flow through it). Unlike the standard DQN, the network has two additional output heads. The first is a *reconstruction* head which, given the base feature representation of a state  $s_t$  and an embedding of the subsequent action  $a_t$  produces a prediction of the following state  $\hat{s}_{t+1}$ . It's trained to minimize the reconstruction loss

$$\mathcal{L}_s = \sum_{i=1}^B \|s_{t+1} - \hat{s}_{t+1}\|_2^2. \quad (6.20)$$

The final output head of the network is an FF/SF prediction, trained using the squared FF error in the former case:

$$\mathcal{L}_\varphi = \sum_{i=1}^B \|\tilde{\phi}(s_t) + \gamma(1 - \tilde{\phi}(s_t))\varphi_{-}(s_{t+1}) - \varphi(s_t)\|_2^2, \quad (6.21)$$

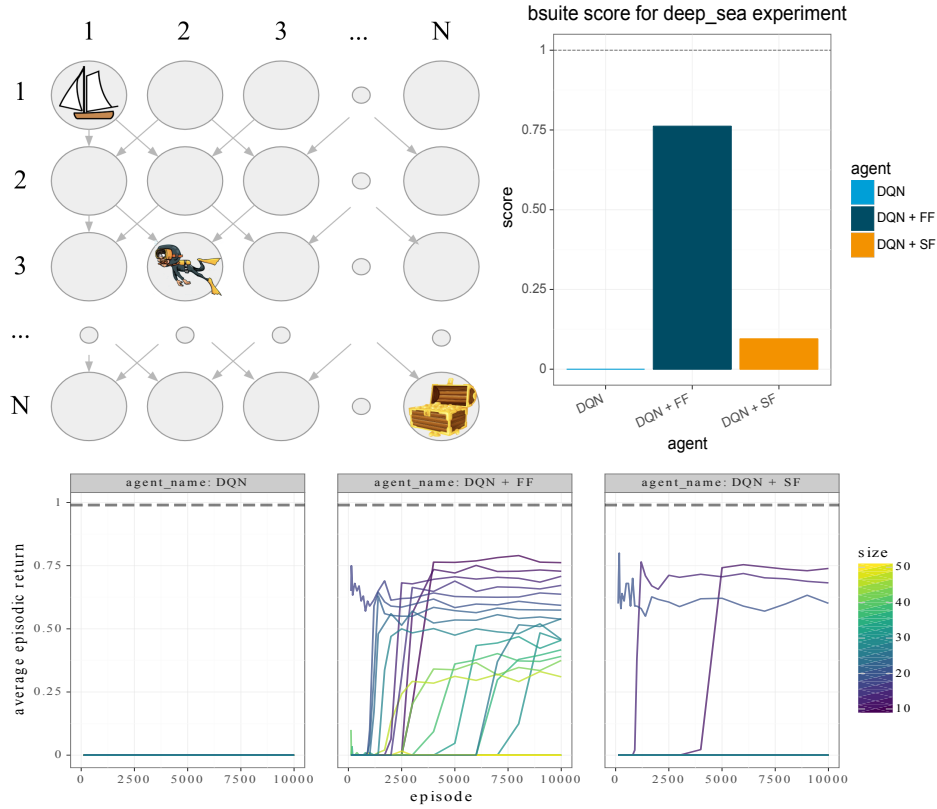
and the squared SF error in the latter

$$\mathcal{L}_\psi = \sum_{i=1}^B \|\phi(s_t) + \gamma\psi_{-}(s_{t+1}) - \psi(s_t)\|_2^2. \quad (6.22)$$

For the FF, the features  $\phi_t$  are passed through a sigmoid function to compress them in the range  $[0, 1]$  and then thresholded at 0.75. The total loss is then given as a weighted combination

$$\mathcal{L} = w_Q \mathcal{L}_Q + w_s \mathcal{L}_s + w_X \mathcal{L}_X, \quad (6.23)$$

where  $X \in \{\varphi, \psi\}$  and  $w_Q, w_s, w_X \in \mathbb{R}$  are fixed weights. As in Machado et al. (2020), gradients from  $\mathcal{L}_Q$  and  $\mathcal{L}_s$ , but not  $\mathcal{L}_X$ , are permitted to flow through to  $\phi$ . Thus,



**Figure 6.10: Exploration with function approximation.** (Top left) Visualization of the DeepSea environment, credit to [Osband et al. \(2020\)](#). (Top right) DQN + FF significantly outperforms standard DQN and DQN + SF. (Bottom) Different runs across problem sizes.

the base features are trained to be both reward-predictive and to carry information about the environment transition dynamics. The norm of the FF/SF vector is then used to compute an intrinsic exploration bonus to the task reward in the same manner as in the tabular setting.

To test this model, we chose the DEEPSEA task from the Behavior Suite (bsuite; [\(Osband et al., 2020\)](#)) set of benchmark tasks. DEEPSEA is a challenging exploration task set up in an  $N \times N$  grid (Fig. 6.10, top left). At the beginning of each episode, the agent starts at the top left of the grid. Each time step, the agent descends one level, and can choose to move either right or left. The episode ends after  $N$  steps, when the agent reaches the bottom level. There is a small negative reward of  $-0.01$  if the agent moves right, but if the agent moves right  $N$  times in a row, there is a large reward  $+1$  located at the bottom right of environment—this is the only policy which nets the agent a positive reward.

In the `bsuite` framework, the agent is separately trained on increasing problem sizes  $N = 5, 6, 7, \dots, 50$  for 10,000 episodes each, with the final score the proportion of  $N$  for which the agent reached an average regret of less than 0.9 faster than  $2^N$  episodes.

We tested the standard DQN, DQN + SF (Machado et al., 2020), and DQN + FF agents on this task, with training hyperparameters described in Table 6.3. For all models, the network consisted entirely of fully-connected layers, with  $\phi(s_t)$  being a 2-layer MLP with 64 units per layer,  $Q(s_t, \cdot)$  being a linear function of  $\phi(s_t)$  with  $|\mathcal{A}| = 2$  units,  $\hat{s}_{t+1}$  consisting of a 64-unit layer followed by an  $N^2$ -unit output layer (the action embedding is 64-dimensional as well), and the SFs/FFs also a 64-dimensional linear layer over  $\phi(s_t)$ . The agent is trained using  $\epsilon$ -greedy action selection. Our DQN implementation was coded in JAX and based off that of Osband et al. (2020). To select the values of  $\beta$ ,  $w_s$ , and  $w_X$  ( $w_Q$  was always kept at 1) we performed a sweep over  $\beta \in \{0.01, 0.05, 0.1\}$  and  $w_X, w_Q \in \{0.001, 0.1, 1, 10, 100, 1000\}$ , choosing the best-performing values for each method.

HYPERPARAMETER		DQN + FF	DQN + SF	DQN
optimizer	Adam (Kingma and Ba, 2014)		Adam	Adam
learning rate		0.001	0.001	0.001
$\beta$		0.05	0.01	—
$w_Q, w_s, w_X$		(1, 100, 1000)	(1, 0.001, 1000)	—
$B$		32	32	32
replay buffer size		10,000	10,000	10,000
target update period		4	4	4
$\gamma$		0.99	0.99	0.99
$\epsilon$		0.05	0.05	0.05

**Table 6.3:** Hyperparameter settings for the DEEPSEA experiment

Results are presented in Fig. 6.10. We can see that DQN+FF significantly outperforms the other methods (top right), with the intuition from the tabular experiments—particularly RIVERSWIM-N—carrying over into the function approximation setting. That is, as  $N$  increases, the norm of the SF approaches its asymptotic value regardless of the degree of exploration. In contrast, for the FF, the maximum bonus scales with the problem size. This enables the bonus to remain effective in environments with larger

state spaces. We hope to investigate this approach and its theoretical properties further in future work.

**MountainCar experiment** In our version of the task, the feature representations are learned in a rewardless environment, and at test time the reward may be located at any location along the righthand hill. We evaluate the performance of a set of policies  $\Pi = \{\pi_i\}$  with a constant magnitude of acceleration and which accelerate in the opposite direction from their current displacement when at rest and in the direction of their current velocity otherwise (see Python code below for details). That is, each  $\pi_i$  will swing back and forth along the track with a fixed power coefficient  $a_i$ . For each possible reward location along the righthand hill, then, the best policy from among this set is the one whose degree of acceleration is such that it arrives at the reward location in the fewest time steps. There is a natural tradeoff—too little acceleration and the cart will not reach the required height. Too much, and time will be wasted traveling too far up the lefthand hill and then reversing momentum back to the right.

*We hypothesized that the FF would be a natural representation for this task, as it would not count the repeated state visits each policy experiences as it swings back and forth to gain momentum.*

We defined a set of policies with acceleration magnitudes  $|a_i| = 0.1i$  for  $i = 1, \dots, 9$ , and learned both their SFs and FFs via TD learning on an “empty” environment without rewards over the course of 100 episodes, each consisting of 200 time steps, with the SFs using just the simple RBF feature functions without thresholds. Python code for the policy class is shown below.

```

1  class FixedPolicy:
2
3  def __init__(self, a):
4      # set fixed acceleration/power
5      self.a = a
6
7  def get_action(self, pos, vel):
8
9      if vel == 0:
10         # if stopped, accelerate to the opposite end of the

```

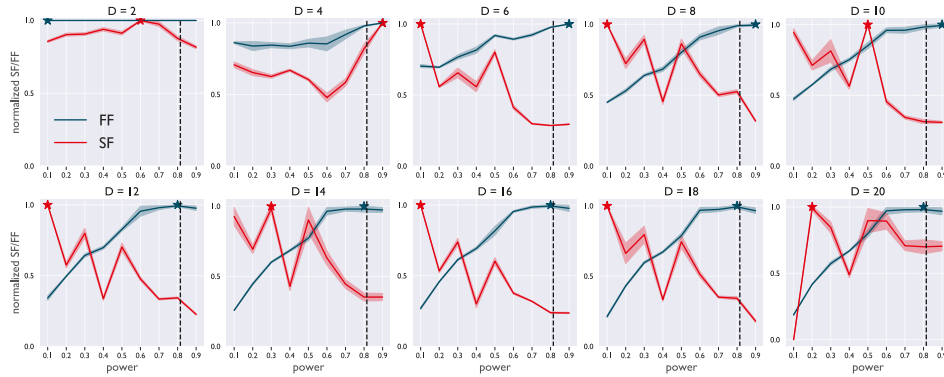
```

environment
11     action = -sign(pos) * self.a
12     else:
13         # otherwise, continue in the current direction of motion
14         action = sign(vel) * self.a
15
16     return action

```

We repeated this process for 20 runs, with the plots in Fig. 6.7 showing the means and standard deviations of the TD learning curves across runs. For the FFs, the thresholds were constant across features at  $\theta_d = \theta = 0.7$ . Because of the nature of the environment, all of the policies spent a significant portion of time coasting back and forth between the hills, causing their SFs to accumulate in magnitude each time states were revisited.

Given the learned representations, we then tested them by using them as features for policy evaluation in different tasks, with each task containing a different rewarded/absorbing state. Note that a crucial factor is that the representations were learned in the environment without absorbing states. This is natural, as in the real world reward may arise in the environment anywhere, and we'd like a representation that can be effective for any potential goal location.



**Figure 6.11: The FF is robust to feature dimensionality.** FF and SF representation strengths for difference feature dimensionalities between the start and goal locations for an example goal in continuous MountainCar. The vertical dashed line marks the power of the optimal policy. We can see that for all but the coarsest feature representation, the FF is highest for the policy closest to the optimal.

**Robotic reaching experiment** We used the custom JACO domain as well as the APS base code from Laskin et al. (2021), located at this link: <https://anonymous.4open.>

[science/r/urlb/README.md](https://github.com/GoogleCloudPlatform/aiplatform/blob/master/samples/colab/training/training_urls/README.md). Both the critic and actor networks were parameterized by 3-layer MLPs with ReLU nonlinearities and 1,024 hidden units. Observations were 55-dimensional with 10-dimensional features  $\phi(\cdot)$ . For all other implementation details, including learning rates, optimizers, etc. see the above link. All hyperparameters and network settings are kept constant from those provided in the linked .yaml files. All experiments were repeated for 10 random seeds.

We now describe each training phase. *Pre-training*: Agents were trained for 1M time steps on the rewardless JACO domain by maximizing the intrinsic reward

$$\begin{aligned} r^{\text{intrinsic}}(s, a, s') &= r^{\text{exploit}}(s, a, s') + r^{\text{explore}}(s, a, s') \\ &= \mathbf{w}^T \phi(s) + \log \left( 1 + \frac{1}{k} \sum_{h^{(j)} \in N_k(\phi(s'))} \|\phi(s') - \phi(s')^{(j)}\|_{n_h}^{n_h} \right), \end{aligned} \quad (6.24)$$

where  $w \in \mathbb{R}^D$ ,  $D = 10$  is a random reward vector sampled from a standard Gaussian distribution and the righthand term is a particle-based estimate of the state-based feature entropy, with  $N_k(\cdot)$  denoting the  $k$  nearest-neighbors (see Liu and Abbeel (2021) for details). In standard APS, this reward is used to train the successor features by constructing the bootstrapped target

$$y^{\text{APS}} = r^{\text{intrinsic}}(s, a, s') + \gamma \mathbf{w}^T \psi(s_{t+1}, a', w), \quad (6.25)$$

where  $a' = \arg\max_a \mathbf{w}^T \psi(s', a, w)$ . For the FF, we make the following modification:

$$y^{\text{APF}} = y^{\text{APF-exploit}} + y^{\text{explore}} \quad (6.26)$$

$$= \underbrace{\mathbf{w}^T \tilde{\phi}(s)}_{:=r^F(s)} + \gamma \mathbf{w}^T (\mathbf{1} - \tilde{\phi}(s)) V(s_{t+1}) + r^{\text{explore}}(s, a, s') + \gamma V(s_{t+1}) \quad (6.27)$$

$$= r^F(s) + r^{\text{explore}} + \gamma [\mathbf{w}^T \mathbf{1} - r^F + 1] V(s_{t+1}), \quad (6.28)$$

with  $V(s_{t+1}) = \max_{a'} \mathbf{w}^T \varphi(s_{t+1}, a', w)$ ,  $\tilde{\phi}(\cdot)$  the thresholded base features, such that  $\tilde{\phi}_d(s_t) = \mathbb{1}(\phi_d(s_t) \geq \theta_d, \{\phi_d(s_{t'})\}_{t'=0:t} < \theta_d)$ ,  $\mathbf{1}$  is the  $D$ -length vector of ones, and



$\phi(s_t) \in [0, 1]$ . Interestingly, if the features are kept in the range  $[0, 1]$ ,  $\phi(\cdot)$  can be thought of encoding a form of “soft” first feature occupancy, rather than the hard threshold given by the indicator function.

The agent is then trained with the off-policy *deep deterministic policy gradient* (DDPG; (Lillicrap et al., 2019)) algorithm, where given a stored replay buffer of transitions  $\mathcal{D} = \{(s_t, a_t, r_t, s_{t+1})\}$ , the (SF/FF) critic  $Q_\omega$  (with  $Q$  formed from either the SF or FF and  $\omega$  being the parameters) is trained to minimize the squared Bellman loss

$$\mathcal{L}_Q(\omega, \mathcal{D}) = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1}) \sim \mathcal{D}} [(y^r - Q_\omega(s_t, a_t))^2], \quad (6.29)$$

where in the pre-training phase  $y^r \in \{y^{\text{APS}}, y^{\text{APF}}\}$  (the target parameters are an exponential moving average of the weights—gradients do not flow through them). The deterministic actor  $\pi_\theta$  is trained using the deterministic policy gradient loss:

$$\mathcal{L}_\pi(\theta, \mathcal{D}) = \mathbb{E}_{s_t \sim \mathcal{D}} [Q_\phi(s_t, \pi_\theta(s_t))]. \quad (6.30)$$

*Fine-tuning:* After pre-training, the agent is fine-tuned on the target task, REACH-TOPLEFT, where the learning proceeds exactly as in the pre-training phase, but instead of intrinsic reward, the agent is given the task reward—that is,  $y^r = r_t^{\text{task}} + \gamma V(s_{t+1})$ . We performed this task-specific training for an additional 1M steps.

As additional baselines, we implemented two versions of the dynamic distance learning method of (Hartikainen et al., 2020), the original DDLUS and an additional variant DDLUS-G. In standard DDLUS, the goals  $g \in \mathcal{S}$  for the pre-training phase are generated according to

$$g^* \in \operatorname{argmax}_{g \in \mathcal{D}} d^\pi(s_0, g),$$

where  $\mathcal{D}$  is a stored set of trajectories. In (Hartikainen et al., 2020), this is useful as a mechanism for encouraging the agent to learn skills which move the agent as far as possible from the start state. However, since the object for manipulation in the fine-tuning phase of the JACO task is not typically especially far from the initial point, we also tested

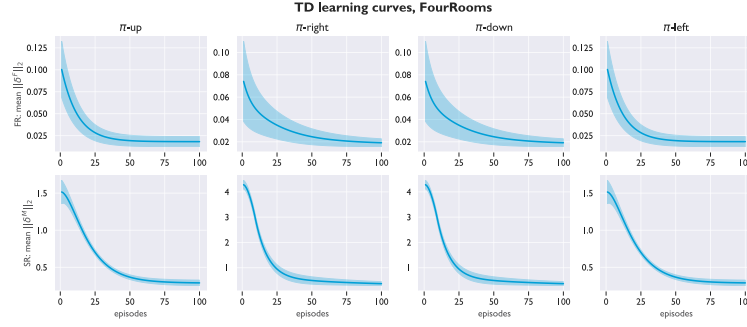
DDLUS-G, which samples goals in the same manner as APS and APF, via

$$g \sim \mathcal{N}(0, I),$$

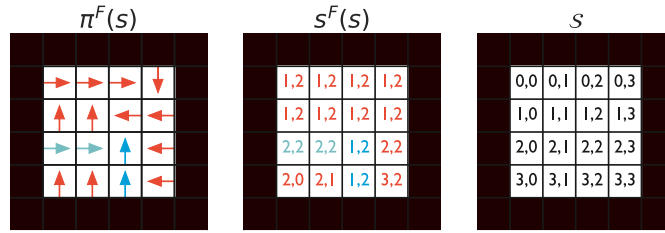
to ensure a more fair comparison. All other hyperparameters match those of [Hartikainen et al. \(2020\)](#), with the exception that the base agent is DDPG, implemented using the same framework as APS and APF, rather than SAC ([Haarnoja et al., 2018](#)). For a more detailed discussion of DDL and its relationship to the FR/FF, see Appendix [6.H](#).

In future work, it would be interesting to explore the interaction of the FF with other off-policy algorithms ([Haarnoja et al., 2018](#); [Fujimoto et al., 2018](#); [Moskovitz et al., 2021a](#)) and whether on-policy learning (e.g., with ([Schulman et al., 2017](#); [Kakade, 2002](#); [Williams, 1992](#); [Hartikainen et al., 2020](#); [Moskovitz et al., 2021a](#))) has different effects.

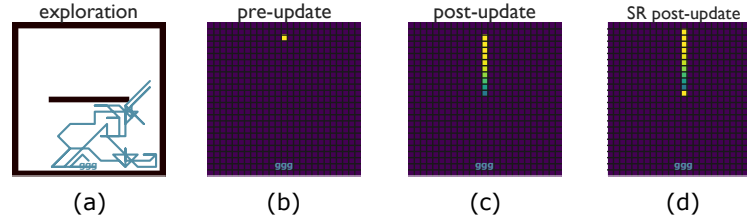
**FourRoom experiments** The FourRoom environment we used was defined on an  $11 \times 11$  gridworld in which the agent started in the bottom left corner and moved to a known goal state. The action space was  $\mathcal{A} = \{\text{up}, \text{right}, \text{down}, \text{left}\}$  with four base policies each corresponding to one of the basic actions. TD Learning curves for the base policies are depicted in Fig. [6.12](#). Once reaching the goal, a new goal was uniformly randomly sampled from the non-walled states. At each time step, the agent received as state input only the index of the next square it would occupy. Each achieved goal netted a reward of +50, hitting a wall incurred a penalty of  $-1$  and kept the agent in the same place, and every other action resulted in 0 reward. There were 75 time steps per episode—the agent had to reach as many goals as possible within that limit. The discount factor  $\gamma$  was 0.95, and the FR learning rate was 0.05. In order to learn accurate FRs for each policy, each policy was run for multiple start states in the environment for 50 episodes prior to training. FRP (for different values of  $K$ ), GPI, and VI were each run for 100 episodes. VI was given the true transition matrix and reward vector in each case. In the stochastic case, for each level of transition noise  $\epsilon = 0.0, 0.1, 0.2, \dots, 1.0$ , both VI and FRP were run to convergence ( $\approx 180$  iterations for VI, 3 iterations for FRP) and then tested for 100 episodes.



**Figure 6.12: FOURROOMS learning curves.** FOURROOMS base policies learning curves (average L2 norm of TD errors over 10 runs; shaded area is one standard deviation); top row is for FRs, bottom is for SRs.



**Figure 6.13: Implicit planning output.** (Left) The planning policies  $\pi^F(s)$  that the agent will elect to follow in each state en route to the goal (see Fig. 6.3(a)). Arrows denote the action taken by the chosen policy in each state. (Middle) The (row, column) subgoals for each state  $s^F(s)$ . (Right) The state space  $\mathcal{S}$ , for reference.



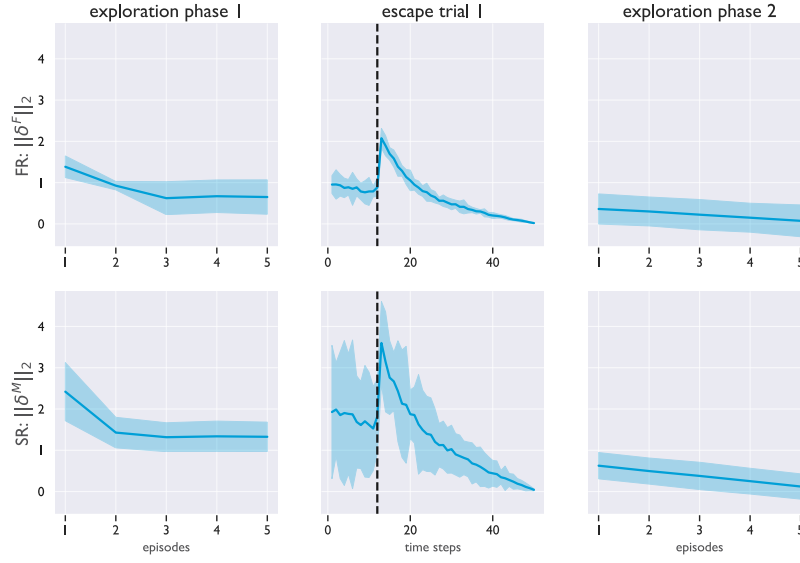
**Figure 6.14: Exploration and escape** (a) A sample trajectory from the “exploration phase” starting from the shelter. (b) Because the agent starts from the shelter during exploration, the first time it is tested starting from the top of the grid, its FR for the down policy for that state is still at initialization. (c) After updating its FR during testing and further exploration, the FR for the down policy from the start state is accurate, stopping at the barrier. (d) We can see that if we were to use the SR instead, the value in the state above the wall would accumulate when it gets stuck.

**Escape experiments** The escape experiments were modeled as a  $25 \times 25$  gridworld with eight available actions,

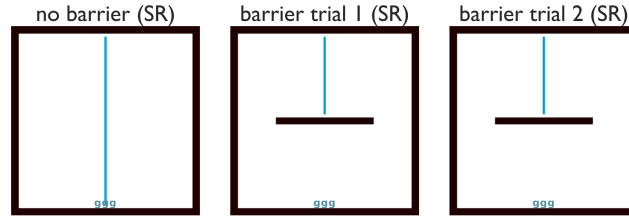
$$\mathcal{A} = \{\text{up, right, down, left, up-right, down-right, down-left, up-left}\} \quad (6.31)$$

and a barrier one-half the width of the grid optionally present in the center of the room. The discount factor  $\gamma$  was 0.99, and the FR learning rate was 0.05. In test settings, the agent started in the top central state of the grid, with a single goal state directly opposite. At each time step, the agent receives only a number corresponding to the index of its current state as input. The base policy set  $\Pi$  consisted of eight policies, one for each action in  $\mathcal{A}$ . Escape trials had a maximum of 50 steps, with termination if the agent reached the goal state. In (Shamash et al., 2021a), mice were allowed to explore the room starting from the shelter location. Accordingly, during “exploration phases,” the agent started in the goal state and randomly selected a base policy at each time step, after which it updated the corresponding FR. Each exploration phase consisted of 5 episodes—a sample trajectory is shown in Fig. 6.14(a). After the first exploration phase, the agent started from the canonical start state and ran FRP to reach the goal state. Because most of its experience was in the lower half of the grid, the FRs for the upper half were incomplete (Fig. 6.14(b)), and we hypothesized that in this case, the mouse should either i) default to a policy which would take it to the shelter in the area of the room which it knew well (the down policy) or ii) default to a policy which would simply take it away from the threat (again the down policy). During the first escape trial, the agent selects the down policy repeatedly, continuing to update its FRs during the testing phase. Upon reaching the wall and getting stuck, the FR for the down policy is eventually updated enough that re-planning with FRP produces a path around the barrier. After updating its FR during the first escape trial and during another exploration period, the FRs for the upper half of the grid are more accurate (Fig. 6.14(c)) and running FRP again from the start state produces a faster path around the barrier on the second escape trial. TD learning curves for the experiment (repeated with the SR for completeness) are plotted in Fig. 6.15.

For completeness, we repeated this experiment with the SR. In this case, the planning algorithm is ill-defined for  $K > 0$ , so we default to GPI ( $K = 0$ ). As expected, without the barrier, the down policy is selected and the goal is reached (Fig. 6.16 left). However, when there is a barrier, while the SR updates when the agent hits it (Fig. 6.12), since there is no single policy that can reach the shelter, GPI fails to find a path around the barrier (Fig. 6.16 middle, right).



**Figure 6.15: Escape learning curves.** Learning curves (norms of TD errors) for the first exploration phase, the first escape trial, and the second exploration phase for the “down” policy. The vertical dotted lines in the escape trial mark the time step at which the agent encounters the barrier. This causes a temporary jump in the TD errors, as representation learning did not reflect the wall at this point. The top row consists of FR results and the bottom row is from SRs, averaged over 10 runs. The shading represents one standard deviation.



**Figure 6.16: An SR cannot effectively escape under the same conditions as an FR agent.**

## Appendix 6.D: Additional Proofs

Below are the proofs for Proposition 6.3.1 and Proposition 6.3.2, which are restated below.

**Proposition 3.1 (Contraction).** *Let  $\mathcal{G}^\pi$  be the operator as defined in Definition 6.3.2 for some stationary policy  $\pi$ . Then for any two matrices  $F, F' \in \mathbb{R}^{|S| \times |S|}$ ,*

$$|\mathcal{G}^\pi F(s, s') - \mathcal{G}^\pi F'(s, s')| \leq \gamma |F(s, s') - F'(s, s')|, \quad (6.32)$$

*with the difference equal to zero for  $s = s'$ .*

*Proof.* For  $s \neq s'$  we have

$$|(\mathcal{G}^\pi - \mathcal{G}^\pi F')_{s,s'}| = \gamma |(P^\pi F - P^\pi F')_{s,s'}| = \gamma |P^\pi (F - F')_{s,s'}| \leq \gamma |(F - F')_{s,s'}|,$$

where we use the notation  $X_{s,s'}$  to mean  $X(s, s')$ , and the inequality is due to the fact that every element of  $P^\pi(F - F')$  is a convex average of  $F - F'$ . For  $s = s'$ , we trivially have  $|(\mathcal{G}^\pi F - \mathcal{G}^\pi F')_{s,s}| = |1 - 1| = 0$ .  $\square$

**Proposition 6.D.1** (Convergence). *Under the conditions assumed above, set  $F^{(0)} = I_{|S|}$ . For  $k = 0, 1, \dots$ , suppose  $F^{(k+1)} = \mathcal{G}^\pi F^{(k)}$ . Then*

$$|F^{(k)}(s, s') - F^\pi(s, s')| < \gamma^k \quad (6.33)$$

for  $s \neq s'$  with the difference for  $s = s'$  equal to zero  $\forall k$ .

*Proof.* We have, for  $s \neq s'$  and using the notation  $X_{s,s'} = X(s, s')$  for a matrix  $X$ ,

$$\begin{aligned} |(F^{(k)} - F^\pi)_{s,s'}| &= |(\mathcal{G}^k F^{(0)} - \mathcal{G}^k F^\pi)_{s,s'}| \\ &\leq \gamma^k |(F^{(0)} - F^\pi)_{s,s'}| \quad (\text{Proposition 6.3.1}) \\ &= \gamma^k F^\pi(s, s') < \gamma^k \quad (F^\pi(s, s') \in [0, 1)). \end{aligned} \quad (6.34)$$

For  $s = s'$ ,  $|(F^{(k)} - F^\pi)_{s,s}| = |1 - 1| = 0 \forall k$ .  $\square$

Below is the proof of Proposition 6.4.1 which is restated below.

