Efficient Online Learning & Adaptive Decision-Making in Artificial & Biological Agents

Veronica Chelu, School of Computer Science
McGill University, Montreal
December, 2024

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of

Doctor of Philosophy

©Veronica Chelu, December 2024

Abstract

Flexible decision-making in biological and artificial systems hinges on using prior knowledge to adapt. With inherent uncertainty and limited computational resources, how can agents effectively leverage experience to learn and act in complex or dynamic environments?

In this thesis, we investigate whether cognitive mechanisms inspired by biological systems– such as internal modelling, inductive biases like selectivity and momentum, and biological constraints arising from excitatory–inhibitory interactions—can enhance the efficiency and adaptability of reinforcement learning (RL) agents in complex or evolving environments. Through four interconnected themes, we explore how these elements enable more flexible and efficient decision-making, drawing parallels between artificial and biological systems.

First, we investigate how agents use internal models to anticipate outcomes and refine decisions, highlighting the importance of planning in temporal credit assignment. By studying both prospective planning and retrospective reasoning we show how deviations between expected and actual outcomes drive behavioural changes.

Building on these studies, we propose retrospective selective models that prioritize learning updates based on salience or attentional biases, mirroring processes observed in biological systems. By emphasizing certain experiences, agents can focus computational resources and improve learning efficiency in environments with noise, sparse rewards, or delayed feedback.

Next, we explore momentum as an extrapolation mechanism in policy-based learning, leveraging successive policy changes to inform future decision-making. We show that momentum can accelerate learning in long-horizon settings and environments with sparse state connectivity. Incorporating momentum into the Policy Mirror Descent framework allows agents to extrapolate from past experiences and navigate optimization landscapes more efficiently.

Finally, we present a multi-level computational framework to examine how excitatory – inhibitory (E/I) interactions may influence adaptive learning and decision-making. Starting with a mean-field model, we illustrate how selective excitation and inhibition can stabilize dynamics or enhance competition among alternatives. Embedding this mechanism into RL agents reveals that inhibitory feedback modulates the behavioural flexibility in a dynamic bandit task. We further implement E/I-constrained recurrent neural networks (RNNs), showing that an architecture respecting Dale's principle (Dale's Artificial Neural Network, DANN) preserves robust performance under sign constraints. These findings highlight how structured E/I dynamics can shape learning and flexibility in changing tasks.

By identifying and leveraging computational mechanisms, such as internal modelling, selective experience prioritization, momentum-based learning, and biologically-plausible mechanisms for enhancing behavioural flexibility, this thesis contributes to a deeper understanding of how agents can learn and adapt effectively in complex or dynamic environments.

Abrégé

La prise de décision flexible dans les systèmes biologiques et artificiels repose sur l'utilisation des connaissances préalables pour s'adapter. Face à l'incertitude inhérente et aux ressources computationnelles limitées, comment les agents peuvent-ils exploiter efficacement leur expérience pour apprendre et agir dans des environnements complexes ou dynamiques ?

Dans cette thèse, nous examinons si des mécanismes cognitifs inspirés des systèmes biologiques—tels que la modélisation interne, les biais inductifs comme la sélectivité et le moment, et les contraintes biologiques issues des interactions excitation–inhibition—peuvent améliorer l'efficacité et l'adaptabilité des agents d'apprentissage par renforcement (RL) dans des environnements complexes ou évolutifs. À travers quatre thématiques interconnectées, nous explorons comment ces éléments permettent une prise de décision plus souple et efficace, en établissant des parallèles entre les systèmes artificiels et biologiques.

Premièrement, nous étudions comment les agents utilisent des modèles internes pour anticiper les résultats et affiner leurs décisions, en soulignant l'importance de la planification dans l'attribution de crédit temporel. En analysant à la fois la planification prospective et le raisonnement rétrospectif, nous montrons comment les écarts entre les résultats attendus et réels entraînent des changements comportementaux.

Sur la base de ces études, nous proposons des modèles rétrospectifs sélectifs qui priorisent les mises à jour d'apprentissage en fonction de la saillance ou des biais attentionnels, reflétant des processus observés dans les systèmes biologiques. En mettant l'accent sur certaines expériences, les agents peuvent concentrer leurs ressources computationnelles et

améliorer l'efficacité de l'apprentissage dans des environnements bruités, à récompenses rares ou à rétroactions différées.

Ensuite, nous explorons le moment comme mécanisme d'extrapolation dans l'apprentissage fondé sur les politiques, en tirant parti des changements successifs de politique pour orienter les décisions futures. Nous montrons que le moment peut accélérer l'apprentissage dans des contextes à long horizon et dans des environnements à connectivité d'états clairsemée. L'intégration du moment dans le cadre de la descente miroir politique permet aux agents d'extrapoler à partir des expériences passées et de naviguer plus efficacement dans les paysages d'optimisation.

Enfin, nous présentons un cadre computationnel multi-niveaux pour examiner comment les interactions excitation–inhibition (E/I) peuvent influencer l'apprentissage adaptatif et la prise de décision. En partant d'un modèle en champ moyen, nous illustrons comment l'excitation et l'inhibition sélectives peuvent stabiliser la dynamique ou renforcer la compétition entre alternatives. L'intégration de ce mécanisme dans des agents RL révèle que le feedback inhibiteur module la flexibilité comportementale dans une tâche de bandit dynamique. Nous implémentons aussi des réseaux neuronaux récurrents (RNN) contraints E/I, montrant qu'une architecture respectant le principe de Dale (Réseau Neuronal Artificiel de Dale, DANN) conserve des performances robustes malgré les contraintes de signe. Ces résultats soulignent comment des dynamiques E/I structurées peuvent façonner l'apprentissage et la flexibilité dans des tâches en constante évolution.

En identifiant et en exploitant des mécanismes computationnels tels que la modélisation interne, la prioritisation sélective de l'expérience, l'apprentissage fondé sur le moment, et des mécanismes biologiquement plausibles améliorant la flexibilité comportementale, cette thèse contribue à une meilleure compréhension de la manière dont les agents peuvent apprendre et s'adapter efficacement dans des environnements complexes ou dynamiques.

Acknowledgements

None of the work in this thesis would have been possible without the incredible support, kindness, and encouragement of many people. First and foremost, I owe everything to my supervisor, Doina Precup, who gave me the extraordinary opportunity to pursue a PhD. Her guidance and unwavering support have shaped my academic career.

I am grateful to Hado van Hasselt for mentorship and encouragement during early stages of my research journey, and for his friendship, which made everything more fun. I would like to thank Blake Richards and Glen Berseth for their mentorship and encouragement, which have been instrumental in my growth as a researcher. I am thankful to Diana Borsa for her kindness, support, and insightful conversations. Much gratitude to my advisors and collaborators: Joelle Pineau, Adam White, Tom Zahavy, Sebastian Flennerhag, Arthur Guez, Shangtong Zhang, Anthony Chen, Nathan de Lara, Arthur Juliani, Laura Graesser, Adam Safron. I thank Emmanuel Bengio for his friendship, engaging and thought-provoking conversations, and camaraderie throughout the years. I am also grateful to Jincheng Mei, Colin Bredenberg, Jonathan Lebensold and Nishanth Anand for support, feedbackm, and insightful conversations.

Boundless gratitude goes to my mum, best friend and biggest supporter, for love and kindness I can never repay. Grateful also to my family for support throughout this journey.

I am deeply thankful to my meditation teachers for wisdom, stillness, clarity and wonder. Lastly, gratitude to the fungi—from which I have learned much, especially to see.

Contributions to Original Knowledge

This thesis advances our understanding of adaptive learning and decision-making through the following contributions:

1. **Forethought and Hindsight in Credit Assignment**
    - Investigated the properties of forward and backward planning, demonstrating their complementarity and highlighting their respective strengths.
    - Justified backward models for efficient credit assignment, exploring design choices in model representation, estimation, and parametrization.
    - Illustrated the impact of state selection in planning, emphasizing the role of backward planning in counterfactual reasoning.
    - Analyzed the interaction between planning and model-free updates, showing how these interactions influence learning efficiency.
    - Conducted an empirical study on illustrative prediction and control tasks to validate theoretical insights and build intuition around planning dynamics [1].
2. **Salience Maps for Selective Credit Assignment**
    - Proposed a unified framework for selective credit assignment in reinforcement learning, leveraging salience maps to mediate backward credit distribution.

[1] Code available at https://github.com/veronicachelu/forethought_and_hindsight

- Extended selective credit assignment to planning algorithms, introducing models of selective traces for counterfactual credit assignment and reducing computational complexity relative to prior methods.
- Addressed issues with naive salience map combinations, identifying conditions for stable credit assignment.
- Characterized conditions under which salience maps yield computational benefits [2].

3. **Acceleration in Policy Optimization**
   - Proposed incorporating acceleration mechanisms into policy optimization, to enable faster convergence with fewer iterations.
   - Introduced momentum-based Policy Mirror Descent (PMD), an algorithm that leverages recent policy updates to accelerate learning.
   - Analyzed the learning dynamics of the proposed algorithm, characterizing conditions for enhanced performance.
   - Explored the interplay between functional acceleration and inexact critics, examining stability and effectiveness.[3]
4. **Excitatory-Inhibitory Dynamics in Adaptive Learning and Decision-Making**
   - Developed a multi-level modelling framework to examine how selective excitation and inhibition influence adaptive behaviour.
   - Showed that decision-selective excitation and inhibition can stabilize or enhance competition among alternative decisions.

[2]Code available at https://github.com/veronicachelu/selective_credit_assignment
[3]Code available at https://github.com/veronicachelu/functional-acceleration-for-pmd

- Integrated an E/I-mechanism into the preferences of an RL agent and showed that inhibitory feedback can regulate the speed–accuracy trade-off in dynamic environments.
- Applied biologically inspired E/I constraints to RNN-based RL agents, showing that a Dale-conformant architecture (DANN) maintains robust learning performance in dynamic environments. [4].

[4]Code available at https://github.com/veronicachelu/EI_dynamics

Statement of Authorship

- **Chapter 1** and **Chapter 2** provide the context and technical background for this thesis and are written specifically for this purpose. **Chapter 7** is also written for this thesis specifically. It contains a discussion of the overarching ideas presented in the thesis, and ends with possible directions of future work.

- **Chapter 3: Forethought and Hindsight in Credit Assignment** is based on the paper *"Forethought and Hindsight in Credit Assignment"* [Chelu et al., 2020] that was published in *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS), 2020*. The material in this dissertation is adapted from the original work with minimal adjustments for clarity or presentation. I devised the original idea, formulated the problem and solution methodology, contributed the theoretical analysis, conducted all the experiments, and wrote the paper. Doina Precup and Hado van Hasselt provided supervision for the project, contributed to the editing of the paper and gave valuable feedback.

- **Chapter 4: Salience Maps for Selective Credit Assignment** follows the paper *"Selective Credit Assignment"* [Chelu et al., 2022]. The material in this dissertation is adapted from the original work with minimal adjustments for clarity and presentation. I devised the idea, formulated the problem and solution methodology, contributed the theoretical analysis, conducted all the experiments, and wrote the paper. Doina Precup, Hado van Hasselt and Diana Borsa provided supervision for the project, contributed to the editing of the paper, provided feedback and valuable comments.

- **Chapter 5: Acceleration in Policy Optimization** is based on the paper "Functional Acceleration for Policy Mirror Descent" [Chelu and Precup, 2024a] (in submission @ Reinforcement Learning Conference (RLC) 2025; previously presented: @ Aligning Reinforcement Learning Experimentalists and Theorists (ARLET) workshop @ ICML 2024, @ European Workshop on Reinforcement Learning (EWRL) 2024, @ Adaptive Learning in Complex Environments workshop (TTIC)), and the abstract "Mood as an Extrapolation Engine for Adaptive Learning & Decision-Making" [Chelu and Precup, 2024b] (presented @ Computational and Systems Neuroscience (COSYNE) 2025). The material in this dissertation is adapted from the original work with minimal adjustments for clarity and presentation. I devised the idea, formulated the problem and solution methodology, contributed the theoretical analysis, conducted all the experiments, and wrote the paper. Doina Precup supervised.

- **Chapter 6: Excitatory-Inhibitory Dynamics in Adaptive Learning and Decision-Making** is based on the paper "Excitatory-Inhibitory Dynamics in Adaptive Learning and Decision-Making" (in submission @ Cognitive Computational Neuroscience (CCN) 2025). The material in this dissertation is adapted from the original work with minimal adjustments for clarity and presentation. I devised the ideas, formulated the problem and solution methodology, reviewed literature, contributed the analysis, conducted all the simulations, and wrote the text. Doina Precup supervised.

Table of Contents

Abstract ii

Abrégé iv

Acknowledgements v

Contributions to Original Knowledge viii

Statement of Authorship x

List of Figures xx

List of Tables xxi

List of Acronyms xxii

Notation xxiv

1 Introduction **1**

1 Thesis Statement 8

2 Outline of the Manuscript 8

3 Excluded Contributions 10

2 Foundations **12**

1 Sequential Decision Making 13

1.1 Markov Decision Processes 14

1.2 Agent's Internal Representations 16

1.3 The Reinforcement Learning Problem 17

2 Planning methods 18

3 Reinforcement Learning (RL) Methods 23

3.1 Learning by Monte-Carlo (MC) sampling . . . 24
3.2 Temporal-Difference (TD) Learning . . . 25
3.2.1 On-Policy Learning . . . 25
3.2.2 Off-Policy Learning . . . 27
3.3 Function Approximation . . . 28
3.3.1 Parametrization . . . 29
3.3.2 Parameter Optimization . . . 29
4 Optimization in RL . . . 31
4.1 Value Function Optimization . . . 31
4.2 Policy Optimization . . . 32
4.3 Continual Reinforcement Learning (CRL) . . . 35

3 Forethought & Hindsight in Credit Assignment **36**
1 Online Planning for Credit Assignment . . . 36
2 Background and Preliminaries . . . 39
3 Planning Backward in Time . . . 41
3.1 Backward models . . . 43
3.1.1 Time-extended backward models . . . 43
3.2 Backward planning . . . 44
3.3 Model estimation . . . 45
4 Empirical Studies . . . 47
5 Related Work . . . 54
6 Discussion . . . 56

4 Salience Maps for Selective Credit Assignment **58**
1 Motivation . . . 58
2 Background and Preliminaries . . . 61
2.1 Multi-Step Online Credit Assignment . . . 62
2.1.1 Learning and Planning for Evaluation . . . 62

2.1.2 Learning and Planning for Control . . . 64
3 Selective Credit Assignment . . . 65
3.1 Numerical Illustration: On-Policy TD Can Diverge . . . 66
3.2 Selectivity for Stable Off-Policy Credit Assignment . . . 67
3.2.1 Numerical Illustration . . . 68
3.3 Selectivity for Stable On-Policy Credit Assignment . . . 68
3.3.1 Numerical Illustration . . . 69
4 Planning selectively . . . 71
4.1 Online Learning for Off-Policy Counterfactual Models of Traces . . . 71
4.1.1 Numerical Illustration . . . 72
4.2 Selectively Using and Learning Models of Traces . . . 73
4.2.1 Numerical Illustration . . . 75
4.3 Sparse Expected Eligibility Traces . . . 75
4.3.1 Numerical Illustration . . . 76
5 Discussion . . . 77

5 Acceleration in Policy Optimization **79**
1 Background and Preliminaries . . . 82
1.1 Online Convex Optimization . . . 82
1.1.1 Mirror Descent (MD) . . . 83
1.1.2 Acceleration Methods in Optimization . . . 85
1.2 Policy Optimization as Online Learning . . . 86
2 Functional Acceleration for Policy Mirror Descent . . . 88
2.1 Approximate Functional Acceleration for Parametric Policies . . . 90
3 Numerical Studies . . . 91
3.1 When is Acceleration Possible? . . . 92
3.1.1 Experimental Setting . . . 92
3.1.2 Metrics . . . 92
3.2 Policy Dynamics in Value Space . . . 94

3.3 Functional Acceleration with an Inexact Critic 95
4 Related Work 96
5 Discussion 98

6 Excitatory-Inhibitory Dynamics in Adaptive Learning and Decision-Making 100
1 Motivation 101
2 Background and Preliminaries 103
3 Methods and Results 107
3.1 A Mean-Field E/I Model of Decision-Making 107
3.2 E/I in Value-based Control 111
3.3 E/I in Recurrent Neural Networks 118
4 Discussion 126

7 Conclusion 128
1 Summary of Contributions 128
2 Discussion 130
2.1 Interdisciplinary Perspectives: Biological Inspiration and Computational Frameworks 130
2.2 Continual Learning Agents: Necessity and Challenges 131
2.3 Adaptive Learning Through Integrated Model-Based and Model-Free Approaches 132
2.4 Cognitive Constructs for Efficiency in Resource-Limited Agents . . . 134
2.5 Extrapolation in Learning Behaviour 135
2.6 E/I Dynamics and the Neural Substrate of Continual Reinforcement Learning 135
3 Limitations 137
4 Directions for Future Work 137
4.1 Computational Modelling 138
4.2 Information Gathering 139

4.3 Evaluation . . . 140
5 Closing Remarks . . . 140

Appendices

A Forethought & Hindsight in Credit Assignment **141**
1 Theoretical Background and Derivations . . . 141
1.1 Backward Models are Policy Dependent . . . 141
1.2 Action Conditioned Backward Models . . . 141
1.3 Multi-step models . . . 142
1.4 Off-policy Backward Models . . . 145
2 Forward-Backward Equivalences . . . 145
2.1 Forward-Backward Views – Equivalence of Temporal Difference Learning . . . 146
2.2 Relationship Forward-Backward Markov Chains . . . 146
2.3 Equivalence between Forward and Backward Planning . . . 147
3 Model-Learning Objectives . . . 148
3.1 Expected Linear Backward Models . . . 148
3.2 Maximum Likelihood Estimation for Expectation Models . . . 150
4 Empirical Studies Details . . . 151
4.1 Direct effect of backward planning on prediction . . . 151
4.2 Indirect effect of backward planning on control . . . 152

B Salience Maps for Selective Credit Assignment **156**
1 Background and preliminaries (details) . . . 156
1.1 Expected eligibility traces . . . 156
2 Proofs and derivations . . . 158
2.1 New derivation of off-policy Q(λ) . . . 158
2.2 Sparse expected eligibility traces . . . 160
2.3 Weightings for distribution correction: off-policy expected emphasis . . . 160

2.4 Weightings for on-policy learning: on-policy expected emphasis . . . 163
2.4.1 Stability & convergence . . . 166
3 Details on empirical illustrations . . . 172
3.1 Weightings for off-policy distribution correction . . . 172
3.2 Weightings for on-policy distribution correction – Atari Ms.Pac-Man experiments . . . 174
3.3 Off-policy counterfactual evaluation . . . 177
3.4 Sparse expected eligibility traces . . . 180

C Acceleration in Policy Optimization **182**
1 Proofs and derivations . . . 182
1.1 Proofs and Derivations for Sec.1: Background & Preliminaries . . . 182
1.1.1 Functional Policy Gradient . . . 182
1.1.2 Helpful Lemmas for Policy Mirror Descent . . . 184
1.1.3 Derivation of the Suboptimality Decomposition and Convergence of PMD . . . 186
1.2 Proofs for Sec. 2: Functional Acceleration for PMD . . . 189
1.2.1 Extrapolation from the past: derivation of the lazy momentum-based PMD algorithm . . . 191
2 Details on PMD updates for Sec. 2: Functional Acceleration for PMD . . . 201
3 Details on Algorithmic Implementation for Sec. 2.1: Approximate Functional Acceleration for Parametric Policies . . . 201
4 Acceleration Methods in Optimization . . . 204
5 Approximate Policy Mirror Descent as Projected Gradient Descent (PGD) . 205
6 Newton's method . . . 206
7 Experimental details for Sec. 3: Numerical Studies . . . 209
7.1 Details of two-state Markov Decision Processes . . . 209
7.2 Details of Random Markov Decision Processes . . . 210
7.3 Details of Experimental Setup for Sec. 3.1 . . . 210

8 Supplementary Results for Sec. 3: Numerical Studies 212
8.1 Supplementary results for Sec. 3.1 212
8.2 Supplementary results for Sec. 3.2 212
8.3 Supplementary results for Sec. 3.3 214

D Excitatory-Inhibitory Dynamics in Adaptive Learning and Decision-Making 222

1 Details for Sec. 3.1 222
1.1 Nullcline Derivation 222
1.2 Fixed Point and Stability Analysis 224
1.3 Noise generation 228
1.4 Evidence Integration 228
1.5 Parameters 229
2 Details for Sec. 3.2 230
2.1 Agent-Environment Interaction 230
2.2 Parameters 230
3 Details for Sec. 3.3 231
3.1 Initialization details 231
3.2 Parameters 232
4 Additional Results for Sec. E/I in Recurrent Neural Networks 232

List of Figures

2.1 Agent–environment interaction in MDPs 15

3.1 Complementary properties of forward and backward planning 48
3.2 Random Chain 48
3.3 Inflection point in planning forward vs. backward 49
3.4 Planning frame of reference 53
3.5 Information propagation in stochastic settings 54

4.1 Mean Return of Planning in Atari Ms. Pac-Man 60
4.2 Illustrative MDPs 66
4.3 Value Error in Five-state MRP 68
4.4 Mean Return in Atari Ms. Pac-Man 70
4.5 Mazes for Temporal Abstraction 72
4.6 Value error in Open World 73
4.7 Online Learning & Planning with Temporal Abstraction 76

5.1 Extrapolation 81
5.2 When is Acceleration Possible? 93
5.3 Policy Dynamics in Value Space 94
5.4 Functional Acceleration with an Inexact Critic 96

6.1 E/I mean-field model/Evidence integration 109
6.2 Recurrent excitation in a mean-field model 111

6.3 Selective inhibition in a mean-field model . . . 112
6.4 Performance in high precision setting . . . 115
6.5 Entropy in high precision setting . . . 116
6.6 Performance in low precision setting . . . 116
6.7 Entropy in low precision setting . . . 117
6.8 Inhibitory modulation of the policy in adaptive decision-making . . . 118
6.9 E/I RNN . . . 119
6.10 Performance of RNNs on Setting (i) . . . 122
6.11 RNN Activity Traces . . . 123
6.12 Performance of RNNs on Setting (ii) . . . 126

A.1 Random Chain . . . 151
A.2 Maze Navigation . . . 152

B.1 The 5-state chain MDP . . . 173
B.2 Open World . . . 179

C.1 When is Acceleration Possible? (final regret) . . . 212
C.2 When is Acceleration Possible? (learning curves) . . . 213
C.3 Policy Dynamics (PMD vs. PI) . . . 214
C.4 Policy Dynamics (PMD, k) . . . 215
C.5 Policy Dynamics (PMD(+loo)) . . . 216
C.6 Policy Dynamics (PMD(+loo), k) . . . 216
C.7 Policy Dynamics (PMD vs. PMD(+loo)) . . . 217
C.8 Policy Dynamics (PI vs. PMD vs. PMD(+mom))—example (i) . . . 218
C.9 Policy Dynamics (PI vs. PMD vs. PMD(+mom))—example (ii) . . . 219
C.10 Policy Dynamics (PI vs. PMD vs. PMD(+mom))—example (iii) . . . 220
C.11 Policy Dynamics (PI vs. PMD vs. PMD(+mom))—example (iv) . . . 221

D.1 Parameter Sensitivity Analysis: Heatmaps for Setting (i) . . . 233

D.2 Parameter Sensitivity Analysis: U-shaped curves for Setting (i) 233
D.3 Parameter Sensitivity Analysis: Learning curves for Setting (i) 234
D.4 Parameter Sensitivity Analysis for Setting (ii) 234
D.5 Performance of RNNs on Setting (ii) 235
D.6 Parameter Sensitivity Analysis: U-shaped curves for Setting (ii) 235
D.7 Parameter Sensitivity Analysis: Learning Curves for Setting (ii) 236

List of Tables

1 Notation xxiii

A.1 Hyperparameters for tabular control on the Maze Gridworld 155

C.1 The parameters used for the optimization in Sec. 3.1. 211

D.1 Parameters for the Wilson-Cowan mean-field model 229

D.2 Parameters for Value-based Control 231

D.3 Parameters for RNN Networks in Setting(i) 232

D.4 Parameters for RNN Networks in Setting(ii) 233

List of Acronyms

5-HT	Serotonin (5-Hydroxytryptamine)
AC	Actor-Critic
AI	Artificial Intelligence
CRL	Continual Reinforcement Learning
DA	Dopamine
E/I	Excitation-Inhibition
EC	Entorhinal Cortex
HPC	Hippocampus
MDP	Markov Decision Process
mPFC	Medial Prefrontal Cortex
NAc	Nucleus Accumbens
OFC	Orbitofrontal Cortex
PFC	Prefrontal Cortex
PGD	Projected Gradient Descent
PI	Policy Iteration
PMD	Policy Mirror Descent
RPE	Reward Prediction Error
RL	Reinforcement Learning
SN	Substantia Nigra
TD	Temporal Difference
VTA	Ventral Tegmental Area

Notation

We will generally follow the notation from Sutton and Barto [2018], and provide specific notation for each chapter wherever necessary.

Table 1: Notation

$\mathcal{S}$	the set of states
$\mathcal{A}$	the set of actions
$\mathcal{P}(s'\|s,a)$	the probability of transitioning into state s' upon taking action a in state s
$r(s,a)$	the (expected) reward for taking action a in state s
γ	discount factor
S_t	state encountered at time t
A_t	action taken at time t
$R_{t+1} \sim r(S_t, A_t)$	the reward upon taking action A_t at state S_t
δ_t	temporal difference error at time t.
G_t	return from time t forward: $\sum_{k=t}^{\infty} \gamma^{k-t} R_{k+1}$.
$G_{t:t+n}$	bootstrapped return from time t until $t+n$: $\sum_{k=t}^{t+n} \gamma^{k-t} R_{k+1} + \gamma V_{t+n+1}$
G_t^λ	λ-return from time t: $(1-\lambda)\sum_{n=0}^{\infty} \lambda^n G_{t:t+n+1}$
$\pi(a\|s)$	policy specifying the probability of taking a in state s
V_π, Q_π	value function, action-value/Q function under policy π
V^*, Q^*	optimal value functions
$\mathcal{T}^\pi$	Bellman operator for policy π

$\mathcal{T}$	Bellman optimality operator
$\mathcal{T}^n$	n-step Bellman operator
$\mathcal{T}^\lambda$ Bellman operator	
λ	trace-decay parameter
$\mathbf{e}$	eligibility trace
α	learning rate or step-size in updates
$\mathbf{w}, \theta$	parameters
$\mathbf{x}$	feature representation
ρ	initial state distribution
d_π	steady state distribution for policy π
h	mirror map
D_h	Bregman divergence induced by h
$\doteq$	equal by definition
$\|\cdot\|$	supremum norm
$\mathbf{1}_{a=b}$	indicator function that returns 1 if $a = b$ and 0 otherwise
$\mathbf{0}$	vector of zero-components

1 Introduction

"Somewhere, something incredible is waiting to be known."

Carl Sagan

In both artificial and biological systems, the capacity to learn from experience and adapt to new situations is critical for survival and efficiency. Whether an animal is navigating its environment in search of food or an artificial agent is optimizing its behaviour in a simulated world, adaptive decision-making hinges on learning from past experiences and generalizing to new contexts. Reusing knowledge, integrating new information, and adjusting behavioural strategies are essential skills for responding effectively to a dynamic and ever-changing world.

A well-supported theory across psychology, neuroscience, and behavioural economics proposes that decision-making involves multiple systems, each optimal under different circumstances [Daw et al., 2005, Kahneman, 2013]. The *habitual*, *model-free* system is responsible for fast, automatic decisions driven by reinforcement, where actions leading to rewards are more likely to be repeated [Thorndike, 1911]. *Temporal-Difference (TD)* learning [Sutton, 1988], a central concept in *reinforcement learning (RL)* [Sutton and Barto, 2018], formalizes this by updating action-values based on the difference between expected and actual rewards or values. This discrepancy, known as the *reward prediction error (RPE)* [Barto, 1995, Schultz et al., 1997] or temporal-difference error (TD error) [Sutton, 1988], is hypothesized to be signalled by *dopamine (DA)*—a neurotransmitter and neuromodulator

produced in the brain's ventral tegmental area (VTA) and substantia nigra (SN)—and plays a vital role in guiding learning and decision-making. A key area in this process is the *striatum*, where dopamine signals update action propensities [Suri and Schultz, 1999].

However, *habitual decision-making* alone falls short in complex and rapidly changing environments. Both humans and animals can engage in *goal-directed*, *model-based decision-making*, which involves using an *internal model* of the environment to simulate future outcomes and evaluate possible actions. This concept was first proposed in Tolman's theory of *cognitive maps*, confirmed through behavioural experiments [Tolman, 1948], and further validated by the discovery of *place and grid cells* in the *hippocampus (HPC)* and *entorhinal cortex (EC)*, an achievement recognized by the 2014 Nobel Prize in Physiology [Burgess, 2014]. The idea of *"space as away of perceiving, not as a thing to be perceived"* [O'Keefe and Nadel, 1978] lead to one of the most important ideas in computational neuroscience today—*neural manifolds* [Barack and Krakauer, 2021, Langdon et al., 2023, Vyas et al., 2020], i.e. low-dimensional representations within the high-dimensional space of behavioural signals in neural population activity, where coordinated patterns of neuron firing conceptually reside. This perspective aligns with the view that the brain performs *distributed computation* [Findling et al., 2023, Laboratory et al., 2023] (similarly to artificial neural networks) and information is represented by the coordinated activity of populations of neurons, providing a more holistic understanding of neural processing.

From a computational standpoint, model-free and model-based approaches to RL provide complementary strategies [Daw et al., 2005, Niv et al., 2006]. While model-free systems like TD learning are computationally efficient and suited for stable environments, model-based systems support adaptive decision-making by allowing agents to plan ahead based on predictions of future states and outcomes [Kool et al., 2018]. *Model-based RL (MBRL)* leverages *internal models* to simulate the effects of different actions, supporting *prospective planning* and enabling more flexible behaviour, particularly in uncertain or changing environments, as long as learned models can be reused [Dayan, 2009].

A key challenge faced by both artificial and biological agents is the problem of *temporal credit assignment* [Minsky, 1961]—the ability to determine which past actions or decisions were responsible for future outcomes, especially when rewards are delayed [Sutton and Barto, 2018]. TD learning handles this through incremental updates based on immediate feedback, but this approach is slow when rewards are significantly delayed. *Multi-step updates* accelerate this process by propagating credit across sequences of actions that lead to a reward. Discoveries like the concept of *"preplay"* in the HPC [Dvorak et al., 2021, Farrell et al., 2024], highlight the brain's ability to simulate future events to facilitate planning. Computational models simulating these processes rely on *successor representations* [Dayan, 1993] to represent *cognitive maps* [GX-Chen et al., 2022], or on *latent space models* [Schrittwieser et al., 2019], demonstrating that more efficient learning is possible with *model-based planning*.

Our **first contribution** is an investigation of the mechanisms by which agents can leverage internal models to anticipate outcomes and adapt their decisions [Chelu et al., 2020]. By exploring both prospective planning and retrospective reasoning, we demonstrate that these mechanisms are complementary and can enhance learning efficiency under different circumstances. Using *retrospective models*—which simulate what could have happened—agents can assign credit not just to actions taken but also to *counterfactual actions*, improving their learning efficiency [Chelu et al., 2020, 2022, Pitis, 2018, van Hasselt et al., 2020]. This integration of *forethought and hindsight* [Chelu et al., 2020] allows agents to adapt their strategies based on both actual experiences and alternative possibilities, both *in anticipation and in retrospect*.

Real-world environments often present additional challenges for credit assignment in both biological and artificial agents, such as delayed or sparse rewards, as well as noisy or ambiguous observations. Research in neuroscience highlights the involvement of the *prefrontal cortex (PFC)*—particularly the *orbitofrontal cortex (OFC)* [Goudar et al., 2023, Hattori et al., 2023, Walton et al., 2010] and the *medial prefrontal cortex (mPFC)* [Harlow,

1949, Wang et al., 2018]—in model-based processes. These brain regions, along with the HPC, are essential for proper identification of *internal state representations* [Courellis et al., 2024]. Downstream projections from these areas to the *nucleus accumbens (NAc)* integrate contextual information with RPEs to modulate *goal-directed behaviour* [Grace, 2000, Iyer et al., 2024, Mogenson et al., 1988]. Dysfunction in these pathways can lead to impaired outcome-contingent learning and decision-making, contributing to psychiatric disorders like depression and anxiety.

Addressing the complexities of long-term credit assignment and its impact on behaviour in artificial agents remains an active area of research in RL [Arjona-Medina et al., 2019, Chelu et al., 2020, 2022, Harutyunyan et al., 2019b, Hung et al., 2019, Ke et al., 2018b, Mesnard et al., 2020]. Computational models have proposed ways to enhance this process by prioritizing learning updates based on recency or predictability [Moore and Atkeson, 1992, Schaul et al., 2015].

Building on this, our **second contribution** introduces *retrospective selective models* [Chelu et al., 2022] that prioritize learning updates based on the salience of experiences or attentional biases. This approach is rooted in cognitive science and mirrors the phenomenological aspects of human cognition, whereby *memory, mood, and emotion* can directly impact salience/attention and bias decision-making, influencing the perception of outcomes (e.g. emotional tagging and selective memory consolidation [Richter-Levin and Akirav, 2003, Zhu et al., 2022]). Similarly, our approach emphasizes certain experiences over others to improve learning efficiency in noisy environments with sparse or delayed rewards. This selective focus allows agents to concentrate computational resources more effectively.