**Proposition 4.1** (Planning optimality). *Consider a deterministic, finite MDP with a single goal state  $s_g$ , and a policy set  $\Pi$  composed of policies  $\pi : \mathcal{S} \rightarrow \mathcal{A}$ . We make the following coverage assumption, there exists some sequence of policies that reaches  $s_g$  from a given start state  $s_0$ . Under these conditions, Algorithm 7 converges such that  $\Gamma(s_0) = \gamma^{L_\pi^*}$ , where  $L_\pi^*$  is the shortest path length from  $s_0$  to  $s_g$  using  $\pi \in \Pi$ .*

*Proof.* Since the MDP is deterministic, we use a deterministic transition function  $\rho : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ . We proceed by induction on  $L_\pi^*$ .

Base case:  $L_\pi^* = 1$

If  $L_\pi^* = 1$ ,  $s_0$  must be one step from  $s_g$ . The coverage assumption guarantees that  $\exists \pi \in$

$\Pi$  such that  $\rho(s_0, \pi(s_0)) = s_g$ . Note also that when both the MDP and policies in  $\Pi$  are deterministic,  $F^\pi(s, s') = \gamma^{L_\pi}$ , where  $L_\pi$  is the number of steps from  $s$  to  $s'$  under  $\pi$ , and we use the abuse of notation  $L_\pi = \infty$  if  $\pi$  does not reach  $s'$  from  $s$ .

Then following Algorithm 7,

$$\Gamma_1(s_0) = \max_{\pi \in \Pi} F^\pi(s_0, s_g) = \gamma \quad (\text{guaranteed by coverage of } \Pi)$$

$$\Gamma_1(s_g) = \max_{\pi \in \Pi} F^\pi(s_g, s_g) = 1 \quad (\text{by definition of } F^\pi).$$

Moreover,

$$\begin{aligned} \Gamma_2(s_0) &= \max_{\pi \in \Pi, s' \in \mathcal{S}} F^\pi(s_0, s') \Gamma_1(s') \\ &= \max_{\pi \in \Pi} \{F^\pi(s_0, s_0) \Gamma_1(s_0), F^\pi(s_0, s_g) \Gamma_1(s_g)\} \\ &= \max\{1 \cdot \gamma, \gamma \cdot 1\} \\ &= \gamma. \end{aligned}$$

Then  $\Gamma_2(s) = \Gamma_1(s) \forall s$  and Algorithm 7 terminates. Thus,  $\Gamma(s_0) = \gamma = \gamma^{L_\pi^*}$  and the base case holds.

Induction step: Assume Proposition 6.4.1 holds for  $L_\pi^* = L$

Given the induction assumption, we now need to show that Proposition 6.4.1 holds for  $L_\pi^* = L + 1$ . By the induction and coverage assumptions, there must exist at least one state within one step of  $s_g$  that the agent can reach in  $L$  steps, such that the discount for this state or states is  $\gamma^L$ . Moreover, the coverage assumption guarantees that  $\exists \pi \in \Pi$  such that for at least one such state  $s_L$ ,  $\rho(s_L, \pi(s_L)) = s_g$ .

Then this problem reduces to the base case—that is, Algorithm 7 will select the policy  $\pi \in \Pi$  that transitions directly from  $s_L$  to  $s_g$ —and the proof is complete.  $\square$

## Appendix 6.E: Explicit Planning

Below we describe a procedure for constructing an explicit plan using  $\pi^F$  and  $s^F$ .

**Algorithm 8** ConstructPlan

---

```

1: input: goal state  $s_g$ , planning policy  $\pi^F$ , subgoals  $s^F$ 
2:  $\Lambda \leftarrow []$  {init. plan}
3:  $s \leftarrow s_0$  {begin at start state}
4: while  $s \neq s_g$  do
5:    $\Lambda.append((\pi^F(s), s^F(s)))$  {add policy-subgoal pair for current state to plan}
6:    $s \leftarrow s^F(s)$ 
7: end while
8: return  $\Lambda$ 

```

---

**Appendix 6.F: FRP with Multiple Goals**

Here we consider the application of FRP to environments with multiple goals  $\{g_1, \dots, g_n\}$ . To find the shortest path between them given the base policy set  $\Pi$ , we first run FRP for each possible goal state in  $\{g_i\}$ , yielding an expected discount matrix  $\Gamma^\Pi \in [0, 1]^{|S| \times n}$ , such that  $\Gamma^\Pi(s, g_i)$  is the expected discount of the shortest path from state  $s$  to goal  $g_i$ . We denote by  $g_\sigma = [s_0, \sigma(g_1), \sigma(g_2), \dots, \sigma(g_n)]$  a specific ordering of the goals in  $\{g_i\}$  starting in  $s_0$ . The expected discount of a sequence of goals is then

$$\Xi(g_\sigma) = \prod_{i=1}^n \Gamma^\Pi(g_\sigma(i-1), g_\sigma(i)), \quad (6.35)$$

with the optimal goal ordering  $g_{\sigma^*}$  given by

$$g_{\sigma^*} = \operatorname{argmax}_{g_\sigma \in G} \Xi(g_\sigma), \quad (6.36)$$

where  $G$  is the set of all possible permutations of  $\{g_i\}$ , of size  $n!$ . This is related to a form of the travelling salesman problem, and we refer the reader to [Zahavy et al. \(2019\)](#) for a formal investigation of the use of local policies to construct shortest paths. Fortunately, in most settings we don't expect the number of goals  $n$  to be particularly large.

**Appendix 6.G: Connections to options**

The options framework ([Sutton et al., 1999](#)) is a method for temporal abstraction in RL, wherein an option  $\omega$  is defined as a tuple  $(\pi_\omega, \tau_\omega)$ , where  $\pi_\omega$  is a policy and  $\tau_\omega \in \Delta(\mathcal{S})$  is a state-dependent termination distribution (or function, if deterministic). Executing



an option at time  $t$  entails sampling an action  $a_t \sim \pi_\omega(\cdot|s_t)$  and ceasing execution of  $\pi_\omega$  at time  $t + 1$  with probability  $\tau_\omega(s_{t+1})$ . The use of options enlarges an MDP's action space, whereby a higher-level policy selects among basic, low-level actions and options.

Options are connected to FRP (Algorithm 7) in that by outputting a set of policies and associated subgoals  $\{(\pi^F, s^F)\}$ , FRP effectively converts each base policy to an option with a deterministic termination function, i.e., the agent will follow  $\pi^F$  until terminating at  $s^F$ . One of the key difficulties in the options literature is how to learn the best options to add to the available action set. For the class of problems considered in this paper, FRP then provides a framework for generating *optimal* (in the sense of finding the fastest path to a goal) options from a set of standard policies, subject to the fulfillment of the coverage assumption. Importantly, the associated FRs (which can be learned via simple TD updating) can be reused across tasks, so that FRP can re-derive optimal options for a new goal.

FRP can also be seen as related to the work of Silver and Ciosek (2012), which demonstrates that value iteration performed on top of a set of task-specific options converges more quickly than value iteration performed on the default state space of the MDP. One critical difference to note is that the FR/FRP is transferable to any MDP with shared transition dynamics. While value iteration on a set of options for a given MDP is more efficient than value iteration performed directly on the underlying MDP, this process must be repeated every time the reward function changes. However, the FR enables an implicit representation of the transition dynamics to be cached and reused.

## Appendix 6.H: Further connections to related work

We now describe the connection between the FR/FF and several related approaches in the literature. Table 6.4 summarizes a high-level view of these connections.

**The Dynamic Distance Function** The *dynamic distance function* (DDF; (Hartikainen et al., 2020)) is defined as

$$d^\pi(s, s') = \mathbb{E}_\pi \left[ \sum_{k=0}^{j-1} \gamma^k c(s_{t+k}, s_{t+k+1}) \middle| s_t = s, s_j = s' \right], \quad (6.37)$$

	FR/FF	SR/SF	DDL	TDM	DG
On- v. Off-policy	Both	Both	On-policy	Off-policy	Off-policy
Eval. v. Control	Eval.	Eval.	Eval.	Control	Control
Finite v. Inf. Horizon	Both	Both	Finite	Finite	Finite
State representation?	Yes	Yes	No	No	No

**Table 6.4:** Overview of basic points of comparison between the FR/FF, SR/SF (Dayan, 1993; Barreto et al., 2017), DDL (Hartikainen et al., 2020), TDMs (Pong et al., 2018), and DG (Kaelbling, 1993).

where  $c : \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}$  is a local cost function. In practice,  $c(s_{t+k}, s_{t+k+1}) = \gamma = 1$ , giving

$$d^\pi(s, s') = \mathbb{E}_\pi \left[ \sum_{k=0}^{j-1} 1 \mid s_t = s, s_j = s' \right]. \quad (6.38)$$

In practice,  $d^\pi(s, s')$  is parameterized via a neural network with parameters  $\psi$  trained on-policy from full trajectories  $\tau$  using the loss

$$\mathcal{L}(\psi) = \frac{1}{2} \mathbb{E}_{\tau \sim \mathcal{D}, i \sim [0, T], j \sim [i, T]} (d_\psi^\pi(s_i, s_j) - (j - i))^2, \quad (6.39)$$

where  $\mathcal{D}$  is a buffer of stored trajectories. On a downstream navigation task, the agent policy is trained to minimize

$$\mathcal{L}_\pi(\phi) = \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t d_\psi^\pi(s_t, g), \right], \quad (6.40)$$

where  $g \in \mathcal{S}$  is a goal state and  $\phi$  are the policy parameters.

There are several important differences between the DDF and the FR, both in theory and practical application.

This form of the DDF, as discussed by Hartikainen et al. (2020), is also naturally restricted to on-policy learning from full trajectories, and cannot be updated off-policy and/or via one-step temporal difference learning. A natural additional consequence is that it is only applicable to finite-horizon MDPs. This confinement to finite horizons is significant because it leads to the conditioning problem described by Hartikainen et al.

(2020). This problem occurs because the DDF is conditioned on the policy successfully reaching the goal state—when this doesn’t happen, it can lead to significant value estimation errors.

A second difference is that the policy is trained, via Eq. (6.40), to minimize the *cumulative* discounted distance to the goal rather than via greedy distance minimization.

Another difference is that Eqs. (6.37) and (6.38) don’t explicitly require time step  $j$  to be the first time the agent enters  $s'$ , although this may have been the authors’ intention. In fact, this definition is more closely related to the SR than the FR (with equality in the infinite horizon setting), but the implementation of the DDF in practice is more closely related to the FR. It’s also important to note that when  $\gamma = 1$ , the number of steps  $k$  appears linearly within the expectation rather than exponentially (this is important, as in general  $\gamma^{\mathbb{E}[k]} \neq \mathbb{E}[\gamma^k]$ ).

Another significant difference, then, is that the DDF as presented cannot be in a meaningful sense be considered a state *representation*, but rather a function mapping pairs of states to expected distances. That is,  $d^\pi(s, \cdot)$  has no meaningful semantics when implemented as a mapping  $d^\pi : \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}$  and trained using Eq. (6.39). In practice, it is used to support policy optimization, rather than evaluation. In contrast, the FF representation, like the SF representation, maps a single state to a fixed length vector-valued encoding,  $\varphi^\pi : \mathcal{S} \rightarrow \mathbb{R}^d$ . This difference has significant implications for the applications of the DDF. In particular, it is unclear how the DDF could be used as an exploration bonus in the manner of the FF/SF, and the scalar value would make it challenging to implement the type of parallel updates useful for efficient planning.

**Dynamic Goal Learning** Another related approach is that of *dynamic goal* (DG) learning (Kaelbling, 1993), a method for optimal control related to  $Q$ -learning. The optimal DG function  $G^* : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$  is defined recursively for a goal state  $g \in \mathcal{S}$  as

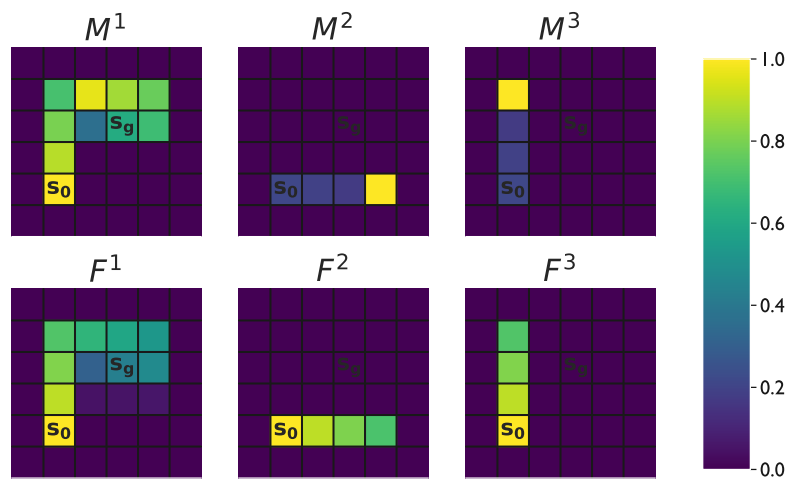
$$G^*(s, a, g) = 1 + \mathbb{E}_{s' \sim P(\cdot | s, a)} \left[ \min_{a' \in \mathcal{A}} G^*(s', a', g) \right], \quad (6.41)$$

where  $G^*(g, a, g) := 0$ . There are several important differences between this approach and the FR. First, as mentioned above, DG learning is a method for optimal control, rather than policy *evaluation*. That is,  $D^*(s, a, g)$  converges to the expected number

of steps for the optimal policy for reaching  $g$  starting from  $s$ . It cannot be reused for policy evaluation for a policy  $\pi \neq \pi^*$ , and also implicitly assumes that  $g$  is reachable in finite time. DG learning is thus susceptible to the same conditioning problem discussion above for the DDF, and is only studied by Kaelbling (1993) in a gridworld environment. A second difference is that the DG function scales linearly with  $k$ , the number of steps for the optimal policy to reach  $g$ , while the FR scales exponentially at a rate of  $\gamma$ . This is important, as we can note (with a slight abuse of notation) in general that  $\gamma^{\mathbb{E}_\pi[k]} \neq \mathbb{E}_\pi[\gamma^k]$ —it is nontrivial to recover the FR for the optimal policy from the DG function (and vice-versa).

**Temporal Difference Models** *Temporal difference models* (TDMs; (Pong et al., 2018)) are another related approach. TDMs are an optimal control method motivated by the observation that goal-conditioned  $Q$ -functions can be used to construct an implicit dynamics model when the discount factor  $\gamma = 0$ . For  $\gamma > 0$ , the authors introduce a horizon variable  $\tau$  to interpolate between model-based and model-free learning. Like the FR, this formulation allows agent to consider multiple goals in parallel. As in DG learning, and unlike the FR, TDMs represent a method for optimal control, not evaluation. The emphasis is on efficiently improving a policy, rather than learning a state representation which can be leveraged for multiple uses (e.g., policy evaluation or as an exploration bonus). Another difference from the FR is that, like the DDF and DG learning, TDMs rely on a finite-horizon setting.

## Appendix 6.I: FR vs. SR Visualization

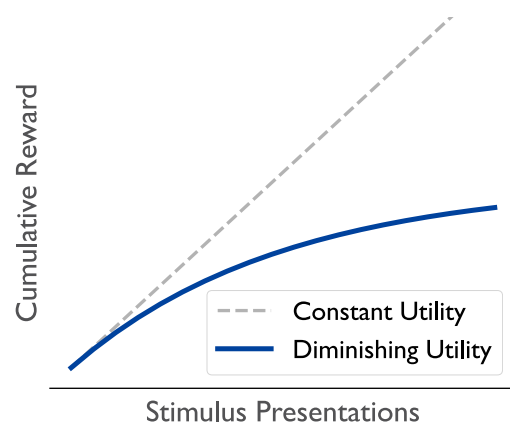


**Figure 6.17: SR vs. FR visualization** The SRs and FRs from the start state for the policies in Fig. Fig. 6.3. For the SRs (Fig. 6.17, top row), we can see that states that are revisited (or in which the policy simply stays) are more highly weighted, while for the FRs (Fig. 6.17, bottom row), the magnitude of  $F(s_0, s')$  is higher for states  $s'$  that are closer along the path taken by the policy.

## Chapter 7

# The $\lambda$ -Occupancy Representation

The previous chapter introduced a state representation which can be used to efficiently evaluate policies and plan in situations for which transitions are consistent across tasks and reward is depleted after its initial presentation. This setting, along with the standard MDP task description in which rewards can be accessed an infinite number of times, represents two extremes of reward persistence. However, the degree to which stimuli are rewarding often diminishes at varying rates with repeated exposures.



**Figure 7.1:** Diminishing rewards.

The second ice cream cone rarely tastes as good as the first, and once all the most accessible brambles have been picked, the same investment of effort yields less fruit. In everyday life, the availability and our enjoyment of stimuli is sensitive to our past interactions with them. Thus, to evaluate different courses of action and act accordingly, we might expect our brains to form representations sensitive to the non-stationarity of rewards. Evidence in fields from behavioral economics (Kahneman and Tversky, 1979; Rabin, 2000) to neuroscience (Pine et al., 2009) supports this hypothesis. Surprisingly, however, most of reinforcement learning (RL) takes place under the assumptions of the Markov Decision Process (MDP; Puterman, 1994), where rewards and optimal decision-making remain stationary.

In this chapter, adapted from Moskowitz et al. (2024), we seek to bridge this gap by studying the phenomenon of *diminishing marginal utility* (Gossen and Blitz, 1983) in the context of RL. Diminishing marginal utility (DMU) is the subjective phenomenon by which repeated exposure to a rewarding stimulus reduces the perceived utility one experiences. While DMU is thought to have its roots in the maintenance of homeostatic equilibrium (too much ice cream can result in a stomach ache), it also manifests itself in domains in which the collected rewards are abstract, such as economics (\$10 vs. \$0 is perceived as a bigger difference in value than \$1,010 vs. \$1,000), where it is closely related to risk aversion (Arrow, 1996; Pratt, 1978). While DMU is well-studied in other fields, relatively few RL studies have explored diminishing reward functions (Wisniewski et al., 2023; Shuvaev et al., 2020), and, to our knowledge, none contain a formal analysis of DMU within RL. Here, we seek to characterize both its importance and the challenge it poses for current RL approaches (Section 7.1).

Surprisingly, we find that evaluating policies under diminishing rewards requires agents to learn a novel state representation which we term the  $\lambda$  representation ( $\lambda$ R, Section 7.2). The  $\lambda$ R generalizes several state representations from the RL literature: the *successor representation* (SR; Dayan, 1993), the *first-occupancy representation* (FR; Moskowitz et al., 2022c), and the *forward-backward representation* (FBR; Touati and Olivier, 2021b), adapting them for non-stationary environments. Interestingly, despite the non-stationarity of the underlying reward functions, we show that the  $\lambda$ R still admits a Bellman recursion, allowing for efficient computation via dynamic programming (or approximate dynamic programming) and prove its convergence. We demonstrate the scalability of the  $\lambda$ R to large and continuous state spaces (Section 7.2.1), show that it supports policy evaluation, improvement, and composition (Section 7.3), and show that the behavior it induces is consistent with optimal foraging theory (Section 7.4).

## 7.1 Diminishing Marginal Utility

**Problem Statement** Motivated by DMU’s importance in decision-making, our goal is to understand RL in the context of the following class of non-Markov reward functions:

$$r_\lambda(s, t) = \lambda(s)^{n(s,t)} \bar{r}(s), \quad \lambda(s) \in [0, 1], \quad (7.1)$$

where  $n(s, t) \in \mathbb{N}$  is the agent’s visit count at  $s$  up to time  $t$  and  $\bar{r}(s)$  describes the reward at the first visit to  $s$ .  $\lambda(s)$  therefore encodes the extent to which reward diminishes after each visit to  $s$ . Note that for  $\lambda(s) = \lambda = 1$  we recover the usual stationary reward given by  $\bar{r}$ , and so this family of rewards strictly generalizes the stationary Markovian rewards typically used in RL.

**DMU is Challenging** An immediate question when considering reward functions of this form is whether or not we can still define a Bellman equation over the resulting value function. If this is the case, standard RL approaches still apply. However, the following result shows otherwise.

**Lemma 7.1.1** (Impossibility; Informal). *Given a reward function of the form Eq. (7.1), it is impossible to define a Bellman equation solely using the resulting value function and immediate reward.*

We provide a more precise statement, along with proofs for all theoretical results, in Appendix 7.B. This result means that we can’t write the value function corresponding to rewards of the form Eq. (7.1) recursively only in terms of rewards and value in an analogous manner to Eq. (7.18). Nonetheless, we found that it *is* in fact possible to derive a recursive relationship in this setting, but only by positing a novel state representation that generalizes the SR and the FR, which we term the  $\lambda$  *representation* ( $\lambda$ R). In the following sections, we define the  $\lambda$ R, establish its formal properties, and demonstrate its necessity for RL problems with diminishing rewards.

## 7.2 The $\lambda$ Representation

**A Representation for DMU** We now derive the  $\lambda$ R by decomposing the value function for rewards of the form Eq. (7.1) and show that it admits a Bellman recursion. To



simplify notation, we use a single  $\lambda$  for all states, but the results below readily apply to non-uniform  $\lambda$ . We have

$$V^\pi(s) = \mathbb{E} \left[ \sum_{k=0}^{\infty} \gamma^k r_\lambda(s_{t+k}, k) \middle| s_t = s \right] = \bar{\mathbf{r}}^\top \underbrace{\mathbb{E} \left[ \sum_{k=0}^{\infty} \gamma^k \lambda^{n_t(s_{t+k}, k)} \mathbf{1}(s_{t+k}) \middle| s_t = s \right]}_{\triangleq \Phi_\lambda^\pi(s)}, \quad (7.2)$$

where we call  $\Phi_\lambda^\pi(s)$  the  $\lambda$  representation ( $\lambda$ R), and  $n_t(s, k) \triangleq \sum_{j=0}^{k-1} \mathbb{1}(s_{t+j} = s)$ , is the number of times state  $s$  is visited from time  $t$  up to—but not including—time  $t + k$ . Formally:

**Definition 7.2.1** ( $\lambda$ R). *For an MDP with finite  $\mathcal{S}$  and  $\lambda \in [0, 1]^{|\mathcal{S}|}$ , the  $\lambda$  representation is given by  $\Phi_\lambda^\pi$  such that*

$$\Phi_\lambda^\pi(s, s') \triangleq \mathbb{E} \left[ \sum_{k=0}^{\infty} \lambda(s')^{n_t(s', k)} \gamma^k \mathbb{1}(s_{t+k} = s') \middle| s_t = s \right] \quad (7.3)$$

where  $n_t(s, k) \triangleq \sum_{j=0}^{k-1} \mathbb{1}(s_{t+j} = s)$  is the number of times state  $s$  is visited from time  $t$  until time  $t + k - 1$ .

We can immediately see that for  $\lambda = 0$ , the  $\lambda$ R recovers the FR (we take  $0^0 = 1$ ), and for  $\lambda = 1$ , it recovers the SR. For  $\lambda \in (0, 1)$ , the  $\lambda$ R interpolates between the two, with higher values of  $\lambda$  reflecting greater persistence of reward in a given state or state-action pair and lower values of  $\lambda$  reflecting more ephemeral rewards.

The  $\lambda$ R admits a recursive relationship:

$$\begin{aligned} \Phi_\lambda^\pi(s, s') &= \mathbb{E} \left[ \sum_{k=0}^{\infty} \lambda^{n_t(s', k)} \gamma^k \mathbb{1}(s_{t+k} = s') \middle| s_t = s \right] \\ &\stackrel{(i)}{=} \mathbb{E} \left[ \mathbb{1}(s_t = s') + \lambda^{n_t(s', 1)} \gamma \sum_{k=1}^{\infty} \lambda^{n_{t+1}(s', k)} \gamma^{k-1} \mathbb{1}(s_{t+k} = s') \middle| s_t = s \right] \\ &= \mathbb{1}(s_t = s') (1 + \gamma \lambda \mathbb{E}_{s_{t+1} \sim p^\pi} \Phi_\lambda^\pi(s_{t+1}, s')) \\ &\quad + \gamma (1 - \mathbb{1}(s_t = s')) \mathbb{E}_{s_{t+1} \sim p^\pi} \Phi_\lambda^\pi(s_{t+1}, s'), \end{aligned} \quad (7.4)$$

where (i) follows from  $n_t(s', k) = n_t(s', 1) + n_{t+1}(s', k-1)$ . A more detailed derivation is provided in Appendix [7.A](#). Thus, we can define a tractable Bellman operator:

**Definition 7.2.2** ( $\lambda$ R Operator). *Let  $\Phi \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$  be an arbitrary real-valued matrix, and let  $\mathcal{G}_\lambda^\pi$  denote the  $\lambda$ R Bellman operator for  $\pi$ , such that*

$$\mathcal{G}_\lambda^\pi \Phi \triangleq I \odot (\mathbf{1}\mathbf{1}^\top + \gamma\lambda P^\pi \Phi) + \gamma(\mathbf{1}\mathbf{1}^\top - I) \odot P^\pi \Phi, \quad (7.5)$$

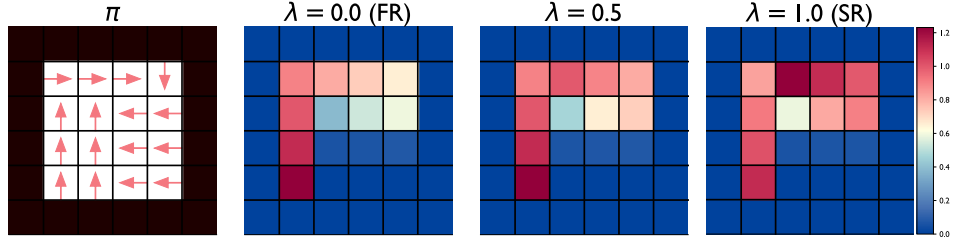
where  $\odot$  denotes elementwise multiplication and  $I$  is the  $|\mathcal{S}| \times |\mathcal{S}|$  identity matrix. In particular, for a stationary policy  $\pi$ ,  $\mathcal{G}_\lambda^\pi \Phi_\lambda^\pi = \Phi_\lambda^\pi$ .

The following result establishes that successive applications of  $\mathcal{G}_\lambda^\pi$  converge to the  $\lambda$ R.

**Proposition 7.2.1** (Convergence). *Under the conditions assumed above, set  $\Phi^{(0)} = (1 - \lambda)I$ . For  $k = 1, 2, \dots$ , suppose that  $\Phi^{(k+1)} = \mathcal{G}_\lambda^\pi \Phi^{(k)}$ . Then*

$$|(\Phi^{(k)} - \Phi_\lambda^\pi)_{s,s'}| \leq \frac{\gamma^{k+1}}{1 - \lambda\gamma}.$$

While such analysis is fairly standard in MDP theory, it is noteworthy that the analysis extends to this case despite Lemma [7.1.I](#). That is, for a ethologically relevant class of *non-Markovian* reward functions, it is possible to define a Markovian Bellman operator and prove that repeated applications of it converge to the desired representation. Furthermore, unlike in the stationary reward case, where decomposing the value function in terms of the reward and the SR is “optional” to perform prediction or control, in this setting the structure of the problem *requires* that this representation be learned. To get an intuition for the  $\lambda$ R consider the simple gridworld presented in Fig. [7.2](#), where we visualize the  $\lambda$ R for varying values of  $\lambda$ . For  $\lambda = 0$ , the representation recovers the FR, encoding the expected discount at first occupancy of each state, while as  $\lambda$  increases, effective occupancy is accumulated accordingly at states which are revisited. We offer a more detailed visualization in Fig. [7.10](#).



**Figure 7.2: The  $\lambda$ R interpolates between the FR and the SR.** We visualize the  $\lambda$ Rs of the bottom left state for the depicted policy for  $\lambda \in \{0.0, 0.5, 1.0\}$ .

### 7.2.1 Continuous State Spaces

When the state space is large or continuous, it becomes impossible to use a tabular representation, and we must turn to function approximation. There are several ways to approach this, each with their own advantages and drawbacks.

**Feature-Based Representations** For the SR and the FR, compatibility with function approximation is most commonly achieved by simply replacing the indicator functions in their definitions with a *base feature function*  $\phi : \mathcal{S} \mapsto \mathbb{R}^D$  to create *successor features* (SFs; Barreto et al., 2017, 2020) and *first-occupancy features* (FFs; Moskovitz et al., 2022c), respectively. The intuition in this case is that  $\phi$  for the SR and the FR is just a one-hot encoding of the state (or state-action pair), and so for cases when  $|\mathcal{S}|$  is too large, we can replace it with a compressed representation. That is, we have the following definition

**Definition 7.2.3** (SFs). *Let  $\phi : \mathcal{S} \mapsto \mathbb{R}^D$  be a base feature function. Then, the successor feature (SF) representation  $\varphi_1^\pi : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}^D$  is defined as  $\varphi_1^\pi(s, a) \triangleq \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k \phi(s_{t+k}) \middle| s_t, a_t \right]$  for all  $s, a \in \mathcal{S} \times \mathcal{A}$ .*

One key fact to note, however, is that due to this compression, all notion of state “occupancy” is lost and these representations instead measure feature accumulation. For any feasible  $\lambda$  then, it is most natural to define these representations using their recursive forms:

**Definition 7.2.4** ( $\lambda$ F). *For  $\lambda \in [0, 1]$  and bounded base features  $\phi : \mathcal{S} \mapsto [0, 1]^D$ , the  $\lambda$ -feature ( $\lambda$ F) representation of state  $s$  is given by  $\varphi_\lambda^\pi$  such that*

$$\varphi_\lambda^\pi(s) \triangleq \phi(s) \odot (1 + \gamma \lambda \mathbb{E}_{s' \sim p^\pi(\cdot|s)} \varphi_\lambda^\pi(s')) + \gamma(1 - \phi(s)) \odot \mathbb{E}_{s' \sim p^\pi(\cdot|s)} \varphi_\lambda^\pi(s'). \quad (7.6)$$

In order to maintain their usefulness for policy evaluation, the main requirement of the base features is that the reward should lie in their span. That is, a given feature function  $\phi$  is most useful for an associated set of reward functions  $\mathcal{R}$ , given by

$$\mathcal{R} = \{r \mid \exists \mathbf{w} \in \mathbb{R}^D \text{ s.t. } r(s, a) = \mathbf{w}^\top \phi(s, a) \forall s, a \in \mathcal{S} \times \mathcal{A}\}. \quad (7.7)$$

However, Barreto et al. (2018) demonstrate that good performance can still be achieved for an arbitrary reward function as long as it's sufficiently close to some  $r \in \mathcal{R}$ .

**Set-Theoretic Formulations** As noted above, computing expectations of accumulated abstract features is unsatisfying because it requires that we lose the occupancy-based interpretation of these representations. It also restricts the agent to reward functions which lie within  $\mathcal{R}$ . An alternative approach to extending the SR to continuous MDPs that avoids this issue is the *successor measure* (SM; Blier and Ollivier, 2018), which treats the distribution of future states as a measure over  $\mathcal{S}$ :

$$M^\pi(s, a, X) \triangleq \sum_{k=0}^{\infty} \gamma^k \mathbb{P}(s_{t+k} \in X \mid s_t = s, a_t = a, \pi) \quad \forall X \subset \mathcal{S} \text{ measurable}, \quad (7.8)$$

which can be expressed in the discrete case as  $M^\pi = I + \gamma P^\pi M^\pi$ . In the continuous case, matrices are replaced by their corresponding measures. Note that SFs can be recovered by integrating:  $\varphi_1^\pi(s, a) = \int_{\mathcal{S}'} M^\pi(s, a, ds') \phi(s')$ . We can define an analogous object for the  $\lambda$ R as follows

$$\Phi_\lambda^\pi(s, X) \triangleq \sum_{k=0}^{\infty} \lambda^{n_t(X, k)} \gamma^k \mathbb{P}(s_{t+k} \in X \mid s_t = s, \pi) \quad (7.9)$$

where  $n_t(X, k) \triangleq \sum_{j=0}^{k-1} \delta_{s_{t+j}}(X)$ . However, this is not a measure because it fails to satisfy additivity for  $\lambda < 1$ , i.e., for measurable disjoint sets  $A, B \subseteq \mathcal{S}$ ,  $\Phi_\lambda^\pi(s, A \cup B) < \Phi_\lambda^\pi(s, A) + \Phi_\lambda^\pi(s, B)$  (Lemma 7.B.4). For this reason, we call Eq. (7.9) the  $\lambda$  operator ( $\lambda O$ ). We then minimize the following squared Bellman error loss for  $\Phi_\lambda^\pi$  (dropping sub-

/superscripts for concision):

$$\begin{aligned} \mathcal{L}(\Phi) = \mathbb{E}_{s_t, s_{t+1} \sim \rho, s' \sim \mu} \left[ (\varphi(s_t, s') - \gamma \bar{\varphi}(s_{t+1}, s'))^2 \right] - 2\mathbb{E}_{s_t \sim \rho} [\varphi(s_t, s_t)] \\ + 2\gamma(1 - \lambda) \mathbb{E}_{s_t, s_{t+1} \sim \rho} [\mu(s_t) \varphi(s_t, s_t) \bar{\varphi}(s_{t+1}, s_t)], \end{aligned} \quad (7.10)$$

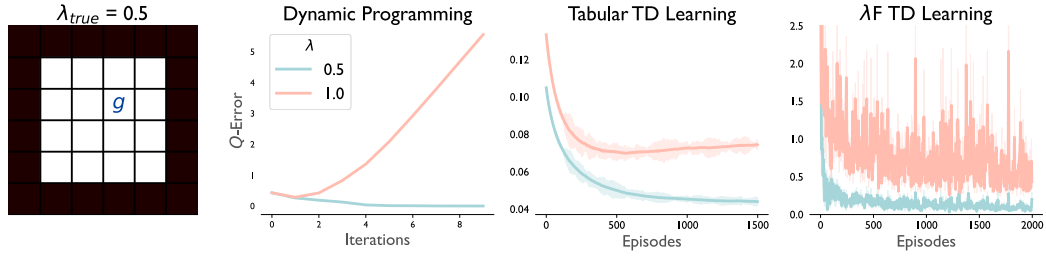
where  $\rho$  and  $\mu$  are densities over  $\mathcal{S}$  and  $\Phi_\lambda^\pi(s) \triangleq \varphi_\lambda^\pi(s) \text{diag}(\mu)$  in the discrete case, with  $\varphi_\lambda^\pi$  parameterized by a neural network.  $\bar{\cdot}$  indicates a stop-gradient operation, i.e., a target network. A detailed derivation and discussion are given in Appendix 7.H. While  $\rho$  can be any training distribution of transitions we can sample from, we require an analytic expression for  $\mu$ . Eq. (7.10) recovers the SM loss of Touati and Ollivier (2021b) when  $\lambda = 1$ .

## 7.3 Policy Evaluation, Learning, and Composition under DMU

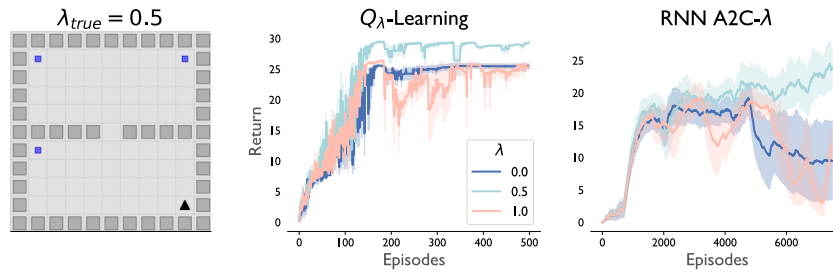
In the following sections, we experimentally validate the formal properties of the  $\lambda$ R and explore its usefulness for solving RL problems with DMU. The majority of our experiments center on navigation tasks, as we believe this is the most natural setting for studying behavior under diminishing rewards. However, in Appendix 7.I we also explore potential for the  $\lambda$ R’s use other areas, such as continuous control, even when rewards do not diminish. There is also the inherent question of whether the agent has access to  $\lambda$ . In a naturalistic context,  $\lambda$  can be seen as an internal variable that the agent likely knows, especially if the agent has experienced the related stimulus before. Therefore, in subsequent experiments, treating  $\lambda$  as a “given” can be taken to imply the agent has prior experience with the relevant stimulus. Further details for all experiments can be found in Appendix 7.E.

### 7.3.1 Policy Evaluation

In Section 7.2, we showed that in order to perform policy evaluation under DMU, an agent is required to learn the  $\lambda$ R. In our first experimental setting, we verify this analysis empirically for the policy depicted in Fig. 7.2 with a rewarded location in the state indicated by a  $g$  in Fig. 7.3 with  $\lambda_{true} = 0.5$ . We then compare the performance for agents



**Figure 7.3: The  $\lambda$ R is required for accurate policy evaluation.** Policy evaluation of the policy depicted in Fig. 7.2 using dynamic programming, tabular TD learning, and  $\lambda$ F TD learning produces the most accurate value estimates when using the  $\lambda$ R with  $\lambda = \lambda_{true}$ . Results are averaged over three random seeds. Shading indicates standard error.



**Figure 7.4: The  $\lambda$ R is required for strong performance.** We apply a tabular  $Q$ -learning-style algorithm and deep actor-critic algorithm to policy optimization in the TwoRooms domain. The blue locations indicate reward, and the black triangle shows the agent's position. Results are averaged over three random seeds. Shading indicates standard error.

using different values of  $\lambda$  across dynamic programming (DP), tabular TD learning, and  $\lambda$ F TD learning with Laplacian features. For the latter two, we use a linear function approximator with a one-hot encoding of the state as the base feature function. We then compute the  $Q$ -values using the  $\lambda$ R with  $\lambda \in \{0.5, 1.0\}$  (with  $\lambda = 1$  corresponding to the SR) and compare the resulting value estimates to the true  $Q$ -values. Consistent with our theoretical analysis, Fig. 7.3 shows that the  $\lambda$ R with  $\lambda = \lambda_{true}$  is required to produce accurate value estimates.

### 7.3.2 Policy Learning

To demonstrate that  $\lambda$ R is useful in supporting policy improvement under diminishing rewards, we implemented modified forms of  $Q$ -learning (Watkins and Dayan, 1992) (which we term  $Q_\lambda$ -learning) and advantage actor-critic (A2C; Mnih et al., 2016) and applied them to the TwoRooms domain from the NeuroNav benchmark task set (Juliani

et al., 2022) (see Fig. 7.4). For a transition  $(s_t, a_t, s_{t+1})$ , we define the following operator:

$$\mathcal{T}_\lambda^* \Phi \triangleq \mathbf{1}(s_t) \odot (1 + \lambda \gamma \Phi(s_{t+1}, a_{t+1})) + \gamma(1 - \mathbf{1}(s_t)) \odot \Phi(s_{t+1}, a_{t+1}), \quad (7.11)$$

where  $a_{t+1} = \operatorname{argmax}_a Q_\lambda(s_t, a) = \operatorname{argmax}_a \mathbf{r}^\top \Phi(s_t, a)$ . This is an improvement operator for  $Q_\lambda$ . The results in Fig. 7.4 show that  $Q_\lambda$ -learning outperforms standard  $Q$ -learning ( $\lambda = 1$ ) for diminishing rewards, and that the “correct”  $\lambda$  produces the best performance. To implement A2C with a  $\lambda R$  critic, we modified the standard TD target in a similar manner as follows:

$$\mathcal{T}_\lambda V(s_t) = r(s_t) + \gamma(V(s_{t+1}) + (\lambda - 1)\mathbf{w}^\top(\phi(s_t) \odot \varphi_\lambda(s_{t+1}))), \quad (7.12)$$

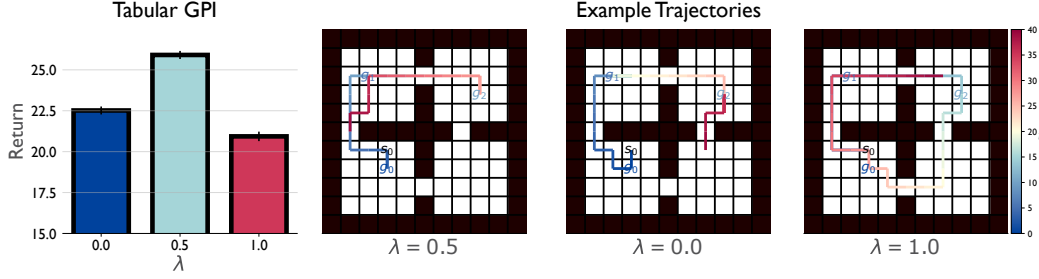
where  $\phi$  were one-hot state encodings, and the policy, value function, and  $\lambda F$  were output heads of a shared LSTM network (Hochreiter and Schmidhuber, 1997b). Note this target is equivalent to Definition 7.2.4 multiplied by the reward (derivation in Appendix 7.E). Fig. 7.4 shows again that correct value targets lead to improved performance. Videos of agent behavior can be found at [lambdarepresentation.github.io](https://lambdarepresentation.github.io).

### 7.3.3 Policy Composition

As we’ve seen, DMU problems of this form have an interesting property wherein solving one task requires the computation of a representation which on its own is task agnostic. In the same way that the SR and FR facilitate generalization across reward functions, the  $\lambda R$  facilitates generalization across reward functions with different  $\bar{r}$ s. The following result shows that there is a benefit to having the “correct”  $\lambda$  for a given resource.

**Theorem 7.3.1 (GPI).** *Let  $\{M_j\}_{j=1}^n \subseteq \mathcal{M}$  and  $M \in \mathcal{M}$  be a set of tasks in an environment  $\mathcal{M}$  and let  $Q^{\pi_j^*}$  denote the action-value function of an optimal policy of  $M_j$  when executed in  $M$ . Assume that the agent uses diminishing rate  $\hat{\lambda}$  that may differ from the true environment diminishing rate  $\lambda$ . Given estimates  $\tilde{Q}^{\pi_j}$  such that  $\|Q^{\pi_j^*} - \tilde{Q}^{\pi_j}\|_\infty \leq \epsilon$  for all  $j$ , define*

$$\pi(s) \in \operatorname{argmax}_a \max_j \tilde{Q}^{\pi_j}(s, a).$$



**Figure 7.5: Tabular GPI.** (Left) Average returns obtained by agents performing GPE+GPI using  $\lambda$ Rs with  $\lambda \in \{0.0, 0.5, 1.0\}$  over 50 episodes. Error bars indicate standard error. (Right) Sample trajectories. Agents with  $\lambda$  set too high overstay in rewarding states, and those with  $\lambda$  too low leave too early.

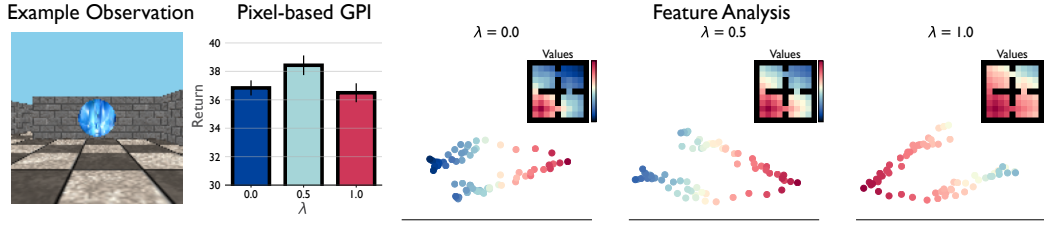
Then,

$$Q^\pi(s, a) \geq \max_j Q^{\pi_j^*}(s, a) - \frac{1}{1-\gamma} \left( 2\epsilon + |\lambda - \hat{\lambda}| \|r\|_\infty + \frac{\gamma(1-\lambda)r(s, a)}{1-\lambda\gamma} \right).$$

Note that for  $\lambda = 1$ , we recover the original GPI bound due to Barreto et al. (2018) with an additional term quantifying error accrued if incorrectly assuming  $\lambda < 1$ .

**Tabular Navigation** We can see this result reflected empirically in Fig. 7.5, where we consider the following experimental set-up in the classic FourRooms domain (Sutton et al., 1999). The agent is assumed to be given or have previously acquired four policies  $\{\pi_0, \pi_1, \pi_2, \pi_3\}$  individually optimized to reach rewards located in each of the four rooms of the environment. There are three reward locations  $\{g_0, g_1, g_2\}$  scattered across the rooms, each with its own initial reward and all with  $\lambda = 0.5$ . At the beginning of each episode, an initial state  $s_0$  is sampled uniformly from the set of available states. An episode terminates either when the maximum reward remaining in any of the goal states is less than 0.1 or when the maximum number of steps  $H = 40$  is reached (when  $\lambda = 1$ , the latter is the only applicable condition). For each of the four policies, we learn  $\lambda$ Rs with  $\lambda \in \{0.0, 0.5, 1.0\}$  using dynamic programming and record the returns obtained while performing GPE+GPI with each of these representations over 50 episodes. Bellman error curves for the  $\lambda$ Rs are shown in Fig. 7.11, and demonstrate that convergence is faster for lower  $\lambda$ . In the left panel of Fig. 7.5, we can indeed see that using the correct  $\lambda$  (0.5) nets the highest returns. Example trajectories for each agent  $\lambda$  are shown in the remaining panels.





**Figure 7.6: Pixel-Based GPI.** Performance is strongest for agents using the correct  $\lambda = 0.5$ . PCA on the learned features in each underlying environment state shows that the  $\lambda$ Fs capture the value-conditioned structure of the environment.

**Pixel-Based Navigation** We verified that the previous result is reflected in larger scales by repeating the experiment in a partially-observed version of FourRooms in which the agent receives  $128 \times 128$  RGB egocentric observations of the environment (Fig. 7.6, left) with  $H = 50$ . In this case, the agent learns  $\lambda$ Fs for each policy for  $\lambda \in \{0.0, 0.5, 1.0\}$ , where each  $\lambda$ F is parameterized by a feedforward convolutional network with the last seven previous frames stacked to account for the partial observability. The base features were Laplacian eigenfunctions normalized to  $[0, 1]$ , which were shown by Touati et al. (2023) to perform the best of all base features for SFs across a range of environments including navigation.

## 7.4 Understanding Natural Behavior

Naturalistic environments often exhibit diminishing reward and give insight into animal behavior. The problem of foraging in an environment with multiple diminishing food patches (i.e., reward states) is of interest in behavioral science (Hayden et al., 2011; Yoon et al., 2018; Gabay and Apps, 2021). The cornerstone of foraging theory is the marginal value theorem (MVT; Charnov, 1976; Gabay and Apps, 2021), which states that the optimal time to leave a patch is when the patch’s reward rate matches the average reward rate of the environment. However, the MVT does not describe which patch to move to once an agent leaves its current patch. We show that the  $\lambda$ O recovers MVT-like behavior in discrete environments and improves upon the MVT by not only predicting *when* agents should leave rewarding patches, but also *where* they should go. Moreover, we provide a scheme for *learning*  $\lambda$  alongside the  $\lambda$ O using feedback from the environment. To learn the  $\lambda$ O, we take inspiration from the FBR (Touati and Ollivier,

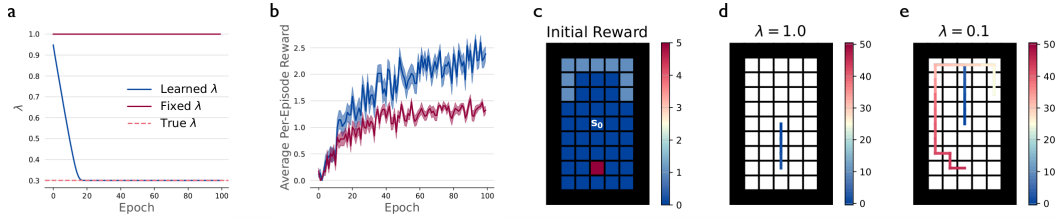
2021b) and use the following parametrization:  $\Phi_{\lambda}^{\pi_z}(s, a, s') = F(s, a, z)^{\top} B(s') \mu(s')$  and  $\pi_z(s) = \operatorname{argmax}_a F(s, a, z)^{\top} z$ , where  $\mu$  is a density with full support on  $\mathcal{S}$ , e.g., uniform. We then optimize using the the loss in Eq. (7.10) under this parameterization (details in Appendix 7.E). Given a reward function  $r : \mathcal{S} \rightarrow \mathbb{R}$  at evaluation time, the agent acts according to  $\operatorname{argmax}_a F(s, a, z_R)^{\top} z_R$ , where  $z_R = \mathbb{E}_{s \sim \mu}[r(s)B(s)]$ . Because the environment is non-stationary,  $z_R$  has to be re-computed at every time step. To emulate a more realistic foraging task, the agent learns  $\lambda$  by minimizing the loss in Eq. (7.20) in parallel with the loss

$$\mathcal{L}(\lambda) = \mathbb{E}_{s_t, s_{t+1} \sim \rho} [\mathbb{1}(s_t = s_{t+1}) (r(s_{t+1}) - \lambda r(s_t))^2],$$

which provably recovers the correct value of  $\lambda$  provided that  $\rho$  is sufficiently exploratory. In Section 7.H.9 we provide experiments showing that using an incorrect value of  $\lambda$  leads to poor performance on tabular tasks. In Fig. 7.7 we show that the agent learns the correct value of  $\lambda$ , increasing its performance. We illustrate the behavior of the  $\lambda$ O in an asymmetric environment that has one large reward state on one side and many small reward states (with higher total reward) on the other. Different values of  $\lambda$  lead to very different optimal foraging strategies, which the  $\lambda$ O recovers and exhibits MVT-like behavior (see Section 7.H.10 for a more detailed analysis). Our hope is that the  $\lambda$ R may provide a framework for new theoretical studies of foraging behavior and possibly mechanisms for posing new hypotheses. For example, an overly large  $\lambda$  may lead to overstay in depleted patches, a frequently observed phenomenon (Nonacs, 2001).

## 7.5 Conclusion

**Limitations** Despite its advantages, there are several drawbacks to the representation which are a direct result of the challenge of the DMU setting. First, the  $\lambda$ R is only useful for transfer across diminishing reward functions when the value of  $\lambda$  at each state is consistent across tasks. In natural settings, this is fairly reasonable, as  $\lambda$  can be thought of as encoding the type of the resource available at each state (i.e., each resource has its own associated decay rate). Second, as noted in Section 7.2.1, the  $\lambda$ R does not admit a measure-theoretic formulation, which makes it challenging to define a principled, occupancy-



**Figure 7.7:  $\lambda$ O trained via FB.** **a)**  $\lambda$  values of two agents in FourRooms, one which learns  $\lambda$  and one which does not. **b)** Performance of the two agents from (a). Learning  $\lambda$  improves performance. **c)** Reward structure and starting state of the asymmetric environment. **d)** Trajectory of an agent with  $\lambda = 1$ . The optimal strategy is to reach the high reward state and exploit it *ad infinitum*. **e)** Trajectory of an agent with  $\lambda = 0.1$ . The optimal strategy is to exploit each reward state for a few time steps before moving to the next reward state.

based version compatible with continuous state spaces. Third, the  $\lambda$ R is a prospective representation, and so while it is used to correctly evaluate a policy’s future return under DMU, it is not inherently memory-based and so performs this evaluation as if the agent hasn’t visited locations with diminishing reward before. Additional mechanisms (i.e., recurrence or frame-stacking) are necessary to account for previous visits. Finally, the  $\lambda$ R is dependent on an episodic task setting for rewards to reset, as otherwise the agent would eventually consume all reward in the environment. An even more natural reward structure would include a mechanism for reward *replenishment* in addition to depletion. We describe several such candidates in Appendix [7.J](#), but leave a more thorough investigation of this issue to future work.

In this work, we aimed to lay the groundwork for understanding policy evaluation, learning, and composition under diminishing rewards. To solve such problems, we introduced—and showed the necessity of—the  $\lambda$  *representation*, which generalizes the SR and FR. We demonstrated its usefulness for rapid policy evaluation and by extension, composition, as well as control. We believe the  $\lambda$ R represents a useful step in the development of state representations for naturalistic environments.

## Appendix

### Appendix 7.A: Derivation of $\lambda$ R Recursion

We provide a step-by-step derivation of the  $\lambda$ R recursion in Eq. (7.4):

$$\begin{aligned}
\Phi_{\lambda}^{\pi}(s, s') &= \mathbb{E} \left[ \sum_{k=0}^{\infty} \lambda^{n_t(s', k)} \gamma^k \mathbb{1}(s_{t+k} = s') \middle| s_t = s \right] \\
&= \mathbb{E} \left[ \mathbb{1}(s_t = s') + \sum_{k=1}^{\infty} \lambda^{n_t(s', k)} \gamma^k \mathbb{1}(s_{t+k} = s') \middle| s_t = s \right] \\
&\stackrel{(i)}{=} \mathbb{E} \left[ \mathbb{1}(s_t = s') + \lambda^{n_t(s', 1)} \gamma \sum_{k=1}^{\infty} \lambda^{n_{t+1}(s', k)} \gamma^{k-1} \mathbb{1}(s_{t+k} = s') \middle| s_t = s \right] \\
&\stackrel{(ii)}{=} \mathbb{E} \left[ \mathbb{1}(s_t = s') + \mathbb{1}(s_t = s') \lambda \gamma \sum_{k=1}^{\infty} \lambda^{n_{t+1}(s', k)} \gamma^{k-1} \mathbb{1}(s_{t+k} = s') \right. \\
&\quad \left. + \gamma(1 - \mathbb{1}(s_t = s')) \sum_{k=1}^{\infty} \lambda^{n_{t+1}(s', k)} \gamma^{k-1} \mathbb{1}(s_{t+k} = s') \middle| s_t = s \right] \\
&= \mathbb{1}(s_t = s') + \mathbb{1}(s_t = s') \lambda \gamma \mathbb{E}_{s_{t+1} \sim p^{\pi}} \Phi_{\lambda}^{\pi}(s_{t+1}, s') \\
&\quad + \gamma(1 - \mathbb{1}(s_t = s')) \mathbb{E}_{s_{t+1} \sim p^{\pi}} \Phi_{\lambda}^{\pi}(s_{t+1}, s') \\
&= \mathbb{1}(s_t = s') (1 + \gamma \lambda \mathbb{E}_{s_{t+1} \sim p^{\pi}} \Phi_{\lambda}^{\pi}(s_{t+1}, s')) \\
&\quad + \gamma(1 - \mathbb{1}(s_t = s')) \mathbb{E}_{s_{t+1} \sim p^{\pi}} \Phi_{\lambda}^{\pi}(s_{t+1}, s'),
\end{aligned} \tag{7.13}$$

where (i) is because  $n_t(s', k) = n_t(s', 1) + n_{t+1}(s', k)$  and (ii) is because

$$\lambda^{n_t(s', 1)} = \lambda^{\mathbb{1}(s_t = s')} = \mathbb{1}(s_t = s') \lambda + (1 - \mathbb{1}(s_t = s')).$$

### Appendix 7.B: Theoretical Analysis

Here, we provide proofs for the theoretical results in the main text.

**Lemma 7.I.1** (Impossibility; Informal). *Given a reward function of the form Eq. (7.1), it is impossible to define a Bellman equation solely using the resulting value function and immediate reward.*

Before providing the formal statement and proof for Lemma 7.I.1, we introduce a

definition for a Bellman operator.

**Definition 7.B.1** (Bellman Operator). *A Bellman operator is a contractive operator  $\mathbb{R}^{|\mathcal{S}|} \rightarrow \mathbb{R}^{|\mathcal{S}|}$  that depends solely on  $\bar{\mathbf{r}}$ , one-step expectations under  $p^\pi$ , and learning hyperparameters (in our case  $\gamma$  and  $\lambda$ ).*

We can now give a formal statement of Lemma 7.1.1:

**Lemma 7.B.1** (Impossibility; Formal). *Assume that  $|\mathcal{S}| > 1$ . Then, there does not exist a Bellman operator  $T$  with fixed point  $V^\pi$ .*

*Proof.* Assume for a contradiction that  $T$  is a Bellman operator. By the Banach fixed-point theorem,  $V^\pi$  must be the unique fixed point of  $T$ . Hence,  $TV^\pi$  must take on the following form (see the proof of Lemma 3.1 in Appendix B): for  $s \in \mathcal{S}$ ,

$$(TV^\pi)(s) = \bar{r}(s) + \gamma \mathbb{E}_{s' \sim p^\pi(\cdot|s)} V^\pi(s') + \gamma(\lambda - 1) \bar{r}(s) \mathbb{E}_{s' \sim p^\pi(\cdot|s)} \Phi_\lambda^\pi(s', s).$$

For the assumption to hold, there must exist a function  $f$  such that, for any  $s \in \mathcal{S}$ ,

$$\mathbb{E}_{s' \sim p^\pi(\cdot|s)} \Phi_\lambda^\pi(s', s) = \mathbb{E}_{p^\pi(\cdot|s)} f(\bar{\mathbf{r}}, \mathbf{V}^\pi, \gamma, \lambda, s).$$

Now, by definition,

$$V^\pi(s) = \sum_{s' \in \mathcal{S}} \Phi_\lambda^\pi(s, s') \bar{r}(s').$$

$\bar{\mathbf{r}}$  is a vector in  $\mathbb{R}^{|\mathcal{S}|}$ , so as long as  $\mathcal{S} > 1$ ,  $\bar{\mathbf{r}}^\perp$  is non-trivial. Fix any  $\bar{\mathbf{r}}, \mathbf{V}^\pi, s$ . Pick a vector  $\mathbf{w} \in \bar{\mathbf{r}}^\perp \setminus \{\mathbf{0}\}$  and define

$$\tilde{\Phi}_\lambda^\pi(s, s') := \Phi_\lambda^\pi(s, s') + w(s')$$

for any  $s, s' \in \mathcal{S}$ . Note that

$$\sum_{s' \in \mathcal{S}} \tilde{\Phi}_\lambda^\pi(s, s') \bar{r}(s') = \sum_{s' \in \mathcal{S}} \Phi_\lambda^\pi(s, s') \bar{r}(s') + \sum_{s' \in \mathcal{S}} w(s') \bar{r}(s') = V^\pi(s),$$

as  $\mathbf{w} \perp \bar{\mathbf{r}}$ . However,

$$\mathbb{E}_{s' \sim p^\pi(\cdot|s)} \tilde{\Phi}_\lambda^\pi(s', s) = \sum_{s' \in \mathcal{S}} p^\pi(s'|s) \Phi_\lambda^\pi(s', s) + \sum_{s' \in \mathcal{S}} p^\pi(s'|s) w(s).$$

The final term evaluates to  $w(s)$ . Because  $\mathbf{w} \neq \mathbf{0}$ , there must exist some  $s$  such that  $w(s) \neq 0$ . For this  $s$ , we have a single input  $(\bar{\mathbf{r}}, \mathbf{V}^\pi, \gamma, \lambda, s)$  to  $f$  that corresponds to two distinct outputs:  $\mathbb{E}_{s' \sim p^\pi(\cdot|s)} \Phi_\lambda^\pi(s', s)$  and  $\mathbb{E}_{s' \sim p^\pi(\cdot|s)} \tilde{\Phi}_\lambda^\pi(s', s)$ .