The *striatum* plays a pivotal role in RL, with its *dorsal* and *ventral* subdivisions contributing to distinct aspects of action selection (dorsal) and reward processing (ventral), respectively. *Neural Actor-Critic (AC)* algorithms, which separate the learning of a behavioural strategy or *policy* (the *actor*) from value-based evaluations (the *critic*), offer an influential conceptual model for understanding how the brain might implement these processes. This model

provides insight into *habit learning*—experimentally associated with the *dorsolateral striatum (DLS)* [Daw et al., 2005, 2011, Huang et al., 2020, Schreiner et al., 2020, Yin and Knowlton, 2006, Yin et al., 2004, 2005]—but also highlights the need for integrating model-based approaches to account for *goal-directed decision-making*, which is experimentally shown to be supported by the *dorsomedial striatum (DMS)* [Balleine, 2005, Corbit and Balleine, 2005, Daw et al., 2005, 2011, Huang et al., 2020, Schreiner et al., 2020] (in addition to cortical areas).

Recent advancements in policy-based RL algorithms, such as *Policy Mirror Descent (PMD)* [Agarwal et al., 2019, Bhandari and Russo, 2021, Johnson et al., 2023, Xiao, 2022], suggest ways to bridge value- and policy-based control, offering novel perspectives on efficient decision-making in both biological and artificial systems. Specifically, PMD can be interpreted as using *probabilistic inference* to obtain a *goal-directed policy* by starting from a *prior policy*—possibly a habitual behaviour strategy—and then conditioning on the desired goals or outcomes to estimate a *posterior policy* that approximately meets those outcomes. Positive or negative outcomes relative to performance expectations drive changes in behaviour, creating biases that shape behaviour strategies.

In biological systems, behaviour is shaped not only by fast reinforcement signals, such as immediate rewards or punishments, but also by slower, longer-lasting modulatory influences. Emotional states, which emerge from complex brain circuitry with slow dynamics, play a crucial role in biasing learning, decision-making, and motivation over time. Experimental evidence in cognitive science [Bakic et al., 2014, Bennett et al., 2022, Eldar et al., 2016, Jangraw et al., 2023, Rutledge et al., 2014] suggests *mood*—a prolonged emotional state based on recent experience—influences decision-making processes by biasing perceptions of future outcomes. Evidence suggests that mood serves an evolutionary advantage, and functions as an *extrapolation* mechanism, tracking how recent experience deviates from prior expectations to inform future behaviour. Beyond emotions, a range of physiological processes also contribute. Homeostatic mechanisms, such as hunger, thirst, fatigue, and

hormonal cycles, create internal drives that influence behaviour and learning priorities. Neuromodulators like dopamine, serotonin, norepinephrine, and acetylcholine act over diverse timescales to regulate arousal, attention, and plasticity. Even circadian rhythms and immune system signals can affect cognition and mood. These systems often have longer time constants and can bias learning and decision-making over extended periods, influencing things like exploration, risk sensitivity, and motivation. Together, these slower processes interact with fast reinforcement signals, creating a rich, multi-scale framework for shaping adaptive behaviour.

Taking inspiration from these findings, we propose a similar mechanism that can use prior experience to accelerate learning. For our **third contribution**, we propose a momentum-based update within the *Policy Mirror Descent (PMD)* framework [Agarwal et al., 2019, Bhandari and Russo, 2021, Johnson et al., 2023, Vaswani et al., 2021, Xiao, 2022], effectively incorporating the influence of recent policy changes into future updates. This approach accelerates learning in environments with extended time horizons and sparse state connectivity.

Biological organisms must navigate dynamic and uncertain environments, where adaptive learning and decision-making are essential for survival. Rapid behavioural adjustments to shifting contingencies enable efficient continual learning. A core organizing principle of neural circuits is Dale's principle [Dale, 1935], which posits that individual neurons release either excitatory or inhibitory neurotransmitters, but not both [Eccles et al., 1954]. Neural circuits throughout the brain rely on the interplay of excitatory (E) and inhibitory (I) neurons to achieve both stable and flexible computations [Dale, 1935, Isaacson and Scanziani, 2011, Pouille et al., 2009]. At the neural level, behavioural flexibility is influenced by E/I dynamics [Mante et al., 2013, Roach et al., 2023, Sussillo and Barak, 2013, Wakhloo et al., 2024]. Continual RL aims to equip artificial agents with analogous adaptive capabilities in dynamic environments [Abel et al., 2023, Dohare et al., 2024, Khetarpal et al., 2020]. Yet, these frameworks neglect biological constraints such as Dale's law.

In our **fourth and final contribution** we investigate how E/I dynamics may contribute to flexible behaviour. Using a mean-field model of decision-making, we find that decision-selective excitation (E) and inhibition (I) can stabilize or amplify competition between alternative choices. Building on these insights, we integrate an E/I mechanism into the preferences of an RL agent, illustrating how inhibitory feedback can modulate the behavioural flexibility. To scale these mechanistic principles to biologically-plausible models, we investigate the feasibility of RL agents with E/I-constrained recurrent neural networks (RNNs) in dynamic environments, showing that a carefully structured E/I partitioning maintains sign constraints while preserving robust performance.

1 Thesis Statement

The central question investigated in this thesis is the following:

How can agents adapt flexibly their decision-making in complex or dynamic environments?

First, we show that using *internal models for anticipation and adaptation—through prospective planning and retrospective reasoning*—enhances learning efficiency. **Second**, we introduce *retrospective selective models* that prioritize salient experiences, mirroring selective memory consolidation mechanisms, to improve learning in uncertain settings with sparse or delayed rewards. **Third**, we investigate the role of *momentum in policy learning*, demonstrating that it can accelerate learning. **Finally**, we develop a computational framework to explore the computational role of *excitation-inhibition (E/I)* dynamics in adaptive decision-making, illustrating how E/I interactions can influence behavioural flexibility in dynamic environments.

Collectively, these contributions advance our understanding of integrating learning and planning in both biological and artificial agents, highlighting pathways to achieve flexible and efficient decision-making.

2 Outline of the Manuscript

Chapter 1 This chapter sets the stage by providing the context and motivation for the research presented in this thesis. We introduce the thesis statement, outline the central research questions, and provide a summary of the contributions.

The rest of the thesis is organized as follows:

Chapter 2: Foundations This chapter establishes the theoretical and computational background necessary for understanding the thesis. It includes discussions on reinforcement

learning (RL), sequential decision-making, function approximation, and the interplay between model-based and model-free approaches. Key mathematical frameworks and optimization methods relevant to policy learning are introduced.

Chapter 3: Forethought and Hindsight in Credit Assignment This chapter explores how agents use internal models for anticipation and adaptation, leveraging both prospective planning and retrospective reasoning for credit assignment. We conduct an empirical investigation of planning dynamics, highlighting the complementary nature of forward and backward models in enhancing learning efficiency. The chapter demonstrates how integrating forethought and hindsight allows agents to adapt their strategies based on both actual experiences and alternative possibilities.

Chapter 4—Salience Maps for Selective Credit Assignment This chapter introduces the concept of salience in prioritizing learning updates, inspired by salience processing in biological systems [Cowan et al., 2021, Schimmelpfennig et al., 2023, Seeley, 2019, Zhu et al., 2022]. We develop a unified framework for selective credit assignment in RL, leveraging salience maps to quantify the contribution of learning updates and mediate backward credit distribution. Theoretical analysis and numerical illustrations demonstrate how salience maps improve stability and convergence in RL, especially in environments with sparse or delayed rewards.

Chapter 5: Acceleration in Policy Optimization This chapter investigates the role of momentum and acceleration mechanisms in policy learning. We introduce a novel RL framework that incorporates momentum into mirror descent policy optimization, effectively accelerating learning and enhancing decision-making efficiency. Theoretical and practical insights are presented on how momentum influences learning speed and stability, with numerical studies validating the proposed framework. This approach amplifies or stabilizes behaviour by recognizing dependencies across successive policies, improving learning in certain settings characterized by long horizons and sparse state connectivity.

Chapter 6: Excitatory-Inhibitory Dynamics in Adaptive Learning and Decision-Making This chapter explores how excitatory–inhibitory (E/I) dynamics contribute to adaptive learning and decision-making across multiple modelling levels. Starting with a mean-field model of decision circuits, it shows how recurrent excitation and selective inhibition shape attractor dynamics. These principles are then embedded into a reinforcement learning (RL) agent, where inhibitory feedback regulates learning flexibility in dynamic environments. Finally, biologically constrained recurrent neural networks (RNNs) are evaluated in continual RL tasks, revealing that naïve E/I separation hinders learning, while a structured architecture (DANN) preserves both biological plausibility and adaptive performance. Together, the results demonstrate that E/I interactions can offer a computational mechanism for modulating flexibility in dynamic environments.

Chapter 7: Conclusion This chapter summarizes the thesis contributions, highlighting key insights and implications. We discuss limitations of the current work and suggest directions for future research.

3 Excluded Contributions

This thesis focuses on mechanisms of flexible decision-making in biological and artificial systems through the integration of internal modelling, salience processing, mood mechanisms, and neural inhibition. However, some contributions, which were collaborative or fall outside the scope of this work, were excluded. Below is an overview of these excluded contributions:

- In Jiang et al. [2021], we address the challenges posed by the "deadly triad" in reinforcement learning—off-policy sampling, function approximation, and bootstrapping—which often lead to instability and high variance. The paper introduces X-ETD(n), a method for incorporating expected emphatic traces into temporal-difference learning, specifically designed for non-sequential data sampled from replay buffers. This

feature enables more stable learning with theoretical guarantees for convergence and variance reduction, significantly improving performance in deep RL tasks such as Atari 2600 games.

- In GX-Chen et al. [2022], we propose an implicit planning method, which effectively leverages a successor features model in the construction of the bootstrap target for TD methods. We show the intuition of our method in tabular illustrations, and demonstrate that it can be scaled up to non-linear function approximation, enabling faster policy evaluation and better control performance in Deep RL.

- In [de Lara et al.], we highlight two failure modes of recurrent models trained for continual reinforcement learning—embedding collapse and drift—and propose strategies to mitigate them and enhance adaptability in dynamic environments.

- In [Chelu et al., 2023], we present a framework for accelerating policy optimization in reinforcement learning (RL) by using adaptive-optimistic predictions for policy updates. By connecting policy iteration and policy gradient methods, we reinterpret various algorithms as updates guided by gradient predictions, which are adjusted to improve post-update performance. This approach includes model-based forward search and optimistic meta-learning methods. We propose an optimistic policy gradient algorithm with meta-gradient learning and analyze its properties in an illustrative task.

- In [Juliani et al., 2024], we developed a computational dual-receptor model of serotonergic psychedelics, exploring the neuromodulatory mechanisms of 5-HT2A and 5-HT1A receptor agonism on illustrative neural dynamics. This framework and simulations contribute theoretical insights into how psychedelics might enhance cognitive flexibility.

2 Foundations

"If I have seen further, it is by standing on the shoulders of giants."

Isaac Newton

Cognitive science is a multidisciplinary field that examines the processes underlying thought, learning, and decision-making through the integration of psychology, neuroscience, computer science, and philosophy. By exploring how humans and other agents acquire, represent, and utilize knowledge, cognitive science provides essential frameworks for understanding adaptive behaviour. Within this context, artificial intelligence (AI) and reinforcement learning (RL) bridge the gap between computational principles and psychological theories, offering a formalism to study how agents navigate complex environments. This thesis builds upon these principles to investigate the computational substrates of decision-making which mirror neural processes, leveraging insights from cognitive science to explore how internal representations and biases shape adaptive behaviour.

Artificial Intelligence (AI) has become a crucial field for advancing our understanding of cognitive processes. As Pamela McCorduck describes, AI embodies "that audacious effort to duplicate in an artifact what we humans consider to be our most important, our identifying, property—our *intelligence*" [McCorduck, 1979]. From the standpoint of cognitive science, AI is instrumental in modelling and simulating the processes underlying human thought. By examining AI systems, we gain insights into essential cognitive functions such as learning, memory, and decision-making. This exploration not only sheds

light on how intelligence can be replicated in artificial systems but also enhances our understanding of cognitive processes in biological entities.

Reinforcement learning (RL) provides a robust framework for interpreting how organisms learn from interactions with their environment—a concept central to both cognitive science and neuroscience. Unlike supervised learning, RL does not depend on predefined labels, but models how agents adapt through trial and error, aiming to maximize cumulative rewards without explicit guidance. This mechanism is directly linked to cognitive phenomena like habit formation, goal-directed behaviour, and adaptability. RL aligns with cognitive science's broader goal of linking neural mechanisms to observable behaviour, facilitating a deeper understanding of the interplay between habitual and goal-directed actions.

The RL framework [Sutton and Barto, 2018] refers to the problem of solving sequential decision-making tasks under uncertainty, together with a class of solution methods tailored for it. The RL problem has found applications in games [Bellemare et al., 2017, Hessel et al., 2017, Mnih et al., 2013, 2016, Schrittwieser et al., 2019, Silver et al., 2014, 2017a, Tesauro, 1994, Zahavy et al., 2023], robotic manipulation [Haarnoja et al., 2018, Schulman et al., 2015, 2017], medicine [Jumper et al., 2021, Nie et al., 2020, Schaefer et al., 2004] and is described through the formalism Markov Decision Processes (MDP) [Puterman, 1994].

1 Sequential Decision Making

Sequential decision-making, central to RL, reflects how organisms balance short- and long-term goals by making choices that impact future outcomes. This is essential for cognitive science as it mirrors human learning strategies and adaptation. In neuroscience, RL principles align with the brain's reward systems, such as dopamine circuits, which adjust behaviour based on reward prediction errors (RPEs). By studying RL, we gain insights into the neural mechanisms of decision-making, learning, and adaptation, linking cognitive processes with their biological underpinnings.

In RL, the concepts of the *agent*, *environment*, and their *interaction loop* are fundamental, mirroring real-world cognitive processes. The agent perceives and acts within the environment, receiving feedback in the form of rewards or penalties. This continuous interaction drives learning, as the agent adjusts its actions to maximize long-term outcomes. In cognitive science, this loop reflects how humans and animals adapt, make decisions, and learn from experience. By understanding this dynamic, we gain insight into how cognitive systems develop flexibility and goal-directed behaviour in complex environments.

An **agent** is an algorithm that maps observations as input to output actions, utilizing internal and evolving representations. The **environment** comprises all external factors influencing the outcomes of the agent's actions. It provides the agent with observations or states, accepts actions, and returns feedback in the form of rewards and subsequent states. Operating on underlying dynamics typically unknown to the agent, the environment dictates how actions impact future states and rewards.

1.1 Markov Decision Processes

A *Markov Decision Process (MDP)* [Bellman, 1957] is a mathematical model describing the interaction between a *decision-making agent* and its *environment*. The core assumption is the *Markov property*, asserting that the next state depends solely on the current state and action, not on the sequence of events that preceded it. Formally, if $\tau_t = \{S_0, A_0, S_1, A_1, \ldots, S_t, A_t\}$ is the history, then the transition probability to the next state S_{t+1} is:

$$\Pr(S_{t+1} \mid \tau_t) = \Pr(St + 1 \mid S_t, A_t) \quad \text{(Markov property)}$$

This property simplifies analysis and algorithm design, enabling efficient solutions even in complex scenarios.

Definition 1 (Markov Decision Process). An *MDP* is defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$, where:

- $\mathcal{S}$ is the set of possible states,
- $\mathcal{A}$ is the set of possible actions,
- $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the state transition probability function, with $\mathcal{P}(s' \mid s, a)$ representing the probability of transitioning to state s' from state s by taking action a,
- $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, assigning rewards to state-action pairs,
- $\gamma \in [0, 1)$ is the discount factor, weighting future rewards in the agent's objective.

Definition 2 (Markov Reward Process (MRP)). An *MRP* is defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{P}, r, \gamma)$. An MDP is an MRP with decisions.

At each discrete time step t, the agent observes a state $S_t \in \mathcal{S}$, selects an action $A_t \in \mathcal{A}$, receives a reward $R_{t+1} = r(S_t, A_t)$, and transitions to the next state $S_{t+1} \sim \mathcal{P}(\cdot \mid S_t, A_t)$ (Fig. 2.1).

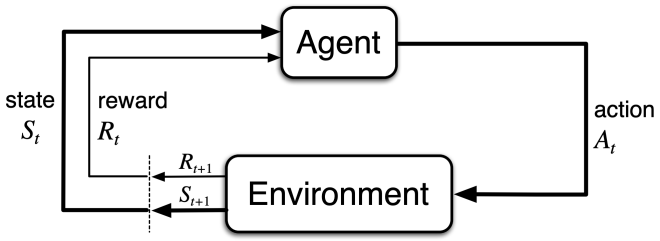


Figure 2.1: The agent–environment interaction in an MDP (Figure 3.1. from Sutton and Barto [2018]

Assumption 3 (Setting). We assume $\mathcal{A}$ is finite and discrete, and the r is bounded ($|r|\infty \leq R\max$). We focus on the discounted setting ($\gamma < 1$), which defines the agent's planning horizon and simplifies the mathematical treatment. While our theoretical analysis considers finite $\mathcal{S}$, practical applications may involve function approximation techniques discussed in Section 3.3.

In the *episodic* setting, in every episode, the learner acts for some finite number of steps, starting from a fixed starting state $S_0 \sim \rho$. It samples a trajectory, and the state resets to $S_0 \sim \rho$. This episodic model of feedback is applicable to both the finite horizon and infinite horizon settings. In the *finite horizon* setting, each episode lasts for H-steps, and then the state is reset to $S_0 \sim \rho$. For the *infinite horizon* setting, it is often natural to assume the episode will terminate at each step with probability $1-\gamma$. After termination, we again assume that the state is reset to $S_0 \sim \rho$. If each step in an episode is terminated with probability $1-\gamma$, then the observed cumulative reward in an episode of a policy provides an unbiased estimate of the infinite-horizon, discounted value of that policy [Shwartz, 2001].

Herein, we constrain the **agent** to follow the principle of *bounded rationality*.

Definition 4 (Bounded rationality). In RL, *bounded rationality*—a concept from cognitive psychology and decision theory—accounts for the limited ability of agents to act optimally due to resources limitations, such as computational power and memory [Gershman et al., 2015, Kumar et al., 2023, Simon, 1955].

1.2 Agent's Internal Representations

Intelligent agents require internal models or representations that evolve over time. These are central to RL research. The agent's goal is to maximize the cumulative reward over time, quantified by the *return*:

$$G_t \doteq \sum_{i=t}^{\infty} \gamma^{i-t} R_{i+1} \qquad \text{(return)} \tag{2.1}$$

The agent's behaviour is governed by a *policy*:

> **Definition 5 (Policy).** A (stationary, memoryless) *policy* $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ specifies the probability $\pi(a \mid s)$ of selecting action a in state s.

The **objective of RL** is to find an **optimal policy** π^* that maximizes the expected return. The **action-value function** (or **Q-function**) under policy π is defined as:

$$Q_\pi(s, a) \doteq \mathbb{E}\Big[\underbrace{\textstyle\sum_{i=t}^{\infty} \gamma^{i-t} R_{i+1}}_{G_t} \mid S_t = s, A_t = a; \pi\Big] \qquad \text{(action-value function)} \qquad (2.2)$$

Similarly, the **state-value function** (or **V-function**) is:

$$V_\pi(s) \doteq \mathbb{E}\left[\textstyle\sum_{i=t}^{\infty} \gamma^{i-t} R_{i+1} \mid S_t = s; \pi\right] = \textstyle\sum_{a \in \mathcal{A}} \pi(a \mid s) Q_\pi(s, a) \qquad \text{(value function)} \qquad (2.3)$$

Let d_π be the discounted visitation distribution $d_\pi(s) = (1-\gamma) \sum_{i=0}^{\infty} \gamma^i \Pr(S_i = s | S_0 \sim \rho, A_j \sim \pi_{s_j}, \forall j \leq i)$.

1.3 The Reinforcement Learning Problem

The central challenge in RL [Sutton and Barto, 2018] is to determine the optimal policy $\pi^*(s)$ from the set of all policies mapping states to actions $\Pi \doteq \Delta_{\mathcal{A}}^{|\mathcal{S}|}$, that maximizes $V_\pi(s)$ for all $s \in \mathcal{S}$.

> **Definition 6 (Policy Optimization — The Control problem).** For a distribution ρ over states, the task of finding the best policy which maximizes the policy's value is $\pi^* = \operatorname{argmax}_{\pi \in \Pi} \mathbb{E}_{S \sim \rho} V_\pi(S)$.

To formalize optimality, we define the *optimal value functions*:

$$Q^*(s, a) \doteq \max_{\pi \in \Pi} Q_\pi(s, a) \qquad \text{(optimal action-value function)} \qquad (2.4)$$

$$V^*(s) \doteq \max_{\pi \in \Pi} V_\pi(s) = \max_{a \in \mathcal{A}} Q^*(s, a) \qquad \text{(optimal value function)} \qquad (2.5)$$

The optimal policy π^* can be derived directly from these value functions. It is guaranteed that an optimal deterministic stationary policy π^* exists such that $V_{\pi^*} = V^*$ [Bertsekas and Tsitsiklis, 1996, Puterman, 1994].

Solving the control problem often involves addressing the *policy evaluation (prediction) problem*.

> **Definition 7 (Policy Evaluation — The Prediction Problem).** The process of computing the value function V_π for a given policy π is called policy evaluation.

2 Planning methods

We now explore classic solution methods for the RL problem, known as *planning methods*, which presuppose knowledge of the environment's dynamics and reward structure.

Dynamic Programming (DP) leverages recursive decompositions to solve complex problems by breaking them into simpler subproblems. The cornerstone of DP in RL is the *Bellman equation* [Bellman, 1957], which provides a recursive relationship for value functions.

The *Bellman equation* for policy evaluation is:

$$Q_\pi(s,a) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s' \mid s,a) \sum_{a' \in \mathcal{A}} \pi(a' \mid s') Q_\pi(s',a') \quad \text{(Bellman Equation)} \tag{2.6}$$

For the optimal policy, the *Bellman optimality equation* is:

$$Q^*(s,a) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s' \mid s,a) \max_{a' \in \mathcal{A}} Q^*(s',a') \quad \text{(Bellman Optimality Equation)} \tag{2.7}$$

These equations can be represented using operators.

Definition 1 (Bellman Operator). The *Bellman operator* $\mathcal{T}_\pi$ for a policy π acts on a function $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ as:

$$\mathcal{T}_\pi Q(s,a) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s' \mid s,a) \sum_{a' \in \mathcal{A}} \pi(a' \mid s') Q(s',a') \tag{2.8}$$

We can represent the Bellman equations in matrix-vector form. Given a policy π, let $\mathcal{P}_\pi$ be

$$\mathcal{P}_\pi Q(s,a) \doteq \sum_{s' \in \mathcal{S}} \sum_{a' \in \mathcal{A}} \mathcal{P}(s' \mid s,a) \pi(a' \mid s') Q(s',a') \quad \text{(transition dynamics in vector form)} \tag{2.9}$$

As a result, Eq. 2.8 can be represented in matrix-vector form as

$$\mathcal{T}_\pi Q \doteq r + \gamma \mathcal{P}_\pi Q \qquad \text{(Bellman operator)} \tag{2.10}$$

The Bellman operator possesses properties of *monotonicity* and *contraction*.

Definition 2 (Monotonicity). For any $Q, Q' : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, where $Q \leq Q'$, we also have $\mathcal{T}_\pi Q \leq \mathcal{T}_\pi Q'$

Definition 3 (γ-Contraction). For any pair of $Q, Q' : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, we have the following relation in L_∞ norm:

$$\|\mathcal{T}_\pi Q - \mathcal{T}_\pi Q'\|_\infty \leq \gamma \|Q - Q'\|_\infty$$

When the operator is iteratively applied, the properties of *monotonicity* and *contraction* ensure convergence to the fixed point Q_π

Definition 4 (Fixed point). A function f is said to be a fixed point of an operator $\mathcal{T}$, if $\mathcal{T} f = f$.

The Bellman equation Eq.2.10 tells us that Q_π is the fixed point of the Bellman operator

$$\mathcal{T}_\pi Q_\pi = Q_\pi = \sum_{t=0}^{\infty} (\gamma \mathcal{P}_\pi)^t r = (I - \gamma \mathcal{P}_\pi)^{-1} r \qquad \text{(analytical solution)} \tag{2.11}$$

where for any operator $\mathcal{T}$, $\mathcal{T}_t$ denotes t successive applications of $\mathcal{T}$, and the last equality is a standard result for stochastic operators and matrices [Kemeny and Snell, 1960].

This solution can be computed exactly if the state space is not too large. The key strength of DP is that when that is not the case, iterative methods, based on repeated applications of $\mathcal{T}_\pi$ converge to Q_π.

The approach in Eq. 2.11 might not be feasible for large state spaces, in which case iterative algorithms are useful, where we repeatedly apply the operator until we reach the fixed point. This approach is principled, based on the Banach fixed-point theorem [Banach, 1922].

To reach the solution iteratively, we can maintain a sequence of value functions $(Q_k)_{k \in \mathbb{N}}$. The Bellman operator $\mathcal{T}_\pi$ is the operator underlying **iterative policy evaluation**, an algorithm provided in Alg. 1, which applies the Bellman operator for a number of iterations $k \leq K$.

Algorithm 1 Policy Evaluation

1: **Input:** MDP $M = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$ and policy π.
2: Initialize $Q_0 \doteq \mathbf{0}^{|\mathcal{S}| \times |\mathcal{A}|}$.
3: Compute $\mathcal{P}_\pi$.
4: **for** $k = 1, 2 \dots K$ **do**
5: $\quad Q_{k+1} = \mathcal{T}_\pi Q_k \qquad \{\mathcal{T}_\pi Q \doteq r + \gamma P_\pi Q, \forall Q \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{A}|}\}$
6: **return** Q_K

All of the convergence properties carry over, and one may use the repeated n-step operator $(T_\pi)^n$ to converge to Q_π

$$(\mathcal{T}_\pi)^n Q = \sum_{t=0}^{n} (\gamma \mathcal{P}_\pi)^t r + (\gamma \mathcal{P}_\pi)^{n+1} Q \qquad (n\text{-step Bellman operator}) \tag{2.12}$$

This reduces the number of iterations required for convergence but makes each iteration more involved [Bertsekas and Tsitsiklis, 1996]. A more flexible form of multi-step operators can be obtained by considering a geometrically weighted sum of the n-step operators

$$\mathcal{T}_\pi^\lambda \doteq (1-\lambda) \sum_{n=0}^{\infty} \lambda^n \left((\mathcal{T}_\pi)^{n+1} Q \right) \qquad (\lambda \text{ Bellman operator}) \tag{2.13}$$

Naturally, Q_π remains the fixed point of $\mathcal{T}_\pi^\lambda$. Taking $\lambda = 0$ yields the usual one-step Bellman operator $\mathcal{T}_\pi$, and $\lambda \to 1$ corresponds to an infinite number of applications of $\mathcal{T}_\pi$, which in turn corresponds to the closed form solution from Eq. 2.11.

Value Iteration (VI) is a fundamental DP algorithm that updates value estimates using the Bellman optimality operator. This process converges to the optimal value function Q^*. Alg. 2 provides the pseudocode.

Algorithm 2 Value Iteration

Input: MDP $M = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$ and policy π.
Initialize $Q_0 \doteq \mathbf{0}^{|\mathcal{S}| \times |\mathcal{A}|}$.
Compute $\mathcal{P}_\pi$.
for $k = 1, 2 \dots K$ **do**
    $Q_{k+1} = \mathcal{T} Q_k \quad \{\mathcal{T} Q \doteq r + \gamma \max_{\pi \in \Pi} P_\pi Q, \forall Q \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{A}|}\}$
return $\operatorname{argmax}_{a \in \mathcal{A}} Q_{\pi_k}(s, a) \quad \forall s \in \mathcal{S}$

Policy Iteration (PI) alternates between *policy evaluation* and *policy improvement* steps as described in Alg. 3. This iterative process converges to the optimal policy π^*.

Algorithm 3 Policy Iteration

Input: MDP $M = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$ and random initial policy $\pi_0 \doteq \frac{\mathbf{1}}{|\mathcal{A}|}^{|\mathcal{S}|\times|\mathcal{A}|}$.

for $k = 1, 2 \dots K$ **do**

  Policy Evaluation: Compute $Q_{\pi_{k+1}}$ using the analytic solution in Eq. 2.11 or Value Iteration (Alg.1)

  Policy Improvement: $\pi_{k+1}(s) \in \operatorname{argmax}_{a\in\mathcal{A}} Q_{\pi_k}(s,a) \quad \forall s \in \mathcal{S}$

return π_K

A simple analysis of PI shows that it converges to the optimal policy at an *exponential (geometric, or as more commonly termed, "linear") rate*. This results follows by using the monotonicity and contraction properties of the Bellman operators. Observe that

$$Q_\pi = \mathcal{T}_\pi Q_\pi \prec \mathcal{T} Q_\pi = \mathcal{T}_{\pi'} Q_\pi \tag{2.14}$$

Inductively applying $\mathcal{T}_{\pi'}$ to each side and using the monotonicity property yields a policy improvement property,

$$Q_\pi \prec \mathcal{T}_{\pi'} Q_\pi \prec (T_{\pi'})^2 Q_\pi \prec \cdots \prec Q_{\pi'} \tag{2.15}$$

Here we used the definition that $Q_{\pi'} = \lim_{k\to\infty} \mathcal{T}_{\pi'} Q$ for any $Q \in \mathbb{R}^n$. Since $Q_\pi \prec \mathcal{T} Q_\pi \prec Q_{\pi'} \prec Q^*$ we have,

$$\|Q_{\pi'} - Q^*\|_\infty \leq \|\mathcal{T} Q_\pi - Q^*\|_\infty = \|\mathcal{T} Q_\pi - \mathcal{T} Q^*\|_\infty \tag{2.16}$$

$$\leq \gamma \|Q_\pi - Q^*\|_\infty \qquad \text{(improvement guarantee)} \tag{2.17}$$

using the contraction property. From this, we conclude that policy iteration converges to the optimal policy at a *linear rate* [1]. Let $\{\pi_k\}_{k\in\mathbb{N}}$ be the set of policies produced by policy

[1] Any sequence that is asymptotically equivalent to a convergent geometric sequence may be said to "converge geometrically" or "converge exponentially" with respect to the absolute difference from its limit,

iteration. Then iterating over Eq.2.17 shows

$$\|Q_{\pi_k} - Q^*\|_\infty \leq \gamma \|Q_{\pi_{k-1}} - Q^*\|_\infty \leq \cdots \leq \gamma^k \|Q_{\pi_0} - Q^*\|_\infty \qquad \text{(linear convergence)}$$

3 Reinforcement Learning (RL) Methods

When the environment's dynamics and reward function are unknown, agents must learn optimal behaviour through interaction. This constitutes the domain of *Reinforcement Learning (RL)* methods.

In RL, the agent interacts with the environment through the process described in Alg. 4 and depicted in Fig.2.1.

Algorithm 4 Interaction protocol: RL

1: **Input:** MDP $M = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma, \rho)$
2: Initialize $s_0 \sim \rho$
3: **for** $k = 1, 2 \ldots$ **do**
4: Choose action $A_t \in \mathcal{A}$
5: Collect feedback $R_{t+1} \sim r(S_t, A_t)$
6: Transition to next state $S_{t+1} \sim \mathcal{P}(\cdot|S_t, A_t)$

RL algorithms are categorized as:

> **Definition 1. Model-based** algorithms estimate models of $\mathcal{P}$ and r, then apply planning methods.

> **Definition 2. Model-free** algorithms directly estimate value functions or policies from experience without modelling the environment.

or it may be said to "converge linearly" relative to a logarithm of the absolute difference such as the "number of decimals of precision". The latter is standard in numerical analysis and optimization.

3.1 Learning by Monte-Carlo (MC) sampling

In RL, Monte-Carlo (MC) methods are used to estimate value functions by averaging returns from sampled trajectories, also called *rollouts* or *episodes*, of experience. MC methods provide unbiased estimates of the true value function, and are commonly applied in episodic tasks, where interactions naturally terminate, making it possible to calculate the full return for each episode.

Let $G_t^{(k)} \doteq \sum_{i=t}^{\infty} \gamma^i R_{i+1}$ be the return (Eq. 2.1) with respect to the k^{th} trajectory of experience that starts with S_t, A_t and then continues by sampling actions from π, $\{S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}, R_{t+2}, \ldots\}$, with $A_i \sim \pi(\cdot \mid S_i), \forall i > k$. The sampled return $G_t^{(k)}$ is an unbiased sample of $Q_\pi(S_t, A_t)$ By the law of large numbers, the expectation $Q_\pi(S_t, A_t)$ can be estimated iteratively by taking a mixture of the independent samples $(G_t^{(k)})_{k\in\mathbb{N}}$

$$Q_{k+1}(S_t, A_t) \doteq (1 - \alpha_k)\, Q_k(S_t, A_t) + \alpha_k G_t^{(k)}$$

where α_k is the *step-size* or *learning rate*. If the sequence of step-sizes $(\alpha_k)_{k\in\mathbb{N}}$ is appropriately decreasing, the sequence $(Q_k(S_t, A_t))_{k\in\mathbb{N}}$ converges to $Q_\pi(S_t, A_t)$ almost surely, as $k \to \infty$ [Robbins and Monro, 1951]. A minimal requirement for the step-sizes to be appropriately decreasing is formalized in the following assumption.

> **Assumption 3 (Robbins-Monro step-size condition).** The sequence of step-sizes $(\alpha_k)_{k\in\mathbb{N}}$ satisfies:
>
> $$\sum_{k=0}^{\infty} \alpha_k = \infty \quad \text{and} \quad \sum_{k=0}^{\infty} \alpha_k^2 < \infty$$

Unfortunately, the variance of the multi-step returns G_π^k involved in this computation, can be very high, increasing exponentially with the trajectory length.