Hence,  $f$  is a one-to-many mapping: for fixed input, there is more than one output. Therefore,  $f$  is not a function, yielding a contradiction. □

The following establishes  $\mathcal{G}_\lambda^\pi$  as a contraction.

**Lemma 7.B.2 (Contraction).** *Let  $\mathcal{G}_\lambda^\pi$  be the operator as defined in Definition 7.2.2 for some stationary policy  $\pi$ . Then for any two matrices  $\Phi, \Phi' \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$ ,*

$$|\mathcal{G}_\lambda^\pi \Phi(s, s') - \mathcal{G}_\lambda^\pi \Phi'(s, s')| \leq \gamma |\Phi(s, s') - \Phi'(s, s')|.$$

*Proof.* We have

$$\begin{aligned} |(\mathcal{G}_\lambda^\pi \Phi - \mathcal{G}_\lambda^\pi \Phi')_{s,s'}| &= |(I \odot (\mathbf{1}\mathbf{1}^\top + \gamma\lambda P^\pi \Phi) + \gamma(\mathbf{1}\mathbf{1}^\top - I) \odot P^\pi \Phi \\ &\quad - I \odot (\mathbf{1}\mathbf{1}^\top + \gamma\lambda P^\pi \Phi') - \gamma(\mathbf{1}\mathbf{1}^\top - I) \odot P^\pi \Phi')_{s,s'}| \\ &= |(I \odot \gamma\lambda P^\pi (\Phi - \Phi') + \gamma(\mathbf{1}\mathbf{1}^\top - I) \odot P^\pi (\Phi - \Phi'))_{s,s'}| \\ &= |((I \odot \lambda \mathbf{1}\mathbf{1}^\top + \mathbf{1}\mathbf{1}^\top - I) \odot \gamma P^\pi (\Phi - \Phi'))_{s,s'}| \\ &\stackrel{(i)}{\leq} |(\gamma P^\pi (\Phi - \Phi'))_{s,s'}| \\ &= \gamma |(P^\pi (\Phi - \Phi'))_{s,s'}| \\ &\leq \gamma |(\Phi - \Phi')_{s,s'}|, \end{aligned}$$

where (i) comes from using  $\lambda \leq 1$  and simplifying. □

Note that we can actually get a tighter contraction factor of  $\lambda\gamma$  for  $s = s'$ . Given this contractive property, we can prove its convergence with the use of the following lemma.

**Lemma 7.B.3** (Max  $\lambda R$ ). *The maximum possible value of  $\Phi_\lambda^\pi(s, s')$  is*

$$\frac{\mathbb{1}(s = s') + (1 - \mathbb{1}(s = s'))\gamma}{1 - \lambda\gamma}.$$

*Proof.* For  $s = s'$ ,

$$\Phi_\lambda^\pi(s, s) = 1 + \lambda\gamma \mathbb{E}_{s_{t+1} \sim p^\pi(\cdot | s_t)} \Phi_\lambda^\pi(s_{t+1}, s).$$

This is just the standard SR recursion with discount factor  $\lambda\gamma$ , so the maximum is

$$\sum_{k=0}^{\infty} (\lambda\gamma)^k = \frac{1}{1 - \lambda\gamma}. \quad (7.14)$$

For  $s \neq s'$ ,  $\mathbb{1}(s_t = s') = 0$ , so

$$\Phi_\lambda^\pi(s, s') = \gamma \mathbb{E}_{s_{t+1} \sim p^\pi(\cdot | s_t)} \Phi_\lambda^\pi(s_{t+1}, s').$$

Observe that  $\Phi_\lambda^\pi(s, s) \geq \Phi_\lambda^\pi(s, s')$  for  $s' \neq s$ , so the maximum is attained for  $s_{t+1} = s'$ .

We can then use the result for  $s = s'$  to get

$$\Phi_\lambda^\pi(s, s') = \gamma \left( \frac{1}{1 - \lambda\gamma} \right). \quad (7.15)$$

Combining Eq. (7.14) and Eq. (7.15) yields the desired result.  $\square$

**Proposition 7.2.1** (Convergence). *Under the conditions assumed above, set  $\Phi^{(0)} = (1 - \lambda)I$ . For  $k = 1, 2, \dots$ , suppose that  $\Phi^{(k+1)} = \mathcal{G}_\lambda^\pi \Phi^{(k)}$ . Then*

$$|(\Phi^{(k)} - \Phi_\lambda^\pi)_{s,s'}| \leq \frac{\gamma^{k+1}}{1 - \lambda\gamma}.$$

*Proof.* Using the notation  $X_{s,s'} = X(s, s')$  for a matrix  $X$ :

$$\begin{aligned}
|(\Phi^{(k)} - \Phi_\lambda^\pi)_{s,s'}| &= |(\mathcal{G}_\lambda^k \Phi^{(0)} - \mathcal{G}_\lambda^k \Phi_\lambda^\pi)_{s,s'}| \\
&= |(\mathcal{G}_\lambda^k \Phi^{(0)} - \Phi_\lambda^\pi)_{s,s'}| \\
&\stackrel{(i)}{\leq} \gamma^k |(\Phi^{(0)} - \Phi_\lambda^\pi)_{s,s'}| \\
&\stackrel{(ii)}{=} \gamma^k \Phi_\lambda^\pi(s, s') \\
&\stackrel{(iii)}{\leq} \frac{\gamma^{k+1}}{1 - \lambda\gamma}
\end{aligned} \tag{7.16}$$

where (i) is due to Lemma 7.B.2, (ii) is because  $\Phi^{(0)}(s, s') = 0$  for  $s \neq s'$ , and (iii) is due to Lemma 7.B.3  $\square$

**Lemma 7.B.4** (Subadditivity). *For any  $s \in \mathcal{S}$ , policy  $\pi$ ,  $\lambda \in [0, 1)$ , and disjoint measurable sets  $A, B \subseteq \mathcal{S}$ ,*

$$\Phi_\lambda^\pi(s, A \cup B) < \Phi_\lambda^\pi(s, A) + \Phi_\lambda^\pi(s, B).$$

*Proof.* Note that for disjoint sets  $A, B$ , we have  $n_t(A \cup B, k) = n_t(A, k) + n_t(B, k)$ . Hence, conditioned on some policy  $\pi$  and  $s_t = s$ ,

$$\begin{aligned}
&\lambda^{n_t(A \cup B, k)} \mathbb{P}(s_{t+k} \in A \cup B) \\
&= \lambda^{n_t(A, k)} \lambda^{n_t(B, k)} \mathbb{P}(s_{t+k} \in A) + \lambda^{n_t(A, k)} \lambda^{n_t(B, k)} \mathbb{P}(s_{t+k} \in B) \\
&\leq \lambda^{n_t(A, k)} \mathbb{P}(s_{t+k} \in A) + \lambda^{n_t(B, k)} \mathbb{P}(s_{t+k} \in B),
\end{aligned}$$

where the first line follows from  $\mathbb{P}(s_{t+k} \in A \cup B) = \mathbb{P}(s_{t+k} \in A) + \mathbb{P}(s_{t+k} \in B)$ . Equality holds over all  $A, B, t, k$  if and only if  $\lambda = 1$ . Summing over  $k$  yields the result.  $\square$

### 7.B.1 Proof of Theorem 7.3.1

We first prove two results, which rely throughout on the fact that  $\Phi_\lambda(s, a, s') \leq \frac{1}{1-\lambda\gamma}$  for all  $s, a, s'$ , which follows from Lemma 7.B.3. For simplicity, we also assume throughout that all rewards are non-negative, but this assumption can easily be dropped by taking ab-



solute values of rewards. The proofs presented here borrow ideas from those of (Barreto et al., 2017).

**Lemma 7.B.5.** *Let  $\{M_j\}_{j=1}^n \subseteq \mathcal{M}$  and  $M \in \mathcal{M}$  be a set of tasks in an environment  $\mathcal{M}$  with diminishing rate  $\lambda$  and let  $Q^{\pi_j^*}$  denote the action-value function of an optimal policy of  $M_j$  when executed in  $M$ . Given estimates  $\tilde{Q}^{\pi_j}$  such that  $\|Q^{\pi_j^*} - \tilde{Q}^{\pi_j}\|_\infty \leq \epsilon$  for all  $j$ , define*

$$\pi(s) \in \operatorname{argmax}_a \max_j \tilde{Q}^{\pi_j}(s, a).$$

*Then,*

$$Q^\pi(s, a) \geq \max_j Q^{\pi_j^*}(s, a) - \frac{1}{1 - \gamma} \left( 2\epsilon + \frac{\gamma(1 - \lambda)r(s, a)}{1 - \lambda\gamma} \right),$$

*where  $r$  denotes the reward function of  $M$ .*

*Proof.* Define  $\tilde{Q}_{\max}(s, a) := \max_j \tilde{Q}^{\pi_j}(s, a)$  and  $Q_{\max}(s, a) := \max_j Q^{\pi_j^*}(s, a)$ . Let  $T^\nu$  denote the Bellman operator of a policy  $\nu$  in task  $M$ . For all  $(s, a) \in \mathcal{S} \times \mathcal{A}$  and all

$j$ ,

$$\begin{aligned}
& T_i^\pi \tilde{Q}_{\max}(s, a) - r(s, a) \\
&= \gamma \sum_{s'} p(s'|s, a) \left( (\lambda - 1)r_i(s, a)\Phi^\pi(s', \pi(s'), s) + \tilde{Q}_{\max}(s', \pi(s')) \right) \\
&= \gamma \sum_{s'} p(s'|s, a) \left( (\lambda - 1)r_i(s, a)\Phi^\pi(s', \pi(s'), s) + \max_b \tilde{Q}_{\max}(s', b) \right) \\
&\geq \gamma \sum_{s'} p(s'|s, a) \left( (\lambda - 1)r_i(s, a)\Phi^\pi(s', \pi(s'), s) + \max_b Q_{\max}(s', b) \right) - \gamma\epsilon \\
&\geq \gamma \sum_{s'} p(s'|s, a) \left( (\lambda - 1)r_i(s, a)\Phi^\pi(s', \pi(s'), s) + Q_{\max}(s', \pi_j^*(s')) \right) - \gamma\epsilon \\
&\geq \gamma \sum_{s'} p(s'|s, a) \left( (\lambda - 1)r_i(s, a)\Phi^\pi(s', \pi(s'), s) + Q_i^{\pi_j^*}(s', \pi_j^*(s')) \right) - \gamma\epsilon \\
&= \gamma \sum_{s'} p(s'|s, a) \left( (\lambda - 1)r_i(s, a)\Phi^{\pi_j^*}(s', \pi_j^*(s'), s) + Q_i^{\pi_j^*}(s', \pi_j^*(s')) \right) - \gamma\epsilon \\
&\quad + \gamma(\lambda - 1)r(s, a) \sum_{s'} p(s'|s, a) \left( \Phi^\pi(s', \pi(s'), s) - \Phi^{\pi_j^*}(s', \pi_j^*(s'), s) \right) \\
&\geq T_i^{\pi_j^*} Q_i^{\pi_j^*}(s, a) - \gamma\epsilon - \frac{\gamma(1 - \lambda)r(s, a)}{1 - \lambda\gamma} - r(s, a) \\
&= Q_i^{\pi_j^*}(s, a) - \gamma\epsilon - \frac{\gamma(1 - \lambda)r(s, a)}{1 - \lambda\gamma} - r(s, a).
\end{aligned}$$

This holds for any  $j$ , so

$$\begin{aligned}
T^\pi \tilde{Q}_{\max}(s, a) &\geq \max_j Q_i^{\pi_j^*}(s, a) - \gamma\epsilon - \frac{\gamma(1 - \lambda)r(s, a)}{1 - \lambda\gamma} \\
&= Q_{\max}(s, a) - \gamma\epsilon - \frac{\gamma(1 - \lambda)r(s, a)}{1 - \lambda\gamma} \\
&\geq \tilde{Q}_{\max}(s, a) - \epsilon - \gamma\epsilon - \frac{\gamma(1 - \lambda)r(s, a)}{1 - \lambda\gamma}.
\end{aligned}$$

Next, note that for any  $c \in \mathbb{R}$ ,

$$\begin{aligned}
T^\pi(\tilde{Q}_{\max}(s, a) + c) &= T^\pi \tilde{Q}_{\max}(s, a) + \gamma \sum_{s'} p(s'|s, a)c \\
&= T^\pi \tilde{Q}_{\max}(s, a) + \gamma c.
\end{aligned}$$

Putting everything together, and using the fact that  $T^\nu$  is monotonic and contractive,

$$\begin{aligned}
Q_i^\pi(s, a) &= \lim_{k \rightarrow \infty} (T^\pi)^k \tilde{Q}_{\max}(s, a) \\
&\geq \lim_{k \rightarrow \infty} \left[ \tilde{Q}_{\max}(s, a) - \left( \epsilon(1 + \gamma) - \frac{\gamma(1 - \lambda)r(s, a)}{1 - \lambda\gamma} \right) \sum_{j=0}^k \gamma^j \right] \\
&\geq \tilde{Q}_{\max}(s, a) - \frac{1}{1 - \gamma} \left( \epsilon(1 + \gamma) - \frac{\gamma(1 - \lambda)r(s, a)}{1 - \lambda\gamma} \right) \\
&\geq Q_{\max}(s, a) - \epsilon - \frac{1}{1 - \gamma} \left( \epsilon(1 + \gamma) - \frac{\gamma(1 - \lambda)r(s, a)}{1 - \lambda\gamma} \right) \\
&\geq Q^{\pi_j^*}(s, a) - \frac{1}{1 - \gamma} \left( 2\epsilon + \frac{\gamma(1 - \lambda)r(s, a)}{1 - \lambda\gamma} \right).
\end{aligned}$$

This holds for every  $j$ , hence the result.  $\square$

**Lemma 7.B.6.** *Let  $\nu$  be any policy,  $\lambda, \hat{\lambda} \in [0, 1]$ , and  $Q_\lambda$  denote a value function with respect to diminishing rate  $\lambda$ . Then,*

$$\|Q_\lambda^\nu - Q_{\hat{\lambda}}^\nu\|_\infty \leq \frac{|\lambda - \hat{\lambda}| \|r\|_\infty}{1 - \gamma}.$$

*Proof.* The proof follows from the definition of  $Q$ : for every  $(s, a) \in \mathcal{S} \times \mathcal{A}$ ,

$$\begin{aligned}
|Q_\lambda^\nu(s, a) - Q_{\hat{\lambda}}^\nu(s, a)| &= \left| \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k \left( \lambda^{n_t(s_{t+k}, k)} - \hat{\lambda}^{n_t(s_{t+k}, k)} \right) \right. \right. \\
&\quad \left. \left. \cdot r(s_{t+k}) \middle| s_t = s, a_t = a \right] \right| \\
&\leq \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k \left| \lambda^{n_t(s_{t+k}, k)} - \hat{\lambda}^{n_t(s_{t+k}, k)} \right| \right. \\
&\quad \left. \cdot r(s_{t+k}) \middle| s_t = s, a_t = a \right] \\
&= \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k r(s_{t+k}) \left| \lambda - \hat{\lambda} \right| \right. \\
&\quad \left. \cdot \sum_{j=0}^{n_t(s_{t+k}, k)-1} \lambda^{n_t(s_{t+k}, k)-1-j} \hat{\lambda}^j \middle| s_t = s, a_t = a \right] \\
&\leq |\lambda - \hat{\lambda}| \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k r(s_{t+k}) \middle| s_t = s, a_t = a \right] \\
&\leq \frac{|\lambda - \hat{\lambda}| \|r\|_\infty}{1 - \gamma}.
\end{aligned}$$

□

**Theorem 7.3.1 (GPI).** Let  $\{M_j\}_{j=1}^n \subseteq \mathcal{M}$  and  $M \in \mathcal{M}$  be a set of tasks in an environment  $\mathcal{M}$  and let  $Q^{\pi_j^*}$  denote the action-value function of an optimal policy of  $M_j$  when executed in  $M$ . Assume that the agent uses diminishing rate  $\hat{\lambda}$  that may differ from the true environment diminishing rate  $\lambda$ . Given estimates  $\tilde{Q}^{\pi_j}$  such that  $\|Q^{\pi_j^*} - \tilde{Q}^{\pi_j}\|_\infty \leq \epsilon$  for all  $j$ , define

$$\pi(s) \in \operatorname{argmax}_a \max_j \tilde{Q}^{\pi_j}(s, a).$$

Then,

$$Q^\pi(s, a) \geq \max_j Q^{\pi_j^*}(s, a) - \frac{1}{1 - \gamma} \left( 2\epsilon + |\lambda - \hat{\lambda}| \|r\|_\infty + \frac{\gamma(1 - \lambda)r(s, a)}{1 - \lambda\gamma} \right).$$

*Proof.* Let  $Q_\lambda$  denote a value function with respect to diminishing constant  $\lambda$ . We wish

to bound

$$\max_j Q_{\hat{\lambda}}^{\pi_j^*}(s, a) - Q_{\lambda}^{\pi}(s, a),$$

i.e., the value of the GPI policy with respect to the true  $\lambda$  compared to the maximum value of the constituent policies  $\pi_j^*$  used for GPI, which were used assuming  $\hat{\lambda}$ . By the triangle inequality,

$$\begin{aligned} \max_j Q_{\hat{\lambda}}^{\pi_j^*}(s, a) - Q_{\lambda}^{\pi}(s, a) &\leq \max_j Q_{\lambda}^{\pi_j^*}(s, a) - Q_{\lambda}^{\pi}(s, a) \\ &\quad + |\max_j Q_{\lambda}^{\pi_j^*}(s, a) - \max_j Q_{\hat{\lambda}}^{\pi_j^*}(s, a)| \\ &\leq \underbrace{\max_j Q_{\lambda}^{\pi_j^*}(s, a) - Q_{\lambda}^{\pi}(s, a)}_{(1)} \\ &\quad + \underbrace{\max_j |Q_{\lambda}^{\pi_j^*}(s, a) - Q_{\hat{\lambda}}^{\pi_j^*}(s, a)|}_{(2)}. \end{aligned}$$

We bound (1) by Lemma 7.B.5 and (2) by Lemma 7.B.6 to get the result.  $\square$

### 7.B.2 An Extension of Theorem 7.3.1

Inspired by (Barreto et al., 2018), we prove an extension of Theorem 7.3.1:

**Theorem 7.B.1.** *Let  $M \in \mathcal{M}$  be a task in an environment  $\mathcal{M}$  with true diminishing constant  $\lambda$ . Suppose we perform GPI assuming a diminishing constant  $\hat{\lambda}$ :*

*Let  $\{M_j\}_{j=1}^n$  and  $M_i$  be tasks in  $\mathcal{M}$  and let  $Q_i^{\pi_j^*}$  denote the action-value function of an optimal policy of  $M_j$  when executed in  $M_i$ . Given estimates  $\tilde{Q}_i^{\pi_j}$  such that  $\|Q_i^{\pi_j^*} - \tilde{Q}_i^{\pi_j}\|_{\infty} \leq \epsilon$  for all  $j$ , define  $\pi(s) \in \arg\max_a \max_j \tilde{Q}_i^{\pi_j}(s, a)$ .*

*Let  $Q_{\hat{\lambda}}^{\pi}$  and  $Q_{\lambda}^{\pi^*}$  denote the action-value functions of  $\pi$  and the  $M$ -optimal policy  $\pi^*$  when executed in  $M$ , respectively. Then,*

$$\begin{aligned} \|Q_{\hat{\lambda}}^{\pi} - Q_{\lambda}^{\pi^*}\|_{\infty} &\leq \frac{2}{1-\gamma} \left( \frac{1}{2} |\lambda - \hat{\lambda}| \|r\|_{\infty} + \epsilon \right. \\ &\quad \left. + \|r - r_i\|_{\infty} + \min_j \|r_i - r_j\|_{\infty} \right) + \frac{1-\lambda}{1-\lambda\gamma} C, \end{aligned}$$

where  $C$  is a positive constant not depending on  $\lambda$ :

$$\begin{aligned} \frac{1-\gamma}{\gamma}C &= 2\|r - r_i\|_\infty + 2\min_j \|r_i - r_j\|_\infty + \min(\|r\|_\infty, \|r_i\|_\infty) \\ &\quad + \min(\|r_i\|_\infty, \|r_1\|_\infty, \dots, \|r_n\|_\infty) \end{aligned}$$

Note that when  $\lambda = 1$ , we recover Proposition 1 of (Barreto et al., 2018) with an additional term quantifying error incurred by  $\hat{\lambda} \neq \lambda$ . The proof relies on two other technical lemmas, presented below.

**Lemma 7.B.7.**

$$\|Q^{\pi^*} - Q_i^{\pi_i^*}\|_\infty \leq \frac{\|r - r_i\|_\infty}{1-\gamma} + \gamma(1-\lambda) \frac{\min(\|r\|_\infty, \|r_i\|_\infty) + \|r - r_i\|_\infty}{(1-\gamma)(1-\lambda\gamma)}.$$

*Proof.* Define  $\Delta_i := \|Q^{\pi^*} - Q_i^{\pi_i^*}\|_\infty$ . For any  $(s, a) \in \mathcal{S} \times \mathcal{A}$ ,

$$\begin{aligned} &|Q^{\pi^*}(s, a) - Q_i^{\pi_i^*}(s, a)| \\ &= \left| r(s, a) + \gamma \sum_{s'} p(s'|s, a) ((\lambda - 1)r(s, a)\Phi^{\pi^*}(s', \pi^*(s'), s) + Q^{\pi^*}(s', \pi^*(s'))) \right. \\ &\quad \left. - r_i(s, a) - \gamma \sum_{s'} p(s'|s, a) ((\lambda - 1)r_i(s, a)\Phi^{\pi_i^*}(s', \pi_i^*(s'), s) + Q_i^{\pi_i^*}(s', \pi_i^*(s'))) \right| \\ &\leq |r(s, a) - r_i(s, a)| + \gamma \sum_{s'} p(s'|s, a) |Q^{\pi^*}(s, a) - Q_i^{\pi_i^*}(s, a)| \\ &\quad + \gamma(\lambda - 1) \sum_{s'} p(s'|s, a) |r(s, a)\Phi^{\pi^*}(s', \pi^*(s'), s) - r_i(s, a)\Phi^{\pi_i^*}(s', \pi_i^*(s'), s)| \\ &\leq \|r - r_i\|_\infty + \gamma\Delta_i + \gamma(1-\lambda)\|r\Phi^{\pi^*} - r_i\Phi^{\pi_i^*}\|_\infty. \end{aligned}$$

The third term decomposes as

$$\begin{aligned} \|r\Phi^{\pi^*} - r_i\Phi^{\pi_i^*}\|_\infty &\leq \|r\Phi^{\pi^*} - r\Phi^{\pi_i^*}\|_\infty + \|r\Phi^{\pi_i^*} - r_i\Phi^{\pi_i^*}\|_\infty \\ &\leq \frac{\|r\|_\infty + \|r - r_i\|_\infty}{1-\lambda\gamma}. \end{aligned}$$

We could equivalently use the following decomposition:

$$\begin{aligned} \|r\Phi^{\pi^*} - r_i\Phi^{\pi_i^*}\|_{\infty} &\leq \|r\Phi^{\pi^*} - r_i\Phi^{\pi^*}\|_{\infty} + \|r_i\Phi^{\pi^*} - r_i\Phi^{\pi_i^*}\|_{\infty} \\ &\leq \frac{\|r_i\|_{\infty} + \|r - r_i\|_{\infty}}{1 - \lambda\gamma}, \end{aligned}$$

and so

$$\|r\Phi^{\pi^*} - r_i\Phi^{\pi_i^*}\|_{\infty} \leq \frac{\min(\|r\|_{\infty}, \|r_i\|_{\infty}) + \|r - r_i\|_{\infty}}{1 - \lambda\gamma}.$$

The inequalities above hold for all  $s, a$  and so

$$\begin{aligned} \Delta_i &\leq \|r - r_i\|_{\infty} + \gamma\Delta_i + \gamma(1 - \lambda) \frac{\min(\|r\|_{\infty}, \|r_i\|_{\infty}) + \|r - r_i\|_{\infty}}{1 - \lambda\gamma} \\ \Rightarrow \Delta_i &\leq \frac{\|r - r_i\|_{\infty}}{1 - \gamma} + \gamma(1 - \lambda) \frac{\min(\|r\|_{\infty}, \|r_i\|_{\infty}) + \|r - r_i\|_{\infty}}{(1 - \gamma)(1 - \lambda\gamma)}. \end{aligned}$$

Hence the result. □

**Lemma 7.B.8.** *For any policy  $\pi$ ,*

$$\|Q_i^{\pi} - Q^{\pi}\|_{\infty} \leq \frac{\|r - r_i\|_{\infty}}{1 - \gamma} + \gamma(1 - \lambda) \frac{\|r - r_i\|_{\infty}}{(1 - \gamma)(1 - \lambda\gamma)}.$$

*Proof.* Write  $\Delta_i := \|Q_i^{\pi} - Q^{\pi}\|_{\infty}$ . Proceeding as in the previous lemma, for all  $(s, a) \in$

$\mathcal{S} \times \mathcal{A}$ , we have

$$\begin{aligned}
& |Q_i^\pi(s, a) - Q^\pi(s, a)| \\
&= \left| r_i(s, a) + \gamma \sum_{s'} p(s'|s, a) ((\lambda - 1)r_i(s, a)\Phi^\pi(s', a', s) + Q_i^\pi(s', a')) \right. \\
&\quad \left. - r(s, a) - \gamma \sum_{s'} p(s'|s, a) ((\lambda - 1)r(s, a)\Phi^\pi(s', a', s) + Q^\pi(s', a')) \right| \\
&\leq \gamma \sum_{s'} p(s'|s, a) (1 - \lambda) |r(s, a) - r_i(s, a)| \Phi^\pi(s', a', s) \\
&\quad + \gamma \sum_{s'} p(s'|s, a) |Q_i^\pi(s', a') - Q^\pi(s', a')| + |r(s, a) - r_i(s, a)| \\
&\leq \|r - r_i\|_\infty + \gamma(1 - \lambda) \|r - r_i\|_\infty \frac{1}{1 - \lambda\gamma} + \gamma\Delta'_i \\
\Rightarrow \Delta'_i &\leq \|r - r_i\|_\infty + \frac{\gamma(1 - \lambda) \|r - r_i\|_\infty}{1 - \lambda\gamma} + \gamma\Delta'_i \\
\Rightarrow \Delta'_i &\leq \frac{\|r - r_i\|_\infty}{1 - \gamma} + \frac{\gamma(1 - \lambda) \|r - r_i\|_\infty}{(1 - \gamma)(1 - \lambda\gamma)}.
\end{aligned}$$

where  $a' \sim \pi(s')$ . □

Finally, we prove Theorem [7.B.I](#).

*Proof of Theorem [7.B.I](#)* By the triangle inequality,

$$\|Q_\lambda^{\pi^*} - Q_{\hat{\lambda}}^\pi\|_\infty \leq \|Q_\lambda^{\pi^*} - Q_\lambda^\pi\|_\infty + \|Q_\lambda^\pi - Q_{\hat{\lambda}}^\pi\|_\infty.$$

By Lemma [7.B.6](#), the second term is bounded above by

$$\frac{|\lambda - \hat{\lambda}| \|r\|_\infty}{1 - \gamma}.$$

The first term decomposes as follows (dropping the  $\lambda$  subscript on all action-value functions for clarity):

$$\|Q^{\pi^*} - Q^\pi\|_\infty \leq \underbrace{\|Q^{\pi^*} - Q_i^{\pi_i^*}\|_\infty}_{(1)} + \underbrace{\|Q_i^{\pi_i^*} - Q_i^\pi\|_\infty}_{(2)} + \underbrace{\|Q_i^\pi - Q^\pi\|_\infty}_{(3)}.$$

Applying Lemma [7.B.5](#) to (2) (but with respect to  $M_i$  rather than  $M$ ), we have that for



any  $j$ ,

$$\begin{aligned}
Q_i^{\pi^*}(s, a) - Q_i^\pi(s, a) &\leq Q_i^{\pi^*}(s, a) - Q_i^{\pi^j}(s, a) + \frac{1}{1-\gamma} \left( 2\epsilon + \frac{\gamma(1-\lambda)r_i(s, a)}{1-\lambda\gamma} \right) \\
\Rightarrow \|Q_i^{\pi^*} - Q_i^\pi\|_\infty &\leq \underbrace{\|Q_i^{\pi^*} - Q_j^{\pi^*}\|_\infty}_{(2.1)} + \underbrace{\|Q_j^{\pi^*} - Q_i^{\pi^j}\|_\infty}_{(2.2)} \\
&\quad + \frac{1}{1-\gamma} \left( 2\epsilon + \frac{\gamma(1-\lambda)\|r_i\|_\infty}{1-\lambda\gamma} \right).
\end{aligned}$$

We bound (2.1) using Lemma 7.B.7 and (2.2) using Lemma 7.B.8 (but with respect to  $M_j$  rather than  $M$ ):

$$\begin{aligned}
\|Q_i^{\pi^*} - Q_j^{\pi^*}\|_\infty + \|Q_j^{\pi^*} - Q_i^{\pi^j}\|_\infty &\leq \frac{2\|r_i - r_j\|_\infty}{1-\gamma} \\
&\quad + \gamma(1-\lambda) \frac{\min(\|r_i\|_\infty, \|r_j\|_\infty) + 2\|r_i - r_j\|_\infty}{(1-\gamma)(1-\lambda\gamma)}.
\end{aligned}$$

We then apply Lemma 7.B.7 to (1) and Lemma 7.B.8 to (3) to get the result.