3.2 Temporal-Difference (TD) Learning

Temporal-Difference (TD) Learning [Sutton, 1988] combines ideas from MC methods and DP, updating value estimates based on **bootstrapping**.

> **Definition 4 (Off-Policy vs On-Policy Learning).** Learning is said to be *off-policy* when samples generated by a *behaviour* policy μ are used to learn about a distinct *target* policy π. If $\mu = \pi$, the learning is *on-policy*.

Policy evaluation can be either on- or off-policy, while value-based control is off-policy.

3.2.1 On-Policy Learning

Given an estimate Q_k of the Q-function, and a tuple of sampled experience $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$, the TD update for the Q-function is:

$$\begin{aligned} Q_{k+1}(S_t, A_t) &\doteq (1-\alpha_k)Q_k(S_t, A_t) + \alpha_k G^{(k)}_{t:t+1} \\ &= (1-\alpha_k)Q_k(S_t, A_t) + \alpha_k(R_{t+1} + \gamma Q_k(S_{t+1}, A_{t+1})) \\ &= Q_k(S_t, A_t) + \alpha_k \underbrace{(R_{t+1} + \gamma Q_k(S_{t+1}, A_{t+1}) - Q_k(S_t, A_t))}_{\text{TD error } \delta_t} \quad \text{(SARSA(0))} \end{aligned} \tag{2.18}$$

where δ_t is the *temporal difference (TD) error*, due to the difference between the learning target $R_{t+1} + Q_k(S_{t+1}, A_{t+1})$, which is bootstrapped from $Q_k(S_{t+1}, A_{t+1})$ at a future time step $t+1$ and the estimate $Q_k(S_t, A_t)$ of the current time step t. When $Q_k(S_t, A_t)$ has converged to $Q_\pi(S_t, A_t)$, the expected value of the TD error becomes 0, $\mathbb{E}_\pi[R_{t+1} + \gamma Q_\pi(S_{t+1}, A_{t+1}) - Q_\pi(S_t, A_t)] = 0$, cf. the Bellman equation. Convergence is guaranteed if the step-sizes obey Assumption 3 and all states and actions are visited infinitely often.

When estimating values, this algorithm is known as TD(0).

Multi-Step On-Policy TD Learning To update multiple steps from the sampled trajectory, the n-step return $G_{t:t+n}$ uses $\{S_t, A_t, R_{t+1}, \dots R_{t+n+1}\}$, and is a sample of $(\mathcal{T}_\pi)^n Q$ in Eq. 2.12,

and the λ-return G^λ uses $\{S_t, A_t, R_{t+1}, \ldots\}$, and is a sample of $\mathcal{T}_\pi^\lambda Q$ in Eq. 2.13

$$G_{t:t+n} \doteq \sum_{k=0}^{n} \gamma^k R_{t+k+1} + \gamma^n Q(S_{t+n+1}, A_{t+n+1}) \qquad (n\text{-step return}) \tag{2.19}$$

$$\begin{aligned} G_t^\lambda &\doteq (1-\lambda) \sum_{n=0}^{\infty} \lambda^n G_{t:t+n+1} \\ &= R_{t+1} + \gamma[(1-\lambda)Q(S_{t+1}, A_{t+1}) + \lambda G_{t+1:}^\lambda] \qquad (\lambda\text{-return}) \end{aligned} \tag{2.20}$$

From the specification of G^λ in Eq. 2.20, the offline SARSA(λ) learning update writes

$$Q(S_t, A_t) \doteq Q(S_t, A_t) + \alpha[G_t^\lambda - Q(S_t, A_t)] \qquad (\text{Offline SARSA}(\lambda)) \tag{2.21}$$

When estimating values, this algorithm is known as TD(λ). When $\lambda = 0$ we recover the one-step TD update from Eq. 2.18, whereas $\lambda \to 1$ removes bootstrapping on the approximate Q-function, and instead relies on the MC return G. Here, λ trades off the variance of sampling multi-step returns and the bias from bootstrapping with an approximate Q-function [Kearns and Singh, 2000]. Intermediate values of λ thus usually perform best in practice.

On-Policy Eligibility Traces The update in Eq. 2.21 is sometimes called the "forward-view". Forward-view algorithms are not considered online algorithms, as they require access to n steps (SARSA(n)), or the full trajectory of samples (MC, SARSA(λ)) at every update step, the exception being SARSA(0), for which the forward and backward views are the same.

So far, we saw TD(λ) and SARSA(λ) need to be carried out offline. In the absence of a finite horizon, this is as infeasible as plain MC estimation. An online mechanism to implement the update relies on **eligibility traces** [Sutton and Barto, 2018]. Let $\mathbf{e}$ denote these traces, and consider the following update at time t

$$\mathbf{e}(s,a) \doteq \lambda_t \gamma_t \mathbf{e}(s,a) + \mathbf{1}_{\{(S_t,A_t)=(s,a)\}}, \quad \forall s \in \mathcal{S}, a \in \mathcal{A}$$

where $\mathbf{1}_{\text{predicate}}$ is an indicator function ($\mathbf{1}_{\text{predicate}} \doteq 1$ if the predicate is true, else 0).

The trace-decay parameter $\lambda_t \in [0, 1]$ interpolates between one-step TD learning and Monte-Carlo (MC) methods

$$\delta_t = R_{t+1} + \gamma Q_k(S_{t+1}, A_{t+1}) - Q_k(S_t, A_t)) \qquad \text{(TD error } \delta_t\text{)} \tag{2.22}$$

$$Q_{k+1}(s, a) \doteq Q_k(s, a) + \alpha_k \delta_t \mathbf{e}(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A} \qquad \text{(online SARSA}(\lambda)\text{)} \tag{2.23}$$

3.2.2 Off-Policy Learning

Consider the snippet of experience $S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}$ used as an input to all of the procedures above. Action A_{t+1} can be sampled from a different policy than the *target* policy π. For instance, if A_{t+1} is sampled from the greedy policy with respect to Q, then we obtain *the control problem*, and the algorithm becomes **Watkins's Q-Learning** [Watkins, 1989], which convergences to Q^*,

$$\begin{aligned} Q_{k+1}(S_t, A_t) &\doteq (1 - \alpha_k) Q_k(S_t, A_t) + \alpha_k (R_{t+1} + \gamma \max_a Q_k(S_{t+1}, a)) \\ &= Q_k(S_t, A_t) + \alpha_k \underbrace{(R_{t+1} + \gamma \max_a Q_k(S_{t+1}, a) - Q_k(S_t, A_t))}_{\text{TD error } \delta_t} \qquad \text{(Q-learning)} \end{aligned} \tag{2.24}$$

Multi-Step Off-Policy Learning The multi-step analog of SARSA(λ) for off-policy value-based control is Q(λ) [Peng and Williams, 1996]. The λ-return G^λ is a sample of $\mathcal{T}_\pi^\lambda Q$ which uses experience $\{S_t, A_t, R_{t+1}, \dots\}$ sampled from μ

$$G_t^\lambda = R_{t+1} + \gamma_{t+1}(1 - \lambda_{t+1}) \max_a Q_{\mathbf{w}}(S_{t+1}, a) + \gamma_{t+1} \lambda_{t+1} G_{t+1}^\lambda \tag{2.25}$$

Changing the specification of G^λ with the one in Eq. 2.25, the Q(λ) offline learning update has the same expression as SARSA(λ)

$$Q(S_t, A_t) \doteq Q(S_t, A_t) + \alpha[G_t^\lambda - Q(S_t, A_t)] \qquad \text{(offline Q}(\lambda)\text{)} \tag{2.26}$$

Standard Q-learning corresponds to $\lambda_t = 0, \forall t$.

Off-Policy Eligibility Traces An alternative derivation of the Q(λ) algorithm allows for online credit assignment despite the infinite horizon setting. The mechanism is based on eligibility traces. If e denotes a vector of such traces

$$\delta_t = R_{t+1} + \gamma_{t+1} \max_a Q_k(S_{t+1}, a) - Q_k(S_t, A_t) \tag{2.27}$$

$$\mathbf{e}(s, a) = \gamma_t \lambda_t \mathbf{e}(s, a) + \mathbf{1}_{(S_t, A_t)=(s,a)} \tag{2.28}$$

$$Q_{k+1}(s, a) = Q_k(s, a) + \alpha_k \delta_t \mathbf{e}(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A} \tag{2.29}$$

Exploration vs Exploitation A key challenge in RL is balancing *exploration*—gathering information about the environment, and *exploitation*—-using known information to maximize reward. Strategies like ϵ-greedy policies introduce randomness to ensure sufficient exploration by taking a random action with probability ϵ.

3.3 Function Approximation

In practical applications, the state or action spaces may be too large for exact methods or continuous. *Function approximation* techniques approximate value functions or policies using parametrized functions.

3.3.1 Parametrization

Linear Approximation The action-value function or the policy can be modelled as a linear combination of a set of features $\mathbf{x} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^m$

$$Q_{\mathbf{w}}(s,a) \doteq \mathbf{w}^\top \mathbf{x}(s,a), \quad \pi_\theta(s) \doteq \text{softmax}(\theta^\top \mathbf{x}(s,a))$$

where $\mathbf{w}$ and θ are parameters or weights, and we used the softmax transform for the policy.

Deep Reinforcement Learning (DRL) When the features are unknown, action-value functions or policies can use Neural Networks (NN) [LeCun et al., 2015] as function approximators. Commonly used are feed-forward networks with multiple hidden layers between the input and output that can capture complex non-linear relationships [Rosenblatt, 1958, Rumelhart et al., 1986], and, given enough capacity, can represent any function. DRL leverages deep learning techniques to handle high-dimensional inputs and complex function representations. While powerful, DRL methods may lack theoretical convergence guarantees [Baird, 1995, Tsitsiklis and Van Roy, 1996] and often require extensive training data and careful tuning.

3.3.2 Parameter Optimization

Convex Optimization We now introduce some concepts on convexity, which are used in the analysis throughout.

> **Definition 5 (Convex set).** A set $\mathcal{X}$ in a vector space is a *convex* set if the convex combination any two points in $\mathcal{X}$ is also in $\mathcal{X}$:
>
> $$\forall x, y \in \mathcal{X}, \forall \alpha \in [0,1], \qquad \alpha x + (1-\alpha) y \in \mathcal{X}.$$

Definition 6 (Convex function—zeroth-order condition). A function $f : \mathcal{X} \to \mathcal{X}'$, is *convex* iff the domain $\mathcal{X}$ is a convex set and $\forall x, y \in \mathcal{X}, \forall \alpha \in [0, 1]$,

$$f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha)f(y)$$

with strict inequality for **strictly convex** functions.

For a convex function f, its opposite function $-f$ is defined as a concave function.

Definition 7 (Convex function—first-order condition). A differentiable function $f : \mathcal{X} \to \mathcal{X}'$ is convex iff $\mathcal{X}$ is convex and $\forall x, y \in \mathcal{X}$

$$f(y) \geq f(x) + \nabla f(x)^\top (y - x)$$

with strict inequality for **strictly convex** functions.

Definition 8. Convex function—second-order condition A twice differentiable function $f : \mathcal{X} \to \mathcal{X}'$ is convex iff $\mathcal{X}$ is convex and $\forall x \in \mathcal{X}$ its Hessian must be positive semi-definite

$$\nabla^2 f(x) \succeq 0$$

and strictly positive semi-definite $\succ$ for **strictly convex** functions.

We will make consistent use of **Gradient Descent (GD)** as the optimization method throughout the thesis. *Gradient descent* is an iterative optimization algorithm that starts from an initial point, and moves in the direction of the *steepest descent*. This makes the algorithm especially useful for convex optimization, since for a convex function, any local minimum is also a global minimum.

Definition 9 (Gradient Descent). Starting from an initial guess x_1, the algorithm generates the sequence $x_1, x_2, \ldots, x_T \in \mathbb{R}^d$ to approach the *minimum* of a differentiable function $f : \mathcal{X} \equiv \mathbb{R}^m \to \mathbb{R}$ using the following *update rule*

$$x_{k+1} = x_k - \alpha \nabla f(x_k)$$

with α a *learning rate* or *step-size*.

If f is convex and α decays at an appropriate rate, then it is guaranteed that as $T \to \infty$, $x_T \to x^*$, where $x^* \in \operatorname{argmin}_{x \in \mathcal{X}} f(x)$ is an optimal value [Bubeck, 2015].

4 Optimization in RL

4.1 Value Function Optimization

We now present the necessary changes to the algorithms presented so far in order for them to work with function approximation. We start with **credit assignment** algorithms used for learning value functions of a given policy—*the policy evaluation setting*—after which we look at methods that adapt to maximize performance—*the control setting*.

Given experience $S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}$, the SARSA($\lambda$) update can be written as

$$\delta_t \doteq G_t^\lambda - Q_{\mathbf{w}_k}(S_t, A_t) \tag{2.30}$$

$$\mathbf{w}_{k+1} \doteq \mathbf{w}_k + \alpha_k \delta_t \nabla_{\mathbf{w}_t} Q_{\mathbf{w}_k}(S_t, A_t), \qquad \text{(Approx Offline SARSA}(\lambda)\text{)} \tag{2.31}$$

Since gradients are not backpropagated through bootstrap targets G_t^λ, these methods are not true gradient methods, and are sometimes referred to as *"semi-gradient"*.

The online SARSA(λ) counterpart writes

$$\mathbf{e}_t = \gamma_t \lambda_t \mathbf{e}_{t-1} + \nabla_{\mathbf{w}} Q_{\mathbf{w}_k}(S_t, A_t) \qquad \text{(Eligibility Trace)}$$

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \alpha_k \mathbf{e}_t \delta_t \qquad \text{(Approx Online SARSA}(\lambda)\text{)}$$

Several variations exist [e.g., Maei, 2011, Sutton et al., 2014, van Hasselt and Sutton, 2015, van Seijen and Sutton, 2014]. For clarity, we focus on the canonical variant above. Similar changes carry over from the exact setting to approximation for Q(λ) as well.

4.2 Policy Optimization

Rather than representing the optimal greedy policy implicitly through the optimal value function, *policy optimization* algorithms encode an agent's behaviour using an **explicit stochastic policy** π. Assuming an episodic setting with ρ an initial distribution over states, *policy optimization algorithms* search for an *optimal policy* by formulating **RL as an optimization problem**

$$\pi^* \in \text{argmin}_{\pi \in \Pi} -V_\pi, \text{ with } V_\pi \doteq \mathbb{E}_{S_0 \sim \rho}\left[V_\pi(S_0)\right] \tag{2.32}$$

By defining our objective with respect to an initial distribution over states—ρ, the resulting set of maximizing policies obtained by policy optimization methods in the discounted case can only be optimal for one distribution over initial states, but not for a different one. This limitation also arises in constrained MDPs [Altman, 1999], or in partially observable settings [Bertsekas, 2000, Singh et al., 1994]. In the undiscounted, average reward case, the assumption on the existence of a stationary distribution dispenses us with the dependence on the initial distribution in the resulting policies.

From this class of methods that search for an optimal policy, increased interest has been devoted to the study of **policy-gradient (PG)** methods, based on optimizing the objective V_{π_θ} over a class of *parametrised* policies $\pi_\theta \in \Pi_\theta$, with $\Pi_\theta \doteq \{\pi_\theta | \theta \in \Theta \subset \mathbb{R}^m\}$ [Agarwal

et al., 2019, Bhandari and Russo, 2019, 2021, Kakade, 2001, Konda and Borkar, 1999, Mei et al., 2020a,b, Sutton et al., 1999b, Williams, 1992].

Under an appropriate choice of policy parameterization, policy optimization methods can benefit from regularities in the policy space, even when the underlying value function space is much more complex [Bertsekas, 2000]. Moreover, in some settings, occurring inevitably when introducing function approximation, namely having imperfect knowledge about the state (partial observability), optimality can only be achieved with a stochastic policy. Stochastic gradients can be generated by Monte Carlo simulation, even in complex environments and with policies represented by deep neural networks [Schulman et al., 2015, 2017]. Direct policy search has been used in control problems in robotics [Peters and Schaal, 2006]. Policy gradient methods seek to minimize

$$\theta^* \in \operatorname{argmin}_{\theta\in\Theta} -V_{\pi_\theta}, \text{ with } V_{\pi_\theta} \doteq \mathbb{E}_{S_0\sim\rho}\left[V_{\pi_\theta}(S_0)\right] \tag{2.33}$$

Unfortunately, although PG methods can be applied to a broad class of problems, it is not always clear whether they adequately are able to solve them. A key challenge is that the objective V_π is a non-convex function of the chosen policy.

PG methods aim to update the *policy parameters* $\theta \in \Theta$ using first order optimization methods, i.e. they optimize surrogate objectives that are local approximations of the original policy performance, and correspond to the sub-problems solved by policy iteration methods. [Bhandari and Russo, 2019] identify the following properties which guarantee that (despite non-convexity) the *optimization landscape* does not suffer from spurious local optima, thereby implying convergence of policy gradient methods to globally optimal solutions.

1. **Approximate Closure:** The policy class is (approximately) *closed under policy improvement*, i.e. starting with a linear policy and performing policy iteration yields another

linear policy or that the policy iteration update can be solved in the given policy class up to a small error;

2. **No Suboptimal Stationary Points:** The sub-problem defining the (weighted) policy iteration update has *no suboptimal stationary points*, i.e. the objective is a simple convex function that can be easily optimized using first-order methods. If the latter satisfies a Polyak-Lojasiewich (PL) condition known as "gradient dominance"—a relaxation of (strong) convexity—this property is inherited by the policy gradient objective, and guarantees (fast) global convergence of first-order methods for non-convex objectives [Polyak, 1963].

In the tabular setting, the direct parametrisation associates a parameter to each state-action pair, allowing the shorthand notation $\pi \doteq \pi_\theta$. The gradient of the performance V_π with respect to the *direct representation* π of *function* pi [Agarwal et al., 2019, Bhandari and Russo, 2019, Sutton et al., 1999b]—which we call the ***"functional" gradient***, to distinguish it from the gradient $\nabla_\theta V_{\pi_\theta}$ relative to a policy parameter θ, is

$$\nabla_{\pi(s)} V_\pi = \frac{1}{(1-\gamma)} d_\pi(s) Q_\pi(s,a) \in \mathbb{R}^{|\mathcal{A}|} \tag{2.34}$$

Then, we define $\nabla V_\pi \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}$ as the concatenation of $\nabla_{\pi_s} V_\pi, \forall s \in \mathcal{S}$ (yielding a PGT for directional derivatives in Lemma 2 in Appendix 1). Using the functional gradient in combination with a particular *policy parametrization*, one may obtain the policy gradient relative to a policy parameter θ using simply the *chain rule of derivatives*

$$\nabla_\theta V_{\pi_\theta} \doteq \frac{1}{1-\gamma} \mathbb{E}_{S,A\sim d_{\pi_\theta}} \left[\nabla_\theta \log \pi_\theta(A|S) Q_{\pi_\theta}(S,A)\right] \tag{2.35}$$

Actor-Critic If the quantity Q_π is obtained by MC rollouts, then the resulting algorithm is known as REINFORCE [Williams, 1992]. MC methods typically require finite horizon settings or utilize truncated trajectories of experience. If the quantity Q_π is replaced by a bootstrapped estimate $\hat{Q}_\pi$, the resulting policy gradient becomes biased, but has lower

variance compared to MC sampling. The resulting methods were introduced as known as *neural action-critic* methods, and later became just *action-critic* (AC). Here π denotes the actor that generates the experience, and the value function estimator $\hat{Q}_\pi$ denotes the critic that assesses the performance of the actor.

4.3 Continual Reinforcement Learning (CRL)

We are interested in the setting of *continual reinforcement learning (CRL)*, characterized through the following properties:

i the agent has **a single stream of experience**, and

ii no parallel copies of the environment are possible, which roughly translates to **online learning**; the agent is allowed to use other mechanisms of persistence (e.g. parametric or non-parametric models), but these are limited in terms of memory and/or compute, excluding, e.g, the use of replay buffers storing unrealistic amounts of experience, or planners using unrealistic amounts of computational power.

In CRL, the learner ought to adapt to raw experience efficiently by reusing previous knowledge. Fast learning entails coping with change, extrapolating current events to other potential experiences that could have happened, but did not, as well as imagining what *could* happen.

3 Forethought & Hindsight in Credit Assignment

"Words create worlds. Reality is silent."

Nisargadatta Maharaj

In this chapter, we address the problem of **credit assignment** in reinforcement learning and explore fundamental questions regarding the way in which an agent can best use additional computation to propagate new information, by *planning with internal models of the world* to improve its predictions. Particularly, we work to understand the gains and peculiarities of planning employed as **forethought via forward models** or as **hindsight operating with backward models**. We establish the relative merits, limitations and complementary properties of both planning mechanisms in carefully constructed scenarios. Further, we investigate the best use of models in planning, primarily focusing on the selection of states in which predictions should be (re)-evaluated. Lastly, we discuss the issue of *model estimation* and highlight a spectrum of methods that stretch from environment dynamics predictors to planner-aware models.

1 Online Planning for Credit Assignment

Credit assignment—determining how to correctly associate delayed rewards with states or state-action pairs, is a crucial problem for *value-based reinforcement learning (RL)* agents [Sutton and Barto, 2018]. *Model-based RL (MBRL)* agents [Barto et al., 1989] gradually learn a *model of the rewards and transition dynamics* through interaction with their environments

and use the estimated model to find better policies or predictions [e.g., Abachi et al., 2020, Deisenroth et al., 2015, Farahmand, 2018, Farahmand et al., 2017, Ha and Schmidhuber, 2018, Hester and Stone, 2013, McMahan and Gordon, 2005b, Moore and Atkeson, 1993, Oh et al., 2017, Peng and Williams, 1993b, Schrittwieser et al., 2019, Silver et al., 2017b, Sutton, 1990b, Sutton et al., 2012, Wan et al., 2019]. However, the efficiency of MBRL depends on learning a useful model for its purpose. In this chapter, we focus on the use of **internal models** for value credit assignment.

Broadly, we refer to **models** as *knowledge about the internal workings of the environment*, which can be routinely re-used through *planning*. **Planning** refers to any internal processing that an agent can perform without additional experience to improve prediction and/or performance. One way of using models is by **forethought** or *trying things in your head* [Sutton and Barto, 1981], which requires learning to predict aspects of the future and planning *forward* to achieve goals. **Dyna-style planning** [Sutton, 1990b] chooses a (possibly hypothetical) state and action and predicts the resulting reward and next state, which are then used for credit assignment.

The origins of the chosen state and action are referred to as **search control strategies**. Lin [1992] proposed to use states actually experienced, and introduced the idea of replaying prior experience [Lin, 1992, Mnih et al., 2015]. Combinations of these two approaches result in **prioritized sweeping** variants [McMahan and Gordon, 2005b, Moore and Atkeson, 1993, Peng and Williams, 1993b], which generalize the idea of replaying experience in backward order [Lin, 1992] by prioritizing states based on the potential improvement in the value function estimate upon re-evaluation. From high priority states, **forward models** are used to perform additional value function updates, increasing computational efficiency [e.g., van Seijen and Sutton, 2013]. An investigation into *source-control strategies* [Pan et al., 2018] revealed the utility of additional prioritization for guiding learning towards relevant, causal or interesting states.

In this chapter, we work to understand how different phenomena caused by the structure of an environment favour the use of forward or backward planning mechanisms for credit assignment. We define **backward models** as learning to predict potential predecessors of observed states, either through explicit predictors of the environment or via **planner-aware models**, which account for how the planner performs credit assignment.

Backward models are interesting from two standpoints: (i) they can be used to causally change predictions or behaviour in hindsight, thereby naturally prioritizing states where predictions need to be (re)-evaluated; (ii) modelling errors in backward models can sometimes be less detrimental, because updating misspecified imaginary states with real experience is not as problematic as the reverse [Jafferjee et al., 2020, van Hasselt et al., 2019]. We hope that additional understanding of the mechanisms of backward planning paves the way for new, principled algorithms that use models to seamlessly integrate both forethought and hindsight [as had been the case in traditional planning methods – LaValle, 2006].

The estimation and usage of models can be done in many ways [Parr et al., 2008, van Hasselt et al., 2019, Van Seijen and Sutton, 2015, Wan et al., 2019]. The conventional approach is to learn explicit predictors of the environment which, if accurate enough, lead to good policies. However, no model is perfect. Model error is contingent on the choice of predictors and whether the true environment dynamics can be represented. *Planner-aware model learning* (PAML) suggests learning instead only aspects of the environment relevant to the way in which the model is going to be used by the planner. This class of methods [Abachi et al., 2020, D'Oro et al., 2019, Farahmand, 2018, Farahmand et al., 2017, Farquhar et al., 2017, Joseph et al., 2013, Luo et al., 2019, Oh et al., 2017, Silver et al., 2017b] incorporates knowledge about the value function or policy when learning the model. We describe a spectrum of methods for model estimation. On one end, we have environment predictors that rely on maximum likelihood estimation based on supervised learning. Towards the opposite end, constraints can be progressively relaxed by accounting how

planners use the models, ultimately leading to fully abstract models – i.e. learnable black boxes.

Contributions

- We investigate the emergent properties of planning forward and backward with learned models of the world.
- We justify the use of backward models in identifying relevant states from which to recompute prediction errors, for which we establish available design choices to be made with respect to what the model represents, how it is estimated, and how it is parametrized.
- We review these in the broader context of model estimation.
- Finally, we conduct an empirical study on illustrative prediction and control tasks, which builds intuition and provides evidence for our findings.

2 Background and Preliminaries

Setting We investigate both the **episodic, finite-horizon**, and the **continuing, infinite-horizon** setting (experience is not broken into episodes).

Value Approximation In general, the agent does not directly observe the true state of the environment s, and instead observes or constructs a feature vector $\mathbf{x}(s)$. We use a parametrized function to estimate the expected return obtained by following π, $V_{\mathbf{w}}(s) \approx V_\pi(s)$ with $\mathbf{w} \in \mathbb{R}^m$ the parameter vector. This estimate can be linear: $V_{\mathbf{w}}(s) \doteq \mathbf{w}^\top \mathbf{x}(s)$, or a non-linear arbitrary function in the general case. As a shorthand, we use $\mathbf{x}_t = \mathbf{x}(s_t)$.

Usually, the true model $\mathcal{P}$ and reward function r are not known to the agent. Instead, the agent interacts with the environment to collect samples and update the value estimate

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \underbrace{\alpha \left[Y_t - V_{\mathbf{w}_k}(S_t)\right] \nabla_{\mathbf{w}_k} V_{\mathbf{w}_k}(S_t)}_{\doteq \Delta \mathbf{w}_k} \qquad \text{(TD update)} \tag{3.1}$$

where Y_t is a *learning target*. For instance, we could use Monte Carlo returns $Y_t \doteq G_t$, or a *bootstraped* truncated return $Y_t \doteq R_{t+1} + \gamma V_{\mathbf{w}_k}(S_{t+1})$. For control, an optimal action-value function $Q_{\mathbf{w}}$ can be learned using *Q-learning* [Watkins and Dayan, 1992b] updates

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \alpha \left[R_{t+1} + \gamma \max_a Q_{\mathbf{w}_k}(S_{t+1}, a) - Q_{\mathbf{w}_k}(S_t, A_t)\right] \nabla_{\mathbf{w}_k} Q_{\mathbf{w}_k}(S_t) \quad \text{(Q-learning update)}$$

In the online setting we assume $k \doteq t$.

Internal Models Through the use of models, an agent can acquire knowledge about the world before knowing how best to use it. Models can be used to *transfer knowledge from one problem to another*, or to *compose knowledge* through **prospective planning** or **retrospective reasoning**. We refer to an agent's ability to either *plan ahead of experience*, or to *reason in hindsight of experience*, as any *internal processing* that an agent can perform without additional interaction to improve prediction and/or performance.

Models can have multiple uses, such as:

i *planning ahead to achieve goals* without needing to interact with the environment ("planning in the now"),

ii *structured exploration*,

iii *assigning credit by "planning" ahead to anticipate the future, or reasoning retrospectively about the past*.

In this chapter, we explore the last category (iii).

Determined by their usage, there are multiple choices available for **models**, related to class (*parametric* or *non-parametric*), to function (*implicit* or *explicit*), to output (*probability distributions*, *generative*, or *expectation models*), to parametrization (*linear* or *non-linear*) [Parr et al., 2008, Van Seijen and Sutton, 2015, Wan et al., 2019].

A **MBRL** agent learns an estimate $\vec{p}$ of the true model, denoted $\mathcal{P}$ and $\vec{r}$ of the reward function r, a process called **model learning**. The agent can then employ a planning algorithm, that uses additional computation without additional experience to improve its predictions and/or behaviour. Usually, $\overrightarrow{\texttt{Planner}}$ uses models that look forward in time and anticipate a future state and reward conditioned on their input. A different option is a retrospect $\overleftarrow{\texttt{Planner}}$, which uses models that look backward in time and predict a predecessor state and corresponding reward.

Conventional approaches to model learning focus on learning good predictors of the environment, and ignore how $\overrightarrow{\texttt{Planner}}$ uses the model. Limited capacity and sampling noise can lead to model approximation errors, and the model can potentially choose to pay attention to aspects of the environment irrelevant to $\overrightarrow{\texttt{Planner}}$ (e.g., trying to predict a noisy TV). To mitigate this, value-aware model learning methods [Farahmand, 2018, Farahmand et al., 2017, Silver et al., 2017b] attempt to find a model such that performing value-based planning on the value function has a similar effect to applying the true environment model. Policy-aware model learning methods [Abachi et al., 2020] similarly look at the effect of planning on the policy, rather than values. In both cases, this means the model can focus on the aspect most important for the associated planning algorithm.

3 Planning Backward in Time

As a **thought experiment**, consider a simple model that looks forward or backward for one time-step to predict the next or the previous state. An agent takes action a in s and transitions to s', experiencing a TD-error that changes the value prediction for s. To propagate this information backward to a predecessor state $\tilde{s}$ of s, forward models

can face difficulties, because finding a good predecessor $\tilde{s}$ of s is non-trivial, and model misspecifications can cause a damaging update, pushing the value prediction estimate of a real state further away from its true value.

Dyna-style planning methods [Sutton, 1990b] perform credit assignment by planning forward from previously visited states (or hypothetical states). This requires additional search-control and prioritization mechanisms. Otherwise: (i) the sampled state might be unrelated to the current state whose estimate has been recently updated; (ii) if the model is poor, planning steps can update the value of a real state with an erroneous imagined transition.

Backward models naturally sidestep these issues: (i) they can directly predict predecessor states $\tilde{s}$ of a newly updated state s; (ii) if the planning update of the imagined state $\tilde{s}$ solely uses s as target for the update, a poor model will only damage the value prediction estimate of a *fictitious* state $\tilde{s}$.

We start our analysis with a treatment of **backward planning** which, in contrast with forward planning, operates using models running **backward in time**.

We henceforth use the shorthand notation $\mathcal{P}_\pi(s_{t+1}, a_t|s_t) \doteq \mathcal{P}_\pi(S_{t+1}=s_{t+1}, A_t = a_t|S_t = s_t)$.

Assumptions Throughout the paper, we make a *stationarity assumption*: for any policy π, the Markov chain induced by π, $\mathcal{P}_\pi(s_{t+1}|s_t) = \sum_{a\in\mathcal{A}} \mathcal{P}(s_{t+1}|s_t, a)\pi(a|s_t)$, is irreducible and aperiodic. We denote by $d_{\pi,t}(s)$ the probability of observing state s at time t when following π. Under the ergodicity assumption, each policy π induces a unique stationary distribution of observed states $d_\pi(s) = \lim_{t\to\infty} d_{\pi,t}(s)$, as well as a stationary joint state-action distribution $d_\pi(s, a) = \pi(a|s)d_\pi(s)$.

3.1 Backward models

A *backward transition model* identifies predecessor states of its input state. In formalizing *backward models*, we highlight some interesting **properties** (for which we defer additional details to Appendix 1).

To begin, *backward models are tethered to a policy*. Formally, we use $\overleftarrow{\mathcal{P}}_{\pi,t}$ and $\overleftarrow{\mathcal{P}}_{\pi}$ to refer to the dynamics of the *time-reversed* Markov chain induced by a policy π at time-step t

$$\overleftarrow{\mathcal{P}}_{\pi,t}(s_t, a_t|s_{t+1}) \doteq d_{\pi,t+1}(s_{t+1})^{-1} d_{\pi,t}(s_t)\pi(a_t|s_t)\mathcal{P}(s_{t+1}|s_t, a_t)$$
$$\overleftarrow{\mathcal{P}}_{\pi}(s_t, a_t|s_{t+1}) \doteq \lim_{t\to\infty} \overleftarrow{\mathcal{P}}_{\pi,t}(s_t, a_t|s_{t+1})$$

One might hope that action-conditioning would relieve this policy dependence. Alas, it does not. Nonetheless, this may be less inconvenient than it appears, on account of one-step models being ill-fated for their lack of foresight anyway. An action-conditioned backward model for policy π is defined as:

$$\overleftarrow{\mathcal{P}}_{\pi}(s_t|s_{t+1}, a_t) \doteq \frac{\pi(a_t|s_t)}{\overleftarrow{\pi}(a_t|s_{t+1})} \frac{d_\pi(s_t)}{d_\pi(s_{t+1})} \mathcal{P}(s_{t+1}|s_t, a_t)$$

where $\overleftarrow{\pi}(a_t|s_{t+1})$ is the marginal probability of an action knowing the future state.

3.1.1 Time-extended backward models

Policy-conditioned models hold many shapes and have the potential to be useful in reasoning over larger timescales. Specifically, given a backward transition model $\overleftarrow{\mathcal{P}}_{\pi}$, we define

$$\overleftarrow{\mathcal{P}}_{\pi}^{(n)}(\cdot|s) \doteq (\overleftarrow{\mathcal{P}}_{\pi})^n(\cdot|s)$$

as the predecessor state distribution of having followed policy π for n steps, arriving at state s. Similarly, we denote the associated n-step reward model as

$$\overleftarrow{r}_{\pi}^{(n)}(\tilde{s}, s) = \mathbb{E}\left[\sum_{i=0}^{n-1} \gamma^i R_{i+1} | S_0 = \tilde{s}, S_n = s, A_{i+1} \sim \pi(\cdot|S_i)\right]$$

Other time-extended variants exist, such as λ-models [Sutton, 1995] or option models, [Precup et al., 1998] and can be learned counterfactually (see Appendix 1.2).

We primarily focus on the *policy evaluation* setting, aka the *prediction* setting, in which the goal is to evaluate a given π, and simplify notation by removing the policy subscript from models and value functions. We define a value function model $V_{\mathbf{w}} \approx V$ parametrized by the parameter vector $\mathbf{w} \in \mathbb{R}^m$. This estimate can be linear: $V_{\mathbf{w}}(s) = \mathbf{w}^\top \mathbf{x}(s)$, or a non-linear arbitrary function in the general case. As a shorthand, we use $\mathbf{x}_t = \mathbf{x}(s_t)$.

Algorithm 5 Backward Planning

Input policy π, n
$s \sim \text{env}()$
for each interaction $\{1, 2 \dots T\}$ **do**
  $a \sim \pi(s), \quad r, \gamma, s' \sim \text{env}(a)$
  $\overleftarrow{p}, \overleftarrow{r} \leftarrow \text{update_model}(s, a, r, s')$
  $V \leftarrow \text{update_value}(s, a, r, \gamma, s')$
  for each planning step $\{1, 2 \dots N\}$ **do**
    $\tilde{s} \sim \overleftarrow{p}(s), \tilde{r} \sim \overleftarrow{r}(\tilde{s}, s)$
    $V \leftarrow \overleftarrow{\texttt{Planner}}(\tilde{s}, \tilde{r}, \gamma, s)$
  $s \leftarrow s'$

Backward models are a pair of reward and transition models: $(\overleftarrow{p}, \overleftarrow{r})$ (single or multi-step, and policy-dependent). The reward model takes in two endpoint states and outputs the estimated reward. Depending on its class, the transition model can output *a distribution of predecessor states*, *a sample predecessor* or *an expectation over prior states of its input*.

3.2 Backward planning

The hindsight planning mechanism we consider uses a *backward model* to identify the predecessor states $\tilde{s}$ of a particular state s. $\overleftarrow{\texttt{Planner}}$ projects backward in time, and from the projected states, performs forward-looking TD updates that end back in s. These

corrections are used to re-evaluate the value predictions at states $\tilde{s}$. Such updates attempt to do credit assignment counterfactually by making parameter corrections in hindsight, given the new information gathered at the current step (the TD error of the transition $s \xrightarrow{a} s'$).

Forward view corrections can be reformulated as backward corrections under the backward Markov chain (see Appendix 2). For instance, an n-step learning update from any state s can be formulated as

$$\overleftarrow{\Delta}(s) \doteq \mathbb{E}\left[\left(Y^{(n)}(S_{t-n}, S_t) - V_{\mathbf{w}}(S_{t-n})\right) \nabla_{\mathbf{w}} V_{\mathbf{w}}(S_{t-n}) | S_t = s, S_{t-n} \sim \overleftarrow{p}^{(n)}(\cdot|S_t = s)\right] \quad (3.2)$$

where $Y^{(n)}(S_{t-n}, S_t) \doteq \overleftarrow{r}^{(n)}(S_{t-n}, S_t) + \gamma^n V_{\mathbf{w}}(S_t)$ is the n-step learning target. For simplicity, in the following we use single-step models, and henceforth drop the horizon reference from the notations. Algorithm 5 sketches the generic steps of hindsight planning with backward models in a full framework of simultaneous learning and planning.

Both forward and backward planning objectives for value-based methods try to find an approximation $V_{\mathbf{w}}$ to V_π by applying one step of semi-gradient model-based TD update.

3.3 Model estimation

The choice for model estimation instantiates the above-mentioned algorithmic template. The options are the following.

The most explicit way of computing $\overleftarrow{\Delta}$ is by learning *full probability distributions* – i.e. estimating the backward distribution $\overleftarrow{p}(\cdot|s)$. Then, one can either (i) sample the model $\tilde{s} \sim \overleftarrow{p}(\cdot|s)$ and do a stochastic update (or many), $\forall k \leq K$,

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \alpha \left(\overleftarrow{r}(\tilde{s}, s) + \gamma V_{\mathbf{w}_k}(s) - V_{\mathbf{w}_k}(\tilde{s})\right) \nabla_{\mathbf{w}_k} V_{\mathbf{w}_k}(\tilde{s})$$

or (ii) perform a *distributional* backward planning update $\forall \tilde{s} \in \mathcal{S}$ in proportion to the probability given by the **backward distribution model**

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \alpha \overleftarrow{p}(\tilde{s}|s) \left(\overleftarrow{r}(\tilde{s}, s) + \gamma V_{\mathbf{w}_k}(s) - V_{\mathbf{w}_k}(\tilde{s}) \right) \nabla_{\mathbf{w}_k} V_{\mathbf{w}_k}(\tilde{s})$$

In the general case, learning a full distribution model of all possible transitions is infeasible ([Sutton and Barto, 2018], Chapter 5). Alternatively, one can learn a **backward generative model**, sample predecessor features $\tilde{\mathbf{x}} \sim \overleftarrow{p}(\cdot|\mathbf{x})$ and do one or more *sample* backward planning updates.

We would like to think that maybe in the *linear* setting, where the gradient has the special form $\nabla_{\mathbf{w}} V_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}, \forall \mathbf{x} \in \mathbb{R}^m$, one can get away with learning **backward expectation models** over features, and then perform an *expected* backward planning update. We find however that a direct counterpart of the forward expectation models is not a valid update, as it involves a product of two (possibly) dependent random variables (the TD error and the gradient of the value function evaluated at the predecessor features given by the model). Using a bilinear reward model: $\tilde{r} \doteq \tilde{\mathbf{x}}^\top \theta_r \mathbf{x}$, where $\theta_r \in \mathbb{R}^{m \times m}$ is a matrix of reward parameters, we find that learning an unusual type of model still results in valid parameter corrections:

$$\mathbf{w} = \mathbf{w} + \alpha \left(\overleftarrow{r}_{\mathbf{x}}(\mathbf{x}) + \left(\gamma \overleftarrow{p}_{\mathbf{x}}(\mathbf{x}) \mathbf{x}^\top - \overleftarrow{p}_{\mathbf{x}^2}(\mathbf{x}) \right) \mathbf{w} \right),$$

where $\overleftarrow{p}_{\mathbf{x}}(\mathbf{x}) \doteq \mathbb{E}\left[\tilde{\mathbf{x}}|\mathbf{x}\right]$ is the regular expectation model, $\overleftarrow{p}_{\mathbf{x}^2}(\mathbf{x}) \doteq \mathbb{E}\left[\tilde{\mathbf{x}}\tilde{\mathbf{x}}^\top|\mathbf{x}\right]$ is a covariance matrix of the predecessor features and $\overleftarrow{r}_{\mathbf{x}}(\mathbf{x}) \doteq \mathbb{E}\left[\tilde{\mathbf{x}}\tilde{\mathbf{x}}^\top|\mathbf{x}\right] \theta_r \mathbf{x}$ is a vector reward model with parameters θ_r (see Appendix 3). Note that this model requires estimating three quantities. This formulation only conditions on the current features $\mathbf{x}$, and not directly on both $\mathbf{x}$ and the previous features $\tilde{\mathbf{x}}$.

There are several approaches to estimating $\overleftarrow{p}$, which can be characterized based on the constraints that they impose on the model. The standard approach is Maximum Likelihood

Estimation (MLE)

$$\overleftarrow{p} \leftarrow \text{argmin}_{\overleftarrow{p}^{\dagger} \in \Delta_{\mathbb{S}}} \frac{1}{n} \sum_{S_i \in \mathcal{D}_n} \log \overleftarrow{p}^{\dagger}(S_i)$$

where we used $\Delta_{\mathbb{S}}$ to denote the model space and $\mathcal{D}_n = \{(S_i, A_i, R_i, S_i')\}_{i=1}^n$ to represent collected data from interaction. Learning $\overleftarrow{p}$ that minimizes a negative-log loss or other probabilistic losses leads to a model that tries to estimate all aspects of the environment. Estimating the reward model $\overleftrightarrow{r}$ defaults to a regression problem (see Appendix 3).

If however the true model does not belong to the model estimator's space and approximation errors exist, a planner-aware method can choose a model with minimum error with respect to $\overleftarrow{\texttt{Planner}}$'s objective.

A planner-aware model-learning objective is less constrained than the MLE objective in that it only tries to ensure that replacing the true dynamics with the model is inconsequential for the internal mechanism of $\overleftarrow{\texttt{Planner}}$. In the extreme case, we note that instead of planner-aware model learning objective one can potentially directly parametrize and estimate the expected parameter corrections or updates, thus learning a fully *abstract model*. Learning of this kind shadows the internal arrow of time of the model. The ultimate unconstrained objective could meta-learn the model, such that, after a model learning update, the model would be useful for planning. We defer further investigation of such methods to future work.

4 Empirical Studies

Our empirical studies work to uncover the distinctions between planning in anticipation and in retrospect. With the aim of understanding the underlying properties of these approaches, we ask the following questions:

(i) How are the two planning algorithms distinct? When does it matter?

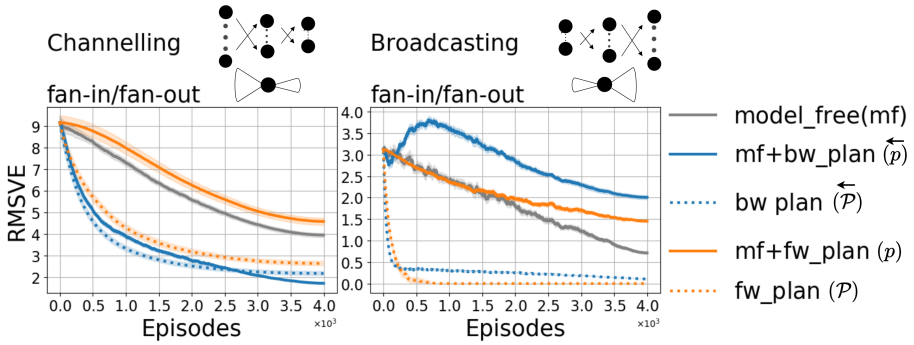


Figure 3.1: Complementary properties of forward and backward planning: Backward models work well in *channelling* structures, with large fan-in and small fan-out, while forward models are better suited for *broadcasting* state formations. y-axis shows the RMSVE: $\sqrt{(|V_\pi - V|_2^2)}$. Lower is better. Solid lines use estimated models, whereas dotted lines use true models. We used *"mf"* to denote algorithms that use *"model-free"* learning in addition to planning, as specified in the parenthesis.

To understand which structural attributes of the environment are important for the two mechanisms and how such aspects interact with planning in online prediction settings we use the following experimental setup.

Experimental setup We explore the first question in a prediction setting using Markov Reward Processes where the states are organised as bi-graphs with with one (or more) sets of states (or levels) $\{x_i\}_{i\in[0:n_x]}$ and $\{y_j\}_{j\in[0:n_y]}$, where we vary n_x and n_y in our experiments (Fig. 3.2). We additionally experiment with adding intermediary levels: $\{z_k^l\}_{k\in[0:n_z^l],l\in[1:L]}$, where L is the number of levels and n_z^l is the size of level l. The states from a particular level transition only to states in the next level, thus establishing a particular flow and stationary structure of the Markov Chain under study.

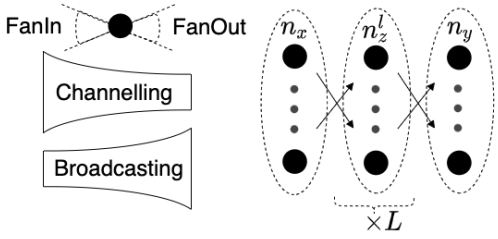


Figure 3.2: Random Chain: Illustration of the Markov Reward Process used in the prediction experiments. The chain flows from left to right.

The experiments are ablation studies of the effects of varying the *fan-in* (n_x), the *fan-out* (n_y) and the number of levels l with their corresponding sizes n_z^l. We refer to fan-in/fan-out

as the number of predecessors/successors a state might have in the state space. Results shown in the main text are averaged over 20 seeds and show the standard error over runs. See Appendix 4 for additional details.

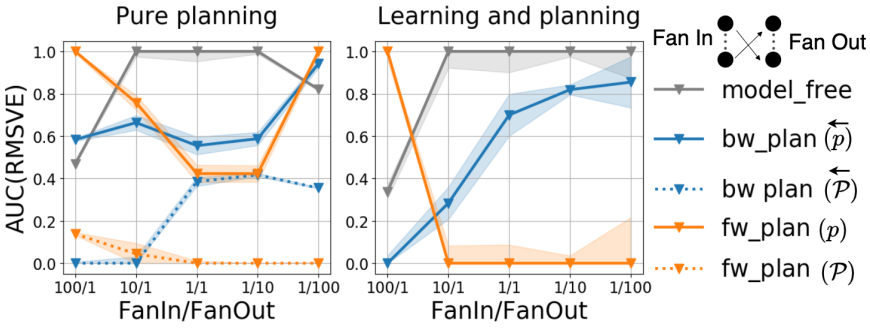


Figure 3.3: Inflection point in planning forward vs. backward: As we shift from channelling patterns (left) to broadcasting ones (right), the gain from one type of planning switches to the other, for both true and learned models. y-axis shows the area under the curve (AUC) of the RMSVE (results are normalized by zero-centering and re-scaling by the $\max - \min$). Lower is better. Solid lines use estimated models, whereas dotted lines use true models.

For this investigation we performed two types of experiments

(i) on 3-level bipartite graphs as illustrated in figure 3.1

(ii) on 2-level bipartite graphs as shown in figure 3.3. Thumbnails depicts the phenomena of transitioning from a larger number of predecessors that funnel into a smaller number of successors, and vice-versa.

For (i) for the channelling structure we use: $L = 1, n_x = 500, n_z^1 = 50, n_y = 5$ and the opposite for the broadcasting structure $L = 1, n_x = 5, n_z^1 = 50, n_y = 500$. For (ii) the results reported are for $(n_x, n_y) \in \{(500, 5), (50, 5), (5, 5), (5, 50), (5, 500)\}$, where we labelled the x-axis with the simplified ratio.

Algorithms For the purposes of this experiment we use backward planning for value prediction. We include a complete pseudo-code for the backward planning algorithm used for this experiment in Algorithm 10 in Appendix 4. We learn action-independent models because we are in the prediction setting, any rely on reward models conditioned on both the input and output states, since the environments we use require it. For any transition

$s \xrightarrow{a} s'$, we use the following reference frame for planning: (i) for backward models – the current state s' of a transition, (ii) for forward models – the previous state s of a transition. The exact definition of reference frames is given in the following question we explore, corresponding to the next experiment.

Context & Observations Our studies identify an interesting phenomenon: the gain of the two planning mechanisms is contingent on two state attributes, which we call *fan-in* and *fan-out*. We illustrate this observation in the prediction setting presented above, depicted in the diagrams of Fig 3.2 and detailed in Appendix 4.

We discover that large fan-in and small fan-out is better suited for backward planning, since backward models propagate lower-variance updates in these settings (variance is low due to small fan-out); moreover, backward planning updates many prior states at once (due to the large fan-in) – see Fig. 3.1(Left). Intuitively, when many trajectories end up with the same outcome, all prior states' values can be updated with the new information available at the current state. This pattern, which we call *channelling*, also abates in part the vulnerability of backward models in updating states in a *sample-based* way (i.e. states from which we correct predictions use a single sample, instead of the whole expectation as is the case with forward models).

In contrast, forward models fit better a *broadcasting* formation (Fig. 3.1(Right)). A backward model for this regime would be closer to factual TD and less efficient. Its predicted past states would need updates from many different successor states to construct accurate predictions. As we shift from the pattern of large fan-in/small fan-out to the opposite end, we notice a shift in the performance of the two planning mechanisms (Fig. 3.3). We observe that planning with a learned forward model is useful as we decrease the fan-in and increase the fan-out, but for large fan-out is performance will still degrade as the model becomes harder to estimate (right-end of the subplots in Fig. 3.3).

Implications These results highlight one aspect of the problem central to the success of planning: the breadth of backward and forward projections; namely, we find anticipation to be sensible when the future is wide-ranging and predictable, and hindsight to work well when new discoveries affect many prior beliefs with certainty and to a great extent. Concurrently, these insights lay the groundwork for the development of new planning algorithms that dynamically choose where to plan *to* and *from*, seamlessly blending forethought and hindsight.

(ii) Does it matter where the agent plans from? What is the effect of shifting the frame of reference used in planning?

Experimental setup In this experiment, as well as the following one, we perform ablation studies on the discrete navigation task from [Sutton and Barto, 2018] illustrated in Fig. 3.5 (Top-Right) (for details see Appendix 4).

Algorithms We operate in the control setting, for which we describe the algorithms we use, *Online Forward-Dyna* and *Online Backward-Dyna* [similar in nature to Sutton, 1990a, van Hasselt et al., 2019] in algorithms 6 and 7, respectively (details in Appendix 4). In brief, both algorithms interlace additional steps of model learning and planning in-between steps of model-free Q-learning [1]. The fact that backward models are tethered to a policy makes the control setting challenging, remaining a currently open issue. Here, we apply a naive approach in which we learn a backward model for the current policy using online experience. In effect the model tracks the implicit online greedy policy with respect to the Q-values.

Context & Observations We now ask ourselves whether the frame of reference (input state of $\overrightarrow{\texttt{Planner}}$ and $\overleftarrow{\texttt{Planner}}$, respectively), from which the agent starts planning, matters and if so, why. More precisely, consider a transition $s \xrightarrow{a} s'$ and note that we

[1] N.B. despite the tabular setting, learning is online and planning uses parametric models only in reference to the current transition. This is because we are interested in insights that transfer to more complex environments.

could use either s or s' as input to each planning algorithm. To show the effects of changing this frame of reference, we consider the control setting described at the beginning of the section and compare the action-value function variants that employ each of the planning mechanisms proposed, namely *Online Forward-Dyna* and *Online Backward-Dyna* (see Appendix 4 for details).

We compare the pure model-based setting and the full learning framework (learning & planning). In the full learning framework shown in Fig. 3.4-Left, planning backward from s implies the use of new knowledge about the current prediction, as we bootstrap on the value at s that has recently been updated: $\max_{\bar{a}} q(x, \bar{a})$; in contrast, applying planning from s' achieves a somewhat different effect: it complements model-free learning (as it bootstraps on $q(s', \cdot)$, which has not changed, but may still benefit from the reward information $r(s')$).

Algorithm 6 Online Forward-Dyna: Learning, Acting & Forward Planning

Input policy $\pi, n, \quad s \sim \text{env}()$
for each interaction $\{1, 2 \dots T\}$ **do**
  $a \sim \text{argmax}_a \, Q(s, a)$
  $r, \gamma, s' \sim \text{env}(a)$
  $\vec{p}, \overleftarrow{r}, \bar{\gamma} \leftarrow$ update_model(s, a, s')
  $Q \leftarrow$ update_value(s, a, r, γ, s')
  $s_{\text{ref}} \leftarrow$ planning_reference_state(s, s')
  for each $a \in \mathcal{A}$ **do**
    for each $s' \in \mathcal{S}$ **do**
      $Y = \overleftarrow{r}(s') + \bar{\gamma}(s') \max_{a'} Q(s', a')$
      $\Delta(s_{\text{ref}}, a) \mathrel{+}= \vec{p}(s' | s_{\text{ref}}, a)(Y - Q(s_{\text{ref}}, a))$
    $q(s_{\text{ref}}, a) \leftarrow q(s_{\text{ref}}, a) + \alpha \Delta(s_{\text{ref}}, a))$
  $s \leftarrow s'$

Algorithm 7 Online Backward-Dyna: Learning, Acting & Backward Planning

Input policy $\pi, n, \quad s \sim \text{env}()$
for each interaction $\{1, 2 \dots T\}$ **do**
  $a \sim \text{argmax}_a \, Q(s, a)$
  $r, \gamma, s' \sim \text{env}(a)$
  $\overleftarrow{p}, \overleftarrow{r} \leftarrow$ update_model(s, a, s')
  $Q \leftarrow$ update_value(s, a, r, γ, s')
  $s_{\text{ref}} \leftarrow$ planning_reference_state(s, s')
  for each $\tilde{s} \in \mathcal{S}, \tilde{a} \in \mathcal{A}$ **do**
    $Y = \overleftrightarrow{r}(s_{\text{ref}}) + \gamma \max_{\bar{a}} Q(s_{\text{ref}}, \bar{a})$
    $\overleftarrow{\Delta}(\tilde{s}, \tilde{a}) = \overleftarrow{p}(\tilde{s}, \tilde{a} | s_{\text{ref}}) \, (Y - Q(\tilde{s}, \tilde{a}))$
    $Q(\tilde{s}, \tilde{a}) \leftarrow Q(\tilde{s}, \tilde{a}) + \alpha \overleftarrow{\Delta}(\tilde{s}, \tilde{a})$
  $s \leftarrow s'$

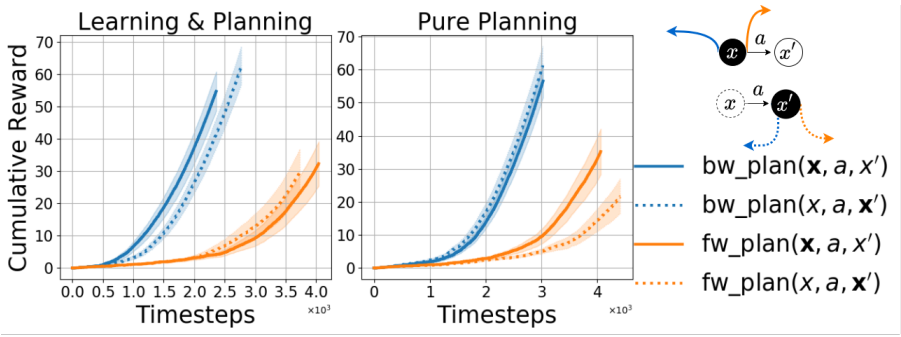


Figure 3.4: Planning frame of reference: **(Left)** In the full learning setting (learning and planning), the agent is more effective by planning backward from s and planning forward from s'. **(Right)** In the pure planning setting, both planning mechanisms assume the role of learning and and gain more by processing the exact same opposite states of the full learning case (Left), remaining in antithesis.

In contrast, forward planning gains from being as anticipatory as possible, by bootstrapping on the current state s'. The effects are reversed in the pure planning setting (Fig 3.4-Right). Specifically, the backward model cannot rely on model-free learning to re-evaluate predecessor predictions with values at s, since these are no longer changed by the learning process; it thus assumes that role. Simultaneously, backward planning is more reliable at state s' since is benefits from the current additional transition $s \xrightarrow{a} s'$. Likewise, forward planning is more reliable in s by the same argument, and also assumes the role of *learning*.

Implications These results emphasize that both planning mechanisms work best when they complement model-free learning, if it is used, and both can take on its role, if it is not.

(iii) How is planning influential on behaviour?

Experimental setup This experiment is done in the same control setting described above[2].

Context & observations Our results provide evidence for the following observations: (i) errors in the backward model, caused by stochastic transitions, are to a lesser extent damaging for credit assignment (Fig. 3.5 Top-Right); (ii) backward planning accelerates

[2]For readers familiar with standard Dyna results, we note that our setting differs from the conventional Dyna setting, as do our results. Whereas classically, tabular Dyna-agents assume access to the entire state space when deciding where to plan from, we deliberately do not make this assumption.

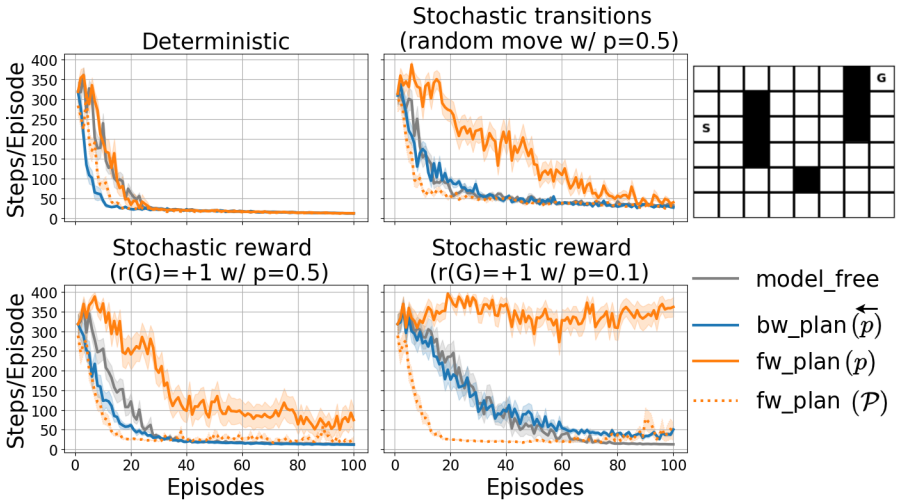


Figure 3.5: Information propagation in stochastic settings: backward models can propagate new information faster in stochastic reward settings and are more robust to randomness in the dynamics. Planning with the true forward model emphasizes the issues with forward planning (planning with the true backward model is omitted, as it depends on the constantly changing policy). Solid lines use estimated models, whereas dotted lines use true models.

the search for the optimal policy in the presence of stochastic rewards (Fig. 3.5-Bottom-Left and Bottom-Right); (iii) for extremely stochastic rewards, even backward models fail to capture the dynamics accurately enough (Fig. 3.5 Bottom-Right); (iv) large variance affects mainly forward planning.

Implications These results emphasize the potential impact, in environments with high stochasticity, of a different pattern of reasoning, related to counterfactual learning. Particularly, an agent can *project back* to a state and *rethink* its decisions after each new experience. More investigation of this idea would be useful.

5 Related Work

Backward models and planning Backward models and prioritized sweeping [McMahan and Gordon, 2005b, Moore and Atkeson, 1993, Peng and Williams, 1993b, Sutton et al., 2012, van Seijen and Sutton, 2013] have been used within the Dyna framework [Sutton, 1990a,b], mostly in combination with heuristic strategies for ordering backups. Other search control

strategies have explored the use of non-parametric models for prioritizing experience [Pan et al., 2018, Schaul et al., 2015]. The experiments of van Hasselt et al. [2019] with backward models for credit assignment have motivated our study; we worked to understand these issues in depth, extending and complementing their results. Specifically, we extend by showing backward planning is more robust in dealing with rare events (e.g. stochastic rewards) and different levels of stochasticity in the transitions. Goyal et al. [2019] also proposes predicting predecessor states on a trajectory, similarly to ideas of episodic control [Lengyel and Dayan, 2007]. However, they use imitation learning on a generative model's outputs so as to improve exploration by incentivizing the agent towards the high value states on which the model was trained; in contrast, we aim to formalize and tease apart the fundamental properties of **online** hindsight planning. The temporal value transport of Hung et al. [2019] uses an attention mechanism over an external memory to jump over irrelevant parts of trajectories and propagate credit backward in time. Ke et al. [2018a] similarly attempts to propagate credit backwards, yet it employs an RNN with learned associative connections to do so.

Counterfactual credit assignment Interestingly, research reveals that planning in the brain might also involve two processes that resemble forward and backward planning [Mattar and Daw, 2018]. Other research into credit assignment with hindsight reasoning, uses either a structural causal model to reason about the causes of events from trajectories [Buesing et al., 2019] or an inverse dynamics model conditioned on future outcomes to determine the relevance of past actions to particular outcomes [Harutyunyan et al., 2019b]. Pitis [2018] addresses the problem of credit assignment by proposing a tabular version of cumulative predecessor representations. Satija et al. [2020] use backward value functions to encode constraints for solving constrained MDPs (CMDPs) with safe policy improvements; though both our work and theirs employ some form of retrospective knowledge, *the contents, purposes and uses differ*. Schroecker et al. [2019] applies generative predecessor

models for imitation learning with policy gradient methods. Harutyunyan et al. [2019a] also connects model entropy and fan-in, albeit for the purpose of option discovery.

Planner-aware model learning Model learning objectives and strategies that ground model prediction in estimates other than environment observations have been been considered in Abachi et al. [2020], D'Oro et al. [2019], Farahmand [2018], Farahmand et al. [2017], Farquhar et al. [2017], Joseph et al. [2013], Luo et al. [2019], Oh et al. [2017], Silver et al. [2017b].

6 Discussion

Summary We explored several questions pertaining to the nature, use and attributes of planning. We provided motivation, formalism and insight for hindsight planning, emphasizing the properties of backward models. Particularly, we looked at how forward and backward planning exhibit complementary gains in opposite settings. Further, we highlighted a spectrum of model learning objectives that increasingly add more flexibility. Lastly, we performed ablation studies that reveal interesting properties about the nature of planning, their instruments – models, and the context in which they operate.

Key Takeaways

- The problem dimension of the transition dynamics, resulting from the world dynamics and the agent's policy is of great importance for the effectiveness of planning; we demonstrated the two planning mechanisms exhibit complementary properties – if the future is broad and predictable, forethought is invaluable, if backward hypotheses are causal and less determined by chance, hindsight planning is effective.
- Planning behaves differently in anticipation and in retrospect.
- The states we pick to plan from matter, and the best states to consider for forward and backward planning differ.

- Planning is complementary to model-free updates, and should be aware of these updates — for instance, the best use of planning can depend on which states are being updated model-free.
- Backward planning can be favourable over forward planning in stochastic environments by proposing a different pattern of reasoning, related to counterfactual learning resulting in efficient credit assignment.

Future Directions & Perspectives Planning is a very broad topic, and much interesting work remains to be done; we name some directions that seem most exciting to us. The interaction of planning with channelling and broadcasting patterns induces interesting questions in relation to time-extended models, for instance in terms of choosing where to plan from and to [e.g., cf. Harutyunyan et al., 2019a]. Planning backwards with time-extended models has received little attention in the literature, but appears promising. Equally, these insights pave the way for future work in developing new planning algorithms that take advantage of this property to seamlessly and optimally integrate both forethought and hindsight. Lastly, we conjecture a potential virtuous cycle of backward planning and generalization, in which the former can complement the latter, either by connecting areas of the latent space where generalization is poor or, ideally, by contributing to the construction of a better representation. Generalization, in turn, can broaden the effect of backward planning through abstraction.

Broader Impact Our work deals with fundamental insights related to the nature of model-based reinforcement learning. The problem of building models of the world and planning with them to achieve desired objectives is of paramount importance for real world applications of intelligent systems. However, in this work, we do not focus on applications, but instead look at the problem from a theoretical and investigative angle and largely treat it conceptually.

4 Salience Maps for Selective Credit Assignment

"Your assumptions are your windows on the world. Scrub them off every once in a while, or the light won't come in."

Isaac Asimov

Efficient credit assignment is essential for reinforcement learning algorithms in both prediction and control settings. We describe a unified view on temporal-difference algorithms for *selective credit assignment*. These selective algorithms weigh experience based on a salience map in order to quantify the contribution of learning updates. We present insights into applying salience maps to value-based learning and planning algorithms, and describe their role in mediating the backward credit distribution in prediction and control. Within this space, we identify some existing online learning algorithms that can assign credit selectively as special cases, as well as add new algorithms that assign credit backward in time counterfactually, allowing credit to be assigned off-trajectory and off-policy.

1 Motivation

In reinforcement learning (RL) [Sutton and Barto, 2018] an agent must assign credit or blame for the rewards it obtains to past states and actions. This problem is difficult because rewards may be sparse and may occur much later than the events that helped cause them. Moreover, the agent's observations are typically noisy or aliased, further complicating its reasoning about the root causes of observed rewards. Effective credit assignment across

long stretches of time in complex environments remains a largely unsolved and actively pursued research problem [Arjona-Medina et al., 2019, Chelu et al., 2020, Harutyunyan et al., 2019b, Hung et al., 2019, Ke et al., 2018a, Mesnard et al., 2020].

In this chapter, we describe a generic way of adding saliency in online credit assignment, leading to a unified view of the space of algorithms available. We present insights into the effect of non-uniform salience maps on the learning updates of value-based algorithms to improve credit assignment.

As an example, consider Fig. 4.1, which contains results of agents playing the Atari game of Ms.Pac-Man. The baseline performance (**brown**) is due to an agent that updates its action values for each transition using a form of Q-learning [Watkins and Dayan, 1992a] with expected eligibility traces [van Hasselt et al., 2020] (algorithm details are in later sections). We then consider a modified version of the game, where some of the observations are very noisy—the idea is that this is similar to a hardware camera on a robot that occasionally adds substantial noise, for instance due to a faulty cable. The same algorithm performs far worse on this new version of the game (**yellow** line at the bottom). However, we recover the baseline performance if we weight the updates (and appropriately modify the algorithm), despite the impoverished input signal (**turquoise** line).

The Q-learning algorithms illustrated in Fig. 4.1 use a form of *temporal difference (TD) learning* [Sutton, 1988] to learn predictions online from sampled experience by *bootstrapping* on other predictions [Sutton and Barto, 2018]. In such algorithms, credit is assigned online via an eligibility trace [Peng and Williams, 1996, Sutton, 1988, Sutton and Barto, 2018, van Hasselt and Sutton, 2015], which keeps track of how the parameters of the value function estimator should be adjusted when a prediction error occurs to appropriately correct predictions made on earlier time steps. Canonical TD algorithms use recency and frequency to determine appropriate credit assignment [Sutton, 1984]; frequent states are updated more, and preceding states that are temporally closer to a prediction error receive more blame for this error.