□

## Appendix 7.C: An $n$ th Occupancy Representation

To generalize the first occupancy representation to account for reward functions of this type, it's natural to consider an  $N$ th occupancy representation—that is, one which accumulates value only for the first  $N$  occupancies of one state  $s'$  starting from another state  $s$ :

**Definition 7.C.2 (NR).** For an MDP with finite  $\mathcal{S}$ , the  $N$ th-occupancy representation (NR) for a policy  $\pi$  is given by  $F^\pi \in [0, N]^{|\mathcal{S}| \times |\mathcal{S}|}$  such that

$$\begin{aligned}
&\Phi_{(N)}^\pi(s, s') \\
&\triangleq \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^{t+k} \mathbb{1}(s_{t+k} = s', \#(\{j \mid s_{t+j} = s', j \in [0, k-1]\}) < N) \mid s_t \right].
\end{aligned}$$

Intuitively, such a representation sums the first  $N$  (discounted) occupancies of  $s'$  from

time  $t$  to  $t + k$  starting from  $s_t = s$ . We can also note that  $\Phi_{(1)}^\pi$  is simply the FR and  $\Phi_{(0)}(s, s') = 0 \forall s, s'$ . As with the FR and the SR, we can derive a recursive relationship for the NR:

$$\begin{aligned} \Phi_{(N)}^\pi(s, s') &= \mathbb{1}(s_t = s')(1 + \gamma \mathbb{E} \Phi_{(N-1)}^\pi(s_{t+1}, s')) \\ &\quad + \gamma(1 - \mathbb{1}(s_t = s')) \mathbb{E} \Phi_{(N)}^\pi(s_{t+1}, s'), \end{aligned} \tag{7.17}$$

where the expectation is wrt  $p^\pi(s_{t+1}|s_t)$ . Once again, we can confirm that this is consistent with the FR by noting that for  $N = 1$ , the NR recursion recovers the FR recursion. Crucially, we also recover the SR recursion in the limit as  $N \rightarrow \infty$ :

$$\begin{aligned} \lim_{N \rightarrow \infty} \Phi_{(N)}^\pi(s, s') &= \mathbb{1}(s_t = s')(1 + \gamma \mathbb{E} \Phi_{(\infty)}^\pi(s_{t+1}, s')) + \gamma(1 - \mathbb{1}(s_t = s')) \mathbb{E} \Phi_{(\infty)}^\pi(s_{t+1}, s') \\ &= \mathbb{1}(s_t = s') + \gamma \mathbb{E} \Phi_{(\infty)}^\pi(s_{t+1}, s'). \end{aligned}$$

This is consistent with the intuition that the SR accumulates every (discounted) state occupancy in a potentially infinite time horizon of experience. While Definition [7.C.2](#) admits a recursive form which is consistent with our intuition, Eq. [\(7.17\)](#) reveals an inconvenient intractability: the Bellman target for  $\Phi_{(N)}^\pi$  requires the availability of  $\Phi_{(N-1)}^\pi$ . This is a challenge, because it means that if we'd like to learn any NR for finite  $N > 1$ , the agent also must learn and store  $\Phi_{(1)}^\pi, \dots, \Phi_{(N-1)}^\pi$ . Given these challenges, the question of how to learn a tractable general occupancy representation remains. From a neuroscientific perspective, a fixed depletion amount is also inconsistent with both behavioral observations and neural imaging ([Pine et al., 2009](#)), which indicate instead that utility disappears at a fixed rate in proportion to the *current remaining utility*, rather than in proportion to the *original utility*. We address these theoretical and practical issues in the next section.

## Appendix 7.D: Additional Related Work

Another important and relevant sub-field of reinforcement learning is work which studies non-stationary rewards. Perhaps most relevant, the setting of DMU can be seen as

a special case of submodular planning and reward structures (Wang et al., 2020; Prajapat et al., 2023). Wang et al. (2020) focus specifically on planning and not the form of diminishment we study, while Prajapat et al. (2023) is a concurrent work which focuses on the general class of submodular reward problems and introduces a policy-based, REINFORCE-like method which is necessarily on-policy. In contrast, we focus on a particular sub-class of problems especially relevant to natural behavior and introduce a family of approaches which exploit this reward structure, are value-based (and which can be used to modify the critic in policy-based methods), and are compatible with off-policy learning. Other important areas include convex (Zahavy et al., 2021) and constrained (Altman, 2021; Moskovitz et al., 2023b) MDPs. In these cases, non-stationarity is introduced by way of a primal-dual formulation of distinct problem classes into min-max games.

## Appendix 7.E: Further Experimental Details

### 7.E.3 Policy Evaluation

We perform policy evaluation for the policy shown in Fig. 7.2 on the  $6 \times 6$  gridworld shown. The discount factor  $\gamma$  was set to 0.9 for all experiments, which were run for  $H = 10$  steps per episode. The error metric was the mean squared error:

$$Q_{error} \triangleq \frac{1}{|\mathcal{S}||\mathcal{A}|} \sum_{s,a} (Q^\pi(s,a) - \hat{Q}(s,a))^2, \quad (7.18)$$

where  $Q^\pi$  is the ground truth  $Q$ -values and  $\hat{Q}$  is the estimate. Transitions are deterministic. For the dynamic programming result, we learned the  $\lambda$ R using Eq. (7.4) for  $\lambda \in \{0.5, 1.0\}$  and then measured the resulting values by multiplying the resulting  $\lambda$ R by the associated reward vector  $\mathbf{r} \in \{-1, 0, 1\}^{36}$ , which was  $-1$  in all wall states and  $+1$  at the reward state  $g$ . We compared the results to the ground truth values. Dynamic programming was run until the maximum Bellman error across state-action pairs reduced below  $5e-2$ . For the tabular TD learning result, we ran the policy for three episodes starting from every available (non-wall) state in the environment, and learned the  $\lambda$ R for

$\lambda \in \{0.5, 1.0\}$  as above, but using the online TD update:

$$\begin{aligned}\Phi_\lambda(s_t, a_t) &\leftarrow \Phi_\lambda(s_t, a_t) + \alpha \delta_t, \\ \delta_t &= \mathbf{1}(s_t) \odot (1 + \gamma \lambda \Phi_\lambda(s_{t+1}, a_{t+1})) \\ &\quad + \gamma(1 - \mathbf{1}(s_t)) \odot \Phi_\lambda(s_{t+1}, a_{t+1}) - \Phi_\lambda(s_t, a_t),\end{aligned}$$

where  $a_{t+1} \sim \pi(\cdot \mid s_{t+1})$ . The learned  $Q$ -values were then computed in the same way as the dynamic programming case and compared to the ground truth. For the  $\lambda F$  result, we first learned Laplacian eigenfunction base features as described in [Touati et al. \(2023\)](#) from a uniform exploration policy and normalized them to the range  $[0, 1]$ . We parameterized the base feature network as a 2-layer MLP with ReLU activations and 16 units in the hidden layer. We then used the base features to learn the  $\lambda F$ s as in the tabular case, but with the  $\lambda F$  network parameterized as a three-layer MLP with 16 units in each of the hidden layers and ReLU activations. All networks were optimized using Adam with a learning rate of  $3e-4$ . The tabular and neural network experiments were repeated for three random seeds, the former was run for 1,500 episodes and the latter for 2,000.

#### 7.E.4 Policy Learning

We ran the experiments for Fig. [7.4](#) in a version of the TwoRooms environment from the NeuroNav benchmark [\(Juliani et al., 2022\)](#) with reward modified to decay with a specified  $\lambda_{true} = 0.5$  and discount factor  $\gamma = 0.95$ . The initial rewards in the top right goal and the lower room goal locations were 5 and the top left goal had initial reward 10. The observations in the neural network experiment were one-hot state indicators. The tabular  $Q_\lambda$  experiments run the algorithm in Algorithm [9](#) for 500 episodes for  $\lambda \in \{0.0, 0.5, 1.0\}$ , with  $\lambda_{true}$  set to 0.5, repeated for three random seeds. Experiments used a constant step size  $\alpha = 0.1$ . There were five possible actions: up, right, down, left, and stay. The recurrent A2C agents were based on the implementation from the BSuite library [\(Osband et al., 2020\)](#) and were run for 7,500 episodes of maximum length  $H = 100$  with  $\gamma = 0.99$  using the Adam optimizer with learning rate  $3e-4$ . The experiment was repeated for three random seeds. The RNN was an LSTM with 128 hidden units and three output heads: one for the policy, one for the value function, and one for the

**Algorithm 9** Online Tabular  $Q_\lambda$ -Learning Update

- 
- 1: **Require:** Current  $\lambda$ R-values  $\Phi_\lambda^{(t)} \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{A}| \times |\mathcal{S}|}$ , current reward vector  $\mathbf{r}^{(t)}$ , observed  $(s_t, a_t, s_{t+1})$  tuple
  - 2: Compute  $Q_\lambda$ -values:  $Q_\lambda^{(t)} \leftarrow (\Phi_\lambda^{(t)})^\top \mathbf{r}^{(t)}$
  - 3: Select greedy action:  $a_{t+1} \leftarrow \operatorname{argmax}_{a \in \mathcal{A}} Q_\lambda^{(t)}(s_{t+1}, a)$
  - 4: Update  $\Phi_\lambda$ :
- 

$$\begin{aligned} \Phi_\lambda^{(t+1)}(s_t, a_t) &\leftarrow \Phi_\lambda^{(t)}(s_t, a_t) + \alpha \delta^{(t)}, \quad \text{where} \\ \delta^{(t)} &= \mathbf{1}(s_t) \odot (1 + \gamma \lambda \Phi_\lambda^{(t)}(s_{t+1}, a_{t+1})) \\ &\quad + \gamma(1 - \mathbf{1}(s_t)) \odot \Phi_\lambda^{(t)}(s_{t+1}, a_{t+1}) - \Phi_\lambda^{(t)}(s_t, a_t). \end{aligned}$$

- 5: **Return** updated  $\Phi_\lambda^{(t+1)}$
- 

$\lambda$ F. The base features were one-hot representations of the current state, 121-dimensional in this case.

**7.E.5 Tabular GPI**

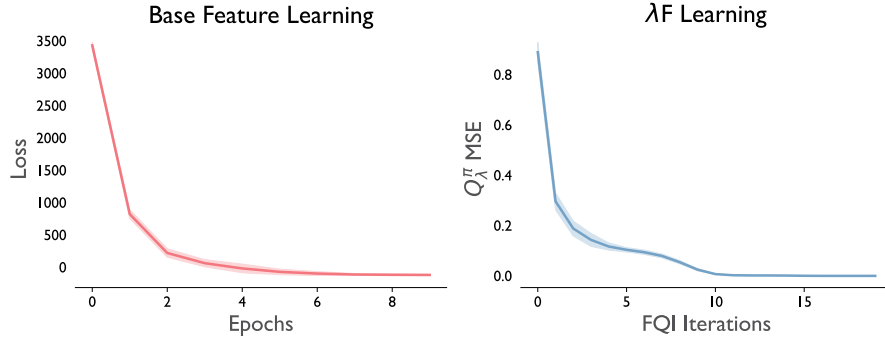
The agent is assumed to be given or have previously acquired four policies  $\{\pi_0, \pi_1, \pi_2, \pi_3\}$  individually optimized to reach rewards located in each of the four rooms of the environment. There are three reward locations  $\{g_0, g_1, g_2\}$  scattered across the rooms, each with its own initial reward  $\bar{r} = [5, 10, 5]$  and all with  $\lambda = 0.5$ . At the beginning of each episode, an initial state  $s_0$  is sampled uniformly from the set of available states. An episode terminates either when the maximum reward remaining in any of the goal states is less than 0.1 or when the maximum number of steps  $H = 40$  is reached. Empty states carry a reward of 0, encountering a wall gives a reward of  $-1$ , and the discount factor is set to  $\gamma = 0.97$ .

For each of the four policies, we learn  $\lambda$ Rs with  $\lambda$  equal to 0, 0.5, and 1.0 using standard dynamic programming (Bellman error curves plotted in Fig. 7.11), and record the returns obtained while performing GPE+GPI with each of these representations over the course of 50 episodes. Bellman error curves for the  $\lambda$ Rs are In the left panel of Fig. 7.5, we can indeed see that using the correct  $\lambda$  (0.5) nets the highest returns. Example trajectories for each of  $\lambda$ R are shown in the remaining panels.

### 7.E.6 Pixel-Based GPI

In this case, the base policies  $\Pi$  were identical to those used in the tabular GPI experiments. First, we collected a dataset consisting of 340 observation trajectories  $(o_0, o_1, \dots, o_{H-1}) \in \mathcal{O}^H$  with  $H = 19$  from each policy, totalling 6,460 observations. Raw observations were  $128 \times 128 \times 3$  and were converted to grayscale. The previous seven observations were stacked and used to train a Laplacian eigenfunction base feature network in the same way as [Touati et al. \(2023\)](#). For observations less than seven steps from the start of an episode, the remaining frames were filled in as all black observations (i.e., zeros). The network consisted of four convolutional layers with  $32 \ 3 \times 3$  filters with strides  $(2, 2, 2, 1)$ , each followed by a ReLU nonlinearity. This was then flattened and passed through a Layer Norm layer ([Ba et al., 2016](#)) and a tanh nonlinearity before three fully fully connected layers, the first two with 64 units each and ReLU nonlinearities and the final, output layer with 50 units. The output was  $L_2$ -normalized as in [Touati et al. \(2023\)](#). This network  $\phi : \mathcal{O}^7 \mapsto \mathbb{R}^D$  (with  $D = 50$ ) was trained on the stacked observations for 10 epochs using the Adam optimizer and learning rate  $1e-4$  with batch size  $B = 64$ . To perform policy evaluation, the resulting features, evaluated on the dataset of stacked observations were collected into their own dataset of  $(s_t, a_{t+1}, s_{t+1}, a_{t+1})$  tuples, where  $s_t \triangleq o_{t-6:t}$ . The “states” were normalized to be between 0 and 1, and a vector  $\mathbf{w}$  was fit to the actual associated rewards via linear regression on the complete dataset. The  $\lambda F$  network was then trained using a form of neural fitted Q-iteration (FQI; [Riedmiller, 2005](#)) modified for policy evaluation with  $\lambda F$ s (Algorithm [10](#)). The architecture for the  $\lambda F$  network was identical to the base feature network, with the exception that the hidden size of the fully connected layers was 128 and the output dimension was  $D|\mathcal{A}| = 250$ . FQI was run for  $K = 20$  outer loop iterations, with each inner loop supervised learning setting run for  $L = 100$  epochs on the current dataset. Supervised learning was done using Adam with learning rate  $3e-4$  and batch size  $B = 64$ . Given the trained networks, GPI proceeded as in the tabular case, i.e.,

$$a_t = \operatorname{argmax}_{a \in \mathcal{A}} \max_{\pi \in \Pi} \mathbf{w}^\top \varphi_\theta^\pi(s_t, a). \quad (7.19)$$



**Figure 7.8: Learning curves for  $\lambda F$  policy evaluation.** Results are averaged over three runs, with shading indicating one unit of standard error.

50 episodes were run from random starting locations for  $H = 50$  steps and the returns measured. Learning curves for the base features and for  $\lambda F$  fitting are shown in Fig. 7.8. The  $\lambda F$  curve measures the mean squared error as in Eq. (7.18).

The feature visualizations were created by performing PCA to reduce the average  $\lambda F$  representations for observations at each state in the environment to 2D. Each point in the scatter plot represents the reduced representation on the  $xy$  plane, and is colored according to the  $\lambda$ -conditioned value of the underlying state.

### 7.E.7 Continuous Control

$\lambda$ -SAC See Appendix 7.I for details.

### 7.E.8 Learning the $\lambda O$ with FB

Training the  $\lambda O$  with the FB parameterization proceeds in much the same way as in (Touati and Ollivier, 2021b), but adjusted for a different norm and non-Markovian environment. We summarize the learning procedure in Algorithm III. The loss function  $\mathcal{L}$  is derived in Appendix 7.H, with the addition of the following regularizer:

$$\|\mathbb{E}_{s \sim \rho} B_{\omega}(s) B_{\omega}(s)^{\top} - I\|^2.$$

This regularizer encourages  $B$  to be approximately orthonormal, which promotes identifiability of  $F_{\theta}$  and  $B_{\omega}$  (Touati and Ollivier, 2021b).

**Algorithm 10** Fitted  $Q_\lambda$ -Iteration

---

```

1: Require: Dataset of base features  $\{\phi(s) \in \mathbb{R}^D\}_{s \in \mathcal{S}}$ , decay rate  $\lambda$ , discount factor
    $\gamma$ , reward feature vector  $\mathbf{w} \in \mathbb{R}^D$ , batch size  $B$ , learning rate  $\alpha$ 
2: Initialize  $\lambda F$   $\varphi_\theta$  parameters  $\theta^{(1)}$  (we drop the subscript  $\lambda$  and superscript  $\pi$  for con-
   cision)
3: for  $k = 1 \dots, K$  do
4:   // Stage 1: Construct dataset
5:    $\mathcal{D} \leftarrow \emptyset$ 
6:   for  $(s, a) \in \mathcal{S} \times \mathcal{A}$  do
7:     for  $(s', a') \in \mathcal{S} \times \mathcal{A}$  do
8:        $\mathcal{D} \leftarrow \mathcal{D} \cup \{((s, a), y(s, a))\}$  where
           
$$y(s, a) = \mathbf{w}^\top [\phi(s) \odot (1 + \lambda \gamma \bar{\varphi}_{\theta^{(k)}}(s', a')) + \gamma(1 - \phi(s)) \odot \bar{\varphi}_{\theta^{(k)}}(s', a')]$$

9:     end for
10:   end for
11:   // Stage 2: Supervised learning
12:   Randomly initialize  $\theta_0$ 
13:   for  $\ell = 1, \dots, L$  do
14:     Randomly shuffle  $\mathcal{D}$ 
15:     for  $\{((s, a), y)\}_{b=1}^B \in \mathcal{D}$  do
16:        $\theta_\ell \leftarrow \theta_{\ell-1} - \alpha \nabla_{\theta} \frac{1}{2B} \sum_{b=1}^B (y_b - \mathbf{w}^\top \varphi_{\theta_{\ell-1}}(s_b, a_b))^2$ 
17:     end for
18:   end for
19:    $\theta^{(k+1)} \leftarrow \theta_L$ 
20: end for

```

---

$$\begin{aligned}
\mathcal{L}(\theta, \omega) = & \frac{1}{2B^2} \sum_{j,k \in J^2} \left( F_\theta(s_j, a_j, z_j)^\top B_\omega(s'_k) \right. \\
& \left. - \gamma \sum_{a \in \mathcal{A}} \pi_{z_j}(a|s_{j+1}) F_{\theta^-}(s_{j+1}, a, z_j)^\top B_{\omega^-}(s'_k) \right)^2 \\
& - \frac{1}{B} \sum_{j \in J} F_\theta(s_j, a_j, z_j)^\top B_\omega(s_j) \\
& + \frac{\gamma(1-\lambda)}{B} \sum_{j \in J} \mu(s_j) F_\theta(s_j, a_j, z_j)^\top B_\omega(s_j) \\
& \cdot \sum_{a \in \mathcal{A}} \pi_{z_j}(a|s_{j+1}) F_{\theta^-}(s_{j+1}, a, z_j)^\top B_{\omega^-}(s_j) \\
& + \beta \left( \frac{1}{B^2} \sum_{j,k \in J^2} B_\omega(s_j)^\top \bar{B}_\omega(\tilde{s}_k) \bar{B}_\omega(s_j)^\top \bar{B}_\omega(\tilde{s}_k) - \frac{1}{B} \sum_{j \in J} B_\omega(s_j)^\top \bar{B}_\omega(s_j) \right)
\end{aligned} \tag{7.20}$$



**Algorithm 11**  $\lambda$ O FB Learning

---

```

1: Require: Probability distribution  $\nu$  over  $\mathbb{R}^d$ , randomly initialized networks  $F_\theta, B_\omega$ ,
   learning rate  $\eta$ , mini-batch size  $B$ , number of episodes  $E$ , number of epochs  $M$ ,
   number of time steps per episode  $T$ , number of gradient steps  $N$ , regularization co-
   efficient  $\beta$ , Polyak coefficient  $\alpha$ , initial diminishing constant  $\lambda$ , discount factor  $\gamma$ ,
   exploratory policy greediness  $\epsilon$ , temperature  $\tau$ 
2: // Stage 1: Unsupervised learning phase
3:  $\mathcal{D} \leftarrow \emptyset$ 
4: for epoch  $m = 1, \dots, M$  do
5:   for episode  $i = 1 \dots, E$  do
6:     Sample  $z \sim \nu$ 
7:     Observe initial state  $s_1$ 
8:     for  $t = 1, \dots, T$  do
9:       Select  $a_t \in \text{greedy}$  with respect to  $F_\theta(s_t, a, z)^\top z$ 
10:      Observe reward  $r_t(s_t)$  and next state  $s_{t+1}$ 
11:       $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r_t(s_t), s_{t+1})\}$ 
12:    end for
13:  end for
14:  for  $n = 1, \dots, N$  do
15:    Sample a minibatch  $\{(s_j, a_j, r_j(s_j), s_{j+1})\}_{j \in J} \subset \mathcal{D}$  of size  $|J| = B$ 
16:    Sample a minibatch  $\{\tilde{s}_j\}_{j \in J} \subset \mathcal{D}$  of size  $|J| = B$ 
17:    Sample a minibatch  $\{s'_j\}_{j \in J} \stackrel{\text{iid}}{\sim} \mu$  of size  $|J| = B$ 
18:    Sample a minibatch  $\{z_j\}_{j \in J} \stackrel{\text{iid}}{\sim} \nu$  of size  $|J| = B$ 
19:    For every  $j \in J$ , set  $\pi_{z_j}(\cdot | s_{j+1}) = \text{softmax}(F_{\theta^-}(s_{j+1}, \cdot, z_j)^\top z_j / \tau)$ 
20:    Update  $\theta$  and  $\omega$  via one step of Adam on  $\mathcal{L}(\theta, \omega)$ 
21:    Sample a minibatch  $\{(s_j, r_j(s_j), s_{j+1}, r_{j+1}(s_{j+1}))\}_{j \in J}$  of size  $|J| = B$  from
       $\mathcal{D}$ 
22:     $\mathcal{L}_\lambda(\lambda) \leftarrow \frac{1}{2B} \sum_{j \in J} \mathbb{1}(s_{j+1} = s_j) (r_{j+1}(s_{j+1}) - \lambda r_j(s_j))^2$ 
23:    Update  $\lambda$  via one step of Adam on  $\mathcal{L}_\lambda(\theta, \omega)$  (Eq. (7.20))
24:  end for
25:   $\theta^- \leftarrow \alpha \theta^- + (1 - \alpha) \theta$ 
26:   $\omega^- \leftarrow \alpha \omega^- + (1 - \alpha) \omega$ 
27: end for
28: // Stage 2: Exploitation phase for a single episode with initial reward  $r_0(s)$ 
29:  $z_R \leftarrow \sum_{s \in \mathcal{S}} \mu(s) r_0(s) B_\omega(s)$ 
30: Observe initial state  $s_1$ 
31: for  $t = 1, \dots, T$  do
32:   $a_t \leftarrow \text{argmax}_{a \in \mathcal{A}} F(s_t, a, z_R)^\top z_R$ 
33:  Observe reward  $r_t(s)$  and next state  $s_{t+1}$ 
34:   $z_R \leftarrow \sum_{s \in \mathcal{S}} \mu(s) r_t(s) B_\omega(s)$ 
35: end for

```

---

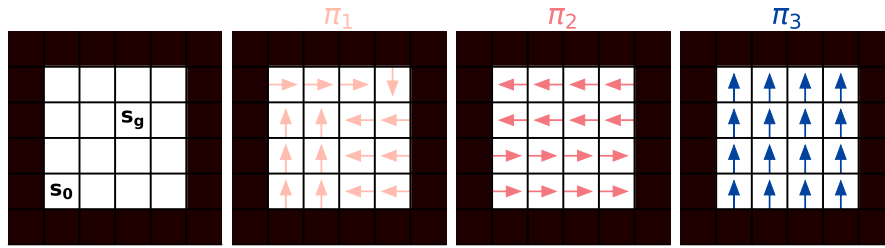


Figure 7.9: A simple grid and several policies.

## Appendix 7.F: Additional Results

See surrounding sections.

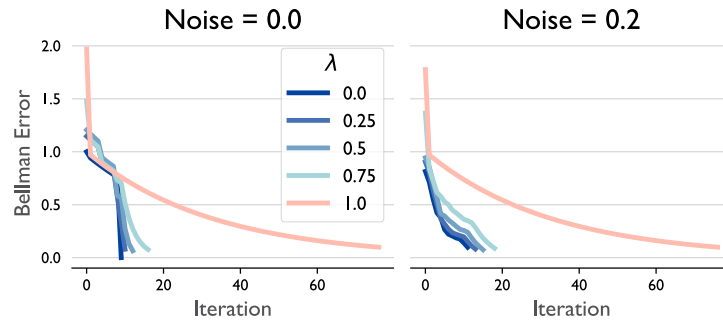
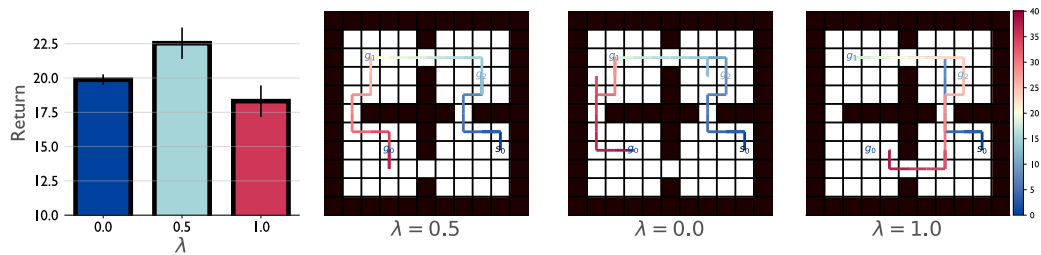


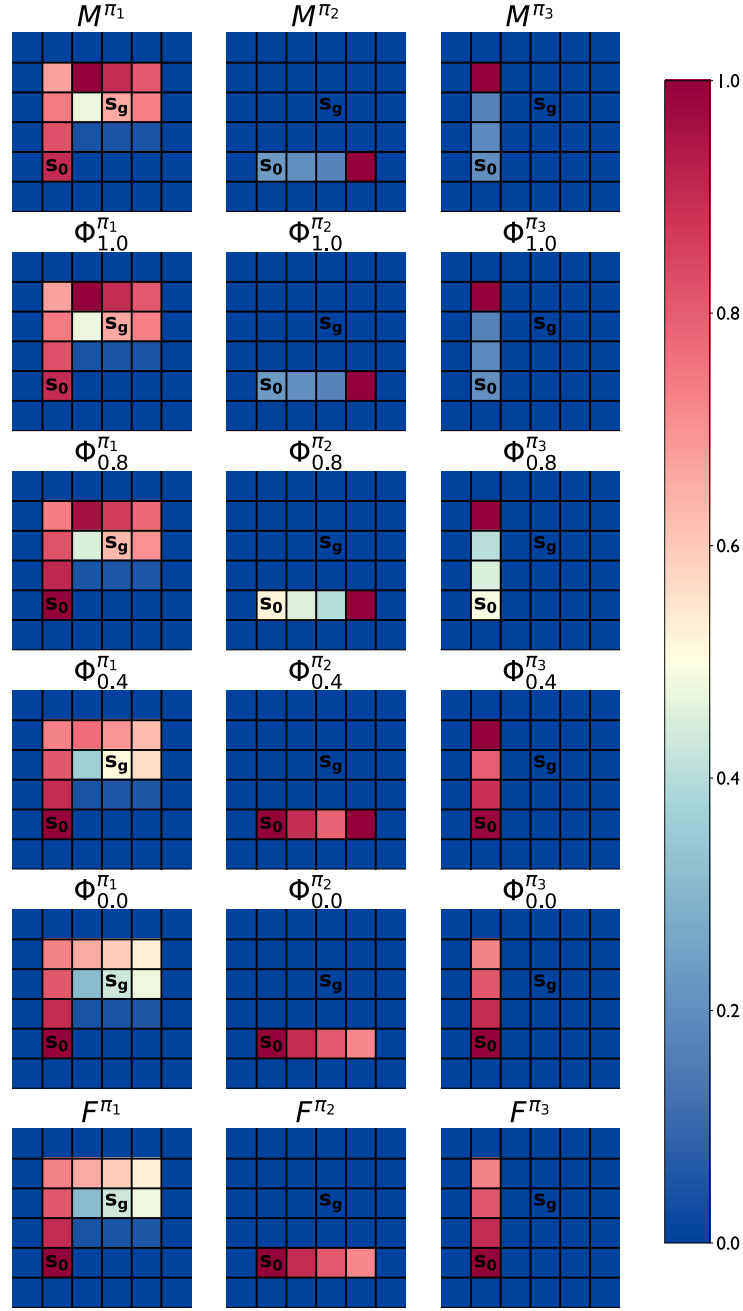
Figure 7.11: Convergence of dynamic programming on FourRooms with and without stochastic transitions.