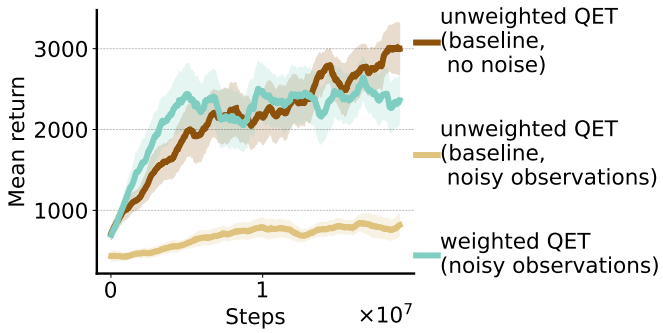


Figure 4.1: Atari Ms. Pac-Man: The curves show mean returns on Ms. Pac-Man for Q-learning with expected eligibility traces (QET), in different settings and with different salience maps. The *baseline* QET algorithm updates its values using a uniform salience map on each transition. In the *noisy* runs, Gaussian noise is added to the observations with probability $\epsilon = 0.5$. The *selective* algorithm down-weights transitions with noisy inputs, and adapts the trace parameter λ to be aware of these non-uniform salience maps. Shaded areas denote standard error over 5 seeds. These experiments demonstrate that uniform salience maps fail in the noisy setting, and performance can be recovered with more careful weighting of experience.

Our **first contribution** is an analysis of stability of arbitrary salience maps for on-policy algorithms, showing how to ensure stable convergent learning. Previous work has mainly focused on salience maps for the off-policy case, or overlooked stability, perhaps because it is not broadly acknowledged that even on-policy algorithms can diverge with a non-uniform salience map. We suggest a simple salience function that ensures convergence and discuss how this on-policy salience is connected to the idea of emphasis [Sutton et al., 2016b] which was proposed to stabilize off-policy TD.

Second, we provide examples of salience functions that greatly enhance learning, e.g., in the face of noisy observations, and show that the analysis mentioned above informs concrete algorithms with improved performance.

In RL, we have *learning* algorithms which use experiential data, and *planning* algorithms which are driven by a model. We consider the online planning algorithm suggested by van Hasselt et al. [2020], using *on-policy expected eligibility traces* (ET) in place of the instantaneous traces of standard TD, to propagate credit backward not just to the states that occur on the current trajectory but also to other *possible* trajectories leading to the current

state. This allows counterfactual credit assignment to states that *could* have happened, but did not occur recently on the current trajectory.

Our **third contribution**, is to provide insights on using selectivity with planning algorithms based on expected traces to improve stability, data efficiency and knowledge reuse. Specifically, we show how selectivity can be used to learn *off-policy expected traces*, online from a single stream of experience and, separately, how it can be used for *sparse credit assignment* which of interest for hierarchical learning. We provide concrete examples of salience functions for expected traces that enhance planning, e.g., in the face of noisy observations. As an **additional contribution**, we provide a more computationally efficient version of the value-based control planning algorithm QET [van Hasselt et al., 2020] (the planning counterpart of the learning algorithm Q(λ)), saving a factor of $|\mathcal{A}|$ — the number of actions, true for all implementations, which can be significant in practice.

2 Background and Preliminaries

Reinforcement Learning Problem Setup We consider the usual RL setting [Sutton and Barto, 2018] in which an agent interacts with an environment, modelled as a (discounted) *Markov decision process* (MDP) [Puterman, 1994] $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$, with state space $\mathcal{S}$, action space $\mathcal{A}$, state-transition distribution $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta_{\mathcal{S}}$ (where $\Delta_{\mathcal{S}}$ is the set of probability distributions on the state space and $\mathcal{P}(s'|s, a)$ denotes the probability of transitioning to state s' from s by choosing action a), reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and discount $\gamma \in (0, 1)$.

A policy π maps states to distributions over actions; $\pi(a|s)$ denotes the probability of choosing action a in state s. Let S_t, A_t, R_t denote the random variables of state, action and reward at time t, respectively.

The return at time t is defined as: $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$. The value function maps each state $s \in \mathcal{S}$ to the expected value of the return: $V_\pi(s) = \mathbb{E}\left[G_t | S_t = s, A_n \sim \pi(\cdot|S_n), \forall n \geq t\right]$. Value-based methods can be used for control by approximating the optimal action-value

function $Q^\star(s, a)$, representing the expected return when following the optimal policy, conditioned on a state-action pair.

2.1 Multi-Step Online Credit Assignment

We start with credit assignment algorithms used for learning value functions of a given policy—the policy evaluation setting—after which we look at methods that adapt to maximize performance—the control setting.

2.1.1 Learning and Planning for Evaluation

TD(λ) A popular and effective algorithm to learn $V_{\mathbf{w}} \approx V_\pi$ online and on-trajectory, is *online TD(λ)* [Sutton, 1988]:

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha_t^{\mathbf{w}} \Delta_t^V \qquad \Delta_t^V \doteq \mathbf{e}_t \delta_t \tag{4.1}$$

$$\mathbf{e}_t \doteq \gamma_t \lambda_t \mathbf{e}_{t-1} + \nabla V_{\mathbf{w}_t}(S_t) \tag{4.2}$$

where $\mathbf{w} \in \mathbb{R}^m$ are parameters of $V_{\mathbf{w}}$ to be updated, $\Delta_t^V = \mathbf{e}_t \delta_t$ is an update with *TD error* $\delta_t = R_{t+1} + \gamma_t V_{\mathbf{w}_t}(S_{t+1}) - V_{\mathbf{w}_t}(S_t)$ and *accumulating eligibility trace* $\mathbf{e}_t$, and $\alpha_t^{\mathbf{w}} \in (0, 1)$ is a (possibly time-varying) step-size parameter. For instance, $\mathbf{w}_t$ could be the weights of a neural network, or of a linear function of a feature mapping $\mathbf{x}(s)$, s.t. $V_{\mathbf{w}}(s) = \mathbf{w}^\top \mathbf{x}(s)$. The trace-decay parameter $\lambda_t \in [0, 1]$ interpolates between one-step TD learning and Monte-Carlo methods. Several variations exist [e.g., Maei, 2011, Sutton et al., 2014, van Hasselt and Sutton, 2015, van Seijen and Sutton, 2014]. For clarity, we focus on the canonical variant above.

ETD(λ) The ETD(λ) algorithm [Sutton et al., 2016b] was introduced for correcting TD algorithms when learning is off-policy, i.e. when the learning distribution differs from the sampling distribution due to a discrepancy between the behaviour and the target policy.

This solution involves weighting the trace using a history-dependent function:

$$\mathbf{e}_t \doteq \gamma_t \lambda_t \mathbf{e}_{t-1} + \rho_t M_t \nabla V_{\mathbf{w}_t}(S_t) \tag{4.3}$$
$$M_t \doteq \lambda_t i_t + (1 - \lambda_t) F_t \qquad F_t \doteq \gamma_t \rho_{t-1} F_{t-1} + i_t \,,$$

where M is the *emphasis*, F is the *follow-on trace*, i is a non-negative arbitrary *interest* function, originally introduced to focus learning [Mahmood et al., 2015, Sutton et al., 2016b], and $\rho_t = \pi(A_t|S_t)/\mu(A_t|S_t)$ is an importance sampling ratio between the target policy π and the behaviour policy μ. ETD algorithms optimize the *excursion* objective, defined as the value error under the stationary distribution of the behaviour policy d_μ. Zhang et al. [2020a] consider learning the expectation of the follow-on trace $f(s) = \mathbb{E}_\mu[F_t|S_t = s]$, and using it directly in place of the history-dependent salience function. Similarly, Jiang et al. [2021] use an expectation of the n-step follow-on trace.

ET(λ) *Expected eligibility trace* (ET) algorithms [van Hasselt et al., 2020] have been introduced for *off-trajectory*, *on-policy* value learning, replacing the instantaneous trace $\mathbf{e}_t$ in Eq. 4.1 with an estimated expectation:

$$\mathbf{z}_{\boldsymbol{\Theta}}(s) \approx \mathbb{E}_\pi \left[\mathbf{e}_t \mid S_t = s \right] \tag{4.4}$$

Expected traces can be thought of as a true expectation model, and approximations thereof can be learned by regressing on the instantaneous eligibility trace $\mathbf{e}_t$:

$$\boldsymbol{\Theta}_{t+1} \doteq \boldsymbol{\Theta}_t + \alpha_t^{\boldsymbol{\Theta}} \Delta_t^{\mathbf{z}} \tag{4.5}$$
$$\Delta_t^{\mathbf{z}} \doteq \frac{\partial \mathbf{z}_{\boldsymbol{\Theta}_t}(S_t)}{\partial \boldsymbol{\Theta}_t} \left(\mathbf{e}_t - \mathbf{z}_{\boldsymbol{\Theta}_t}(S_t) \right)$$

with step-size parameter $\alpha_t^{\boldsymbol{\Theta}}$. Expected traces can also be learned by *backward* or *time-reversed* TD, leading to multi-step updates similar to the TD(λ) version of TD (see Appendix 1.1 for details).

2.1.2 Learning and Planning for Control

For value-based control, we consider multi-step value-based analogues of TD(λ) and ET(λ). The goal is to learn action-values $Q_{\mathbf{w}}(s, a)$, rather than state-values $V_{\mathbf{w}}(s)$ we used thus far in our exposition, so that we can then derive a greedy policy $\text{argmax}_a\, Q_{\mathbf{w}}(s, a)$ with respect to those values.

Q(λ) The Q(λ) algorithm [Peng and Williams, 1996] is the analog of TD(λ) for control. With $\mathbf{w}$ now representing the parameters of the action-value function $Q_{\mathbf{w}}$, the corresponding action-value update replaces Δ_t^V in Eq. 4.1 with

$$\Delta_t^Q \doteq \left(G_t^\lambda - Q_{\mathbf{w}_t}(S_t, A_t)\right) \nabla Q_{\mathbf{w}}(S_t, A_t)$$
$$G_t^\lambda = R_{t+1} + \gamma_{t+1}(1 - \lambda_{t+1}) \max_a Q_{\mathbf{w}}(S_{t+1}, a) + \gamma_{t+1}\lambda_{t+1} G_{t+1}^\lambda$$

Standard Q-learning corresponds to $\lambda_t = 0, \forall t$. An alternative derivation of the Q(λ) algorithm (see 2.1), yields:

$$\Delta_t^Q = \mathbf{e}_{t-1} R_t^\lambda - Q_{\mathbf{w}_t}(S_t, A_t) \nabla Q_{\mathbf{w}_t}(S_t, A_t) \tag{4.6}$$
$$R_t^\lambda = R_{t+1} + \gamma_{t+1}(1 - \lambda_{t+1}) \max_a Q_{\mathbf{w}_t}(S_{t+1}, a)$$
$$\mathbf{e}_t = \gamma_t \lambda_t \mathbf{e}_{t-1} + \nabla Q_{\mathbf{w}_t}(S_t, A_t) \tag{4.7}$$

where R^λ is a *fixed-horizon one-step target*, composed of the reward at the next timestep and the bootstrapped value using a modified discount $\gamma_{t+1}(1 - \lambda_{t+1})$, $\mathbf{e}_t$ is an accumulating eligibility trace, and $\alpha_t^{\mathbf{w}}$ is a step-size parameter.

QET(λ) The QET(λ) algorithm [van Hasselt et al., 2020] is analogous to Q(λ), but using learned models of expected traces in place of the standard instantaneous traces.

3 Selective Credit Assignment

All the aforementioned algorithms apply some sort of selection mechanism over which samples are used for learning. This choice can be implicit—e.g., in TD(λ) it is determined by the behaviour distribution, or explicit—e.g., in ETD(λ).

Saliance map We take a unifying view over all the previous algorithms by separating the learning algorithm from the mechanism that re-weighs experience, which we now capture explicitly in a map defined over the state space, $\omega : \mathcal{S} \to [0, \infty)$. We mostly consider salience as a function of state; other extensions are possible, such as allowing it to be a function of history $\omega : \mathcal{H} \to [0, \infty)$, with $\mathcal{H}$ the space of histories, or a function over some feature representation.

TD(λ, ω) We call *selective TD(λ, ω)* the generic algorithm that uses a salience function for the value update, replacing the standard trace with a *selective eligibility trace* (henceforth using the $\tilde{}$ superscript to denote explicit selectivity)

$$\tilde{\mathbf{e}}_t = \gamma_t \lambda_t \tilde{\mathbf{e}}_{t-1} + \omega(S_t) \nabla V_{\mathbf{w}_t}(S_t) \qquad \text{(selective eligibility trace)} \tag{4.8}$$

Generally, the salience function controls how much credit is received by a state. For instance, if $\omega(s) = 0$, contributions in the trace from s are dropped. Hence, predictions at s are then not corrected, and are learned solely via generalization, if at all. The backward view (4.8) can equivalently be expressed in the forward view as

$$\begin{aligned} \tilde{\Delta}_t^V &= \omega_t (G_t^\lambda - V_{\mathbf{w}_t}(S_t)) \nabla V_{\mathbf{w}_t} \\ G_t^\lambda &= R_{t+1} + \gamma_{t+1}(1 - \lambda_{t+1}) V_{\mathbf{w}_t}(S_{t+1}) + \gamma_{t+1} \lambda_{t+1} G_{t+1}^\lambda \end{aligned} \tag{4.9}$$

Fig. 4.2 (Left) illustrates how salience maps can impact value learning.

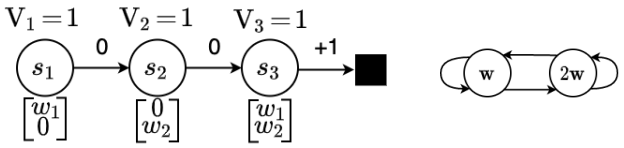


Figure 4.2: **(Left)** The value function $V_{\mathbf{w}}(s) = \mathbf{w}^\top \mathbf{x}(s)$ has two parameters $\mathbf{w} = [w_1, w_2]^\top$. The features are $\mathbf{x}(s_1) = [1, 0]^\top$, $\mathbf{x}(s_2) = [0, 1]^\top$, $\mathbf{x}(s_3) = [1, 1]^\top$. The true values are $V(s_1) = V(s_2) = V(s_3) = 1$, but cannot be represented accurately in all states at the same time. Uniformly weighting the state space can result in a trade-off, such as $w_1 = w_2 = 2/3$, s.t. $V(s_1) = V(s_2) = 2/3$ and $V(s_3) = 4/3$. When desired, weighting selectively can add flexibility, for instance by ignoring s_1 and learning the correct value function only in states s_2 and s_3 (by setting $w_1 = 0$, $w_2 = 1$). We could also reweigh states without ignoring them completely, by having low but non-zero weight on them. **(Right)** Two-state MDP example from [Sutton et al., 2016b]. Consider learning on-policy with a uniform policy and transition dynamics. There is only one feature and one parameter w.

It is well known that in the *off-policy* setting, discrepancies between the behaviour policy used to sample experience and the target policy whose value is being learned can destabilize learning, even with posterior corrections in the form of importance sampling weights [Tsitsiklis and Van Roy, 1996]. Moreover, even when off-policy TD does converge, its solution may be arbitrarily far from the optimal representable value [Kolter, 2011].

3.1 Numerical Illustration: On-Policy TD Can Diverge

Semi-gradient TD algorithms are known to converge in the *on-policy* setting, as long as experience is sampled uniformly [Sutton and Barto, 2018, Tsitsiklis and Van Roy, 1996]. However, divergence can still happen *on-policy* when not carefully considering how experience is weighted. Naively using an arbitrary salience function ω in TD(λ, ω) can lead to divergence. Consider using TD(λ, ω) on the problem in Fig. 4.2 (Right), in which we weight experience non-uniformly by putting a non-zero weight only on the first state ($\omega(s) = 1$) and using a constant trace-decay $\lambda(s) = 0, \forall s$. If $\gamma > 0.5$ and, initially, $w_0 = 1$, then w increases without bounds because we repeatedly update for the transition $w \to 2w$, while ignoring the transition $2w \to w$, causing divergence [Sutton and Barto, 2018, van Hasselt et al., 2018].

Similarly, on-policy TD($\lambda, \omega(\cdot) = 1$) with uniform state weighting and *non-uniform bootstrapping* (e.g., $\lambda(s)$ varies across states) can diverge [White, 2017]. In Fig. 4.2 (Right), this occurs if the weighting is non-zero in both states but we bootstrap only on the second state. In the next section we describe several algorithms that can address this issue.

3.2 Selectivity for Stable Off-Policy Credit Assignment

The aforementioned issues can arise when there is an imbalance between how often a state is used to compute update targets, and how often it is updated itself. Two kinds of salience maps have been proposed to correct for this imbalance—*emphasis* and *distribution ratios*. Both can be seen as instances of TD(λ, ω) for different salience functions ω. Because distribution ratios are generally difficult to estimate, we focus on emphatic weightings. We previously described the emphatic algorithm ETD(λ), which weights experience using a history-dependent salience function $\omega_t \doteq \rho_t M_t$.

A different instance of TD(λ, ω) can be obtained by learning the expectation of the instantaneous follow-on trace $f(s) \doteq \mathbb{E}_\mu[F_t | S_t = s]$, similar to Zhang et al. [2020b] and Jiang et al. [2021]

$$\omega_t = \rho_t m_t \qquad m_t \doteq \lambda_t i_t + \gamma_t (1 - \lambda_t) f(s) \qquad f(s) \doteq \mathbb{E}[F_t | S_t = s] \tag{4.10}$$

and using Eq. (4.10) in Eq. (4.8). The expected follow-on can be estimated with a function $f_\varphi \approx f$, with learnable parameters φ. Canonical learning methods for learning value functions can be applied, by reversing the direction of time in the learning update, similarly to expected eligibility traces[1], e.g., Monte-Carlo regression on the instantaneous follow-on trace, or backward TD (see Appendix 2.3). Because of its similarity to the n-step algorithm X-ETD(n) proposed by Jiang et al. [2021] we will refer to this algorithm as **X-ETD(λ)**—with X(0)-ETD(λ) denoting the variant where the expected follow-on is learned with backward

[1] Estimating a single scalar, instead of a multi-dimensional vector.

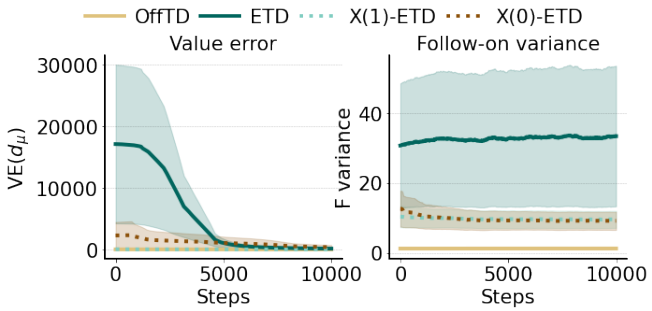


Figure 4.3: Value Error in Five-state MRP: (Left): Value error under the stationary distribution of the behaviour policy for off-policy TD(0), ETD(0), X(1)-ETD(0)—which learns emphasis by regressing on the instantaneous trace, and X(0)-ETD(0)—which learns emphasis by backward TD. Off-policy TD and X(1)-ETD visually overlap on the left, as do X(0)-ETD and X(1)-ETD on the right. The higher error of ETD is due to higher variance, caused by the follow-on trace. **(Right)**: Variance of the follow-on (taken to be 0 for off-policy TD, which does not have this trace). Details on the MRP are in Fig. B.1, Appendix 3.1 and in Sutton et al. [2016b], but do not affect this conclusion: follow-on traces typically add variance to the updates.

TD and X(1)-ETD(λ) the variant where it is learned with regression to the full Monte Carlo follow-on trace.

3.2.1 Numerical Illustration

We illustrate these policy-evaluation algorithms on the 5-state MRP introduced by Sutton et al. [2016b] (also depicted in Fig. B.1 in Appendix 3.1). In Fig. 4.3, we observe that the emphatic algorithms using expected emphasis have lower variance (see 3.1 for more details on the experimental setup).

3.3 Selectivity for Stable On-Policy Credit Assignment

We now describe salience functions which make on-policy learning stable. For constant interest i and constant discount factor γ, we can find a closed-form salience that exactly equals the expected emphasis, without needing to learn it, and thus corrects for a dynamic trace-decay λ_t, thereby avoiding divergence. This salience function is coupled with the

trace-decay λ and the discount factor γ through

$$\omega_t \doteq (1-\gamma\lambda_t)/(1-\gamma) \iff \lambda_t \doteq (\gamma\omega_t + (1-\omega_t))/\gamma \tag{4.11}$$

The constant denominator determined by γ can be folded into the learning rate.

An interesting consequence of this new insight is that we could set the salience in the accumulating trace (4.8) to guarantee convergence, or, if such a quantity is given, we can use this to pick the trace-decay parameter λ_t and the temporal discounting γ to ensure convergence.

If the discount γ_t is dynamic, the expected emphasis cannot be recovered in closed-form. We can instead use a slightly different coupling

$$\omega_t \doteq (1-\gamma_t\lambda_t)/(1-\beta_\lambda) \iff \lambda_t \doteq (1-\omega_t + \beta_\lambda\omega_t)/\gamma_t \tag{4.12}$$

which is stable and convergent $\forall\beta_\lambda \in [0,1)$ under mild conditions. Here, β_λ controls the decay rate of the on-policy follow-on trace, analogous to the one introduced by Hallak et al. [2016] (allowing smooth interpolation between TD and ETD). Appendix 2.4 contains proofs and derivations.

Q(λ,ω) Analogous to TD(λ,ω), we use Q(λ,ω) to refer to the generic algorithm that adds an explicit salience function ω to the trace of the action-value function

$$\tilde{\mathbf{e}}_t \doteq \gamma_t\lambda_t\tilde{\mathbf{e}}_{t-1} + \omega_t\nabla Q_{\mathbf{w}_t}(S_t,A_t) \tag{4.13}$$

to be used in Eq. (4.6) in place of $\mathbf{e}_t$.

3.3.1 Numerical Illustration

To illustrate the importance of this novel connection between salience and bootstrapping, we used Ms.Pac-Man, a canonical Atari game. We designed the following experimen-

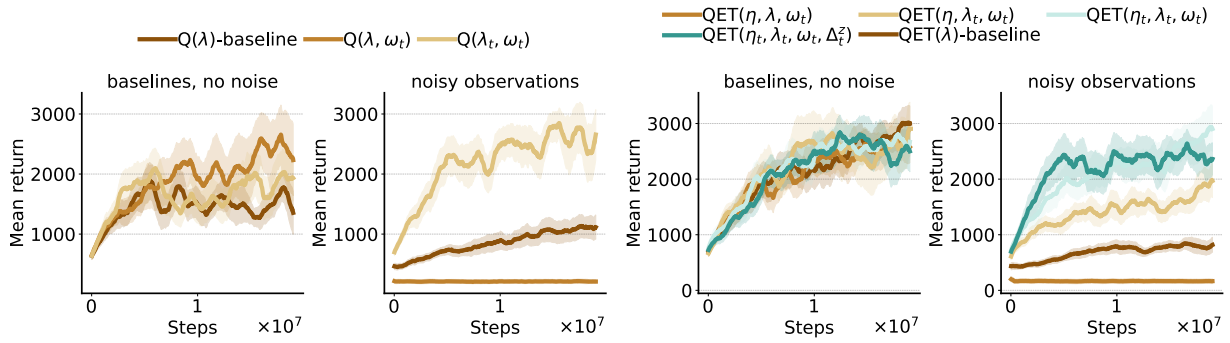


Figure 4.4: Mean Return in Atari Ms. Pac-Man: (Left): with standard observations, $\epsilon = 0$. **(Right)**: noise is added with probability $\epsilon = 0.5$. **(Top)** Algorithms using instantaneous traces: (i) baseline $Q(\lambda = 0.9)$ using uniform salience over states; (ii) $Q(\lambda = 0.9, \omega_t)$, using state-dependent salience: $\omega_t \doteq i_t$ (where $i_t = 0$ for noisy observations, and 1 otherwise); (iii) $Q(\lambda_t, \omega_t)$, additionally using state-dependent trace-decays λ_t with Eq. (4.12), and $\beta_\lambda = 0.9$. **(Bottom)** Algorithms using expected traces: (i) the baseline $QET(\eta = 0, \lambda = 0.9)$ using uniform salience over states; (ii) $QET(\eta = 0, \lambda = 0.9, \omega_t)$, using $\omega_t = i_t$; (iii) $QET(\eta = 0, \lambda_t, \omega_t)$, additionally using a state-dependent λ_t with Eq. (4.12), and $\beta_\lambda = 0.9$; (iv) $QET(\eta_t, \lambda_t, \omega_t)$, in addition applying a state-dependent trace-bootstrapping function η_t using Eq. (4.16), with $\beta_\eta = 0$; (v) $QET(\eta_t, \lambda_t, \omega_t, \tilde{\Delta}^z_t)$, also accounting for the salience in the trace learning process, using Eq. (4.17). Shaded areas show standard error over 5 seeds. These results demonstrate the importance of (1) weighting experience appropriately, and (2) coupling the salience and λ and η, as well as (3) scalability of the ideas to non-linear function approximation such as deep neural networks.

tal setup to test the selective $Q(\lambda_t, \omega_t)$ (with the "t" subscript denoting state or time-dependence), using the trace-decay correction rules in Eq. (4.11) and Eq. (4.12). With probability ϵ, the agent's observation is replaced with random Gaussian noise, to mimic a noisy observation sensor. To simulate access to a module that detects such noisy observations, we provide access to a time-dependent interest i_t, capturing whether an observation is noisy or not, s.t. $i_t = 0$ if the observation at time step t is noisy, and $i_t = 1$ otherwise. Selectivity is entirely dictated by interest, with no other corrections: $\omega_t \doteq i_t$. For the state-dependent trace-decay function λ_t, we use Eq. (4.12). In Fig. 4.4–Top-Left & Top-Right we observed that coupling the salience and the trace-decay function in Eq. (4.12) recovers the baseline's performance, displaying robustness to observation noise. The caption and 3.2 contain further details.

4 Planning selectively

We now provide insights on explicitly adding selectivity to planning algorithms. Expected eligibility traces attempt to capture all possible trajectories coalescing into a state. This allows credit to flow more broadly, not just to states that have happened, but also to states that could have happened, under the same policy. These methods can be interpreted as planning backwards, while their standard backward view counterparts—TD(λ) and Q(λ)—do not plan and only use the current sampled trajectory.

We consider adding explicit selectivity to these planning algorithms, and describe how different choices yield interesting new interpretations for these models of expected traces and their associated algorithms.

4.1 Online Learning for Off-Policy Counterfactual Models of Traces

ET(λ, ω) For evaluation, we call ET(λ, ω), the planning algorithm that estimates and uses a model of the *selective expected eligibility traces* in place of the instantaneous traces:

$$\tilde{\mathbf{z}}_{\boldsymbol{\Theta}}(s) \approx \mathbb{E}_{\pi}\left[\tilde{\mathbf{e}}_t \mid S_t = s\right]$$

with $\boldsymbol{\Theta}$—the model's parameters, and $\tilde{\mathbf{e}}$ from Eq. (4.8).

Expected traces have been described for the on-policy setting. We can instead learn the expected trace for a target policy π under a different behaviour μ by adding importance sampling ratios to the selective trace:

$$\tilde{\mathbf{e}}_t = \rho_t \gamma_t \lambda_t \tilde{\mathbf{e}}_{t-1} + \omega_t \nabla V_{\mathbf{w}_t}(S_t)$$

and choosing a salience function which adds *emphatic corrections*, ω_t as defined in equation (4.3) or (4.10).

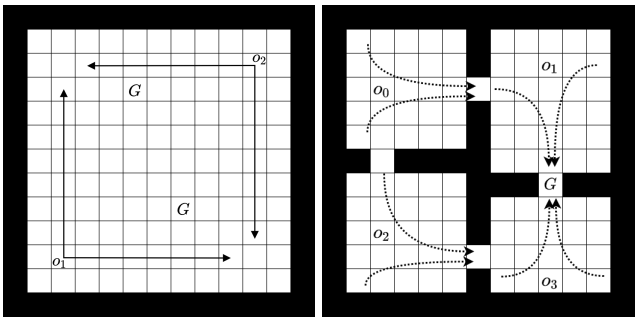


Figure 4.5: (Left) Open World. **(Right)** Four rooms domain.

QET(λ, ω) We call selective QET(λ, ω) the generic algorithm using an expectation model of the selective action-value function trace from Eq.(4.13):

$$\tilde{\mathbf{z}}_{\boldsymbol{\Theta}}(s) \approx \mathbb{E}_{\pi}[\gamma_t \lambda_t \tilde{\mathbf{e}}_{t-1} \mid S_t = s] \tag{4.14}$$

An important difference to prior algorithms is that we estimate the decayed *previous* expected trace $\gamma_t \lambda_t \tilde{\mathbf{e}}_{t-1}$ in Eq. (4.14), instead of $\tilde{\mathbf{e}}_t$. We then use $\tilde{\mathbf{z}}_{\boldsymbol{\Theta}}(s) + \nabla Q_{\mathbf{w}}(s, a)$ as the trace for action a to be used in Eq. (4.6) in place of $\tilde{\mathbf{e}}_t$. This avoids having to condition $\tilde{\mathbf{z}}_{\boldsymbol{\Theta}}(s)$ on the action a, significantly reducing computation in settings with many actions—for instance in Atari this saves a factor $|\mathcal{A}| = 18$, and in many domains $|\mathcal{A}|$ is higher. In addition we get more data per state than for each, more specific, state-action pair, thereby potentially facilitating learning the traces accurately

4.1.1 Numerical Illustration

Consider the Open World environment illustrated in Fig. 4.5-Left. The precise setup is described in 3.3. In short, the agent's behaviour μ is uniformly random and we consider learning about two stochastic policies: one that tends up and right and another that tends down and left. We consider learning two expectation models for the traces associated with those two policies, and then use those expected traces to learn to predict their values. Rewards are, noisily, obtained when bumping into one of the goals (denoted G). Bumping into a goal ends the episode; new episodes start at a random location. In Fig. 4.6, we

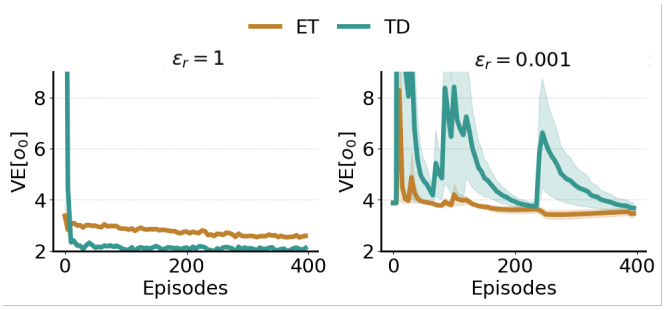


Figure 4.6: Value error in Open World: Value error for policy evaluation using TD($\lambda = 0.98$) and ET($\lambda = 0.98$), for the policy tending to the top right (similar performance for the other policy in Fig.B.2, in 3.3). With deterministic rewards ($\epsilon_r = 1$, left) TD(λ, ω) learned faster than ET(λ, ω). With stochastic rewards ($\epsilon_r = 0.001$, right), we see a slight reduction in the variance when using the off-policy expected traces. Details on hyper-parameters and tuning are in 3.3. Shaded areas show standard errors over 20 seeds.

illustrate the effect of increasing the sparsity of the reward signal, with ϵ_r indicating the probability that the agent receives a reward of $r = 10/\epsilon_r$ in each goal. The reward is zero otherwise. Fig. 4.6 shows off-policy expected traces effectively reduce variance.

4.2 Selectively Using and Learning Models of Traces

To learn expected traces, van Hasselt et al. [2020] propose a mechanism similar to value bootstrapping by introducing *mixture traces* $\mathbf{e}^\eta$ (analogous to λ-returns, with η the counterpart of λ):

$$\mathbf{e}_t^\eta = (1-\eta)\mathbf{z}_{\boldsymbol{\Theta}_t}(S_t) + \eta(\gamma_t\lambda_t\mathbf{e}_{t-1}^\eta + \nabla V_{\mathbf{w}_t}(S_t))\,. \tag{4.15}$$

The trace-bootstrapping parameter η allows us to smoothly interpolate between using expected or instantaneous traces. Using $\mathbf{e}_t^{\eta=0} = \mathbf{z}_{\boldsymbol{\Theta}_t}(S_t)$ results in counterfactual credit assignment based on expected traces, while using $\mathbf{e}_t^{\eta=1} = \gamma_t\lambda_t\mathbf{e}_{t-1}^{\eta=1} + \nabla V_{\mathbf{w}_t}(S_t)$ results in *trajectory-based* learning, relying fully on instantaneous traces. The generic expected trace algorithm ET(λ, η) is then defined by $\Delta_t^V = \delta_t\mathbf{e}_t^\eta$, and smoothly interpolates between these extremes for $\eta \in [0, 1]$.

We now consider how salience could influence using expected traces for value learning. It is reasonable to rely on expected traces more in states where they are more accurate, and more on the instantaneous traces otherwise. Assuming an explicit salience mechanism, we can constrain the trace-bootstrapping parameter η:

$$\eta_t = \beta_\eta \tilde{\omega}_t + (1 - \tilde{\omega}_t)\,, \tag{4.16}$$

where $\beta_\eta \in [0, 1)$ allows for partial trace-bootstrapping, and we used $\tilde{\omega}$ to distinguish this from the ω used in the value learning process, which could be different. For instance, consider the special case in which salience captures partial observability. The value learning could then rely on estimated expected traces more for states that are less aliased, where $\tilde{\omega}_t > 0$, and on the instantaneous eligibility trace otherwise, where $\tilde{\omega}_t = 0$.