**Figure 7.12: GPI with noisy transitions in FourRooms.** To verify that performance was maintained even with stochastic transitions, we added a 20% probability that a given action would result in a random transition to neighboring state. Results are consistent with Fig. 7.5, indicating the having the correct value of  $\lambda$  produces better performance.

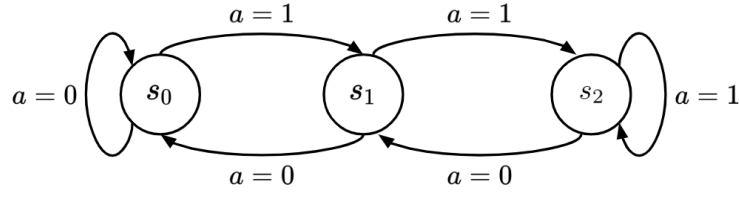
## Appendix 7.G: Advantage of the Correct $\lambda$

Importantly, for GPE using the  $\lambda R$  to work in this setting, the agent must either learn or be provided with the updated reward vector  $\mathbf{r}_\lambda$  after each step/encounter with a rewarded



**Figure 7.10: Visualizing the SR, the  $\lambda$ R and the FR.** We can see that the  $\Phi_1^\pi$  is equivalent to the SR and  $\Phi_0^\pi$  is equivalent to the FR, with intermediate values of  $\lambda$  providing a smooth transition between the two.

state. This is because the  $\lambda$ R is forward-looking in that it measures the (diminished) expected occupancies of states in the future without an explicit mechanism for remembering previous visits. For simplicity in this case, we provide this vector to the agent at each step—though if we view such a multitask agent as simply as a module carrying out the



**Figure 7.13: A 3-state toy environment.**

directives of a higher-level module or policy within a hierarchical framework as in, e.g., Feudal RL (Dayan and Hinton, 1992), the explicit provision of reward information is not unrealistic. Regardless, a natural question in this case is whether there is actually any value in using the  $\lambda$ R with the correct value of  $\lambda$  in this setting: If the agent is provided with the correct reward vector, then wouldn't policy evaluation work with any  $\lambda$ R?

To see that this is not the case, consider the three-state toy MDP shown in Figure Fig. 7.13, where  $\bar{r}(s_1) = 10$ ,  $\bar{r}(s_2) = 6$ ,  $\bar{r}(s_0) = 0$ ,  $\lambda(s_1) = 0$ ,  $\lambda(s_2) = 1.0$ , and  $\gamma = 0.99$ . At time  $t = 0$ , the agent starts in  $s_0$ . Performing policy evaluation with  $\lambda(s_1) = \lambda(s_2) = 1$  (i.e., with the SR) would lead the agent to go left to  $s_1$ . However, the reward would then disappear, and policy evaluation on the second step would lead it to then move right to  $s_0$  and then  $s_2$ , where it would stay for the remainder of the episode. In contrast, performing PI with the correct values of  $\lambda$  would lead the agent to go right to  $s_2$  and stay there. In the first two timesteps, the first policy nets a total reward of  $10 + 0 = 10$ , while the second policy nets  $6 + 5.94 = 11.94$ . (The remaining decisions are identical between the two policies.) This is a clear example of the benefit of having the correct  $\lambda$ , as incorrect value estimation leads to suboptimal decisions even when the correct reward vector/function is provided at each step.

## Appendix 7.H: The $\lambda$ Operator

To learn the  $\lambda$ O, we would like to define  $\Phi_\lambda^\pi(s_t, ds') \triangleq \varphi_\lambda^\pi(s_t, s')\mu(ds')$  for some base policy  $\mu$ . However, this would lead to a contradiction:

$$\Phi_\lambda^\pi(s, A \cup B) = \int_A \varphi_\lambda^\pi(s, ds')\mu(ds') + \int_B \varphi_\lambda^\pi(s, ds')\mu(ds') = \Phi_\lambda^\pi(s, A) + \Phi_\lambda^\pi(s, B)$$

for all disjoint  $A, B$ , contradicting Lemma 7.B.4.

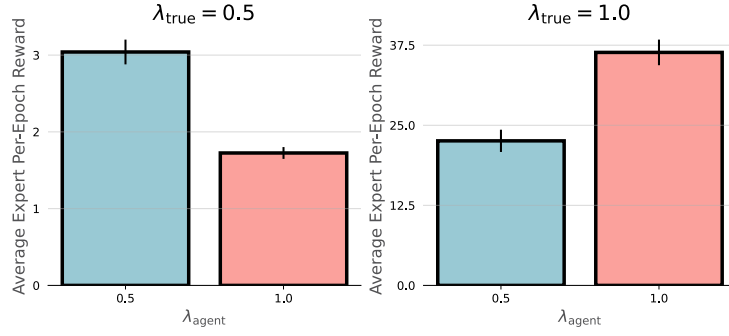
For now, we describe how to learn the  $\lambda$ O for discrete  $\mathcal{S}$ , in which case we have  $\Phi_\lambda^\pi(s, s') = \varphi_\lambda^\pi(s, s')\mu(s')$ , i.e., by learning  $\varphi$  we learn a weighted version of  $\Phi$ . We define the following norm, inspired by [Touati et al. \(2023\)](#):

$$\|\Phi_\lambda^\pi\|_\rho^2 \triangleq \mathbb{E}_{\substack{s \sim \rho \\ s' \sim \mu}} \left[ \left( \frac{\Phi_\lambda^\pi(s, s')}{\mu(s')} \right)^2 \right],$$

where  $\mu$  is any density on  $\mathcal{S}$ . In the case of finite  $\mathcal{S}$ , we let  $\mu$  be the uniform density. We then minimize the Bellman error for  $\Phi_\lambda^\pi$  with respect to  $\|\cdot\|_{\rho, \mu}^2$  (dropping the sub/superscripts on  $\Phi$  and  $\varphi$  for clarity):

$$\begin{aligned} \mathcal{L}(\Phi) &= \|\varphi\mu - (I \odot (\mathbf{1}\mathbf{1}^\top + \lambda\gamma P^\pi \varphi\mu) + \gamma(\mathbf{1}\mathbf{1}^\top + I) \odot P^\pi \varphi\mu)\|_{\rho, \mu}^2 \\ &= \mathbb{E}_{s_t \sim \rho, s' \sim \mu} \left[ \left( \varphi(s_t, s') - \frac{\mathbb{1}(s_t = s')}{\mu(s')} \right. \right. \\ &\quad \left. \left. + \gamma(1 - \lambda) \frac{\mathbb{1}(s_t = s')}{\mu(s')} \mathbb{E}_{s_{t+1} \sim p^\pi(\cdot|s_t)} \bar{\Phi}(s_{t+1}, s') - \gamma \mathbb{E}_{s_{t+1} \sim p^\pi(\cdot|s_t)} \bar{\varphi}(s_{t+1}, s') \right)^2 \right] \\ &\stackrel{+c}{=} \mathbb{E}_{s_t, s_{t+1} \sim \rho, s' \sim \mu} \left[ (\varphi(s_t, s') - \gamma \bar{\varphi}(s_{t+1}, s'))^2 \right] \\ &\quad - 2\mathbb{E}_{s_t, s_{t+1} \sim \rho} \left[ \sum_{s'} \mu(s') \varphi(s_t, s') \frac{\mathbb{1}(s_t = s')}{\mu(s')} \right] \\ &\quad + 2\gamma(1 - \lambda) \mathbb{E}_{s_t, s_{t+1} \sim \rho} \left[ \sum_{s'} \mu(s') \varphi(s_t, s') \bar{\varphi}(s_{t+1}, s') \mu(s') \frac{\mathbb{1}(s_t = s')}{\mu(s')} \right] \\ &\stackrel{+c}{=} \mathbb{E}_{s_t, s_{t+1} \sim \rho, s' \sim \mu} \left[ (\varphi(s_t, s') - \gamma \bar{\varphi}(s_{t+1}, s'))^2 \right] - 2\mathbb{E}_{s_t \sim \rho} [\varphi(s_t, s_t)] \\ &\quad + 2\gamma(1 - \lambda) \mathbb{E}_{s_t, s_{t+1} \sim \rho} [\mu(s_t) \varphi(s_t, s_t) \bar{\varphi}(s_{t+1}, s_t)], \end{aligned}$$

Note that we recover the SM loss when  $\lambda = 1$ . Also, an interesting interpretation is that when the agent can never return to its previous state (i.e.,  $\varphi(s_{t+1}, s_t) = 0$ ), then we also recover the SM loss, regardless of  $\lambda$ . In this way, the above loss appears to “correct” for repeated state visits so that the measure only reflects the first visit.



**Figure 7.14: Performance of the  $\lambda$ O-FB with two values of  $\lambda$ .** Results averaged over six seeds and 10 episodes per seed. Error bars indicate standard error.

$$\begin{aligned}
\mathcal{L}(\Phi) &= \mathbb{E}_{s_t, a_t, s_{t+1} \sim \rho, s' \sim \mu} \left[ \left( F(s_t, a_t, z)^\top B(s') - \gamma \bar{F}(s_{t+1}, \pi_z(s_{t+1}), z)^\top \bar{B}(s') \right)^2 \right] \\
&\quad - 2 \mathbb{E}_{s_t, a_t \sim \rho} \left[ F(s_t, a_t, z)^\top B(s_t) \right] \\
&\quad + 2\gamma(1 - \lambda) \mathbb{E}_{s_t, a_t, s_{t+1} \sim \rho} \left[ \mu(s_t) F(s_t, a_t, z)^\top B(s_t) \bar{F}(s_{t+1}, \pi_z(s_{t+1}), z)^\top \bar{B}(s_t) \right]
\end{aligned} \tag{7.21}$$

Even though the  $\lambda$ O is not a measure, we can use the above loss to the continuous case, pretending as though we could take the Radon-Nikodym derivative  $\frac{\Phi(s, ds')}{\mu(ds')}$ .

### 7.H.9 Experimental Results with the FB Parameterization

To show that knowing the correct value of  $\lambda$  leads to improved performance, we trained  $\lambda$ O with the FB parameterization on the FourRooms task of Fig. 7.5, but with each episode initialized at a random start state and with two random goal states. Average per-epoch reward is shown in Fig. 7.15. We tested performance with  $\lambda_{\text{true}}, \lambda_{\text{agent}} \in \{0.5, 1.0\}$ , where  $\lambda_{\text{true}}$  denotes the true environment diminishing rate and  $\lambda_{\text{agent}}$  denotes the diminishing rate that the agent uses. For the purpose of illustration, we do not allow the agent to learn  $\lambda$ . We see in Fig. 7.15 that using the correct  $\lambda$  leads to significantly increased performance. In particular, the left plot shows that assuming  $\lambda = 1$ , i.e., using the SR, in a diminishing environment can lead to highly suboptimal performance.

Hyperparameters used are given in Table 7.1 (notation as in Algorithm III).

Hyperparameter	Value
$M$	100
$E$	100
$N$	25
$B$	128
$T$	50
$\gamma$	0.99
$\alpha$	0.95
$\eta$	0.001
$\tau$	200
$\epsilon$	1

Table 7.1:  $\lambda$ O-FB hyperparameters.

### 7.H.10 $\lambda$ O and the Marginal Value Theorem

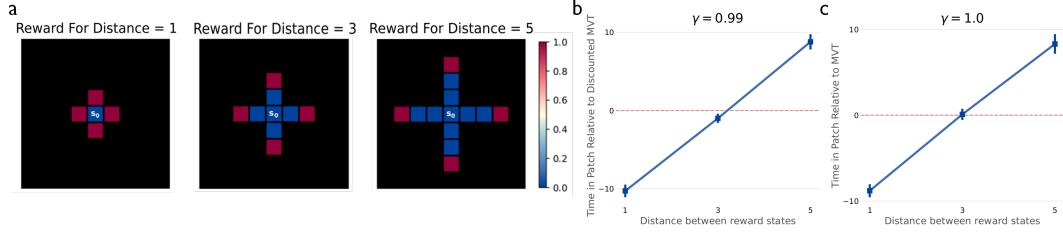
To study whether the agent’s behavior is similar to behavior predicted by the MVT, we use a very simple task with constant starting state and vary the distance between rewards (see Fig. 7.14(a)). When an agent is in a reward state, we define an MVT-optimal leaving time as follows (similar to that of (Wisinski et al., 2023) but accounting for the non-stationarity of the reward).

Let  $R$  denote the average per-episode reward received by a trained agent,  $r(s_t)$  denote the reward received at time  $t$  in a given episode,  $R_t = \sum_{u=0}^t r(s_u)$  denote the total reward received until time  $t$  in the episode, and let  $T$  be episode length. Then, on average, the agent should leave its current reward state at time  $t$  if the next reward that it would receive by staying in  $s_t$ , i.e.,  $\lambda r(s_t)$ , is less than

$$\frac{R - R_t}{T}.$$

In other words, the agent should leave a reward state when its incoming reward falls below the diminished average per-step reward of the environment. We compute  $R$  by averaging reward received by a trained agent over many episodes.

Previous studies have trained agents that assume stationary reward to perform foraging tasks, even though the reward in these tasks is non-stationary. These agents can still achieve good performance and MVT-like behavior (Wisinski et al., 2023). However, be-



**Figure 7.15: Analysis of MVT-like behavior of  $\lambda$ O-FB.** **a)** Three environments with equal start state and structure but different distances between reward states. **b)** Difference between the agent's leave times and MVT-predicted leave times for  $\gamma = 0.99$ , with discounting taken into account. The agent on average behaves similar to the discounted MVT. **c)** Difference between the agent's leave times and MVT-predicted leave times for  $\gamma = 1.0$ , i.e., with no discounting taken into account. The agent on average behaves similar to the MVT. Results for (b) and (c) are averaged over three seeds. Error bars indicate standard error.

cause they target the standard RL objective

$$\mathbb{E}_{\pi} \left[ \sum_{k=0}^{\infty} \gamma^k r(s_{t+k}) \mid s_t = s \right],$$

which requires  $\gamma < 1$  for convergence, optimal behavior is recovered only with respect to the *discounted MVT*, in which  $R$  (and in our case,  $R_t$ ) weights rewards by powers of  $\gamma$  (Wispiński et al., 2023).

In Fig. 7.14(b-c) we perform a similar analysis to that of (Wispiński et al., 2023) and show that, on average over multiple distances between rewards,  $\lambda$ O-FB performs similarly to the discounted MVT for  $\gamma = 0.99$  and the standard MVT for  $\gamma = 1.0$ . An advantage of the  $\lambda$ O is that it is finite for  $\gamma = 1.0$  provided that  $\lambda < 1$ . Hence, as opposed to previous work, we can recover the standard MVT without the need to adjust for discounting.

Hyperparameters used are given in Table 7.1 (notation as in Algorithm II).

## Appendix 7.I: SAC

**Mitigating Value Overestimation** One well-known challenge in deep RL is that the use of function approximation to compute values is prone to overestimation. Standard approaches to mitigate this issue typically do so by using *two* value functions and either taking the minimum  $\min_{i \in \{1,2\}} Q_i^{\pi}(s, a)$  to form the Bellman target for a given



$(s, a)$  pair (Fujimoto et al., 2018) or combining them in other ways (Moskovitz et al., 2021b). However, creating multiple networks is expensive in both computation and memory. Instead, we hypothesized that it might be possible to address this issue by using  $\lambda$ -based values. To test this idea, we modified the Soft Actor-Critic (SAC; Haarnoja et al., 2018) algorithm to compute  $\lambda$ Fs-based values by augmenting the soft value target  $\mathcal{T}_{soft}Q = r_t + \gamma \mathbb{E}V_{soft}(s_{t+1})$ , where  $V_{soft}(s_{t+1})$  is given by the expression

$$\mathbb{E}_{a_{t+1} \sim \pi(\cdot | s_{t+1})} \left[ \bar{Q}(s_{t+1}, a_{t+1}) + (\lambda - 1) \mathbf{w}^\top (\phi(s_t, a_t) \odot \varphi_\lambda(s_{t+1}, a_{t+1})) - \alpha \log \pi(a_{t+1} | s_{t+1}) \right]$$

A derivation as well as pseudocode for the modified loss is provided in Section 7.E.7. Observe that for  $\lambda = 1$ , we recover the standard SAC value target, corresponding to an assumed stationary reward. We apply this modified SAC algorithm, which we term  $\lambda$ -SAC to feature-based Mujoco continuous control tasks within OpenAI Gym (Brockman et al., 2016). We found that concatenating the raw state and action observations  $\tilde{\phi}_t = [s_t, a_t]$  and normalizing them to  $[0, 1]$  make effective regressors to the reward. That is, we compute base features as

$$\phi_t^b = \frac{\tilde{\phi}_t^b - \min_b \tilde{\phi}_t^b}{\max_b \tilde{\phi}_t^b - \min_b \tilde{\phi}_t^b},$$

where  $b$  indexes  $(s_t, a_t)$  within a batch. Let  $X \in [0, 1]^{B \times D}$  be the concatenated matrix of features for a batch, where  $D = \dim(\mathcal{S}) + \dim(\mathcal{A})$ . Then,

$$\mathbf{w}_t = (X^\top X)^{-1} X^\top \mathbf{r},$$

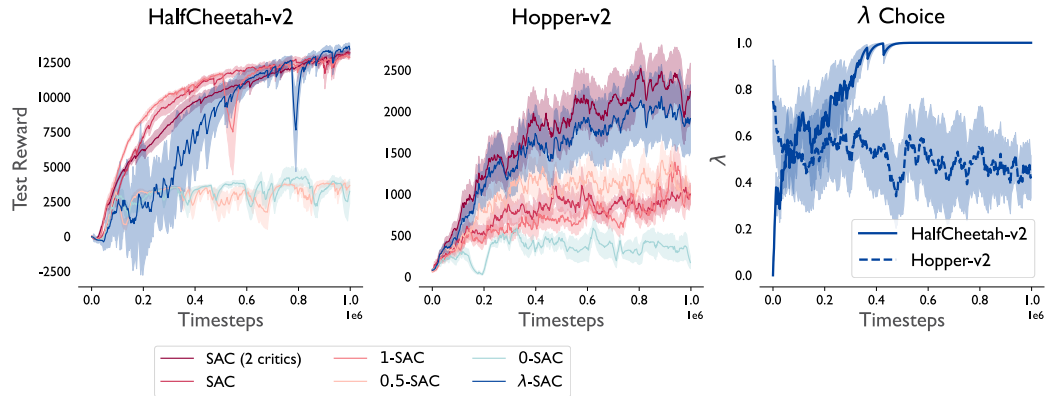
where here  $\mathbf{r}$  denotes the vector of rewards from the batch. In addition to using a fixed  $\lambda$  value, ideally we'd like an agent to adaptively update  $\lambda$  to achieve the best balance of optimism and pessimism in its value estimates. Following Moskovitz et al. (2021b), we frame this decision as a multi-armed bandit problem, discretizing  $\lambda$  into three possible values  $\{0, 0.5, 1.0\}$  representing the arms of the bandit. At the start of each episode, a random value of  $\lambda$  is sampled from these arms and used in the value function update.

The probability of each arm is updated using the Exponentially Weighted Average Forecasting algorithm (Cesa-Bianchi and Lugosi, 2006), which modulates the probabilities in proportion to a feedback score. As in (Moskovitz et al., 2021b), we use the difference in cumulative (undiscounted) reward between the current episode  $\ell$  and the previous one  $\ell - 1$  as this feedback signal:  $R_\ell - R_{\ell-1}$ . That is, the probability of selecting a given value of  $\lambda$  increases if performance is improving and decreases if it's decreasing. We use identical settings for the bandit algorithm as in (Moskovitz et al., 2021b). We call this variant  $\lambda$ -SAC.

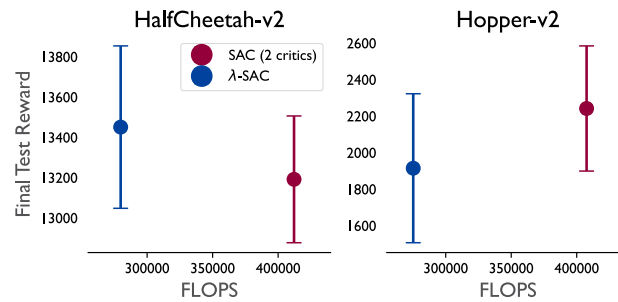
We plot the results for SAC with two critics (as is standard), SAC with one critic, SAC with a single critic trained with  $\lambda$ F-based values (“ $x$ -SAC” denotes SAC trained with a fixed  $\lambda = x$ ), and  $\lambda$ -SAC trained on the HalfCheetah-v2 and Hopper-v2 Mujoco environments. All experiments were repeated over eight random seeds. HalfCheetah-v2 was found by (Moskovitz et al., 2021b) to support “optimistic” value estimates in that even without pessimism to reduce overestimation it was possible to perform well. Consistent with this, we found that single-critic SAC matched the performance of standard SAC, as did 1-SAC (which amounts to training a standard value function with the auxiliary task of SF prediction). Fixing lower values of  $\lambda$  performed poorly, indicating that over-pessimism is harmful in this environment. However,  $\lambda$ -SAC eventually manages to learn to set  $\lambda = 1$  and matches the final performance of the best fixed algorithms. Similarly, in (Moskovitz et al., 2021b) it was observed that strong performance in Hopper-v2 was associated with pessimistic value estimates. Consistent with this,  $\lambda$ -SAC learns to select lower values of  $\lambda$ , again matching the performance of SAC while only requiring one critic and significantly reducing the required FLOPS Fig. 7.17. We consider these results to be very preliminary, and hope to perform more experiments on other environments. We also believe  $\lambda$ -SAC could be improved by using the difference between the current episode’s total reward and the *average* of the total rewards from previous episodes  $R_\ell - (\ell - 1)^{-1} \sum_{i=1}^{\ell-1} R_i$  as a more stable feedback signal for the bandit. There is also non-stationarity in the base features due to the per-batch normalization, which could also likely be improved. Hyperparameters are described in Table 7.2.

Hyperparameter	Value
Collection Steps	1000
Random Action Steps	10000
Network Hidden Layers	256:256
Learning Rate	$3 \times 10^{-4}$
Optimizer	Adam
Replay Buffer Size	$1 \times 10^6$
Action Limit	$[-1, 1]$
Exponential Moving Avg. Parameters	$5 \times 10^{-3}$
(Critic Update:Environment Step) Ratio	1
(Policy Update:Environment Step) Ratio	1
Has Target Policy?	No
Expected Entropy Target	$-\dim(\mathcal{A})$
Policy Log-Variance Limits	$[-20, 2]$
Bandit Learning Rate*	0.1
$\lambda$ Options*	$\{0, 0.5, 1.0\}$

**Table 7.2:** Hyperparameters for SAC Mujoco experiments. \*Only applicable to  $\lambda$ -SAC



**Figure 7.16:**  $\lambda$ -SAC (1 critic) matches the performance of SAC (2 critics) by adapting  $\lambda$  online.



**Figure 7.17:**  $\lambda$ -SAC matches performance while saving in computational cost.

## Appendix 7.J: Replenishing Rewards

We list below a few candidate reward replenishment schemes, which are visualized in Fig. 7.18.

### Time elapsed rewards

$$r(s, t) = \lambda^{n(s, t)/m(s, t)} \bar{r}(s),$$

where  $m(s, t)$  is the time elapsed since the last visit to state  $s$ :

$$m(s, t) \triangleq t - \max\{j | s_{t+j} = s, 0 \leq j \leq t-1\}.$$

Due to the max term in  $m(s, t)$ , the corresponding representation

$$\mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k \lambda^{n(s, t)/m(s, t)} \mathbb{1}(s_{t+k} = s') \middle| s_t = s \right]$$

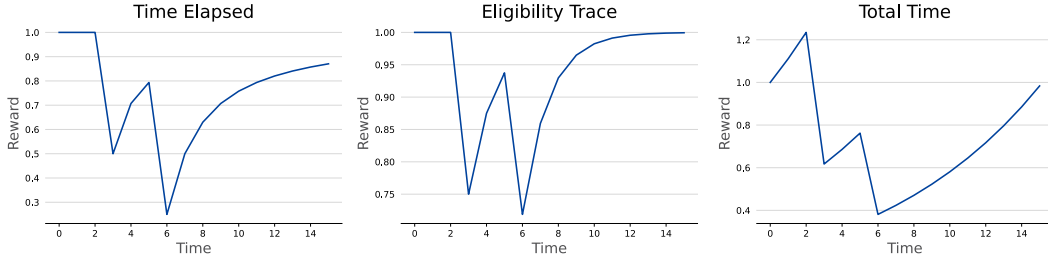
does not admit a closed-form recursion. However, we empirically tested a version of this type of reward with  $Q_\lambda$ -learning in the TwoRooms environment, modified so that the exponent on  $\lambda$  is  $n(s, t)/(0.1m(s, t))$ . This was done so that reward replenishes at a slow rate, reducing the deviation from the standard diminishing setting. Episode length was capped at  $H = 100$ . All other settings are identical to the  $Q_\lambda$  experiment described in Appendix 7.E. Results are presented in Fig. 7.19 and a GIF is included in the supplementary material.

### Eligibility trace rewards

$$r(s, t) = \left( 1 - (1 - \lambda_d) \sum_{j=0}^{t-1} \lambda_r^{t-j} \mathbb{1}(s_{t+j} = s) \right) \bar{r}(s),$$

where  $\lambda_d, \lambda_r \in [0, 1]$  are diminishment and replenishment constants, respectively. Denoting the corresponding representation by  $\Omega^\pi$ , i.e.,

$$\begin{aligned} & \Omega^\pi(s, s') \\ &= \mathbb{E} \left[ \sum_{k=0}^{\infty} \gamma^k \left( 1 - (1 - \lambda_d) \sum_{j=0}^k \lambda_r^{k-j} \mathbb{1}(s_{t+j} = s') \right) \mathbb{1}(s_{t+k} = s') \middle| s_t = s \right], \end{aligned}$$



**Figure 7.18: Visualizing three different replenishment schemes.** For all schemes,  $\bar{r}(s) = 1$  and visits to  $s$  are at  $t = 2, 5$ . (Left) The time elapsed reward with  $\lambda = 0.5$ ; (Middle) The eligibility trace reward with  $\lambda_r = \lambda_d = 0.5$ ; (Right) The total time reward with  $\lambda_d = 0.5, \lambda_r = 0.9$ .

we obtain the following recursion:

$$\begin{aligned} \Omega^\pi(s, s') = \mathbb{1}(s = s')(\lambda_d - \gamma\lambda_r(1 - \lambda_d)\mathbb{E}_{s_{t+1} \sim p^\pi(\cdot|s)} M_{\gamma\lambda_r}^\pi(s_{t+1}, s')) \\ + \gamma\mathbb{E}_{s_{t+1} \sim p^\pi(\cdot|s)} \Omega^\pi(s_{t+1}, s'), \end{aligned}$$

where  $M_{\gamma\lambda_r}^\pi$  denotes the successor representation of  $\pi$  with discount factor  $\gamma\lambda_r$ . This representation could be learned by alternating TD learning between  $\Omega^\pi$  and  $M_{\gamma\lambda_r}^\pi$ . We leave this to future work.

### Total time rewards

$$r(s, t) = \lambda_d^{n(s,t)} \lambda_r^{n(s,t)-t} \bar{r}(s),$$

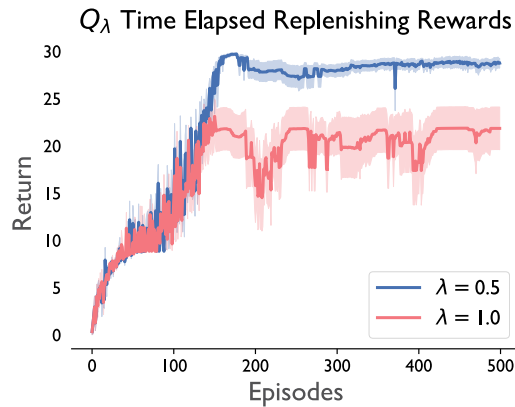
where  $\lambda_d, \lambda_r \in [0, 1]$  are diminishment and replenishment constants, respectively. The corresponding representation is

$$P^\pi(s, s') = \mathbb{E} \left[ \sum_{k=0}^{\infty} \gamma^k \lambda_d^{n_t(s',k)} \lambda_r^{k-n_t(s',k)} \mathbb{1}(s_{t+k} = s') \middle| s_t = s \right],$$

which satisfies the following recursion:

$$\begin{aligned} P^\pi(s, s') = \mathbb{1}(s = s') + \gamma(\lambda_d \mathbb{1}(s = s') + 1 \\ - \mathbb{1}(s = s'))(\lambda_r(1 - \mathbb{1}(s = s')) + \mathbb{1}(s = s')) \mathbb{E}_{s_{t+1} \sim p^\pi(\cdot|s)} P^\pi(s_{t+1}, s'). \end{aligned}$$

While neither the reward nor the representation are guaranteed to be finite,  $P^\pi$  could be learned via TD updates capped at a suitable upper bound.



**Figure 7.19: Performance on TwoRooms with replenishing rewards.** Return is averaged over five runs, with shading indicating one unit of standard error.