So far, we discussed using salience for learning the value function, either through selective value updates or by adapting the mixing parameter η of the mixture trace. But the learning of the expected trace is itself subject to a sampling procedure, in which we can inject salience. We can apply the same procedure we did for value learning in Eq. (4.9):

$$\tilde{\Delta}^{\mathbf{z}}_t \doteq \tilde{\omega}_t \frac{\partial \tilde{\mathbf{z}}_{\boldsymbol{\Theta}_t}(S_t)}{\partial \boldsymbol{\Theta}_t} \left(\tilde{\mathbf{e}}_t - \mathbf{z}_{\boldsymbol{\Theta}_t}(S_t)\right) \tag{4.17}$$

and then using $\tilde{\Delta}^{\mathbf{z}}$ in place of $\Delta^{\mathbf{z}}$ in Eq. (4.5).

When the value learning uses mixture traces, the trace learning process can generally save function approximation resources by also focusing learning the expected trace only in those states in which the expected trace is used. If a mixture trace $\tilde{\mathbf{e}}^\eta$ (similar to Eq. (4.15)) is used in Eq. (4.17) in place of $\tilde{\mathbf{e}}$, we now have a multi-step trace learning process similar to the value learning process in TD(λ), so coupling the dynamic trace-bootstrapping η and the salience map $\tilde{\omega}$ analogously ensures stable learning of the expected traces.

4.2.1 Numerical Illustration

We again consider Ms.Pac-Man to illustrate the effectiveness of using *selective expected eligibility traces*, and the importance of coupling the trace-bootstrapping function η with the salience function $\tilde{\omega}$, and focusing function approximation resources when learning the model for the trace. We use the same experimental setup as before. Fig. 4.4–Bottom-Left illustrates the baseline runs of the algorithms without observational noise, whereas Fig. 4.4–Bottom-Right shows the effect of adding noise to the observations for all the algorithms. We found the algorithms using Eq.(4.16) and Eq.(4.17) to recover the original performance of the baselines, despite needing to rely on substantially noisier observations.

4.3 Sparse Expected Eligibility Traces

Interestingly, learning expectation models of selective traces (c.f. (4.8)) with *binary* salience functions $\omega : \mathcal{S} \rightarrow \{0, 1\}$, results in *sparse expected eligibility traces*. These models are equivalent to expected temporally-extended backward models [Chelu et al., 2020] (proof in 2.2).

In general, backward planning [Chelu et al., 2020, Hasselt et al., 2019, McMahan and Gordon, 2005a, Moore and Atkeson, 2004, Peng and Williams, 1993a, Sutton et al., 2008] propagates credit to events *possibly* responsible for the current outcome, typically using *explicit backward transition models*. Option models Sutton et al. [1999a] describe the long-term effects of temporally-abstract actions (options).

Sparse expected eligibility traces, used in planning algorithms, e.g., selective ET or QET, assign credit in ways akin to jumpy backward planning (cf. Chelu et al. [2020]), but without learning an explicit dynamics model. Specifically, for *binary* salience functions, the selective planning algorithm QET(λ, ω) is similar to planning with *backward option models* [Chelu et al., 2020], skipping parts of the state space as dictated by the salience map.

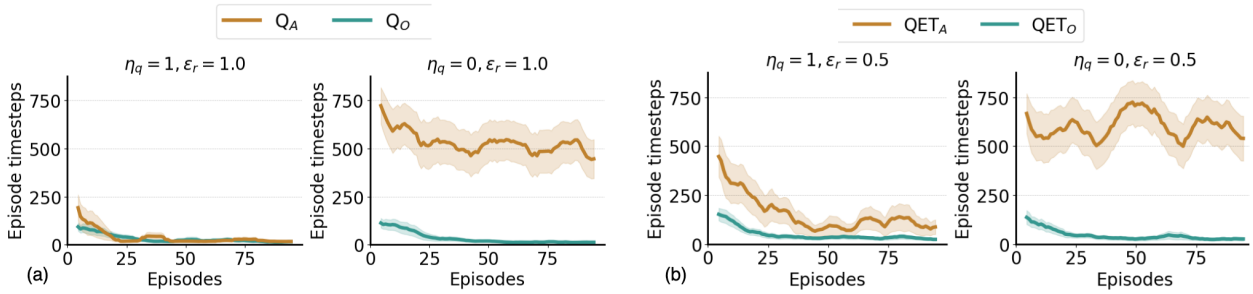


Figure 4.7: Efficiency of Online Learning & Planning with Temporal Abstraction: Number of steps per episode. **(Left)** Q_A: $Q(\lambda = 0.9)$ with dense (standard) eligibility traces over primitive actions; Q_O: $Q(\lambda_t, \omega_t)$ using a sparse eligibility trace for a policy over options (with pre-learned options as illustrated in Fig. 4.5-Right, λ_t using Eq.(4.12), $\beta_\lambda = 0.9$ and $\omega_t = 1$ in hallways, and 0 otherwise). **(Right)** QET_A: $QET(\eta = 0, \lambda = 0.9)$ using dense expected eligibility traces over primitive actions; QET_O: $QET(\eta_t, \lambda_t, \omega_t)$ using sparse expected eligibility traces for a policy over pre-learned options (with η_t using Eq.(4.16), $\beta_\eta = 0$, λ_t using Eq.(4.12), $\beta_\lambda = 0.9$, and $\omega_t = 1$ only in hallways, and 0 otherwise). **(Top)**: $r = 10$ at the goal. **(Bottom)**: sparse reward signals, $r = 20$, with probability $\epsilon_r = 0.5$, and $r = 0$ otherwise. Sparse expected traces were learned faster and coped better with sparse rewards.

Credit assignment using sparse expected traces then happens in a sub-MDP within the original one, where the state space now contains only the states captured by the binary salience map, and the action space is given by an induced option space (containing all the actions/sub-policies in-between these states). In terms of learning, this leads to a way of doing temporally-extended credit assignment in the original MDP. One can even go a step further, and apply selective temporal-discounting, similarly to Harutyunyan et al. [2019c], e.g., using different discounting parameters for the intra-option expected eligibility trace that flows credit inside an option, and the sparse expected trace that flows credit over longer time-spans using options.

4.3.1 Numerical Illustration

In the Four Rooms domain [Sutton et al., 1999a] (Fig. 4.5-Right), the agent aims to navigate to a goal location via options that take it from inside each room to its hallways, as shown in the illustration. The reward is 0 everywhere, except at the goal, where it is $10/\epsilon_r$ with probability ϵ_r, and 0 otherwise. The option policies are pre-learned, and illustrated in

Fig. 4.5-Right. Fig. 4.7 shows the results of using sparse traces for a policy over options, compared with using only primitive actions (see 3.4 for more details).

5 Discussion

We discussed the use of salience maps to stabilize, and focus resources in value-based credit assignment, introducing *selective traces*. We illustrated the importance of trace corrections even for the on-policy case, as well as the significance of adding corrections for off-policy learning. In the context of models of traces, salience can indicate when to rely on the learned models. It can also be applied to selectively learn the traces themselves.

We identified potential issues with naively combining salience maps with instantaneous and expected traces. Using these as motivation, we proposed and investigated different modifications that allow for safe, selective credit assignment, identifying sufficient conditions linking the credit assignment parameters to ensure stable learning. Our computational examples illustrate the potential benefits of adding salience maps to the credit assignment problem, and show that selective learning can substantially improve performance in some settings.

Although specific salience maps have been discussed for off-policy learning before, our definition is generic and does not restrict to a specific salience map, or to off-policy evaluation. Specifically, we consider salience maps to focus function approximation resources, and for sparse or jumpy backward planning in the control setting. Emphatic weightings are one instance of selective traces. Other, perhaps more effective, choices are possible. Finding the "best" salience function for specific problems remains an intriguing open problem for future work, and could be domain-specific.

Anand and Precup [2021] proposed a trace correction similar to the closed-form expected emphasis correction we derived, without relating it to expected emphasis, and employing it slightly differently. Particularly, they replace the trace-decay λ with a preference function

β, and allow zero-step returns rather than the typical n-step return, $n \geq 1$. Our formulation is more generic, subsuming TD, ETD, and X-ETD (expected-emphasis ETD) (3.2), and recognizing the coupling between the salience function ω and the trace-decay λ is strongly related to expected emphasis.

Future work Our examples consider the salience map given, or use emphatic weightings to stabilize learning. One direction for future work is adapting the salience map over time, based on experience. One potential approach is (meta) learning ω, λ, and/or γ Sutton and Singh [1994], White and White [2016], Xu et al. [2018], Zahavy et al. [2020], based on variance or bias minimization, or other proxy objectives Kumar et al. [2020]. Our results can be used to derive stable convergent updates, by respecting the coupling of these parameters.

A second potential approach is to use ideas for learning option terminations [Bacon et al., 2017, Harutyunyan et al., 2019a], which are akin to the credit assignment functions we considered. Inferring controllability Harutyunyan et al. [2019b] could inform salience maps over states and actions, e.g., some states are irrelevant if all actions have the same consequences, and some actions could be irrelevant if they are unlikely to be selected. Salience maps can be learned based on these intuitions. Hindsight conditioning Harutyunyan et al. [2019b] can also help infer policy-related salience maps.

Subsequent work [Zheng et al., 2021] focuses on performance and shows it is often possible to learn weight functions while learning the policy to achieve better performance than competing approaches.

5 Acceleration in Policy Optimization

"We commonly think of animals and plants as matter, but they are really systems through which matter is continually passing."

Merlin Sheldrake

In biological systems, behaviour emerges from the interaction of both fast reinforcement signals and slower modulatory processes that shape learning over time. Neuromodulators such as dopamine, serotonin, and norepinephrine are known to influence synaptic plasticity, learning rates, and motivation. In parallel, internal physiological states—including hunger, fatigue, and hormonal cycles—adjust behavioural priorities by altering attention, arousal, and responsiveness to environmental cues. Emotions themselves, influence perception, memory, and action selection, acting as slow integrators of experience that bias behavioural strategies over extended timescales. Experimental evidence in cognitive science [Bakic et al., 2014, Bennett et al., 2022, Eldar et al., 2016, Jangraw et al., 2023, Rutledge et al., 2014] suggests mood—a prolonged emotional state based on recent experience—influences decision-making processes by biasing perceptions of future outcomes. These studies suggest that mood functions as an *extrapolation* mechanism, tracking how recent experience deviates from prior expectations to inform future behaviour.

Generally, these processes unfold over longer timescales than typical reward signals and can bias behaviour in persistent and structured ways. We hypothesize that such slow dynamics can be interpreted as introducing a form of *momentum* into the learning process—shifting baselines, amplifying or dampening behavioural tendencies. Inspired by

these biological principles, this chapter explores how analogous momentum-like mechanisms can be formalized and exploited to accelerate policy optimization in reinforcement learning.

In this chapter, we explore adding **momentum** to the policy dynamics of a Reinforcement Learning (RL) agent, effectively incorporating the influence of recent policy changes into future updates. Specifically, we focus on *Actor-Critics* [Sutton and Barto, 2018], and propose a momentum-based update within the *Policy Mirror Descent (PMD)* framework [Agarwal et al., 2019, Bhandari and Russo, 2021, Johnson et al., 2023, Vaswani et al., 2021, Xiao, 2022].

Policy Mirror Descent (PMD) is a general family of algorithms, specified by the choice of mirror map covering a wide range of novel and fundamental methods in RL. PMD is a proximal algorithm [Parikh et al., 2014] and an instance of **Mirror Descent (MD)** [Beck and Teboulle, 2003] on the policy simplex [Bhandari and Russo, 2021]. MD is an iterative optimization algorithm that extends *gradient descent (GD)* by adapting to the geometry of the problem using different distance functions, particularly *Bregman divergences* [Amari, 1998, Banerjee et al., 2005, Bubeck, 2015]. PMD applies a *proximal regularization*, with a Bregman divergence, to the improvement step of *Policy Iteration (PI)*, and converges to it as regularization decreases. In the discounted setting, with an *adaptive step-size*, PMD converges linearly at a rate determined by the discount factor, independent of the dimension of the state space or problem instance [Johnson et al., 2023], recovering classical approaches, like PI and VI, as special cases. PMD has been extended to *linear approximation* by Yuan et al. [2023] and to *general function approximation* by Alfano et al. [2024].

Motivation The running time of PMD algorithms scales with the number of iterations, and with a parametrized policy class, each iteration of an approximate PMD method generally requires multiple "inner-loop" updates to the policy parameter [Tomar et al., 2020, Vaswani et al., 2021]. Actor-critic (AC) methods [Konda and Borkar, 1999, Sutton et al., 1999b] additionally require the computation or updating of an inexact critic at each

iteration. It is therefore desirable to design algorithms which converge in a smaller number of iterations, resulting in significant empirical speedups, as has been previously argued by Goyal and Grand-Clement [2021], Johnson et al. [2023], Russo [2022], Xiao [2022].

An agent can prospectively search for an optimal policy by leveraging consistent trends in policy improvement. For policies represented in functional space, momentum ensures that the policy update considers not just the standard policy improvement at each iteration, but also the cumulative effect of prior policy adjustments. This implies a bias toward solutions that align with the long-term trajectory of functional optimization, rather than the per-iteration policy update. Momentum in this context would amplify consistent trends observed over previous updates, making the learning process more efficient. This "momentum" effect can propel the solution toward convergence more quickly (Fig. 5.1).

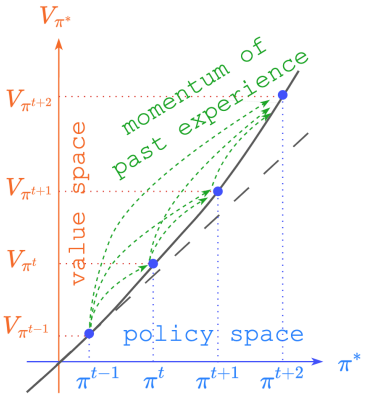


Figure 5.1: Extrapolation

In this chapter, we leverage duality and acceleration to build a novel surrogate objective for momentum-based PMD, leading to faster learning in terms of less iterations necessary to converge. The novelty of our approach is the application of acceleration mechanics to the *direct* or *functional policy representation* π—hence named ***functional acceleration***, as opposed to classic acceleration applied to the policy parameter θ (e.g., Hessel et al. [2017], Mnih et al. [2016], Schulman et al. [2017] use Adam [Kingma and Ba, 2015] or RMSProp [Hinton et al., 2012]). Specifically, we use momentum in the dual policy space to accelerate on "long ravines" or decelerate at "sharp curvatures" at the functional level of the policy optimization objective.

Intuitively, adding momentum to the functional PG (the gradient of the policy performance objective with respect to the direct policy representation π) means applying, to the current directional policy derivative, a weighted version of the previous policy ascent

direction, encouraging the method to adaptively accelerate according to the geometry of the optimization problem.

Contributions

- We illustrate and analyze theoretically the impact of applying functional acceleration on the optimization dynamics of PMD, leading to a practical momentum-based PMD algorithm.
- We characterize the properties of the problem setting, and those intrinsic to the algorithm, for which applying functional acceleration is conducive to faster learning.
- We study the influence of an inexact critic on the acceleration mechanism proposed.

Outline This chapter is organized as follows. After placing our work in existing literature in Sec. 4, and setting up the context in which it operates in Sec. 1, we introduce our main ideas in Sec. 2. We complement with numerical studies in Sec. 3, ending with a discussion in Sec. 5.

1 Background and Preliminaries

One way to design and analyse iterative optimization methods is through the lens of Online Convex Optimization (OCO).

1.1 Online Convex Optimization

In this chapter, we interpret Policy Optimization as *Online Learning* [Hazan, 2017].

> **Definition 1 (Online Learning (Iterative Optimization)).** At iteration t, the learner plays a decision from a convex decision set $\mathcal{X} \subset \mathcal{H}$, given directionally differentiable functions $f_1, f_2, \ldots, f_T : \mathcal{X} \to \mathbb{R}$. The learner algorithm selects an initial point $x_1 \in \mathcal{X}$. For each iteration $t = 1, 2, \ldots, T$, the algorithm observes feedback $g_t \in \mathcal{H}$ and selects the next point $x_{t+1} \in \mathcal{X}$.

Definition 2 (Regret). The goal is to minimize the *regret* after T iterations, against any comparator $x^* \in \mathcal{X}$, defined as $\mathrm{Reg}_T(x^*) \equiv \sum_{t=0}^{t-1} (f_t(x_t) - f_t(x^*))$

1.1.1 Mirror Descent (MD)

In this chapter, the methods and analysis employed rely on the method of Nemirovsky et al. [1983], known as Mirror Descent (MD), which allows to find the minimum of functions defined over Banach spaces[1] $\mathcal{X}$, when norms used to measure various quantities of interest does not derive from an inner product, and the gradients (Frechét derivatives [2]) are elements of the dual space $\mathcal{X}^*$, making the standard gradient computation $x - \eta \nabla f(x)$ nonsensical.

The insight that lead to the discovery of MD is that one can still carry out the gradient computation in the dual space by first mapping the previous iterate to the dual space, compute the update, and them map the resulting output back to the primal space. MD is thus a general gradient descent algorithm, applicable to constrained spaces $\mathcal{C}$, which relies on *Fenchel conjugate duality* to map the iterates of an optimization problem $x^* = \mathrm{argmin}_{x \in \mathcal{X} \cap \mathcal{C}} f(X)$, back and forth between a primal $\mathcal{X}$ and a dual space $\mathcal{X}^*$.

The algorithm uses a *Bregman Divergence* to define an equivalent notion of distances beyond the Euclidean space.

[1] A Banach space is a complete normed vector space, i.e. it is equipped with a norm to measure vector lengths and is complete, ensuring every Cauchy sequence converges within the space. A Cauchy sequence is a sequence where the distance between its terms becomes arbitrarily small as the sequence progresses, ensuring convergence in a complete space.

[2] The Frechét derivative generalizes the concept of a derivative to Banach spaces, providing the best linear approximation of a function at a point using a bounded linear operator.

Definition 3. Bregman Divergence [Amari, 1998, Banerjee et al., 2005, Bubeck, 2015] Let $h : \mathcal{X} \to \mathcal{X}'$ be a strictly convex function, differentiable on the relative interior (rint) of $\mathcal{X}$, assumed to be non-empty, and called *mirror map*. The Bregman Divergence induced by the function h is $D_h : \mathcal{X} \times \text{rint}(\mathcal{X}) \to [0, \infty)$ is defined as

$$D_h(x, y) \doteq h(x) - h(y) - \langle \nabla h(y), x - y \rangle$$

Furthermore, cf. Amari [2016], the divergences derived from the two convex functions are substantially the same, except for the order of the arguments

$$D_h(x, \nabla h^*(\nabla h(y))) = D_{h^*}(\nabla h(y), \nabla h^*(x)) \qquad \text{(duality of Bregman Divergences)}$$

Squared Euclidean distance is perhaps the simplest and most widely used Bregman divergence. The underlying function $h(x) = \langle x, x \rangle$ is strictly convex, differentiable on $\mathbb{R}^m$ and $D_h(x, y) = \|x - y\|^2$. Another widely used Bregman divergence is the KL-divergence. If p is a discrete probability distribution so that $\sum_{i=1}^m p_i = 1$, the negative entropy $h(p) = \sum_{i=1}^m p_i \log p_i$ is a convex function and the corresponding Bregman divergence between two distributions p, q is the KL-divergence $D_h(p, q) = KL(p, q)$.

The *MD algorithm* uses a convex function of the Legendre-type [3], called a mirror map h, to map the MD iterates x to the dual space where the gradient update is performed

$$\nabla h(y) \doteq \nabla h(x) - \eta \nabla f(x) \qquad \text{(dual space gradient update)}$$

with η a step size. A new iterate satisfying the primal constraints $\mathcal{C}$ is obtained using a Bregman projection

$$x' \doteq \text{proj}^h_{\mathcal{C}}(y) \doteq \text{argmin}_{x \in \mathcal{C}} D_h(x, \nabla h^*(\nabla h(y))) \qquad \text{(Bregman projection)}$$

[3] We require h to be strictly convex and essentially smooth (differentiable and $\|\nabla h(x_t)\| \to \infty$ for any sequence x_t converging to a point on the boundary of $\text{dom}\, h$) on the relative interior (rint) of $\text{dom}\, h$.

of the updated dual iterate $\nabla h(y)$ mapped back in the primal space using the conjugate function of h

$$h^*(x^*) = \sup_{x \in \mathcal{X}} \langle x, x^* \rangle - h(x) \qquad \text{(conjugate function of } h\text{)}$$

This projection relies on a Bregman divergence. The proximal formulation of MD (see Lemma 3 in Appendix 1.1) merges the update and projection steps of MD to

$$x' \doteq \operatorname{argmin}_{\bar{x} \in \mathcal{X} \cap \mathcal{C}} \eta \langle \nabla f(x), \bar{x} \rangle + D_h(\bar{x}, x) \qquad \text{(proximal perspective of MD)}$$

1.1.2 Acceleration Methods in Optimization

In convex optimization, **optimism** is a principle applied in online and iterative optimization algorithms, where the update at each step is based not only on the current gradient information, but also on an *optimistic estimate* of the gradient in future iterations. This approach can help reduce the regret and improve convergence rates in online optimization settings.

The goal of **Optimistic Online Learning** algorithms [Rakhlin and Sridharan, 2013a,b] is obtain better performance, and thus guaranteed lower regret, when playing against "easy" (i.e., predictable) sequences of online learning problems, where past information can be leveraged to improve on the decision at each iteration.

Predictability An important property of the above online learning problems is that they are not completely adversarial. In RL, the policy's true performance objective cannot be truly adversarial, as the same dynamics and cost functions are used across different iterations.

See Appendix 4 for an overview of acceleration methods in optimization.

1.2 Policy Optimization as Online Learning

We study policy optimization through the lens of *online learning* [Hazan, 2017], by treating a policy optimization algorithm as the learner in online learning and each intermediate policy that it produces as an online decision. The iterative process of policy optimization into a standard online learning setup is described in Alg. 8. At iteration t, the learner plays a decision $\pi_t \in \Pi$, a per-iteration loss ℓ_t is then selected and some feedback information of ℓ_t is revealed to the learner for making the next decision. The iterations considered can be the episodes of interaction with the MDP, not the timesteps in the MDP.

Algorithm 8 Policy Optimization as Online Learning

1: **Input:** MDP $M = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$ and random initial policy $\pi_0 \doteq \frac{\mathbf{1}}{|\mathcal{A}|}^{|\mathcal{S}|\times|\mathcal{A}|}$.
2: **for** $t = 1, 2 \dots T$ **do**
3: Output π_t, receive feedback ℓ_t
4: $\pi_{t+1} \in \operatorname{argmin}_{\pi\in\Pi} \ell_t(\pi)$
5: **return** π_T

Notation We use the shorthand notation $\langle \cdot, \cdot \rangle$—the dot product, $\nabla f(x) \doteq \nabla_x f(x)$—gradients and partial derivatives, $\nabla f(x, y) \doteq \nabla_x f(x, y)$, $\pi_t \doteq \pi_{\theta_t}$, $Q_t \doteq Q_{\pi_t}$, $V_t \doteq V_{\pi_t}$, $d_t \doteq d_{\pi_t}$.

Policy Mirror Descent (PMD) is an instance of MD [Beck and Teboulle, 2003], applying GD in a non-Euclidean geometry, using the proximal perspective of MD,

$$\pi_{t+1} \doteq \operatorname{argmin}_{\pi\in\Pi} -\langle \nabla V_t^\rho, \pi \rangle + \eta_{\pi_{t+1}}^{-1} D_h(\pi, \pi_t)$$

for some sequence of step-sizes $\eta_{\pi_{t+1}} > 0$ and initial policy π_0. The visitation-distribution d_t^ρ in the gradient of the surrogate objective can lead to vanishing gradients in infrequently visited states under π_t [Bhandari and Russo, 2021, Johnson et al., 2023, Mei et al., 2020a],

so PMD iteratively applies a variant that separates the objective per state

$$\pi_{t+1}(s) \doteq \operatorname{argmin}_{\pi(s)\in\Delta(\mathcal{A})} -\langle Q_t(s), \pi(s)\rangle + \eta_{\pi_{t+1}}^{-1} D_h(\pi(s), \pi_t(s)) \qquad \text{(PMD update)}$$

Inexact PMD replaces Q with a value estimator $\widehat{Q}$.

Using the negative Boltzmann-Shannon entropy [Shannon, 1948] as mirror map yields the Natural Policy Gradient (NPG) [Kakade, 2001]. With a null Bregman divergence, it recovers PI.

Approximate PMD The standard PMD algorithm is adapted by Tomar et al. [2020] to general policy parametrizations π_θ, by updating the parameters θ using a surrogate composite objective

$$\theta_{t+1} = \operatorname{argmin}_{\theta\in\Theta} \mathbb{E}_{s\sim d_t^\rho}[-\mathbb{E}_{a\sim\pi_\theta(s)}[Q_t(s,a)] + \eta_{\pi_{t+1}}^{-1} D_h(\pi_\theta(s), \pi_t(s))] \quad \text{(approximate PMD update)}$$

Alfano et al. [2024] introduces the concept of Bregman policy class

$$\{\pi_\theta : \pi_\theta(s) = \operatorname{proj}^h_{\Delta(\mathcal{A})}(\nabla h^*(f_\theta(s))), s \in \mathcal{S}\} \qquad \text{(Bregman policy class)}$$

and uses a parametrized function f_θ to approximate the dual update of MD

$$f_{t+1}(s) \doteq \nabla h(\pi_t(s)) - \eta_{\pi_{t+1}} \widehat{Q}_t(s) \qquad \text{(dual approximation)}$$

To satisfy the simplex constraint, [Xiong et al., 2024] uses a Bregman projection on the dual approximation mapped back to the policy space

$$\begin{aligned} &\pi_\theta(s) = \operatorname{proj}^h_{\Delta(\mathcal{A})}(\nabla h^*(f_{t+1}(s))) \\ \implies\ &\theta_{t+1} = \operatorname{argmin}_{\theta\in\Theta} D_h(\pi_\theta(s), \nabla h^*(f_{t+1}(s))) \qquad \text{(primal approximation)} \end{aligned}$$

Using the negative Boltzmann-Shannon entropy, yields the softmax policy class

$$\pi_\theta(s,a) \doteq \frac{\exp f_\theta(s,a)}{\|\exp f_\theta(s)\|_1}, \forall s,a \in \mathcal{S}\times\mathcal{A} \qquad \text{(softmax policy)}$$

For the approximate setting, we rely on the proximal formulation by Tomar et al. [2020]. Additional details on other are provided in Appendix 5.

2 Functional Acceleration for Policy Mirror Descent

Extrapolation from the future Consider first an idealized update, denoted **PMD(+loo)**, anticipating one iteration ahead on the optimization path. Given π_t and Q_t, we define the expected lookahead return of acting greedily for one iteration, and following π_t thereafter, $\forall s \in \mathcal{S}, a \in \mathcal{A}$

$$\mathcal{T}_{\mu_t} Q_t(s,a) \doteq \mathbb{E}_{s'\sim P(\cdot|s,a)}[r(s,a) + \gamma \mathbb{E}_{a'\sim\mu_t(s')}[Q_t(s',a')]], \text{ where } \mu_t(s) \doteq \text{greedy}\, Q_t(s) \quad (5.1)$$

Next, $\forall s \in \mathcal{S}$ we define the proximal update

$$\pi_{t+1}(s) \doteq \text{argmin}_{\pi(s)\in\Delta(\mathcal{A})} -\langle (\mathcal{T}_{\mu_t} Q_t)(s), \pi(s)\rangle + \eta^{-1}_{\pi_{t+1}} D_h(\pi(s), \pi_t(s)) \quad (5.2)$$

where $\eta_{\pi_{t+1}}$ is a step size. Thm. 1 indicates that replacing the PMD update with the two updates of PMD(+loo) (Eq. 5.1 & Eq. 5.2) changes the convergence rate of the algorithm, featuring γ^2 in place of γ for PMD[4].

Theorem 1. *(Functional acceleration with exact PMD(+loo)) The policy iterates π_{t+1} of* PMD(+loo) *satisfy $\|V^* - V_t\|_\infty \leq (\gamma^2)^t(\|V^* - V_0\|_\infty + \sum_{i\leq t}(\gamma^2)^i \epsilon^{-1}_{\pi_{i+1}})$, if $\eta_{\pi_{t+1}}$ is an adaptive step-size such that $\eta^{-1}_{\pi_{t+1}} D_h(\text{greedy}(\mathcal{T}_{\mu_t} Q_t)(s), \pi_t(s)) \leq \epsilon_{\pi_{t+1}}$, $\forall \epsilon_{\pi_{t+1}}$ arbitrarily small.*

(Proof. in Thm. 14, Appendix 1.2)

[4] γ^2 corresponds to the one-step lookahead horizon $H=1$, and generalization to multi-step lookahead would yield γ^{H+1}

It is known that the hard greedification operator in the update of μ_t can yield unstable updates if we replace Q_t with a value estimator $\widehat{Q}_t$ [Russo, 2022] (see also Sec.3.3, Fig. 5.4). Consequently, we relax the update of PMD(+loo) by replacing the hard greedification in Eq. 5.1 with a PMD update

$$\mu_t(s) \doteq \text{argmin}_{\pi(s)\in\Delta(\mathcal{A})} -\langle Q_t(s), \pi(s)\rangle + \eta_{\mu_t}^{-1} D_h(\pi(s), \pi_t(s)) \tag{5.3}$$

where $\eta_{\mu_t}^{-1}$ is a step-size. We refer to Eq. 5.3 & 5.2 as **PMD(+ext)** (cf. 'extragradient" or extrapolated gradient method of Nemirovski [2004]). Thm. 2 confirms acceleration is maintained.

Theorem 2. *(Functional acceleration with PMD(+ext)) The policy iterates π_{t+1} of PMD(+ext) satisfy $\|V^* - V_t\|_\infty \leq (\gamma^2)^t(\|V^* - V_0\|_\infty + \sum_{i\leq t} {}^{(\epsilon_{\pi_{t+1}}+\gamma\epsilon_{\mu_t})}/_{(\gamma^2)^i})$, if $\eta_{\mu_t}, \eta_{\pi_{t+1}}$ are adaptive step-sizes such that $\eta_{\mu_t}^{-1} D_h(\text{greedy}(Q_t(s)), \pi_t(s)) \leq \epsilon_{\mu_t}$, and $\eta_{\pi_{t+1}}^{-1} D_h(\text{greedy}(\mathcal{T}_{\mu_t} Q_t)(s), \pi_t(s) \leq \epsilon_{\pi_{t+1}}$, respectively, $\forall \epsilon_{\mu_t}, \epsilon_{\pi_{t+1}}$ arbitrarily small.*

(Proof. in Thm. 15, Appendix 1.2)

Limitations **First**, both PMD(+loo) and PMD(+ext) update two policies per iteration, both from from π_t (μ_t to look ahead and π_{t+1} as the next policy iterate). **Second**, they require two policy evaluation procedures per iteration, for $Q_t(s)$ and $(\mathcal{T}_{\mu_t} Q_t)(s)$. We may use a value estimator for the former, but the latter would then need access to a model. The main issue of PMD(+ext) is that it needs information from two consecutive updates (π_t and μ_t) to compute the update to the policy iterate π_{t+1}. In the next section, we propose a "lazy" practical algorithm which remedies these issues, using only one policy variable and applying a single policy evaluation procedure per iteration, at the expense of extra memory.

Extrapolation from the past The idea is to rewrite the update to use a single evaluation per timestep plus a temporal-difference error between consecutive evaluation updates. Then, assuming past errors reflect future ones, reusing the prior errors (extrapolating from

the past) should have the same effect as extrapolating from the future. We briefly touch on the main steps of algorithm derivation, deferring details to Appendix 1.2.1. **First**, we show the PMD(+ext) update can be rewritten as $\pi_{t+1}(s) = \text{argmin}_{\pi(s)\in\Delta(\mathcal{A})} -\langle (\mathcal{T}_{\mu_t} Q_t)(s) - \eta_{\pi_{t+1}}^{-1}\eta_{\mu_t} Q_t(s), \pi(s)\rangle + \eta_{\mu_t}^{-1} D_h(\pi(s), \mu_t(s))$. **Next**, to remedy the need for two evaluation procedures, we recycle evaluation updates from previous iterations by replacing Q_t with the closest past evaluation, i.e. $\mathcal{T}_{\mu_{t-1}} Q_{t-1}$, twice in the updates of $\pi_{t+1}(s)$ and μ_t (Eq. 5.3). We **then** show the update can be written in terms of a one policy variable and a single evaluation procedure, henceforth denoted π and Q

$$\pi_{t+1}(s) = \text{argmin}_{\pi(s)\in\Delta(\mathcal{A})} -\langle Q_t(s) + \eta_{\pi_{t+1}}^{-1}\eta_{\pi_t}(Q_t(s) - Q_{t-1}(s)), \pi(s)\rangle + \eta_{\pi_{t+1}}^{-1} D_h(\pi(s), \pi_t(s))$$

We refer to the algorithm using this update **PMD(+mom)**, since the temporal-difference is analogous to momentum. Interestingly, a similar update is called "optimistic" mirror descent by Joulani et al. [2020a,b], Rakhlin and Sridharan [2013a, 2014] and a "forward-reflected-backward" method by Malitsky and Tam [2020].

Alg. 12 in Appendix 2 summarizes the updates of all algorithms introduced.

2.1 Approximate Functional Acceleration for Parametric Policies

We are interested in designing algorithms feasible for large-scale optimization, so we further consider parametrized versions of the algorithms introduced, which we illustrate numerically in Sec. 3.