## Appendix 7.K: $\lambda$ vs. $\gamma$

We now briefly discuss the interaction between the temporal discount factor  $\gamma$  commonly used in RL and the diminishing utility rate  $\lambda$ . The key distinction between the two is that all rewards decay in value every time step with respect to  $\gamma$ , regardless of whether a state is visited or not. With  $\lambda$ , however, decay is specific to each state (or  $(s, a)$  pair) and only occurs when the agent visits that state. Thus,  $\gamma$  decays reward in a global manner which is independent of the agent's behavior, and  $\lambda$  decays reward in a local manner which dependent on the agent's behavior. In combination, they have the beneficial effect of accelerating convergence in dynamic programming (Fig. 7.11). This indicates the potential for the use of higher discount factors in practice, as paired with a decay factor  $\lambda$ , similar (or faster) convergence rates could be observed even as agents are able to act with a longer effective temporal horizon.

## Appendix 7.L: Compute Resources

The  $\lambda$ F-based experiments shown were run on a single NVIDIA GeForce GTX 1080 GPU. The recurrent A2C experiments took roughly 30 minutes, base feature learning for policy composition took approximately 45 minutes,  $\lambda$ F learning for policy composition took approximately 10 hours, and the SAC experiments took approximately 8 hours per run. The  $\lambda$ F training required roughly 30GB of memory due to the size of the dataset. All experiments in Section 7.4 and Appendix 7.H were run on a single RTX5000 GPU

and each training and evaluation run took about 30 minutes. All other experiments were run on a 2020 MacBook Air laptop 1.1 GHz Quad-Core Intel Core i5 CPU and took less than one hour to train.

# Conclusion



## Chapter 8

# General Conclusions

### 8.1 Summary

Recalling Chapter [1](#), the question which motivated the work presented in this thesis was as follows:

*How can bounded agents learn and act efficiently in a seemingly unbounded world?*

The answer to this question, in the most general sense, is *structure*: patterns in the world and in the goals an agent wishes to accomplish allow it to identify how to act with much less computational expense than would otherwise be possible. The last five chapters have focused on two forms of structure which are plausibly present in real world decision-making and which can be exploited by decision-makers.

Part I described how shared behavioral structure across tasks can be used by agents to learn new policies more quickly. Specifically, Chapter [3](#) showed that the smaller the largest disagreement among optimal policies is for a group of tasks, the more quickly that an agent can learn to solve new tasks from the same group. This structure is captured in the form of a default policy which asymptotically approaches the barycenter of the optimal policies for the task family faced by the agent. The decision-making control policy is then guided by a KL penalty for deviating from the default policy. While this approach can provide significant speed-ups in learning when such behavioral structure is present, it can fail if the default policy overfits to optimal policies seen early in training whose decisions don't reflect the overall behavioral patterns required for the population of tasks at hand. Chapter [4](#) addresses this shortcoming, deriving a minimum description

length-inspired policy optimization objective which regularizes the complexity of the default policy to prevent it from overfitting in this way. This approach, MDL-C, achieves both theoretical performance guarantees and strong empirical performance in discrete and continuous control tasks. In Chapter 5, MDL-C is applied to a variety of behavioral simulations from cognitive control, reward-based learning, and judgment and decision-making, and is found to qualitatively match human and animal decision-making patterns which are consistent with dual process theories of cognition.

Part II focused on the advantages afforded to agents by a different form of structure: consistent environment dynamics across tasks. Previous work (Dayan, 1993; Barreto et al., 2017) established that this form of structure can be used to learn the successor representation (SR), a state representation which enables an agent to quickly perform policy evaluation and generalized policy improvement (GPI) for new tasks. In Chapter 6, the SR is extended to account for tasks whose rewards are depleted after they are first accessed. While seemingly esoteric, this form of reward structure underpins many artificially and biologically relevant decision-making problems, such as efficient path planning, targeted exploration, foraging, and escape from predation. Like the SR, the resulting representation, the first-occupancy representation (FR), can be used for policy evaluation and GPI for tasks whose reward structures obey this “first occupancy” principle. However, unlike the SR, the FR can also be used as the basis of a shortest-path planning algorithm over policies. A static reward structure (the standard in MDPs) and a first occupancy-based reward structure represent two extremes of reward persistence, while often the degree to which stimuli are rewarding diminishes at intermediate rates across repeated exposures. This principle of diminishing marginal utility, well-studied in behavioral economics and psychology, has received scant attention in machine learning. Chapter 7 shows that solving tasks with this form of reward structure efficiently requires that an agent learn a representation which generalizes the SR and the FR, which we term the  $\lambda$ -representation.

## 8.2 Discussion

A promising direction for future work would be to consider the interactions between these (and other) forms of structure. Agents which could both learn policies more

quickly and compose them efficiently using these strategies would be exciting from the points of view of both machine learning and neuroscience. There are many interesting questions to be explored in creating agents which can both learn and compose policies, such as when composition rather than learning is “good enough” and which policies a bounded agent should maintain. Moreover, as discussed in Chapter 2, the decomposability of an agent’s goals and objectives into separate tasks is itself a form of structure, and one question that this thesis does not address (apart from FR planning for subgoal identification in navigation) is how to identify and exploit this structure, breaking down a long-horizon goal into the optimal sequence of manageable sub-problems. There are already promising works studying this problem in both machine learning (Singh, 1992; McGovern and Barto, 2001; Veeriah et al., 2021; Hafner et al., 2022; Dayan and Hinton, 1992; Vezhnevets et al., 2017; Xie et al., 2021) and neuroscience (Braver and Bongiolatti, 2002; Correa et al., 2020; Shamash et al., 2021b; Correa et al., 2023), and integrating these ideas with those presented here is an exciting prospect.

Finally, it’s worthwhile to consider the implications of the results presented here for the recent advances in developing extremely large-scale models for not only language modeling but also increasingly other modalities such as vision. These so-called *foundation models* (Bommasani et al., 2021), consisting of billions or trillions of parameters and trained on vast amounts of data, are driving rapid advances in machine learning across a variety of domains. There are three main avenues through which the methods discussed in this thesis can be relevant. First, while the majority of the FLOPs expended while training these models are spent in a self-supervised pre-training phase, an important component of this process is to fine-tune the resulting models to solve problems for which defining a clear self-supervised or supervised learning problem is challenging (e.g., to behave like a helpful chatbot) using *reinforcement learning from human feedback* (RLHF; Christiano et al., 2017). A common approach in RLHF is to fine-tune a model using policy optimization with a KL penalty with respect to the same model as it was before the start of RLHF. This is the exact scenario (in that the KL penalty is applied with respect to a default policy that is not necessarily similar to the control policy) studied in the single-task results of Chapter 3, and applying the results derived there to

improve RLHF could constitute a valuable contribution to the training process for these models. Second, an inability to perform consistent, grounded reasoning is a weakness of current models. One promising direction of work on this problem proposes to iteratively prompt these models when deployed to allow them to implement various search or planning strategies for more coherent reasoning (Zhou et al., 2023; Yao et al., 2024). Insights from methods like FR planning (Chapter 6) could be helpful in this pursuit. Finally, there is a major effort underway to make these models more “agentic” (Di Palo et al., 2023)—that is, to give them the ability to take actions within their environments (e.g., web interaction (Nakano et al., 2021) or robotics (Brohan et al., 2022)) and adapt appropriately to sparse, delayed feedback to solve long-horizon tasks. RL likely has much to contribute to this venture.

## References

- Elisa Ciaramelli. The role of ventromedial prefrontal cortex in navigation: a case of impaired wayfinding and rehabilitation. *Neuropsychologia*, 46(7):2099–2105, 2008.
- Kevin J. Miller, Carlos D. Brody, and Matthew M. Botvinick. Identifying model-based and model-free patterns in behavior on multi-step tasks. *bioRxiv*, 2016a. doi: 10.1101/096339.
- Kevin J Miller, Matthew M Botvinick, and Carlos D Brody. From predictive models to cognitive models: Separable behavioral processes underlying reward learning in the rat. *Europe PMC*, 2018.
- Kevin J Miller, Amitai Shenhav, and Elliot A Ludvig. Habits without values. *Psychological review*, 126(2):292, 2019.
- Marcel Binz, Samuel J Gershman, Eric Schulz, and Dominik Endres. Heuristics from bounded meta-learned inference. *Psychological review*, 2022.
- Colin M MacLeod and Kevin Dunbar. Training and stroop-like interference: evidence for a continuum of automaticity. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 14(1):126, 1988.
- Ian Osband, Yotam Doron, Matteo Hessel, John Aslanides, Eren Sezener, Andre Saraiva, Katrina McKinney, Tor Lattimore, Csaba Szepesvari, Satinder Singh, Benjamin Van Roy, Richard Sutton, David Silver, and Hado Van Hasselt. Behaviour suite for reinforcement learning. In *International Conference on Learning Representations*, 2020. URL <https://openreview.net/forum?id=rygf-kSYwH>.
- Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624, 1993. doi: 10.1162/neco.1993.5.4.613.
- Andre Barreto, Will Dabney, Remi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan,

- and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL <https://proceedings.neurips.cc/paper/2017/file/350db081a661525235354dd3e19b8c05-Paper.pdf>.
- Kristian Hartikainen, Xinyang Geng, Tuomas Haarnoja, and Sergey Levine. Dynamical distance learning for semi-supervised and unsupervised skill discovery. In *International Conference on Learning Representations*, 2020. URL <https://openreview.net/forum?id=H1lmhaVtvr>.
- Vitchyr Pong, Shixiang Gu\*, Murtaza Dalal, and Sergey Levine. Temporal difference models: Model-free deep RL for model-based control. In *International Conference on Learning Representations*, 2018. URL <https://openreview.net/forum?id=Skw0n-W0Z>.
- Leslie Pack Kaelbling. Learning to achieve goals. In *Proceedings of IJCAI-93*, pages 1094–1098. Morgan Kaufmann, 1993.
- Ted Moskovitz, Michael Arbel, Jack Parker-Holder, and Aldo Pacchiano. Towards an understanding of default policies in multitask policy optimization. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera, editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 10661–10686. PMLR, 28–30 Mar 2022a. URL <https://proceedings.mlr.press/v151/moskovitz22a.html>.
- Ted Moskovitz, Ta-Chu Kao, Maneesh Sahani, and Matthew Botvinick. Minimum description length control. In *The Eleventh International Conference on Learning Representations*, 2023a. URL <https://openreview.net/forum?id=oX3tGygjW1q>.
- Ted Moskovitz, Kevin Miller, Maneesh Sahani, and Matthew M. Botvinick. A unified theory of dual-process control, 2022b. URL <https://arxiv.org/abs/2211.07036>.
- Ted Moskovitz, Spencer R Wilson, and Maneesh Sahani. A first-occupancy representation for reinforcement learning. In *International Conference on Learning Representations*, 2022c. URL <https://openreview.net/forum?id=JBAZe2yN6Ub>.

- Ted Moskovitz, Samo Hromadka, Ahmed Touati, Diana Borsa, and Maneesh Sahani. A state representation for diminishing rewards. *Advances in Neural Information Processing Systems*, 36, 2024.
- Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018a. URL <http://incompleteideas.net/book/the-book-2nd.html>.
- Rishabh Agarwal, Marlos C. Machado, Pablo Samuel Castro, and Marc G. Bellemare. Contrastive behavioral similarity embeddings for generalization in reinforcement learning, 2021.
- David Abel, Will Dabney, Anna Harutyunyan, Mark K Ho, Michael Littman, Doina Precup, and Satinder Singh. On the expressivity of markov reward. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 7799–7812. Curran Associates, Inc., 2021.
- Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 1994.
- Richard S Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine learning proceedings 1990*, pages 216–224. Elsevier, 1990.
- Richard Bellman. *Dynamic Programming*. Dover Publications, 1957.
- Brian D Ziebart. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University, 2010.
- Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.
- Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.  
<http://www.deeplearningbook.org>.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dhharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. URL <https://doi.org/10.1038/nature14236>.

Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8:293–321, 1992.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. URL <http://arxiv.org/abs/1412.6980>. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

Jiechao Xiong, Qing Wang, Zhuoran Yang, Peng Sun, Lei Han, Yang Zheng, Haobo Fu, Tong Zhang, Ji Liu, and Han Liu. Parametrized deep q-networks learning: Reinforcement learning with discrete-continuous hybrid action space. *arXiv preprint arXiv:1810.06394*, 2018.

Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning with model-based acceleration. In *International conference on machine learning*, pages 2829–2838. PMLR, 2016.

Tim Seyde, Peter Werner, Wilko Schwarting, Igor Gilitschenski, Martin Riedmiller, Daniela Rus, and Markus Wulfmeier. Solving continuous control via q-learning. *arXiv preprint arXiv:2210.12566*, 2022.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya



- Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, January 2016.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.
- Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation, 2018.
- Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.
- Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1928–1937, New York, New York, USA, 20–22 Jun 2016. PMLR. URL <https://proceedings.mlr.press/v48/mniha16.html>.
- Richard Sutton. The reward hypothesis, 2004. URL <http://incompleteideas.net/rlai.cs.ualberta.ca/RLAI/rewardhypothesis.html>.
- David Silver, Satinder Singh, Doina Precup, and Richard S Sutton. Reward is enough. *Artificial Intelligence*, 299:103535, 2021.
- Michael Bowling, John D Martin, David Abel, and Will Dabney. Settling the reward hypothesis. In *International Conference on Machine Learning*, pages 3003–3020. PMLR, 2023.
- Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018b.

- Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.
- Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2019.
- Aldo Pacchiano, Ofir Nachum, Nilseh Tripuraneni, and Peter Bartlett. Joint representation training in sequential tasks with shared structure, 2022.
- David Abel, André Barreto, Benjamin Van Roy, Doina Precup, Hado van Hasselt, and Satinder Singh. A definition of continual reinforcement learning. *arXiv preprint arXiv:2307.11046*, 2023.
- Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards continual reinforcement learning: A review and perspectives. *Journal of Artificial Intelligence Research*, 75:1401–1476, 2022.
- Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.
- Michael Laskin, Denis Yarats, Hao Liu, Kimin Lee, Albert Zhan, Kevin Lu, Catherine Cang, Lerrel Pinto, and Pieter Abbeel. Urlb: Unsupervised reinforcement learning benchmark. In *NeurIPS 2021 Datasets and Benchmarks Track, Submitted*, 2021. URL [https://openreview.net/forum?id=lwrPkQP\\_is](https://openreview.net/forum?id=lwrPkQP_is).
- DJ Strouse, Kate Baumli, David Warde-Farley, Vlad Mnih, and Steven Hansen. Learning more skills through optimistic exploration. *arXiv preprint arXiv:2107.14226*, 2021.
- Satinder Pal Singh. Transfer of learning by composing solutions of elemental sequential tasks. *Machine learning*, 8:323–339, 1992.

Alexandre Galashov, Siddhant M. Jayakumar, Leonard Hasenclever, Dhruva Tirumala, Jonathan Schwarz, Guillaume Desjardins, Wojciech M. Czarnecki, Yee Whye Teh, Razvan Pascanu, and Nicolas Heess. Information asymmetry in kl-regularized RL. *CoRR*, abs/1905.01240, 2019a. URL <http://arxiv.org/abs/1905.01240>.

Yee Whye Teh, Victor Bapst, Wojciech Marian Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distal: Robust multitask reinforcement learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17*, page 4499–4509, 2017a.

Dhruva Tirumala, Hyeonwoo Noh, Alexandre Galashov, Leonard Hasenclever, Arun Ahuja, Greg Wayne, Razvan Pascanu, Yee Whye Teh, and Nicolas Heess. Exploiting hierarchy for learning and transfer in kl-regularized rl. *arXiv preprint arXiv:1903.07438*, 2019.

Dhruva Tirumala, Alexandre Galashov, Hyeonwoo Noh, Leonard Hasenclever, Razvan Pascanu, Jonathan Schwarz, Guillaume Desjardins, Wojciech Marian Czarnecki, Arun Ahuja, Yee Whye Teh, and Nicolas Heess. Behavior priors for efficient reinforcement learning. *arXiv preprint arXiv:2010.14274*, 2020a.

Andre Barreto, Shaobo Hou, Diana Borsa, David Silver, and Doina Precup. Fast reinforcement learning with generalized policy updates. *Proceedings of the National Academy of Sciences*, 117(48):30079–30087, 2020. ISSN 0027-8424. doi: 10.1073/pnas.1907370117. URL <https://www.pnas.org/content/117/48/30079>.

Chen Ma, Dylan R. Ashley, Junfeng Wen, and Yoshua Bengio. Universal successor features for transfer reinforcement learning, 2020.

Eszter V rt s and Maneesh Sahani. A neurally plausible model learns successor representations in partially observable environments. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alch  Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL <https://proceedings.neurips.cc/paper/2019/file/dea184826614d3f4c608731389ed0c74-Paper.pdf>.

Tom Zahavy, Andre Barreto, Daniel J Mankowitz, Shaobo Hou, Brendan O’Donoghue, Iurii Kemaev, and Satinder Singh. Discovering a set of policies for the worst case reward. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=PUkhWz65dy5>.

Alekh Agarwal, Sham Kakade, Akshay Krishnamurthy, and Wen Sun. Flambe: Structural complexity and representation learning of low rank mdps. *Advances in neural information processing systems*, 33:20095–20107, 2020a.

Yuan Cheng, Songtao Feng, Jing Yang, Hong Zhang, and Yingbin Liang. Provable benefit of multitask representation learning in reinforcement learning, 2022. URL <https://arxiv.org/abs/2206.05900>.

Alekh Agarwal, Yuda Song, Wen Sun, Kaiwen Wang, Mengdi Wang, and Xuezhou Zhang. Provable benefits of representational transfer in reinforcement learning, 2022. URL <https://arxiv.org/abs/2205.14571>.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International conference on machine learning*, pages 1312–1320. PMLR, 2015.

Diana Borsa, André Barreto, John Quan, Daniel Mankowitz, Rémi Munos, Hado Van Hasselt, David Silver, and Tom Schaul. Universal successor features approximators. *arXiv preprint arXiv:1812.07626*, 2018.

Burrhus Frederic Skinner. *Science and human behavior*. Number 92904. Simon and Schuster, 1965.

Allan R Wagner and Robert A Rescorla. Inhibition in pavlovian conditioning: Application of a theory. *Inhibition and learning*, pages 301–336, 1972.

Peter Dayan and Laurence F Abbott. *Theoretical neuroscience: computational and mathematical modeling of neural systems*. MIT press, 2005.

Wolfram Schultz. Predictive reward signal of dopamine neurons. *Journal of neurophysiology*, 1998.

- P Read Montague, Steven E Hyman, and Jonathan D Cohen. Computational roles for dopamine in behavioural control. *Nature*, 431(7010):760–767, 2004.
- Mitsuko Watabe-Uchida and Naoshige Uchida. Multiple dopamine systems: weal and woe of dopamine. In *Cold Spring Harbor symposia on quantitative biology*, volume 83, pages 83–95. Cold Spring Harbor Laboratory Press, 2018.
- Francesca Greenstreet, Hernando Martinez Vergara, Sthitapranjya Pati, Laura Schwarz, Matthew Wisdom, Fred Marbach, Yvonne Johansson, Lars Rollik, Theodore Moskovitz, Claudia Clopath, et al. Action prediction error: a value-free dopaminergic teaching signal that drives stable learning. *BiorXiv*, pages 2022–09, 2022.
- Will Dabney, Zeb Kurth-Nelson, Naoshige Uchida, Clara Kwon Starkweather, Demis Hassabis, Rémi Munos, and Matthew Botvinick. A distributional code for value in dopamine-based reinforcement learning. *Nature*, 577(7792):671–675, 2020.
- Edward Thorndike. *Animal intelligence: Experimental studies*. Macmillan, 1911.
- Ann M. Graybiel. Habits, rituals, and the evaluative brain. *Annual Review of Neuroscience*, 31(1):359–387, 2008. doi: 10.1146/annurev.neuro.29.051605.112851. URL <https://doi.org/10.1146/annurev.neuro.29.051605.112851>. PMID: 18558860.
- Nathaniel D Daw, Yael Niv, and Peter Dayan. Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature neuroscience*, 8(12):1704–1711, 2005.
- Ray J Dolan and Peter Dayan. Goals and habits in the brain. *Neuron*, 80(2):312–325, 2013.
- Kevin J. Miller, Amitai Shenhav, and Elliot A. Ludvig. Habits without values. *bioRxiv*, 2016b. doi: 10.1101/067603. URL <https://www.biorxiv.org/content/early/2016/08/03/067603>.
- Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review, 2018a.

Hiroki Furuta, Tadashi Kozuno, Tatsuya Matsushima, Yutaka Matsuo, and Shixiang Shane Gu. Co-adaptation of algorithmic and implementational innovations in inference-based deep reinforcement learning, 2021.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning. *CoRR*, abs/1912.06680, 2019.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures, 2018.

Sham M Kakade. A natural policy gradient. In *Advances in neural information processing systems*, pages 1531–1538, 2002.

Aldo Pacchiano, Jack Parker-Holder, Yunhao Tang, Anna Choromanska, Krzysztof Choromanski, and Michael I Jordan. Learning to score behaviors for guided policy optimization. In *The International Conference on Machine Learning*. JMLR, 2020.

Ted Moskovitz, Michael Arbel, Ferenc Huszar, and Arthur Gretton. Efficient wasserstein natural gradients for reinforcement learning, 2021a. URL <https://arxiv.org/abs/2010.05380>.

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015. URL <http://arxiv.org/abs/1502.05477>.

Joseph Marino, Alexandre Piché, Alessandro Davide Ialongo, and Yisong Yue. Iterative amortized policy optimization, 2020.

Alekh Agarwal, Sham M Kakade, Jason D Lee, and Gaurav Mahajan. Optimality and approximation with policy gradient methods in markov decision processes. In Jacob Abernethy and Shivani Agarwal, editors, *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pages 64–66. PMLR, 2020b. URL <https://proceedings.mlr.press/v125/agarwal20a.html>.

Anirudh Goyal, Riashat Islam, Daniel Strouse, Zafarali Ahmed, Matthew Botvinick, Hugo Larochelle, Yoshua Bengio, and Sergey Levine. Infobot: Transfer and exploration via the information bottleneck, 2019.

Anirudh Goyal, Yoshua Bengio, Matthew Botvinick, and Sergey Levine. The variational bandwidth bottleneck: Stochastic evaluation on an information budget, 2020.

Dhruva Tirumala, Alexandre Galashov, Hyeonwoo Noh, Leonard Hasenclever, Razvan Pascanu, Jonathan Schwarz, Guillaume Desjardins, Wojciech Marian Czarnecki, Arun Ahuja, Yee Whye Teh, and Nicolas Heess. Behavior priors for efficient reinforcement learning, 2020b.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *INPROC. 19TH INTERNATIONAL CONFERENCE ON MACHINE LEARNING*, pages 267–274, 2002.

Jean-Bastien Grill, Florent Altche, Yunhao Tang, Thomas Hubert, Michal Valko, Ioannis Antonoglou, and Remi Munos. Monte-carlo tree search as regularized policy optimization, 2020.

H. Francis Song, Abbas Abdolmaleki, Jost Tobias Springenberg, Aidan Clark, Hubert Soyer, Jack W. Rae, Seb Noury, Arun Ahuja, Siqi Liu, Dhruva Tirumala, Nicolas Heess, Dan Belov, Martin Riedmiller, and Matthew M. Botvinick. V-mpo: On-policy maximum a posteriori policy optimization for discrete and continuous control, 2019.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning, 2021. URL <https://openreview.net/forum?id=ToWi1RjuEr8>.

Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. {AWAC}: Accelerating online reinforcement learning with offline datasets, 2021. URL <https://openreview.net/forum?id=0JiM1R3jAtZ>.

Jan Peters, Katharina Mülling, and Yasemin Altün. Relative entropy policy search. *AAAI Conference on Artificial Intelligence*, 2010.

Emanuel Todorov. Linearly-solvable markov decision problems. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2007. URL <https://proceedings.neurips.cc/paper/2006/file/d806ca13ca3449af72a1ea5aedbed26a-Paper.pdf>.

Marc Toussaint and Amos Storkey. Probabilistic inference for solving discrete and continuous state markov decision processes. In *ICML*, volume 2006, pages 945–952, 01 2006. doi: 10.1145/1143844.1143963.

Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference (extended abstract). In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, IJCAI '13*, page 3052–3056. AAAI Press, 2013.

Roy Fox, Ari Pakman, and Naftali Tishby. Taming the noise in reinforcement learning via soft updates. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence, UAI'16*, page 202–211, Arlington, Virginia, USA, 2016. AUAI Press.

Ahmed Touati, Amy Zhang, Joelle Pineau, and Pascal Vincent. Stable policy optimization via off-policy divergence regularization. In Jonas Peters and David Sontag, editors, *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 124 of *Proceedings of Machine Learning Research*, pages 1328–1337. PMLR, 03–06 Aug 2020. URL <https://proceedings.mlr.press/v124/touati20a.html>.

John Schulman, Xi Chen, and Pieter Abbeel. Equivalence between policy gradients and soft q-learning, 2018.



Ronald Williams and Jing Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3:241–, 09 1991. doi: 10.1080/09540099108946587.

Zafarali Ahmed, Nicolas Le Roux, Mohammad Norouzi, and Dale Schuurmans. Understanding the impact of entropy on policy optimization. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 151–160. PMLR, 09–15 Jun 2019. URL <https://proceedings.mlr.press/v97/ahmed19a.html>.

Martin Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degraeve, Tom Van de Wiele, Volodymyr Mnih, Nicolas Heess, and Jost Tobias Springenberg. Learning by playing - solving sparse reward tasks from scratch, 2018.

Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method. *arXiv e-prints*, April 2000.

Aaron Wilson, Alan Fern, Soumya Ray, and Prasad Tadepalli. Multi-task reinforcement learning: A hierarchical bayesian approach. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, page 1015–1022, New York, NY, USA, 2007. Association for Computing Machinery. doi: 10.1145/1273496.1273624. URL <https://doi.org/10.1145/1273496.1273624>.

Brendan O'Donoghue, Ian Osband, and Catalin Ionescu. Making sense of reinforcement learning and probabilistic inference. In *International Conference on Learning Representations*, 2020. URL <https://openreview.net/forum?id=S1xitgHtvS>.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

David Pollard. *Chapter 3*, pages 54–81. Yale University Press, 2000. URL <http://www.stat.yale.edu/~pollard/Books/Asymptopia>.

- David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *NIPS*, 2017.
- Bodhisattva Sen. A gentle introduction to empirical process theory and applications, 2018.
- Aad W Van der Vaart. *Asymptotic statistics*, volume 3. Cambridge university press, 2000.
- William James, Frederick Burkhardt, Fredson Bowers, and Ignas K Skrupskelis. *The principles of psychology*, volume 1. Macmillan London, 1890.
- Nathaniel D Daw, Samuel J Gershman, Ben Seymour, Peter Dayan, and Raymond J Dolan. Model-based influences on humans' choices and striatal prediction errors. *Neuron*, 69(6):1204–1215, 03 2011.
- Daniel Kahneman. *Thinking, fast and slow*. Farrar, Straus and Giroux, 2011.
- Matthew Botvinick, Ari Weinstein, Alec Solway, and Andrew Barto. Reinforcement learning, efficient coding, and the statistics of natural tasks. *Current Opinion in Behavioral Sciences*, 5:71–77, 08 2015. doi: 10.1016/j.cobeha.2015.08.009.
- Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of generalisation in deep reinforcement learning, 2021. URL <https://arxiv.org/abs/2111.09794>.
- Lucy Lai and Samuel Gershman. Policy compression: An information bottleneck in action selection. *Psychology of Learning and Motivation*, 74:195–232, 01 2021. doi: 10.1016/bs.plm.2021.02.004.
- Jorma Rissanen. Modelling by shortest data description. *Automatica*, 14, 01 1978.
- A. N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1:1–7, 1965.
- Ming Li and Paul M.B. Vitnyi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer Publishing Company, Incorporated, 3 edition, 2008.

- Léonard Blier and Yann Ollivier. The description length of deep learning models. *Advances in Neural Information Processing Systems*, 31, 2018.
- Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13, 1993.
- Antti Honkela and Harri Valpola. Variational learning and bits-back coding: an information-theoretic view to bayesian learning. *IEEE transactions on Neural Networks*, 15(4):800–810, 2004.
- Peter Grunwald. A tutorial introduction to the minimum description length principle, 2004.
- Harold Jeffreys. An invariant form for the prior probability in estimation problems. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 186(1007):453–461, 1946. URL <https://royalsocietypublishing.org/doi/abs/10.1098/rspa.1946.0056>.
- Nicholas G Polson and James G Scott. On the half-cauchy prior for a global scale parameter. *Bayesian Analysis*, 7(4):887–902, 2012.
- Andrew Gelman. Prior distributions for variance parameters in hierarchical models (comment on article by browne and draper). *Bayesian analysis*, 1(3):515–534, 2006.
- Peter Grünwald and Teemu Roos. Minimum description length revisited. *International Journal of Mathematics for Industry*, 11(01), 2019.
- Christos Louizos, Karen Ullrich, and Max Welling. Bayesian compression for deep learning. *Advances in neural information processing systems*, 30, 2017.
- Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. *Advances in neural information processing systems*, 28, 2015.
- Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In *International Conference on Machine Learning*, pages 2498–2507. PMLR, 2017.

- Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL <http://jmlr.org/papers/v15/srivastava14a.html>.
- Bradley Efron and Carl Morris. Stein’s estimation rule and its competitors—an empirical bayes approach. *Journal of the American Statistical Association*, 68(341):117–130, 1973.
- Francesco Orabona. A modern introduction to online learning, 2019.
- Richard S. Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1):181–211, 1999. URL <https://www.sciencedirect.com/science/article/pii/S0004370299000521>.
- Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. Deepmind control suite, 2018.
- Matthew Fellows, Anuj Mahajan, Tim G. J. Rudner, and Shimon Whiteson. Virel: A variational inference framework for reinforcement learning, 2020.
- José M Bernardo. Reference analysis. *Handbook of statistics*, 25:17–90, 2005.
- George Casella. An introduction to empirical bayes data analysis. *The American Statistician*, 39(2):83–87, 1985.
- Eric Nalisnick and Padhraic Smyth. Learning approximately objective priors. *arXiv preprint arXiv:1704.01168*, 2017.
- Eric Nalisnick, Jonathan Gordon, and José Miguel Hernández-Lobato. Predictive complexity priors. In *International Conference on Artificial Intelligence and Statistics*, pages 694–702. PMLR, 2021.
- Andrei Atanov, Arsenii Ashukha, Kirill Struminsky, Dmitry Vetrov, and Max Welling. The deep weight prior. *arXiv preprint arXiv:1810.06943*, 2018.

Shantanu Thakoor, Mark Rowland, Diana Borsa, Will Dabney, Rémi Munos, and André Barreto. Generalised policy improvement with geometric policy composition, 2022. URL <https://arxiv.org/abs/2206.08736>.

Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Evolving curricula with regret-based environment design, 2022. URL <https://arxiv.org/abs/2203.01302>.

Samuel Kessler, Jack Parker-Holder, Philip Ball, Stefan Zohren, and Stephen J. Roberts. Same state, different task: Continual reinforcement learning without interference, 2021. URL <https://arxiv.org/abs/2106.02940>.

Jesse Zhang, Karl Pertsch, Jiefan Yang, and Joseph J Lim. Minimum description length skills for accelerated reinforcement learning. In *Self-Supervision for Reinforcement Learning Workshop - ICLR 2021*, 2021. URL <https://openreview.net/forum?id=r4XxtrIo1m9>.

Sebastian Thrun and Anton Schwartz. Finding structure in reinforcement learning. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7. MIT Press, 1994. URL <https://proceedings.neurips.cc/paper/1994/file/7ce3284b743aefde80ffd9aec500e085-Paper.pdf>.

Emma Brunskill and Lihong Li. Sample complexity of multi-task reinforcement learning, 2013. URL <https://arxiv.org/abs/1309.6821>.

Edward I George, Feng Liang, and Xinyi Xu. Improved minimax predictive densities under kullback-leibler loss. *The Annals of Statistics*, pages 78–91, 2006.

Lawrence D Brown. Admissible estimators, recurrent diffusions, and insoluble boundary value problems. *The Annals of Mathematical Statistics*, 42(3):855–903, 1971.

Dominique Fourdrinier, William E Strawderman, and Martin T Wells. On the construction of bayes minimax estimators. *Annals of Statistics*, pages 660–671, 1998.

- Gabriel Barth-Maron, Matthew W. Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva TB, Alistair Muldal, Nicolas Heess, and Timothy P. Lillicrap. Distributed distributional deterministic policy gradients. *CoRR*, abs/1804.08617, 2018. URL <http://arxiv.org/abs/1804.08617>.
- James Melbourne. Strongly convex divergences. *Entropy (Basel, Switzerland)*, 22(11):1327, 11 2020.
- Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, March 2004.
- Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997a.
- Adele Diamond. Executive functions. *Annual review of psychology*, 64:135–168, 2013.
- Matthew M Botvinick and Jonathan D Cohen. The computational and neural basis of cognitive control: charted territory and new frontiers. *Cognitive science*, 38(6):1249–1285, 2014.
- Omar D Perez and Anthony Dickinson. A theory of actions and habits: The interaction of rate correlation and contiguity systems in free-operant behavior. *Psychological Review*, 127(6):945, 2020.
- Jonathan St BT Evans. Dual-processing accounts of reasoning, judgment, and social cognition. *Annu. Rev. Psychol.*, 59:255–278, 2008.
- Falk Lieder and Thomas L Griffiths. Strategy selection as rational metareasoning. *Psychological review*, 124(6):762, 2017.
- Nicolas P Rougier, David C Noelle, Todd S Braver, Jonathan D Cohen, and Randall C O'Reilly. Prefrontal cortex and flexible cognitive control: Rules without symbols. *Proceedings of the National Academy of Sciences*, 102(20):7338–7343, 2005.
- Amitai Shenhav, Matthew M Botvinick, and Jonathan D Cohen. The expected value of control: an integrative theory of anterior cingulate cortex function. *Neuron*, 79(2):217–240, 2013.

- Mehdi Keramati, Amir Dezfouli, and Payam Piray. Speed/accuracy trade-off between the habitual and the goal-directed processes. *PLoS computational biology*, 7(5):e1002055, 2011.
- Y-Lan Boureau, Peter Sokol-Hessner, and Nathaniel D Daw. Deciding how to decide: Self-control and meta-decision making. *Trends in cognitive sciences*, 19(11):700–710, 2015.
- Katell Mevel, Grégoire Borst, Nicolas Poiriel, Grégory Simon, François Orliac, Olivier Etard, Olivier Houdé, and Wim De Neys. Developmental frontal brain activation differences in overcoming heuristic bias. *cortex*, 117:111–121, 2019.
- Wim De Neys and Vinod Goel. Heuristics and biases in the brain: Dual neural pathways for decision making. In *Neuroscience of decision making*, pages 137–154. Psychology Press, 2011.
- Earl K Miller and Jonathan D Cohen. An integrative theory of prefrontal cortex function. *Annual review of neuroscience*, 24(1):167–202, 2001.
- Hyeon-Ae Jeon and Angela D Friederici. Degree of automaticity and the prefrontal cortex. *Trends in cognitive sciences*, 19(5):244–250, 2015.
- Matthew D Lieberman. Social cognitive neuroscience: a review of core processes. *Annu. Rev. Psychol.*, 58:259–289, 2007.
- Randall C O'Reilly, Ananta Nair, Jacob L Russin, and Seth A Herd. How sequential interactive processing within frontostriatal loops supports a continuum of habitual to controlled processing. *Frontiers in Psychology*, 11:380, 2020.
- Kyle S Smith and Ann M Graybiel. Habit formation. *Dialogues in clinical neuroscience*, 2022.
- J. R. Stroop. Studies of interference in serial verbal reactions. *Journal of Experimental Psychology*, 18(6):643–662, 1935. doi: 10.1037/h0054651.

- Seth A Herd, Marie T Banich, and Randall C O'reilly. Neural mechanisms of cognitive control: An integrative model of stroop task performance and fmri data. *Journal of cognitive neuroscience*, 18(1):22–32, 2006.
- Wouter Kool and Matthew Botvinick. Mental labour. *Nature Human Behaviour*, 2(12): 899–908, 2018. doi: 10.1038/s41562-018-0401-9.
- Nathalie Schouppe, K Richard Ridderinkhof, Tom Verguts, and Wim Notebaert. Context-specific control and context selection in conflict tasks. *Acta psychologica*, 146: 63–66, 2014.
- Alexandre Zenon, Oleg Solopchuk, and Giovanni Pezzulo. An information-theoretic perspective on the costs of cognition. *Neuropsychologia*, 123:5–18, 2019.
- Payam Piray and Nathaniel D Daw. Linear reinforcement learning in planning, grid fields, and cognitive control. *Nature communications*, 12(1):1–20, 2021.
- Ulrik R Beierholm, Cedric Anen, Steven Quartz, and Peter Bossaerts. Separate encoding of model-based and model-free valuations in the human brain. *Neuroimage*, 58(3): 955–962, 2011.
- Jan Gläscher, Nathaniel Daw, Peter Dayan, and John P O'Doherty. States versus rewards: dissociable neural prediction error signals underlying model-based and model-free reinforcement learning. *Neuron*, 66(4):585–595, 2010.
- Bruno Averbeck and John P O'Doherty. Reinforcement-learning in fronto-striatal circuits. *Neuropsychopharmacology*, 47(1):147–162, 2022.
- Nicole Drummond and Yael Niv. Model-based decision making and model-free learning. *Current Biology*, 30(15):R860–R865, 2020.
- Anthony Dickinson. Actions and habits: the development of behavioural autonomy. *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, 308(1135):67–78, 1985.



- Thomas Akam, Rui Costa, and Peter Dayan. Simple plans or sophisticated habits? state, transition and learning interactions in the two-step task. *PLOS Computational Biology*, 11(12):1–25, 12 2015. URL <https://doi.org/10.1371/journal.pcbi.1004648>.
- Carolina Feher da Silva and Todd A Hare. Humans primarily use model-based inference in the two-stage task. *Nature Human Behaviour*, 4(10):1053–1066, 2020.
- Peter Smittenaar, Thomas HB FitzGerald, Vincenzo Romei, Nicholas D Wright, and Raymond J Dolan. Disruption of dorsolateral prefrontal cortex decreases model-based in favor of model-free control in humans. *Neuron*, 80(4):914–919, 2013.
- A Ross Otto, Samuel J Gershman, Arthur B Markman, and Nathaniel D Daw. The curse of planning: dissecting multiple reinforcement-learning systems by taxing the central executive. *Psychological science*, 24(5):751–761, 2013.
- Thomas Akam, Ines Rodrigues-Vaz, Ivo Marcelo, Xiangyu Zhang, Michael Pereira, Rodrigo Freire Oliveira, Peter Dayan, and Rui M Costa. The anterior cingulate cortex predicts future states to mediate model-based action selection. *Neuron*, 109(1):149–163, 2021.
- Kevin J Miller, Matthew M Botvinick, and Carlos D Brody. Dorsal hippocampus contributes to model-based planning. *Nature neuroscience*, 20(9):1269–1276, 2017.
- Anne GE Collins and Jeffrey Cockburn. Beyond dichotomies in reinforcement learning. *Nature Reviews Neuroscience*, 21(10):576–586, 2020.
- Claire M Gillan, A Ross Otto, Elizabeth A Phelps, and Nathaniel D Daw. Model-based learning protects against forming habits. *Cognitive, Affective, & Behavioral Neuroscience*, 15:523–536, 2015.
- Anthony Dickinson. Omission learning after instrumental pretraining. *The Quarterly Journal of Experimental Psychology: Section B*, 51(3):271–286, 1998.
- Henry H Yin, Barbara J Knowlton, and Bernard W Balleine. Inactivation of dorsolateral striatum enhances sensitivity to changes in the action–outcome contingency in instrumental conditioning. *Behavioural brain research*, 166(2):189–196, 2006.

Giovanni Pezzulo, Francesco Rigoli, and Karl J Friston. Hierarchical active inference: a theory of motivated control. *Trends in cognitive sciences*, 22(4):294–306, 2018.

Falk Lieder and Thomas L Griffiths. Resource-rational analysis: Understanding human cognition as the optimal use of limited computational resources. *Behavioral and Brain Sciences*, 43, 2020.

Ben R Newell and David R Shanks. Take the best or look at the rest? factors influencing “one-reason” decision making. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 29(1):53, 2003.

Jonathan Cohen, Kevin Dunbar, and James McClelland. Cohen jd, mclelland jl, dunbar k. on the control of automatic processes: a parallel distributed processing account of the stroop effect. *psychol rev* 97: 332-361. *Psychological review*, 97:332–61, 07 1990. doi: 10.1037/0033-295X.97.3.332.

Reidar Riveland and Alexandre Pouget. A neural model of task compositionality with natural language instructions. *bioRxiv*, 2022.

Randall C O’Reilly, Seth A Herd, and Wolfgang M Pauli. Computational models of cognitive control. *Current opinion in neurobiology*, 20(2):257–261, 2010.

Sebastian Musslick and Jonathan D Cohen. Rationalizing constraints on the capacity for cognitive control. *Trends in Cognitive Sciences*, 25(9):757–775, 2021.

Falk Lieder, Amitai Shenhav, Sebastian Musslick, and Thomas L Griffiths. Rational metareasoning and the plasticity of cognitive control. *PLoS computational biology*, 14(4):e1006043, 2018.

Seth A Herd, Tom E Hazy, Christopher H Chatham, Angela M Brant, Naomi P Friedman, et al. A neural network model of individual differences in task switching abilities. *Neuropsychologia*, 62:375–389, 2014.

Jeremy R Reynolds, Todd S Braver, Joshua W Brown, and Stefan Van der Stigchel. Computational and neural mechanisms of task switching. *Neurocomputing*, 69(10-12):1332–1336, 2006.

- Sam J Gilbert and Tim Shallice. Task switching: A pdp model. *Cognitive psychology*, 44 (3):297–337, 2002.
- Jane X. Wang, Zeb Kurth-Nelson, Dharshan Kumaran, Dhruva Tirumala, Hubert Soyer, Joel Z. Leibo, Demis Hassabis, and Matthew Botvinick. Prefrontal cortex as a meta-reinforcement learning system. *Nature Neuroscience*, 21(6):860–868, 2018. URL <https://doi.org/10.1038/s41593-018-0147-8>.
- F Gregory Ashby, John M Ennis, and Brian J Spiering. A neurobiological theory of automaticity in perceptual categorization. *Psychological review*, 114(3):632, 2007.
- Rafal Bogacz. Dopamine role in learning and action inference. *Elife*, 9:e53262, 2020.
- Raymond Y Cho, Leigh E Nystrom, Eric T Brown, Andrew D Jones, Todd S Braver, Philip J Holmes, and Jonathan D Cohen. Mechanisms underlying dependencies of performance on stimulus history in a two-alternative forced-choice task. *Cognitive, Affective, & Behavioral Neuroscience*, 2:283–299, 2002.
- Rei Akaishi, Kazumasa Umeda, Asako Nagase, and Katsuyuki Sakai. Autonomous mechanism of internal choice estimate underlies decision inertia. *Neuron*, 81(1):195–206, 2014.
- Matthew Balcarras, Salva Ardid, Daniel Kaping, Stefan Everling, and Thilo Womelsdorf. Attentional selection can be predicted by reinforcement learning of task-relevant stimulus features weighted by value-independent stickiness. *Journal of cognitive neuroscience*, 28(2):333–349, 2016.
- Sang Wan Lee, Shinsuke Shimojo, and John P O’Doherty. Neural computations underlying arbitration between model-based and model-free learning. *Neuron*, 81(3):687–699, 2014.
- Daniel Kahneman and Shane Frederick. Representativeness revisited: Attribute substitution in intuitive judgment. *Heuristics and biases: The psychology of intuitive judgment*, 49:81, 2002.

- Nadav Amir, Reut Suliman-Lavie, Maayan Tal, Sagiv Shifman, Naftali Tishby, and Israel Nelken. Value-complexity tradeoff explains mouse navigational learning. *PLoS Computational Biology*, 16(12):e1008497, 2020.
- Gaia Tavoni, Takahiro Doi, Chris Pizzica, Vijay Balasubramanian, and Joshua I Gold. Human inference reflects a normative balance of complexity and accuracy. *Nature human behaviour*, 6(8):1153–1168, 2022.
- Peter D Grünwald. *The minimum description length principle*. MIT press, 2007.
- Rachel A Lerch and Chris R Sims. Policy generalization in capacity-limited reinforcement learning, 2018.
- Nick Chater and Paul Vitányi. Simplicity: a unifying principle in cognitive science? *Trends in cognitive sciences*, 7(1):19–22, 2003.
- David Badre and Derek Evan Nee. Frontal cortex and the hierarchical control of behavior. *Trends in cognitive sciences*, 22(2):170–188, 2018.
- Alexandre Galashov, Siddhant M Jayakumar, Leonard Hasenclever, Dhruva Tirumala, Jonathan Schwarz, Guillaume Desjardins, Wojciech M Czarnecki, Yee Whye Teh, Razvan Pascanu, and Nicolas Heess. Information asymmetry in kl-regularized rl. *arXiv preprint arXiv:1905.01240*, 2019b.
- Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. *Advances in neural information processing systems*, 30, 2017b.
- Andrei A. Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation, 2015. URL <https://arxiv.org/abs/1511.06295>.
- Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018b.

- Zhao Song, Ronald E Parr, Xuejun Liao, and Lawrence Carin. Linear feature encoding for reinforcement learning. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL <https://proceedings.neurips.cc/paper/2016/file/8232e119d8f59aa83050a741631803a6-Paper.pdf>.
- Jane X. Wang, Zeb Kurth-Nelson, Hubert Soyer, Joel Z. Leibo, Dhruva Tirumala, Rémi Munos, Charles Blundell, Dhharshan Kumaran, and Matt M. Botvinick. Learning to reinforcement learn. In *CogSci*, 2017. URL <https://mindmodeling.org/cogsci2017/papers/0252/index.html>.
- AB Atkinson. On the measurement of inequality. *Journal of Economic Inequality*, 6: 277–283, 09 2008. doi: 10.1007/s10888-007-9075-7.
- Kimberly L Stachenfeld, Matthew M Botvinick, and Samuel J Gershman. The hippocampus as a predictive map. *Nature Neuroscience*, 20(11):1643–1653, 2017. URL <https://doi.org/10.1038/nn.4650>.
- I. Momennejad, E. M. Russek, J. H. Cheong, M. M. Botvinick, N. D. Daw, and S. J. Gershman. The successor representation in human reinforcement learning. *Nature Human Behaviour*, 1(9):680–692, 2017.
- Timothy E.J. Behrens, Timothy H. Muller, James C.R. Whittington, Shirley Mark, Alon B. Baram, Kimberly L. Stachenfeld, and Zeb Kurth-Nelson. What is a cognitive map? organizing knowledge for flexible behavior. *Neuron*, 100(2):490–509, 2018. doi: <https://doi.org/10.1016/j.neuron.2018.10.002>. URL <https://www.sciencedirect.com/science/article/pii/S0896627318308560>.
- Samuel J. Gershman. Origin of perseveration in the trade-off between reward and complexity. *bioRxiv*, 2020. doi: 10.1101/2020.01.16.903476. URL <https://www.biorxiv.org/content/early/2020/01/16/2020.01.16.903476>.
- Tejas D. Kulkarni, Ardavan Saeedi, Simanta Gautam, and Samuel J. Gershman. Deep successor reinforcement learning, 2016.

Marlos C. Machado, Marc G. Bellemare, and Michael Bowling. Count-based exploration with the successor representation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):5125–5133, Apr. 2020. doi: 10.1609/aaai.v34i04.5955. URL <https://ojs.aaai.org/index.php/AAAI/article/view/5955>.

Tamas Madarasz and Tim Behrens. Better transfer learning with inferred successor maps. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL <https://proceedings.neurips.cc/paper/2019/file/274a10ffa06e434f2a94df765cac6bf4-Paper.pdf>.

Michael L. Littman, Ufuk Topcu, Jie Fu, Charles Isbell, Min Wen, and James MacGlashan. Environment-Independent Task Specifications via GLTL. *arXiv e-prints*, 2017.

Maor Gaon and Ronen Brafman. Reinforcement learning with non-markovian rewards. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3980–3987, Apr. 2020. doi: 10.1609/aaai.v34i04.5814. URL <https://ojs.aaai.org/index.php/AAAI/article/view/5814>.

Thomas J. Ringstrom and Paul R. Schrater. Constraint satisfaction propagation: Non-stationary policy synthesis for temporal logic planning. *CoRR*, abs/1901.10405, 2019. URL <http://arxiv.org/abs/1901.10405>.

Leonid Peshkin, Nicolas Meuleau, and Leslie Pack Kaelbling. Learning policies with external memory, 2001.

Fahiem Bacchus, Craig Boutilier, and Adam Grove. Rewarding behaviors. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2, AAAI'96*, page 1160–1167. AAAI Press, 1996.

Léonard Blier, Corentin Tallec, and Yann Ollivier. Learning successor states and goal-dependent values: A mathematical viewpoint, 2021.

Ahmed Touati and Yann Ollivier. Learning one representation to optimize all rewards, 2021a.

G. Rummery and Mahesan Niranjan. On-line q-learning using connectionist systems. *Technical Report CUED/F-INFENG/TR 166*, 11 1994.

Alexander L. Strehl and Michael L. Littman. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008. ISSN 0022-0000. doi: <https://doi.org/10.1016/j.jcss.2007.08.009>. URL <https://www.sciencedirect.com/science/article/pii/S0022000008000767>. Learning Theory 2005.

Hao Liu and Pieter Abbeel. Aps: Active pretraining with successor features. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 6736–6747. PMLR, 18–24 Jul 2021. URL <https://proceedings.mlr.press/v139/liu21b.html>.

Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control, 2016.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function, 2018.

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. In *International Conference on Learning Representations*, 2020. URL <https://openreview.net/forum?id=HJgLZR4KvH>.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Yinyu Ye. The simplex and policy-iteration methods are strongly polynomial for the markov decision problem with a fixed discount rate. *Mathematics of Operations*

- Research*, 36(4):593–603, 2011. URL <https://doi.org/10.1287/moor.1110.0516>.
- Steven Lima and Larry Dill. Behavioral decisions made under the risk of predation: A review and prospectus. *Canadian Journal of Zoology-revue Canadienne De Zoologie - CANJ ZOOL*, 68:619–640, 04 1990. doi: 10.1139/z90-092.
- Philip Shamash, Sarah F. Olesen, Panagiota Iordanidou, Dario Campagner, Banerjee Nabhojit, and Tiago Branco. Mice learn multi-step routes by memorizing subgoal locations. *bioRxiv*, 2021a. doi: 10.1101/2020.08.19.256867. URL <https://www.biorxiv.org/content/early/2021/05/08/2020.08.19.256867>.
- Simon S. Du, Akshay Krishnamurthy, Nan Jiang, Alekh Agarwal, Miroslav Dudík, and John Langford. Provably efficient rl with rich observations via latent state decoding, 2019.
- Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games, 2015.
- Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, 2019.
- Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, 2018.
- Tom Zahavy, Avinatan Hassidim, Haim Kaplan, and Yishay Mansour. Planning in hierarchical reinforcement learning: Guarantees for using local policies. *CoRR*, abs/1902.10140, 2019. URL <http://arxiv.org/abs/1902.10140>.
- David Silver and Kamil Ciosek. Compositional planning using optimal option models, 2012.



- Daniel Kahneman and Amos Tversky. Prospect theory: An analysis of decision under risk. *Econometrica*, 47(2):263–292, 1979.
- Matthew Rabin. Risk aversion and expected-utility theory: a calibration theorem. *Econometrica*, 68(5):1281–1292, 2000. doi: 10.2307/2999450.
- Alex Pine, Ben Seymour, Jonathan P Roiser, Peter Bossaerts, Karl J Friston, H Valerie Curran, and Raymond J Dolan. Encoding of marginal utility across time in the human brain. *Journal of Neuroscience*, 29(30):9575–9581, 2009.
- Hermann Heinrich Gossen and Rudolph C Blitz. *The laws of human relations and the rules of human action derived therefrom*. Mit Press Cambridge, MA, 1983.
- Kenneth J Arrow. The theory of risk-bearing: small and great risks. *Journal of risk and uncertainty*, 12:103–111, 1996.
- John W Pratt. Risk aversion in the small and in the large. In *Uncertainty in economics*, pages 59–79. Elsevier, 1978.
- Nathan Wispinski, Andrew Butcher, Kory Wallace Mathewson, Craig S Chapman, Matthew Botvinick, and Patrick M. Pilarski. Adaptive patch foraging in deep reinforcement learning agents. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL <https://openreview.net/forum?id=a0T3n0P9sB>.
- Sergey Shuvaev, Sarah Starosta, Duda Kvitsiani, Adam Kepecs, and Alexei Koulakov. R-learning in actor-critic model offers a biologically relevant mechanism for sequential decision-making. *Advances in neural information processing systems*, 33:18872–18882, 2020.
- Ahmed Touati and Yann Ollivier. Learning one representation to optimize all rewards. *Advances in Neural Information Processing Systems*, 34:13–23, 2021b.
- Andre Barreto, Diana Borsa, John Quan, Tom Schaul, David Silver, Matteo Hessel, Daniel Mankowitz, Augustin Zidek, and Remi Munos. Transfer in deep reinforcement learning using successor features and generalised policy improvement. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference*

- on *Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 501–510. PMLR, 10–15 Jul 2018. URL <http://proceedings.mlr.press/v80/barreto18a.html>.
- Arthur Juliani, Samuel Barnett, Brandon Davis, Margaret Sereno, and Ida Momennejad. Neuro-nav: a library for neurally-plausible reinforcement learning. *arXiv preprint arXiv:2206.03312*, 2022.
- Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997b.
- Ahmed Touati, Jérémy Rapin, and Yann Ollivier. Does zero-shot reinforcement learning exist? In *The Eleventh International Conference on Learning Representations*, 2023. URL [https://openreview.net/forum?id=MYEap\\_0cQI](https://openreview.net/forum?id=MYEap_0cQI).
- Benjamin Y Hayden, John M Pearson, and Michael L Platt. Neuronal basis of sequential foraging decisions in a patchy environment. *Nature neuroscience*, 14(7):933–939, 2011.
- Tehrim Yoon, Robert B Geary, Alaa A Ahmed, and Reza Shadmehr. Control of movement vigor and decision making during foraging. *Proceedings of the National Academy of Sciences*, 115(44):E10476–E10485, 2018.
- Anthony S Gabay and Matthew AJ Apps. Foraging optimally in social neuroscience: computations and methodological considerations. *Social Cognitive and Affective Neuroscience*, 16(8):782–794, 2021.
- Eric L. Charnov. Optimal foraging, the marginal value theorem. *Theoretical Population Biology*, 9(2):129–136, 1976. ISSN 0040-5809.
- Peter Nonacs. State dependent behavior and the marginal value theorem. *Behavioral Ecology*, 12(1):71–83, 2001.
- Ruosong Wang, Hanrui Zhang, Devendra Singh Chaplot, Denis Garagić, and Ruslan Salakhutdinov. Planning with submodular objective functions. *arXiv preprint arXiv:2010.11863*, 2020.

- Manish Prajapat, Mojmír Mutný, Melanie N Zeilinger, and Andreas Krause. Submodular reinforcement learning. *arXiv preprint arXiv:2307.13372*, 2023.
- Eitan Altman. *Constrained Markov decision processes*. Routledge, 2021.
- Ted Moskovitz, Brendan O’Donoghue, Vivek Veeriah, Sebastian Flennerhag, Satinder Singh, and Tom Zahavy. Reload: Reinforcement learning with optimistic ascent-descent for last-iterate convergence in constrained mdps. In *International Conference on Machine Learning*, pages 25303–25336. PMLR, 2023b.
- Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.
- Martin Riedmiller. Neural fitted q iteration—first experiences with a data efficient neural reinforcement learning method. In *Machine Learning: ECML 2005: 16th European Conference on Machine Learning, Porto, Portugal, October 3-7, 2005. Proceedings 16*, pages 317–328. Springer, 2005.
- Peter Dayan and Geoffrey E Hinton. Feudal reinforcement learning. *Advances in neural information processing systems*, 5, 1992.
- Ted Moskovitz, Jack Parker-Holder, Aldo Pacchiano, Michael Arbel, and Michael Jordan. Tactical optimism and pessimism for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 34:12849–12863, 2021b.
- Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006.
- Amy McGovern and Andrew G Barto. Automatic discovery of subgoals in reinforcement learning using diverse density. *University of Massachusetts*, 2001.
- Vivek Veeriah, Tom Zahavy, Matteo Hessel, Zhongwen Xu, Junhyuk Oh, Iurii Kemaev, Hado P van Hasselt, David Silver, and Satinder Singh. Discovery of options via meta-learned subgoals. *Advances in Neural Information Processing Systems*, 34:29861–29873, 2021.

- Danijar Hafner, Kuang-Huei Lee, Ian Fischer, and Pieter Abbeel. Deep hierarchical planning from pixels. *Advances in Neural Information Processing Systems*, 35:26091–26104, 2022.
- Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *International conference on machine learning*, pages 3540–3549. PMLR, 2017.
- Annie Xie, James Harrison, and Chelsea Finn. Deep reinforcement learning amidst continual structured non-stationarity. In *International Conference on Machine Learning*, pages 11393–11403. PMLR, 2021.
- Todd S Braver and Susan R Bongiolatti. The role of frontopolar cortex in subgoal processing during working memory. *Neuroimage*, 15(3):523–536, 2002.
- Carlos G Correa, Mark K Ho, Fred Callaway, and Thomas L Griffiths. Resource-rational task decomposition to minimize planning costs. *arXiv preprint arXiv:2007.13862*, 2020.
- Philip Shamash, Sarah F Olesen, Panagiotia Iordanidou, Dario Campagner, Nabhojit Banerjee, and Tiago Branco. Mice learn multi-step routes by memorizing subgoal locations. *Nature neuroscience*, 24(9):1270–1279, 2021b.
- Carlos G Correa, Mark K Ho, Frederick Callaway, Nathaniel D Daw, and Thomas L Griffiths. Humans decompose tasks by trading off utility and computational cost. *PLoS computational biology*, 19(6):e1011087, 2023.
- Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.
- Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning acting and planning in language models. *arXiv preprint arXiv:2310.04406*, 2023.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

Norman Di Palo, Arunkumar Byravan, Leonard Hasenclever, Markus Wulfmeier, Nicolas Heess, and Martin Riedmiller. Towards a unified agent with foundation models. *arXiv preprint arXiv:2307.09668*, 2023.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.