Q-function Approximation For the *exact setting*, we compute model-based versions of all updates. For the *inexact setting*, we consider approximation errors between $\widehat{Q}_t$ and Q_t (Sec.3.3).

Algorithm 9 Approximate PMD(+mom)

Initialize policy parameter $\theta_0 \in \Theta$, mirror map h, small constant ϵ_0, learning rate β

for $t = 1, 2 \dots T$ **do**

  Find $\widehat{Q}_t$ approximating Q_t (critic update)

  Compute adaptive step-size $\eta_{\pi_{t+1}} = (\gamma^{2(t+1)}\epsilon_0)^{-1} D_h(\text{greedy}(\widehat{Q}_t), \pi_t)$

  Find $\pi_{t+1} \doteq \pi_{\theta_{t+1}}$ by solving the surrogate problem (approximately with k GD updates)

  $\min_{\theta\in\Theta} \ell(\theta) \quad \ell(\theta) \doteq -\mathbb{E}_{s\sim d_t^\rho}[\mathbb{E}_{a\sim\pi_\theta}[Q_t(s,a)] + \eta_{\pi_{t+1}}^{-1} D_h(\pi_\theta(s), \pi_t(s))]$

  (init) $\theta^{(0)} \doteq \theta_t \quad$ *(for* $i \in [0..k-1]$*)* $\theta^{(i+1)} = \theta^{(i)} - \beta\nabla_{\theta^{(i)}}\ell(\theta^{(i)}) \quad$ *(final)* $\theta_{t+1} \doteq \theta^{(k)}$

Policy Approximation We parametrize the policy iterates using a Bregman policy class $\{\pi_\theta : \pi_\theta(s) = \text{proj}^h_{\Delta(\mathcal{A})}(\nabla h^*(f_\theta(s))), s \in \mathcal{S}\}$, a tabular parametrization for the dual policy representation $f_\theta(s,a) \doteq \theta(s,a)$, and the negative Boltzmann-Shannon entropy as mirror map h, which leads to the softmax policy class $\pi_\theta(s) \doteq {\exp\theta(s)}/{\sum_{a\in\mathcal{A}}\exp\theta(s,a)}$. To update the parameter vector θ we use the proximal perspective cf. Alg. 9 [Tomar et al., 2020, Vaswani et al., 2021, 2023] (implementation details in Appendix 3).

3 Numerical Studies

In this section, we investigate numerically the aforementioned algorithms, focusing on the following questions: Sec. 3.1: When is acceleration possible? What properties of the problem make it advantageous? Sec. 3.2: How do the policy optimization dynamics change with acceleration? Does the value improvement path change? Sec. 3.3: Should we expect acceleration to be effective with an inexact critic? What are the implications of value approximation?

In all experiments, results indicate mean values and shades standard deviation (std) over seeds.

3.1 When is Acceleration Possible?

3.1.1 Experimental Setting

We consider randomly constructed finite MDPs—*Random MDP* problems [Archibald et al., 1995], abstract, yet representative of the kind of MDP encountered in practice, which serve as a test-bench for RL algorithms [Goyal and Grand-Clement, 2021, Scherrer and Geist, 2014, Vieillard et al., 2019]. We define a *Random MDP* generator $(|\mathcal{S}|, |\mathcal{A}|, b, \gamma)$ parameterized by number of states $|\mathcal{S}|$, number of actions $|\mathcal{A}|$, branching factor b specifying for each state-action pair the maximum number of possible next states, chosen randomly. We vary b, γ, and $|\mathcal{A}|$ to show how the characteristics of the problem, and the features of the algorithms, impact learning speed with or without acceleration (details in Appendix 7.2).

3.1.2 Metrics

We measure the following quantities. (i) The cumulative *regret* (optimality gap) after T iterations, $\textit{Regret}_t \doteq \sum_{t \leq T} V^* - V_t^\rho$. The relative difference in regret between the PMD baseline and PMD(+mom) shows whether functional acceleration speeds up convergence. To quantify the complexity of the optimization problem and ill-conditioning of the optimization landscape (difference in scaling along dimensions), we use the dual representation form of Wang et al. [2008] for policies (successor representation [Dayan, 1993] or state-visitation frequency), $\Psi_\pi \doteq (\mathbf{I} - \gamma \mathbf{P}_\pi)^{-1}$, with $\mathbf{P}_\pi V(s) = \mathbb{E}_{a \sim \pi, s' \sim P(\cdot|s,a)}[V(s')]$. Policy iteration is known to be equivalent to the Newton-Kantorovich iteration procedure applied to the functional equation of dynamic programming [Puterman and Brumelle, 1979], $V_{\pi_{t+1}} = V_{\pi_t} - \Psi \nabla f(V_{\pi_t})$, where $\nabla f(V) = (I - \mathcal{T})(V)$—with $\mathcal{T}$ the Bellman operator—can be treated as the gradient operator of an unknown function $f : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}$ [Grand-Clément, 2021] (see Appendix 6). From this perspective, the matrix Ψ can be interpreted as a gradient preconditioner, its inverse is the Hessian $\nabla^2 f(V)$, the Jacobian of a gradient operator ∇f. We use the condition number of this matrix, defined as $\kappa(\Psi) \doteq {}^{|\lambda_{\max}|}/_{|\lambda_{\min}|}$, for $\lambda_{\max}$, $\lambda_{\min}$ the max and min eigenvalues in the spectrum $\mathrm{spec}(\Psi)$. We measure (ii) the *condition number* $\kappa_0 = \kappa(\Psi_{\pi_0})$

of a randomly initialized (diffusion) policy π_0 (Fig. 5.2(a-b)) and (iii) the average *condition number* $\kappa_{t\leq T} = {}^1/_T \sum_{t\leq T} \kappa(\Psi_{\pi_t})$, for policies on the optimization path of an algorithm (Fig. 5.2(c)). Lastly, we also measure (iv) the mean *entropy* of a randomly initialized policy $\mathcal{H}_0 \propto \sum_{s,a} \pi_0(s,a) \log \pi_0(s,a)$ (Fig. 5.2(d)), inversely correlated with κ_0.

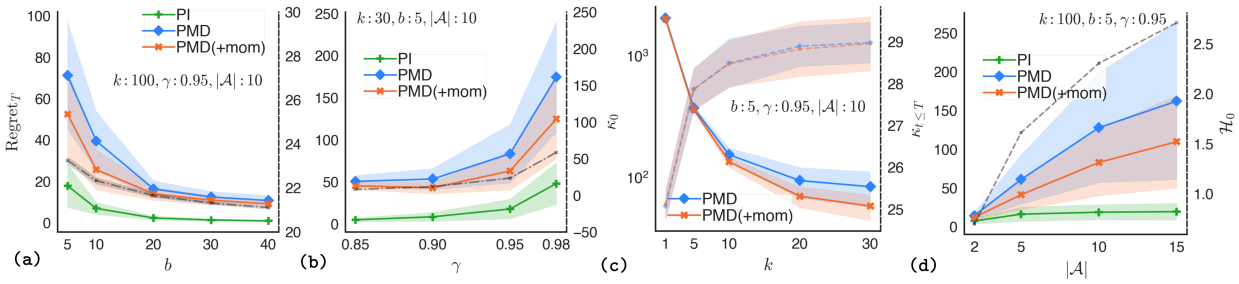


Figure 5.2: The left y-axis shows the optimality gap or cumulative regret at $T = 10$ *(a-c)*, $T = 20$ *(d)*) over 50 randomly sampled MDPs, relative to changing: (a) b—branching factor, (b) γ—discount factor, (c) k—number of parameter updates, (d) $|\mathcal{A}|$—number of actions. The right y-axis and dotted curves measure: *(a-b)*—the condition number κ_0, *(c)* the average condition number $\kappa_{t\leq T}$, *(d)* the entropy $\mathcal{H}_0$.

Hypothesis & Observations In Fig. 5.2 we show the relative difference in regret between PMD(+mom) and the PMD baseline, as we change the features of the algorithms and the complexity of the problem. First, we highlight two cases that lead to ill-conditioning—indicated by the condition number κ_0: *(a)* sparse connectivity of the underlying Markov chain controlled by decreasing the branching factor b, which represents the proportion of next states available at every state-action pair; *(b)* increasing the effective horizon via the discount factor γ. We illustrate the relative difference in optimality gap between the two updates correlates with ill-conditioned policy optimization landscapes, supporting the hypothesis that functional acceleration leads to faster navigation in this case. In *(c)*, we show the relative difference in regret correlates with the number of parameter updates k used in the "inner-loop" optimization procedure. Large k implies the policy iterates approach the analytic solution of the surrogate objective. As k decreases, the added momentum will shrink too, becoming negligible, defaulting to the classic parameter-level momentum in the limiting case of $k = 1$ (the online setting). In *(d)*, as we increase the number of actions, the optimization problem becomes more challenging, entropy and

overall suboptimality increase. However, the relative difference between PMD(+mom) and the baseline PMD also increases, suggesting the increasing advantage of functional acceleration (learning curves and additional statistics in Appendix 8.1).

Implications These studies indicate (i) that it is possible to accelerate PMD, that the advantage of functional acceleration is proportional to: (iii) the ill-conditioning of the optimization surface, induced by the policy and MDP dynamics, and (ii) the quality of the "inner-loop" policy approximation.

3.2 Policy Dynamics in Value Space

To contrast the expected policy dynamics, we rely on the functional mapping from stationary policies to their respective value functions, $\pi \to V_\pi$: $\mathcal{V} \doteq \{V_\pi | \pi \in \Pi\}$. The space of value functions $\mathcal{V}$ the set of all value functions that are attained by some policy, i.e. the image of Π under the mapping, and a possibly self-intersecting, non-convex polytope [Dadashi et al., 2019].

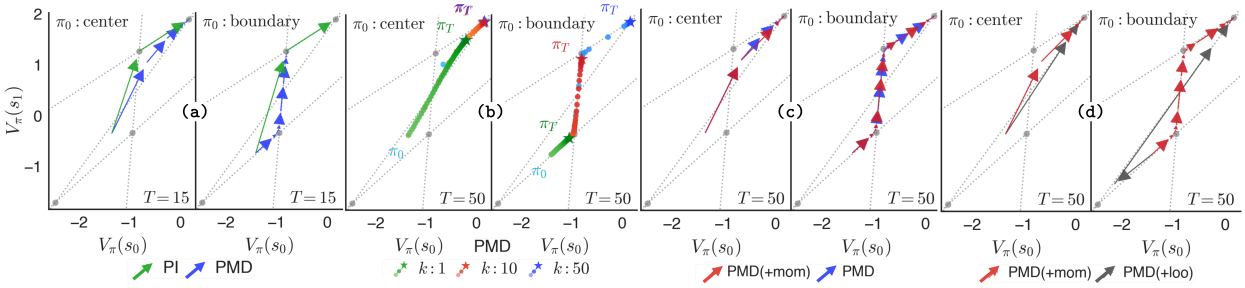


Figure 5.3: Compares and contrasts the policy optimization dynamics on the value polytope. Gray points denote the boundaries—corresponding in this case to value functions, gray dotted lines are hyperplanes circumscribing the polytope. Top-left: policy initialization. *(a, c, d)* Arrows indicate the policy optimization path. PMD and PMD(+mom) use $\beta = 0.1$ and $k = 50$ for the "inner" loop optimization, approximating the analytic solution. *(b)* Points denote policy values on the path, gradient indicates iteration number t, star $\star$ marks π^T.

Experimental Setting We use two-state MDPs (specifics in Appendix 7.1, other MDPs in Appendix 8.2). Policies are initialized with π_0: (i) *center*: interior of the polytope, (ii) *boundary*: near a boundary close to the corner adversarial to the optimum. We use the value polytope to visualize three aspects of the learning dynamics: (1) the policy improvement

path through the polytope, (2) the speed at which they traverse the polytope, and (3) sub-optimal attractors with long escape times that occur along this path, making the policy iterates accumulate (cf. Mei et al. [2020a]). We compute model-based versions of all relevant updates.

Observations & Insights Fig. 5.3 shows the impact of policy approximation through k *(a-b)*, and compares the dynamics of PMD(+mom) relative to the baselines: without acceleration: PMD *(c)*, and with idealized acceleration: PMD(+loo) *(d)*.

We make the following observations: *(a)* PMD's dynamics follow a straight path between the iterates of PI, consistent with the former being an approximation of the latter. The convergence speed depends on the approximation quality through k—the number of parameter updates per iteration *(b)*. PMD with $k = 1$ corresponds to online PG, which depends strongly on initialization, a known issue caused by vanishing gradients at the boundary of the polytope. In contrast, as we increase k, there is faster escape from sub-optimal attractors, and the rate of convergence is higher, becoming more similar to PI by jumping between values of deterministic policies (corners of the polytope *(a)*). *(d)* PMD(+loo) follows a different trajectory through the value polytope. We observe acceleration for PMD(+mom) relative to PMD in *(c)*, and suboptimality relative to the idealized acceleration of PMD(+loo) in *(d)*.

3.3 Functional Acceleration with an Inexact Critic

For the same experimental setting as Sec. 3.2, Fig. 5.4 illustrates the impact of an inexact critic on the relative advantage of functional acceleration, in two settings: *(Left) controlled*—the critic's error is sampled from a random normal distribution with mean 0 and standard deviation τ, such that $\widehat{Q}_t(s) = Q_t(s) + \mathcal{N}(0, \tau)$, $\forall s$. *(Right) natural*—the critic is an empirical estimate of the return obtained by Monte-Carlo sampling, and its error arises naturally from using m truncated trajectories up to horizon $1/1-\gamma$, i.e. $\widehat{Q}_t(s) \doteq 1/m \sum_{i \leq m} G_i(s)/N_i(s)$,

where $G_i(s)$ is the i^{th} empirical return sampled with $\pi_t(s)$ and $N_i(s)$ is the empirical visitation frequency of s.

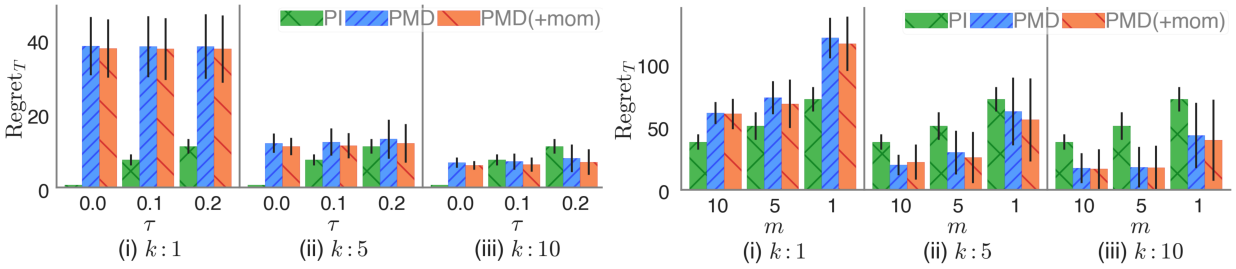


Figure 5.4: Shows cumulative regret on the y-axis, after $T = 50$ iterations, relative to changing k—the number of parameter updates for PMD and PMD(+mom), in the inexact setting. *(Left) controlled*: τ indicates the scale of the critic's error. *(Right) natural*: m is the number of trajectories used in the Monte-Carlo estimation of the return. Error bars denote std. over 50 seeds with policies initialized from a random uniform distribution $\mathcal{U}(0, 1)$.

We observe a larger relative difference in suboptimality on average between PMD(+mom) and PMD for higher values of k, highlighting the difference between functional acceleration (cf. Sec.2) and classic acceleration (applied to the parameter vector θ), corresponding to $k=1$, reinforcing evidence from Sec. 3.1. Further, we confirm PI performs increasingly poor when paired with an inexact critic with large error as previously reported by [Russo, 2022]. Then, we observe a range in which functional acceleration is advantageous, which extends from having negligible benefit, for small k, to more impactful differences in optimality gap for larger k. Beyond a certain sweet spot, when it is maximally advantageous, the critic's error becomes too large, leading to overshooting and oscillations (additional illustrations of this phenomenon in Appendix 8.3).

4 Related Work

Accelerated optimization methods have been at the heart of convex optimization research, e.g., Nesterov's accelerated gradients (NAG) [Nesterov, 1983, Wang and Abernethy, 2018, Wang et al., 2021], extra-gradient (EG) methods [Korpelevich, 1976], mirror-prox [Juditsky et al., 2011, Nemirovski, 2004], optimistic MD [Joulani et al., 2020b, Rakhlin

and Sridharan, 2013b], AO-FTRL [Mohri and Yang, 2015, Rakhlin and Sridharan, 2013a], Forward-Backward-Forward (FBF) method [Tseng, 1991].

Classic acceleration As far as we know, our idea of applying acceleration to the direct (functional) policy representation π_θ—independent of the policy parametrization θ—is novel. This is important because it means universality of the approach to any kind of parametrization and functional form a practitioner requires. Within the context of RL, acceleration has only been applied to value learning [Farahmand and Ghavamzadeh, 2021, Goyal and Grand-Clement, 2021, Nettle and Bateson, 2012, Vieillard et al., 2019], or in the context of PG methods, classic acceleration is applied to the policy parameter θ—all recent deep RL works (e.g. Hessel et al. [2017], Mnih et al. [2016], Schulman et al. [2017]) use some form of adaptive gradient method, like Adam [Kingma and Ba, 2015] or RMSProp [Hinton et al., 2012].

The idea of acceleration generally relies on convexity of the objective relative to the representation of interest. The transformation from parameters θ to functional representation of the policy as probabilities π_θ, can be highly complex, non-linear, and problem-dependent. Proximal algorithms operate on this functional representation, and rely on relative-convexity and relative-smoothness [Lu et al., 2017] (relative to the mirror map h) of the objective with respect to π when constructing surrogate models [Agarwal et al., 2019, Bhandari and Russo, 2019, 2021, Vaswani et al., 2021]. These properties suggests the functional acceleration mechanism is feasible and promising in our setting, since it is able to successfully accelerate convex optimization [Joulani et al., 2020b].

Approximate PMD PMD has been extended to function approximation by Tomar et al. [2020], and later analyzed by Vaswani et al. [2021], who treat the surrogate objective as a nonlinear optimization problem, that of approximately minimizing at each iteration, a composite proximal objective, denoted $\ell(\pi_\theta)$ with respect to the policy parameter θ. In contrast, Alfano et al. [2024], Xiong et al. [2024] apply the PMD update in the dual form, as

(generalized) Projected Gradient Descent (PGD), i.e. a gradient update in the dual space followed by a projection [Bubeck, 2015], rather than in proximal form [Beck and Teboulle, 2003], as used by Tomar et al. [2020]. Here, we use on the proximal objective.

5 Discussion

Conclusions Our findings suggest that momentum can accelerate policy learning in RL agents. Incorporating the influence of recent experiences, agents can better navigate the optimization landscape, avoiding oscillations and re-exploration of suboptimal regions, focusing instead on promising directions indicated by past improvements, thus leading to efficient policy updates and faster convergence to optimal behaviour.

Limitations & Future Work Our focus is on developing a foundation that motivates further study. A translation to practical large-scale implementations and deep RL remains for further investigation, i.e. with non-standard proximal methods, e.g., TRPO [Schulman et al., 2015], PPO [Schulman et al., 2017], MDPO [Tomar et al., 2020], MPO [Abdolmaleki et al., 2018]. Additional guarantees of accelerated convergence for general policy parametrization using the dual policy norm, as well as theoretical analysis for the stochastic setting, are also deferred for future work. We also note the concurrent work of Protopapas and Barakat [2024], who also propose a version of PMD with lookahead (Sec. 2), and show a similar result to Thm. 1, beyond which the works diverge.

Significance Inspired by the idea that neural circuitry with slow dynamics may contribute to the learning process, this study highlights the potential of integrating similar mechanisms into artificial agents. The algorithm introduced provides computational benefits and draws insights on how slow-evolving processes can facilitate learning and adaptation. This work has implications for both experimentalists and theoreticians. For experimentalists studying the neural basis of learning and decision-making, our model provides a computational framework linking value dynamics to behavioural adaptation. For

theoreticians and RL practitioners, this approach introduces a novel algorithm, applicable to large-scale and deep RL systems.

6 Excitatory-Inhibitory Dynamics in Adaptive Learning and Decision-Making

"If your mind is empty, it is always ready for anything, it is open to everything. In the beginner's mind there are many possibilities, but in the expert's mind there are few."

Shunryu Suzuki

Neural circuits rely on excitatory–inhibitory (E/I) interactions to support adaptive learning and decision-making. In this chapter, we investigate how these dynamics may contribute to flexible behaviour across three modelling levels. **First,** using a mean-field model of two-choice decision-making, we examine the computational role of decision-selective excitation (E) and inhibition (I) in stabilizing or amplifying competition between alternative choices. **Next,** building on these insights, we integrate a similar E/I mechanism into the preference function of a reinforcement learning (RL) agent, illustrating how inhibitory feedback can modulate behavioural flexibility when learning in dynamic bandit tasks. **Finally,** to scale these mechanistic principles to biologically-plausible neural network models, we then investigate the feasibility of RL agents with E/I-constrained recurrent neural networks (RNNs) in changing environments. Our results suggest that naively separating a standard network's units into E and I populations hinders adaptive learning, whereas a carefully structured architecture—"Dale's Artificial Neural Network" (DANN)—maintains sign constraints while preserving robust performance. Together, these approaches offer a computational framework linking E/I-constrained dynamics to adaptive behaviour.

1 Motivation

The brain relies on intricate networks of interconnected neuron populations to shape behaviour. Synaptic strengths in these networks constantly change based on sensory input, neural activity, and experience—a process known as synaptic plasticity [Abraham, 2008, Malenka and Bear, 2004]. These adjustments at molecular, cellular, and circuit levels influence higher-order functions such as learning, and memory [Bear et al., 2020].

Biological organisms operate in dynamic and uncertain environments requiring adaptive learning and decision-making to optimize survival. The ability to rapidly adjust behaviour in response to changing contingencies is thus critical for efficient continual learning. At the neural level, this adaptability is governed by the interplay between excitation (E) and inhibition (I), which regulates population-level neural dynamics and modulates the efficiency of information processing [Mante et al., 2013, Roach et al., 2023, Sussillo and Barak, 2013, Wakhloo et al., 2024]. Excitatory (glutamatergic) and inhibitory (GABAergic) neuron populations form recurrent circuits that shape decision-making by updating the tuning and selectivity as learning unfolds [Najafi et al., 2020]. Excitation is mediated by *glutamate*, rapidly activating *N-methyl-D-aspartate (NMDA)* and α*-amino-3-hydroxy-5-methyl-4-isoxazolepropionic acid (AMPA)* receptors, which are crucial for synaptic plasticity and action potential generation. Inhibition, provided by GABAergic neurons, modulates activity through *Gamma-aminobutyric acid (GABA)* receptors [Bear et al., 2020].

A fundamental organizing principle of neural networks is "Dale's principle" [Dale, 1935], which states that neurons release either excitatory or inhibitory neurotransmitters but not both [Eccles et al., 1954] [1].

Within the cortex, a complex interplay of *excitatory (E)* and *inhibitory (I)* neurons processes a rich array of sensory, cognitive, and motor signals. Although each neuron's activity is high-dimensional—influenced by numerous input features and tuned to many task-related

[1] See [Vaaga et al., 2014] (and references therein) for rare exceptions concerning excitatory and inhibitory neurotransmitter co-release.

properties—their collective behaviour can be understood through simpler low-dimensional underlying structures known as *latent circuits* [Barack and Krakauer, 2021, Chung and Abbott, 2021, Langdon and Engel, 2022, Langdon et al., 2023, Levy et al., 2023, Vyas et al., 2020]. These low-dimensional connectivity frameworks highlight how neural populations self-organize into *stable activity patterns*, often described as *attractor states* within a *phase space* [Mante et al., 2013, Sussillo and Barak, 2013]. Recurrent connections are crucial for these attractors to form, and shifts in the balance of excitation and inhibition (E/I) can alter these states and, in turn, affect behaviour [Roach et al., 2023].

Synaptic excitation and inhibition co-occur, even during the simplest sensory responses, and remain balanced during spontaneous cortical activity [Isaacson and Scanziani, 2011, Okun and Lampl, 2008]. Maintaining proper *E/I balance* ensures that cortical networks are neither overly excited nor excessively suppressed, thus achieving stable yet flexible computation. Meta-plasticity mechanisms often involve the modulation of inhibitory circuits to regulate plasticity [Abraham, 2008]. Moreover, this balance can be perturbed by external inputs, like stimulus history, shape, intensity, rate [Isaacson and Scanziani, 2011], or internal modulation, such as adaptation or neuromodulation [Abraham, 2008, Heiss et al., 2008], leading to shifts in circuit function. Internally, *opposing forces within the brain*—mediated by *receptor subtypes* such as *D1/D2 dopamine receptors* and *5-HT1A/5-HT2A serotonin receptors*—further refine these interactions.

Disruption of E/I balance is linked to various neurological and psychiatric disorders, underscoring its importance for healthy function. In contrast, properly balanced circuits can hover *near criticality* where neural networks are maximally sensitive to small perturbations in input and context, which can trigger cascades of activity that spread across networks of neurons [O'Byrne and Jerbi, 2022]. Near-critical dynamics are self-maintained, re-established after disturbances, attenuated with prolonged sleep deprivation, modulated by drugs, and abolished by anesthesia [O'Byrne and Jerbi, 2022]. This *near-criticallity* is hypothesized to be essential for optimizing information processing. Understanding these

processes is fundamental for linking neural circuit mechanisms to cognitive function and behaviour—and for informing interventions targeting disorders of E/I imbalance.

In this chapter, we adopt a three-level modelling strategy to investigate the computational role of *selective excitation–inhibition (E/I)* in *adaptive learning and decision-making*. In Section 3.1, we begin with a *Wilson-Cowan mean-field model* [Wilson and Cowan, 1972], extended to *decision-selective* excitation (E) and inhibition (I). This allows us to explore how selectivity impacts evidence integration and phase dynamics in decision-making. Building on these mechanistic insights, in Section 3.2, we incorporate this E/I mechanism into the preferences of a Reinforcement Learning (RL) agent interacting with a dynamic environment, and find that inhibitory feedback on the preference function modulates behavioural flexibility. To bridge these circuit-level principles with more scalable and biologically-plausible models, in Section 3.3, we apply the same E/I constrains on recurrent neural network (RNN) used by a memory-based RL agent operating in dynamic environments. Our results show that naive partitioning into E and I neurons [Song et al., 2016] hinders adaptive learning by reinforcement, whereas a biologically-motivated architecture—Dale's Artificial Neural Network (DANN) [Li et al., 2023]—supports robust generalization and adaptability. Together, these approaches offer a computational framework linking E/I-constrained dynamics to adaptive behaviour.

2 Background and Preliminaries

To ground our study, we first define key concepts essential to understanding neural circuit function, laying the foundation for the analyses and simulations presented herein.

Neural Population Codes Neural circuits encode information not through individual neurons, but through coordinated activity patterns across populations of neurons [Wakhloo et al., 2024]—a process known as *distributed computation* in AI, and *neural population coding* in computational neuroscience. These collective dynamics enable robust encoding of

sensory, motor, and cognitive information. It is believed that the brain uses *cognitive maps* to solve problems [Behrens et al., 2018], where environmental variables are represented in the population code in a way that preserves task-relevant relationships [Aronov et al., 2017, Bao et al., 2019, Constantinescu et al., 2016, Knudsen and Wallis, 2021, Muhle-Karbe et al., 2023, Nieh et al., 2021]. These patterns represent key task features or behavioural states (e.g., position, auditory stimulus frequency, time, evidence accumulation in decision-making tasks), forming the basis for neural computations. For example, in decision-making, a population code integrates sensory and cognitive inputs, producing a coherent representation of task demands. These codes are shaped by excitatory-inhibitory (E/I) dynamics, with excitation and inhibition sculpting the activity patterns that generate these representations.

Latent Neural Circuits and Low-Dimensional Manifolds Despite the high-dimensional nature of neural activity, task-relevant dynamics often lie on low-dimensional manifolds [Langdon and Engel, 2022, Langdon et al., 2023, Levy et al., 2023, Mante et al., 2013, Sussillo and Barak, 2013, Wakhloo et al., 2024]. These manifolds represent *latent neural circuits*, capturing the dominant activity patterns that drive behaviour. Low-dimensional representations offer an efficient means of encoding complex information while minimizing noise. Thus, understanding brain computations can be achieved by describing the latent, low-dimensional dynamics of neural populations, where each factor reflects a pattern of co-activation across neurons.

Mean Field Theory Mean field theory offers a powerful framework for analyzing the collective behaviour of large neuron populations by reducing complex, high-dimensional dynamics into simpler, averaged descriptions, such as firing rates. Originally developed in physics, this approach enables the study of neural circuits by focusing on macroscopic properties that emerge from interactions among many neurons, while neglecting microscopic details. It has been crucial in understanding phenomena like phase transitions, memory states, and dynamic regimes in neural networks. In this study, we model biological neural

networks by representing the firing rate of a homogeneous neuronal population and approximating their dynamics with a one-dimensional equation describing the evolution of the average firing rate over time.

Firing Rate Models Firing rate models are based on the idea that neural activity arises from a dynamical system defined by:

$$\alpha \frac{\mathrm{d}\mathbf{r}}{\mathrm{dt}} \doteq -\mathbf{r} + f(\mathbf{W}\mathbf{r} + \mathbf{I})$$

where the state of the dynamical system $\mathbf{r}(t)$ is updated by a recurrent function (e.g., a sigmoid function $f(\mathbf{r}; a, b, d) = \frac{1}{1+\exp(-d(a \cdot \mathbf{r} - b))}$, with d—the gain and b—the threshold potential, representing the firing rate output, scaled by a, in response to the stimulus input $\mathbf{I}$ and the recurrent input $\mathbf{r}$. The recurrent input strength is represented by the synaptic weight $\mathbf{W}$, and the parameter α controls the timescale of the population dynamics.

Much of the rich dynamics in the brain are driven by interactions between excitatory and inhibitory neurons. In this chapter, we use the Wilson-Cowan rate model [Wilson and Cowan, 1972] to study the dynamics of two interacting populations of excitatory and inhibitory neurons, whose dynamics evolve according to:

$$\begin{aligned}
\alpha_{\mathrm{E}} \frac{\mathrm{d}\mathbf{r}_{\mathrm{E}}}{\mathrm{dt}} &\doteq -\mathbf{r}_{\mathrm{E}} + f_{\mathrm{E}}(\mathbf{W}_{\mathrm{EE}}\mathbf{r}_{\mathrm{E}} - \mathbf{W}_{\mathrm{IE}}\mathbf{r}_{\mathrm{I}} + \mathbf{I}_{\mathrm{E}}) \\
\alpha_{\mathrm{I}} \frac{\mathrm{d}\mathbf{r}_{\mathrm{I}}}{\mathrm{dt}} &\doteq -\mathbf{r}_{\mathrm{I}} + f_{\mathrm{I}}(\mathbf{W}_{\mathrm{EI}}\mathbf{r}_{\mathrm{E}} - \mathbf{W}_{\mathrm{II}}\mathbf{r}_{\mathrm{I}} + \mathbf{I}_{\mathrm{I}})
\end{aligned}$$

where $\mathbf{r}_{\mathrm{E}}(t)$ and $\mathbf{r}_{\mathrm{I}}(t)$ represent the average activation (firing rate) of the excitatory and inhibitory populations at time t, respectively. The parameters α_{E} and α_{I} control the timescales of each population's dynamics. The connection strengths are given by $\mathbf{W}_{\mathrm{EE}}(\mathrm{E} \to \mathrm{E}), \mathbf{W}_{\mathrm{EI}}(\mathrm{E} \to \mathrm{I}), \mathbf{W}_{\mathrm{IE}}(\mathrm{I} \to \mathrm{E}), \mathbf{W}_{\mathrm{II}}(\mathrm{I} \to \mathrm{I})$, and the transfer functions may differ for excitatory and inhibitory populations.

Behavioural Flexibility In reinforcement learning (RL), an agent interacts with an environment to learn a policy that maximizes cumulative rewards. **Continual RL** extends this paradigm to dynamic environments, where tasks and conditions evolve over time, reward structures shift, new states or goals are introduced. In Continual RL, the central challenge is *lifelong adaptability*, i.e. enabling the agent to continually adapt to new experiences under specific assumptions on distribution shift, as well as generalization or interference with previous experience. Two of the main assumptions are:

i *Future tasks resemble past tasks (generalization)*, in which case the paradigms responsible for evaluating an agent's capabilities focus on *remembering and reusing past strategies* for task continuity.

ii *Future tasks are adversarial to past tasks (interference)*, in which case the paradigms focus on *"unlearning" outdated strategies* to adapt to adversarial or novel tasks.

In the context of biological agents, *behavioural flexibility* is defined as the ability to adapt to changes in the environment by switching task sets, responses, or strategies [Cools, 2015]. Conversely *stability* maintains consistency when changes are irrelevant. Flexibility allows changing behaviour in response to environmental and internal cues. Individuals which score higher at this trait have improved life outcomes, better social functioning, and reduced cognitive decline with age [Zühlsdorff et al., 2022]. Deficits in flexibility have been observed in most (if not all) psychiatric conditions.

Although not fully understood, several behavioural paradigms have been proposed in neuroscience and psychology to study and evaluate adaptive learning, flexibility, and the effects of drugs or brain lesions. Common paradigms encompass *reversal learning*, *attentional set-shifting*, *task-switching*, and *working memory updating* [Cools, 2015]. These are distinct challenges differing in the assumptions they make about the future task distributions, thereby reflecting different aspects of cognitive flexibility and adaptation.

Although all paradigms are essential for understanding how agents, biological or artificial, adapt to dynamic environments, this chapter uses a structured approach to understanding adaptive learning in dynamic environments through *reversal learning* tasks, leaving exploration of the other paradigms for future work.

Reversal Learning This paradigm assumes that future tasks are similar to past tasks in terms of their structure but may involve reversed reward contingencies. Thus, it focuses on forward-looking plasticity and adaptability to dynamically changing reward contingencies. It expects the agent to maintain an ability to adapt strategies dynamically as task demands shift. The agent should be able to *"unlearn"* outdated previously learned associations to avoid perseverative errors.

3 Methods and Results

3.1 A Mean-Field E/I Model of Decision-Making

We employ a *Wilson–Cowan firing rate model* [Wilson and Cowan, 1972] to investigate how E/I interactions, connectivity patterns, and external inputs shape decision-making activity. The model consists of two decision-selective excitatory (E_1, E_2) and likewise two inhibitory (I_1, I_2) populations (Fig. 6.1 Left), governed by the equations:

$$\begin{aligned}\alpha_{\mathrm{E}}\frac{\mathrm{d}\mathbf{r}_{\mathrm{E}}}{\mathrm{dt}} &\doteq -\mathbf{r}_{\mathrm{E}} + f_{\mathrm{E}}(\mathbf{W}_{\mathrm{EE}}\mathbf{r}_{\mathrm{E}} - \mathbf{W}_{\mathrm{IE}}\mathbf{r}_{\mathrm{I}} + \mathbf{I}_{\mathrm{ext}} + \mathbf{I}_{\mathrm{AMPA}})\\ \alpha_{\mathrm{I}}\frac{\mathrm{d}\mathbf{r}_{\mathrm{I}}}{\mathrm{dt}} &\doteq -\mathbf{r}_{\mathrm{I}} + f_{\mathrm{I}}(\mathbf{W}_{\mathrm{EI}}\mathbf{r}_{\mathrm{E}})\end{aligned}$$

where $\mathbf{r}_{\mathrm{E}} \doteq [r_{\mathrm{E}_1}, r_{\mathrm{E}_2}]^\top$ and $\mathbf{r}_{\mathrm{I}} \doteq [r_{\mathrm{I}_1}, r_{\mathrm{I}_2}]^\top$ represent the activity of excitatory and inhibitory populations, $\alpha_{\mathrm{E}}, \alpha_{\mathrm{I}}$ are the timescales for integration, $\{\mathbf{W}_j\}_{j\in\{\mathrm{EE,EI,IE}\}}$, $\mathbf{W}_j = \begin{bmatrix} 1+w_j & 1-w_j \\ 1-w_j & 1+w_j \end{bmatrix}$ represent the connectivity weights across populations, characterizing the relations $\mathrm{EE}(\mathrm{E}\rightarrow$

E), EI($\mathrm{E}\to\mathrm{I}$), IE($\mathrm{I}\to\mathrm{E}$), where $w_{\mathrm{EE}}, w_{\mathrm{EI}} \in [0,1]$ (to enforce recurrent dynamics), $w_{\mathrm{IE}} \in [-1,1]$, and $f_{\mathrm{E}}, f_{\mathrm{I}}$ are tuning (activation) functions.

Unlike standard E/I models, we use decision-specific excitatory and inhibitory subpopulations enabling exploration of how different E/I motifs modulate the phase space. To obtain an analytic solution in this model, we use $f_{\mathrm{E}}(\mathbf{r}), \doteq \frac{(a\mathbf{r}-b)}{(1-\exp(-d(a\mathbf{r}-b)))}$ (ELU), $f_{\mathrm{I}}(\mathbf{r}) \doteq a\mathbf{r}-b$ (linear), with $a=1, b=0, d=1$, without inhibitory recurrence $(\mathrm{I}\to\mathrm{I})$.

Sensory evidence, $\mathbf{I}_{\mathrm{ext}} = [\mathrm{I}_1, \mathrm{I}_2]^\top = [\mathrm{I}_{\mathrm{ext}}(1-c), \mathrm{I}_{\mathrm{ext}}(1+c)]^\top$, provides input to each excitatory population, favouring one population over the other, where $\mathrm{I}_{\mathrm{ext}}$ is a the baseline signal strength, and $c \in [-1,1]$ is the evidence coherence. Noise sweeps in via $\mathbf{I}_{\mathrm{AMPA}}$ following an Ornstein–Uhlenbeck process (as described in Appendix 1).

Decisions are read out when the difference in excitatory activity exceeds a fixed threshold. For instance, in a two-armed bandit task, the excitatory outputs can encode an RL agent's preferences, a strategy we apply in Sec. 3.2 to connect these E/I mechanisms to observed behavioural effects of an RL agent interacting with a dynamical environment, thus providing a computational bridge across modelling levels.

Results & Observations

We analyze *phase-space activity* across different parameter settings. We shows how changes in E/I balance modulate evidence integration, recurrent excitation, and inhibitory feedback. Appendix 1 provides: (i) additional details about the experimental setting, (ii) all the parameters used in the simulations, (iii) explicit derivations of the analytic solutions illustrated in the plots, (iv) analysis of stability conditions and relative alignment of the decision nullclines (v) pseudo-code for simulating activity traces through the system. All numerical simulations were run in Python.

Evidence integration Fig. 6.1 (Center/Right) shows the influence of timescales $(\alpha_{\mathrm{E}}, \alpha_{\mathrm{I}})$ on evidence integration and convergence of the dynamics to one of two attractor states. Specifically, it depicts the phase-space of the excitatory populations (Center), for stimulus

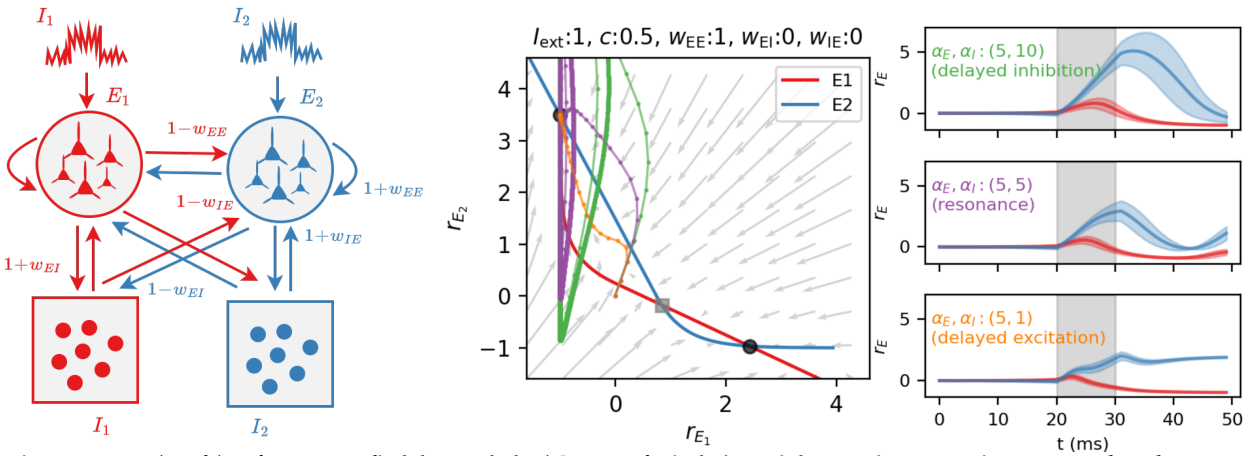


Figure 6.1: (Left) E/I mean-field model. (Center/Right) Evidence integration may display sustained oscillations. **(Center)** Showing three trajectories of evidence integration with different timescales $(\alpha_{\mathrm{E}}, \alpha_{\mathrm{I}})$ over the phase-space of excitatory populations. Nullclines represent equilibrium points where the activity does not change over time ($^{\mathrm{d}\mathbf{r}_{\mathrm{E}}}/_{\mathrm{dt}} = \mathbf{0}$). Stable attractors (black circles) correspond to decision states, while saddle points (gray square) indicate saddle points for transition thresholds. **(Right)** Excitatory activity over time. **(Right-Top)** Delayed inhibition, sustained excitation (small α_{E}, large α_{I}) generates high-amplitude oscillations. **(Right-Center)** Resonance (small $\alpha_{\mathrm{E}} = \alpha_{\mathrm{I}}$) oscillations are amplified instead of dampened. **(Right-Bottom)** Faster inhibition, delayed excitation (large α_{E}, small α_{I}) dampens oscillations and stabilizes faster.

coherence $c = 0.5$, a setting eliciting stronger response from the E_2 population. Red and blue curves denote nullclines associated with the two E populations. Nullclines represent steady-state solutions where neural activity does not change over time with respect to one of the decision variables ($^{\mathrm{d}\mathbf{r}_{\mathrm{E}}}/_{\mathrm{dt}} = \mathbf{0}$), providing insight into the stability of the fixed points in decision-making.

The competition between excitatory and inhibitory forces determines whether the dynamics are stable, oscillatory, or erratic. In this circuit, there are two stable attractors, associated with the two decisions (black circles), separated by a saddle point (gray square). The stability of a fixed point $\mathbf{r}_{\mathrm{E}}$ is determined by analyzing the gradient vector field of the E population

$$\mathbf{g}_{\mathrm{E}}(\mathbf{r}_{\mathrm{E}}) \doteq \frac{1}{\alpha_{\mathrm{E}}}(-\mathbf{r}_{\mathrm{E}} + f_{\mathrm{E}}(\mathbf{W}_{\mathrm{EE}}\mathbf{r}_{\mathrm{E}} - \mathbf{W}_{\mathrm{IE}} f_{\mathrm{I}}(\mathbf{W}_{\mathrm{EI}}\mathbf{r}_{\mathrm{E}}) + \mathbf{I}_{\mathrm{ext}} + \mathbf{I}_{\mathrm{AMPA}}))$$

and specifically the sign of the eigenvalues λ of its Jacobian in the vicinity of the fixed point

$$\mathbf{J}_{\mathrm{E}}(\mathbf{r}_{\mathrm{E}}) \doteq \begin{bmatrix} \frac{\partial g_{\mathrm{E}_1}(\mathbf{r}_{\mathrm{E}})}{\partial r_{\mathrm{E}_1}} & \frac{\partial g_{\mathrm{E}_1}(\mathbf{r}_{\mathrm{E}})}{\partial r_{\mathrm{E}_2}} \\ \frac{\partial g_{\mathrm{E}_2}(\mathbf{r}_{\mathrm{E}})}{\partial r_{\mathrm{E}_1}} & \frac{\partial g_{\mathrm{E}_2}(\mathbf{r}_{\mathrm{E}})}{\partial r_{\mathrm{E}_2}} \end{bmatrix}$$

In Fig. 6.1 (Right), trajectory dynamics are shown for three instances of timescale pairs $\alpha_{\mathrm{E}}, \alpha_{\mathrm{I}}$ in Fig. 6.1 (Center). Fig. 6.1 (Right) further illustrates the activity/time oscillations of the three trajectories. We observe that delayed inhibition and sustained excitatory activity (small α_{E}, large α_{I}) can lead to sustained oscillations with higher amplitude fluctuations, because inhibition is slow, so excitation builds up before being suppressed (green trajectory (Right-Top)). This can lead to runaway excitation if inhibition is too delayed. In contrast, faster inhibition paired with slower excitation (large α_{E}, small α_{I}) leads to damped oscillations with faster stabilization (orange trajectory (Right-Bottom)). When $\alpha_{\mathrm{E}} = \alpha_{\mathrm{I}}$ are small, the system updates quickly. If excitation overshoots before inhibition can fully counteract, the system does not stabilize immediately but instead oscillates around the fixed point. If the feedback loop is strong enough, oscillations can be self-sustaining leading to resonance.

The E/I balance can be modulated by altering the connectivity parameters $w_{\mathrm{EE}}, w_{\mathrm{EI}}, w_{\mathrm{IE}}$.

Recurrent excitation Within the space of circuits, an increase in excitatory self-connections (w_{EE}) within the E populations leads to strong recurrent excitation, which amplifies neural responses. In the phase-space, this increase causes the single fixed point to split into two stable attractor corresponding to the two decisions, separated by the saddle point (Fig. 6.2). Once the circuit settles into one of these attractors, persistent activity endures even if external input is removed—akin to a working-memory mechanism. Working memory refers to the ability to maintain and manipulate items for short periods of time [Baddeley and Hitch, 1974].

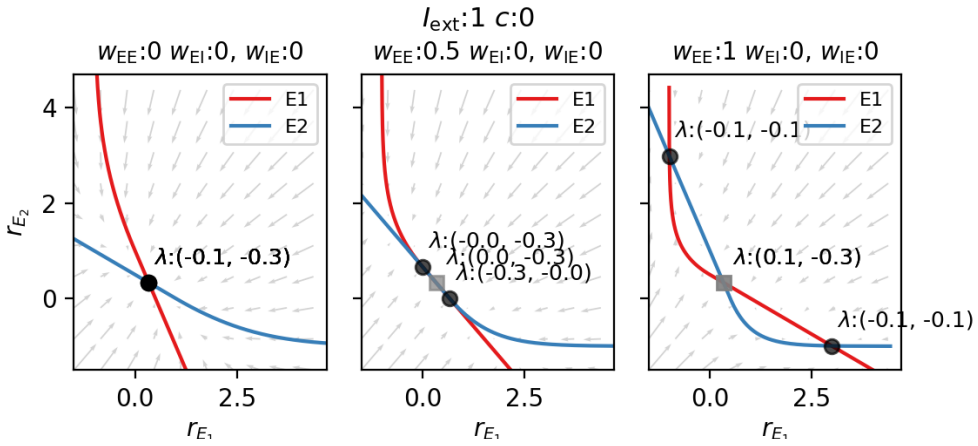


Figure 6.2: Strong recurrent **excitation** can amplify signals and create persistent activity states, crucial for working memory and decision stability. **(Left)** Non-specific recurrent excitation. **(Center)** Excitatory recurrence gives rise to a saddle point between two attractors of persistent neural activity associated with the two decisions. **(Right)** Strong excitatory recurrence increases the separation distance between the two attractors, i.e. the time it takes to switch between them increases (speed decreases), and the system moves further away from criticality or fast responsiveness and into stability. Annotations (λ) represent the two eigenvalues of the Jacobian $\mathbf{J}_{\mathrm{E}}$ of the gradient vector $^{\mathrm{dr_E}}/_{\mathrm{dt}}$ at the fixed points.

Inhibitory feedback Inhibitory feedback (modulated by w_{EI} and w_{IE}) is selective for one of the two decisions if $w_{\mathrm{EI}} > 0$, since the input signal from the E populations becomes distinct for the two I populations. Furthermore, as Fig. 6.3 shows, inhibition influences decision-making by modulating excitatory activity whenever $w_{\mathrm{IE}} \neq 0$. The inhibitory response can either stabilize excitatory activity, preventing runaway activity by suppressing competing choices ($w_{\mathrm{IE}} > 0$, (Center), or enhance competition ($w_{\mathrm{IE}} < 0$, (Right)).

3.2 E/I in Value-based Control

Having analyzed the effects of E/I interactions in decision-making at the circuit level, we now investigate how these principles inform *preference learning* and *adaptive behaviour* in a reinforcement learning (RL) agent. Specifically, we embed separate excitatory and inhibitory decision-selective preferences into an Actor-Critic framework.

Biological Justification The use of separate excitatory and inhibitory preferences is loosely motivated by cortical microcircuits, where excitatory interactions primarily support

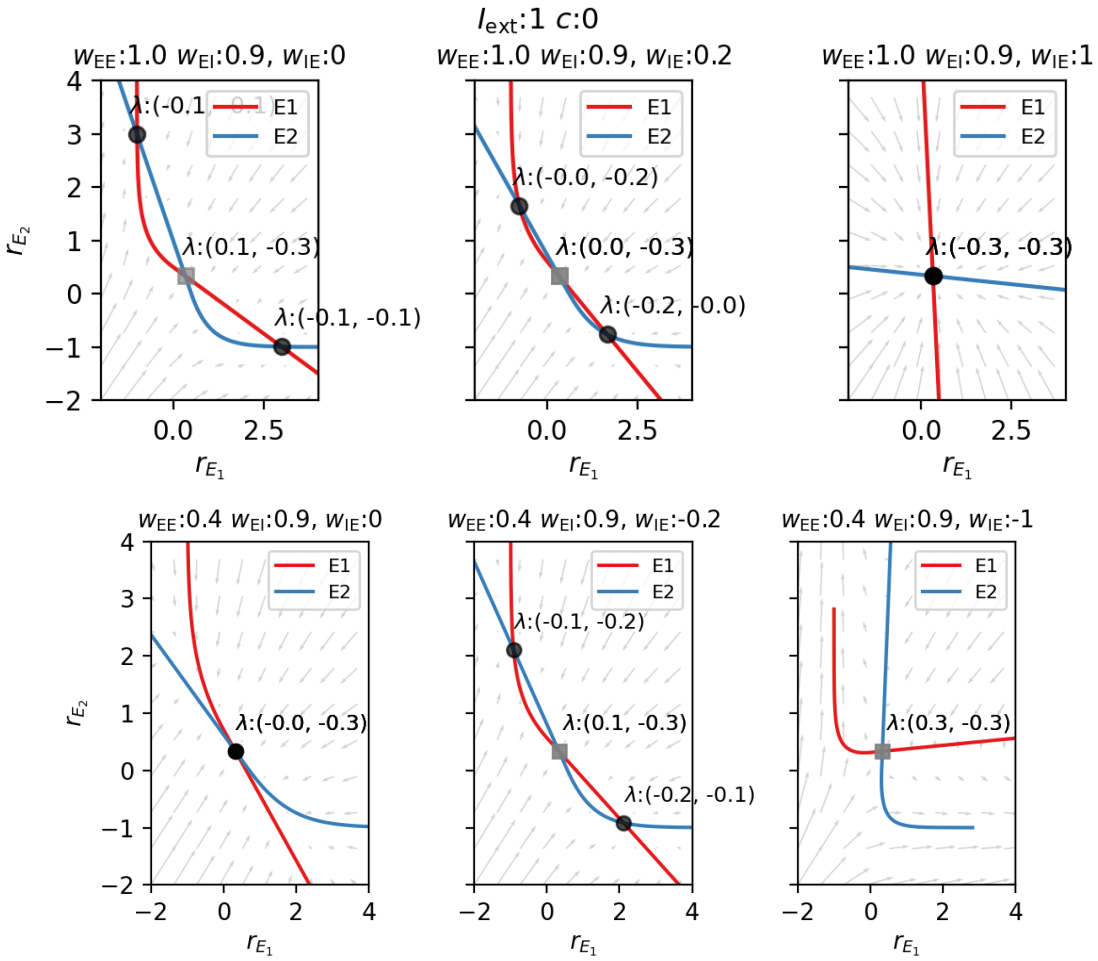


Figure 6.3: Selective Inhibitory modulation of decision-making. Selective inhibition can stabilize or enhance competition. **(Top-Left)** Decision-agnostic inhibition ($w_{\mathrm{EI}} > 0, w_{\mathrm{IE}} = 0$) has no impact on decision-making or the phase-space of the E populations. **(Top-Center)** Ipsi-specific inhibition (i.e. the decision specificity of the excitatory and inhibitory populations is the same) acts antagonistically to recurrent excitation and can push attractors closer together and bring the system closer to criticality—i.e. highly sensitive to small changes in the inputs, which can increase flexibility, but can also lead to instability. **(Top-Right)** Strong ipsi-specific inhibitory feedback can collapse the decision-making circuit to a single attractor. **(Bottom-Left)** Circuit collapses to a single attractor in the absence of strong recurrent excitation. **(Bottom-Center)** Contra-specific inhibition (i.e. the decision specificity of the excitatory populations is the opposite of the inhibitory populations) enhances competition and rescues the saddle point. **(Bottom-Right)** Strong contra-specific inhibition destabilizes the fixed points, and generates diverging dynamics.

goal-directed signals and reward-seeking, while inhibitory mechanisms shape these signals for flexible behaviour (cf. Miller and Cohen 2001, Tremblay et al. 2016, Wang 2002). This functional abstraction captures the core computational antagonism between excitation

and inhibition in an RL setting, and allows us to examine how inhibitory feedback on the preference function modulates behavioural flexibility in dynamic environments.

Setting *Cognitive flexibility*—the ability to adapt to changes in the environment [Cools, 2015]—is crucial for continual learning. Here, we focus on *reversal learning*—a common paradigm employed in behavioural assays designed to asses flexibility. In particular, we consider interrelated Markov Decision Processes (MDPs) (also called tasks), defined as a series of bandit problems varying only in their parameterization. We consider a simple two-armed bandit task. The arm distributions are Bernoulli distributions (rewards are 1 with probability p and 0 with probability $1-p$), where the parameters of each arm (p_1 and p_2) are correlated $p_1 = 1 - p_2$. Initially $p_1 = 0.8, p_2 = 0.2$. At the beginning of each task, the arms probabilities are reversed and held constant for 1K timesteps/trials. The optimal strategy in this setting is rapid adaptation to changed reward contingencies by discarding outdated associations.

Agent The optimal learning algorithm for a particular environment can be determined by creating a probabilistic model of the environment based on the task distribution, and then, using Bayes rule, to infer what outcomes are most likely given previously observed events. In RL, classic methods make the assumption that the reward for an environment is Markovian—i.e. determined by the state the agent is in and the action it takes in that state. In continual RL, different environmental states are not independent, but instead states might get *remapped* as the context of experience (task/MDP) changes, or as they might be affected by other general environmental factors. An efficient learning algorithm would update its expectations of all contextual states that are causally affected by the same factor when experiencing an RPE. We propose to bias the value estimates using a quantity that reflects the perception of outcomes in previous environmental contexts.

Specifically, the agent is an Actor-Critic (AC) with decision-specific excitatory (E) and inhibitory (I) preferences. Only the excitatory preferences are used in the behaviour update, and these track expected reward as is normally the case in standard ACs. To

integrate inhibition into the model, we add an additional modulatory input from inhibitory preferences which integrate a different input over time, namely reward-prediction errors (RPEs).

$$Q_{\mathrm{E}}(A) \leftarrow (1-\alpha)Q_{\mathrm{E}}(A) + \alpha R - w_{\mathrm{I}}Q_{\mathrm{I}}(A)$$
$$Q_{\mathrm{I}}(A) \leftarrow (1-\alpha)Q_{\mathrm{I}}(A) + \alpha\delta$$

where $\delta \doteq R - Q_{\mathrm{E}}(A)$ is the reward prediction error (RPE), α is a step-size, R is the reward, and $A \sim \pi$ is the action sampled from the policy π.

If reward feedback is noisy or dynamically changing, the excitatory preferences can either overfit to recent outcomes or fail to accumulate stable feedback. However, if inhibitory feedback is responsive to such volatility, then the inhibitory connectivity w_{I} modulates how strongly this feedback impacts the excitatory preferences, and the resulting behaviour. The policy uses a recurrent update rule, the entropy-regularized policy mirror descent update [Bhandari and Russo, 2021]

$$\pi'(a) \propto \pi(a)^{1-\tau}\ \exp\big(w_{\mathrm{E}}\, Q_{\mathrm{E}}(a)\big)$$

where $\tau \in [0, 1)$ controls the entropy; entropy regularization prevents policies from picking near deterministic actions and essentially lower bounds $\pi(A^*)$. The precision w_{E} modulates the influence of the excitatory preferences on action selection. Functionally, this controls the integration of preferences into behaviour. Higher values of w_{E} lead to more decisive updates, encouraging greedy behaviour with respect to preferences and rapid adaptation, but at the cost of increased volatility when preferences are inaccurate. In contrast, lower w_{E} values produce more conservative updates, promoting stability at the expense of speed. This mechanism captures a controllable trade-off between decision speed and accuracy.

Results & Observations

We analyze the decision-making performance of the agent under two precision regimes: **high precision** (large w_{E}) and **low precision** (small w_{E}), where w_{E} determines the weight assigned to excitatory Q-values and influences how the agent processes new information. All figures show the mean and standard deviation (std) over 50 seeds.

High precision (large w_{E}) In the high-precision setting, large w_{E} causes the agent to overweigh recent evidence, leading to *greedy* decision-making primarily based on the most recent excitatory preferences (Q_{E}), rather than integrating prior experience. This accelerates learning if preferences are accurate, but may also result in premature decisions when reward estimates fluctuate. In a highly non-stationary task distribution, this strategy can be beneficial for rapid adaptation but risks *instability* and *impulsivity*, as small variations in reward history introduce bias.

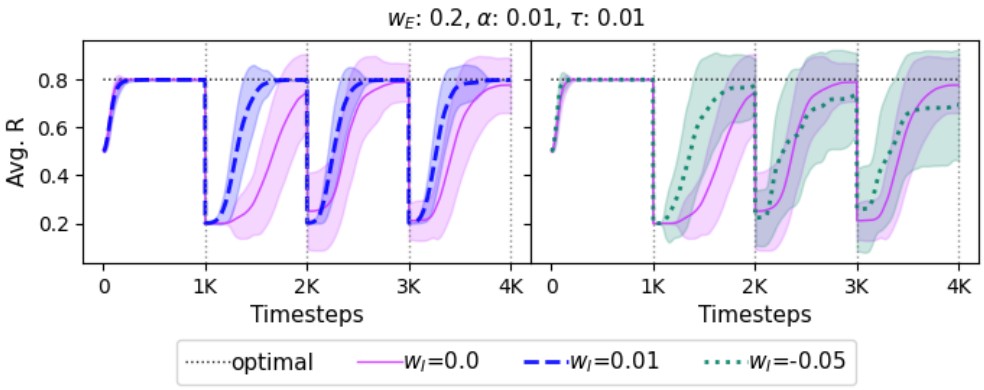


Figure 6.4: High precision (large w_{E}): The performance of the high precision (no-inhibition baseline $w_{\mathrm{I}} = 0$) is sensitive to the inherent volatility of tasks. Dotted lines show the policy performance for two inhibitory feedback settings (w_{I}): **(Top) Ipsi-specific inhibition ($w_{\mathrm{I}} > 0$)** stabilizes learning, reduces variance, and improves average performance in moderation; **(Bottom) Contra-specific inhibition ($w_{\mathrm{I}} < 0$)** increases variance and instability, leading to lower average performance.

As shown in Fig. 6.4, **ipsi-specific inhibition** ($w_{\mathrm{I}} > 0$) counteracts instability by regularizing excitatory preferences, improving robustness (Top). Conversely, **contra-specific inhibition** ($w_{\mathrm{I}} < 0$) enhances adaptability, but it also increases competition and volatility, leading to unstable decision-making (Bottom).

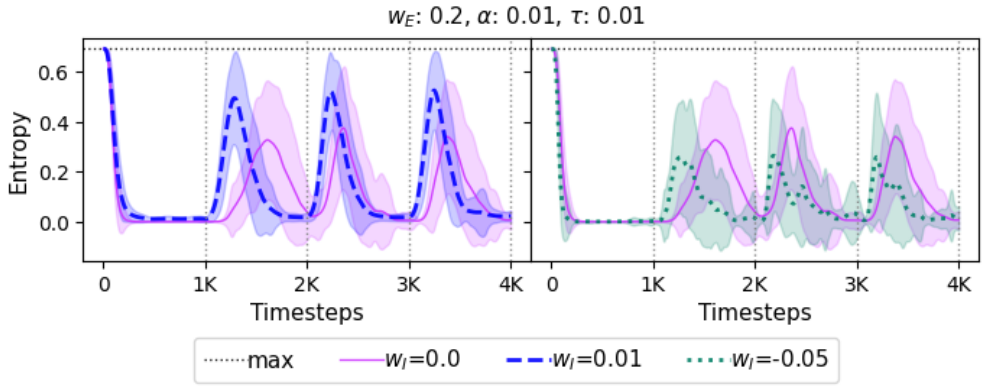


Figure 6.5: Policy entropy (high precision regime: large w_{E}): High precision (no-inhibition baseline $w_{\mathrm{I}} = 0$) results in impulsive decision-making, with excessive reliance on recent evidence. Dotted lines show policy entropy for two inhibitory feedback settings (w_{I}): **(Top) Ipsi-specific inhibition ($w_{\mathrm{I}} > 0$)** initially increases exploration, followed by rapid evidence accumulation. **(Bottom) Contra-specific inhibition ($w_{\mathrm{I}} < 0$)** leads to overall lower entropy and indecisive behaviour.

Fig. 6.5 illustrates that **ipsi-specific inhibition** ($w_{\mathrm{I}} > 0$) facilitates initial exploration but quickly stabilizes decision-making (Top). In contrast, **contra-specific inhibition** ($w_{\mathrm{I}} < 0$) leads to indecisive behaviour with lower entropy across learning phases (Bottom).

Low precision (small w_{E}) At low precision, small w_{E} results in conservative decision-making. Policy updates become sluggish, causing responses to be perseverative, with strong reliance on prior behaviour and reduced sensitivity to new evidence. This weakens the agent's ability to adapt efficiently.

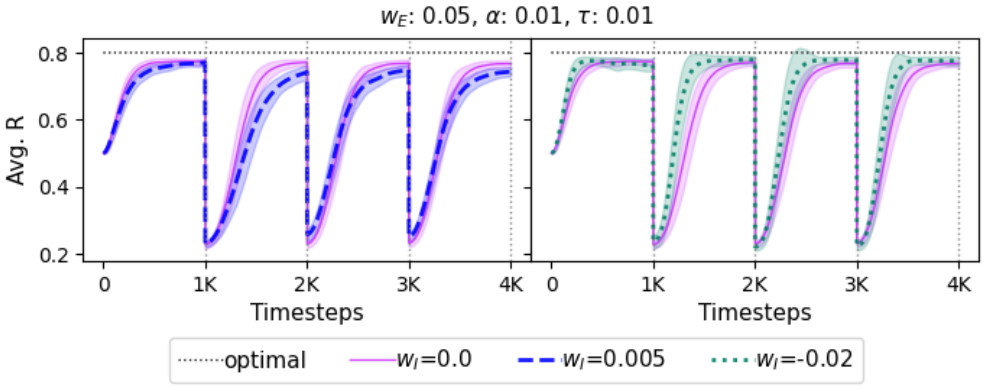


Figure 6.6: Low precision (small w_{E}): In low precision settings (no-inhibition baseline $w_{\mathrm{I}} = 0$) learning is slow. Dotted lines show policy performance for two inhibitory feedback settings (w_{I}): **(Top) Contra-specific inhibition ($w_{\mathrm{I}} < 0$)** increases competition and restores rapid adaptation. **(Bottom) Ipsi-specific inhibition ($w_{\mathrm{I}} > 0$)** exacerbates indecisiveness.

Fig. 6.6 shows that **ipsi-specific inhibition** ($w_{\mathrm{I}} > 0$) further slows learning, reinforcing over-regularization and indecisiveness (Top). In contrast, **contra-specific inhibition** ($w_{\mathrm{I}} < 0$) injects momentum, increasing competition and accelerating adaptation (Bottom).

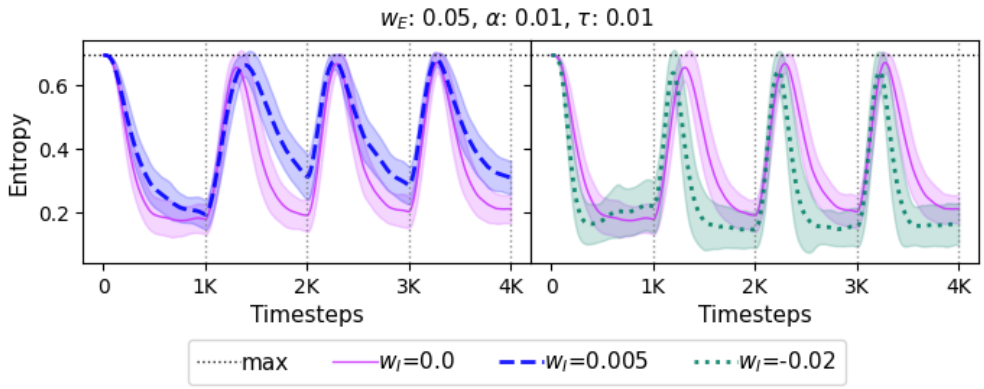


Figure 6.7: Policy entropy (low precision regime (small w_{E})): Low precision (no-inhibition baseline $w_{\mathrm{I}} = 0$) results in indecisive, exploratory policies that fail to fully converge. Dotted lines show policy entropy for two inhibitory feedback settings (w_{I}): **(Top)** **Contra-specific inhibition** **($w_{\mathrm{I}} < 0$)** accelerates learning, increasing contrast between competitive decisions. **(Bottom)** **Ipsi-specific inhibition** **($w_{\mathrm{I}} > 0$)** slows adaptation due to over-regularization.

Fig. 6.7 further demonstrates that positive inhibition ($w_{\mathrm{I}} > 0$) leads to persistent exploration, preventing convergence to a stable strategy (Top). In contrast, negative inhibition ($w_{\mathrm{I}} < 0$) enhances contrast between decisions, accelerating adaptation (Bottom).

Preference Dynamics To better understand how inhibitory feedback influences the agent's learning dynamics, Fig. 6.8 shows the average trajectory of the policy over time (z-axis) and across regimes. We observe that a similar effect as in Fig. 6.3, where ipsi-specific inhibition induces more stability by pushing the policy probabilities closer together, whereas contra-specific inhibition enhances competition.

Speed-accuracy trade-off The speed-accuracy trade-off governs the balance between how quickly an agent adapts to environmental changes and how accurately it maintains performance. If learning occurs too rapidly (high speed—e.g., high precision or contra-specific inhibition), the agent may become overly sensitive to recent evidence, leading to instability and potential divergence. Conversely, if learning is too slow (low speed—e.g., low precision or ipsi-specific inhibition), the agent may struggle to adapt efficiently to

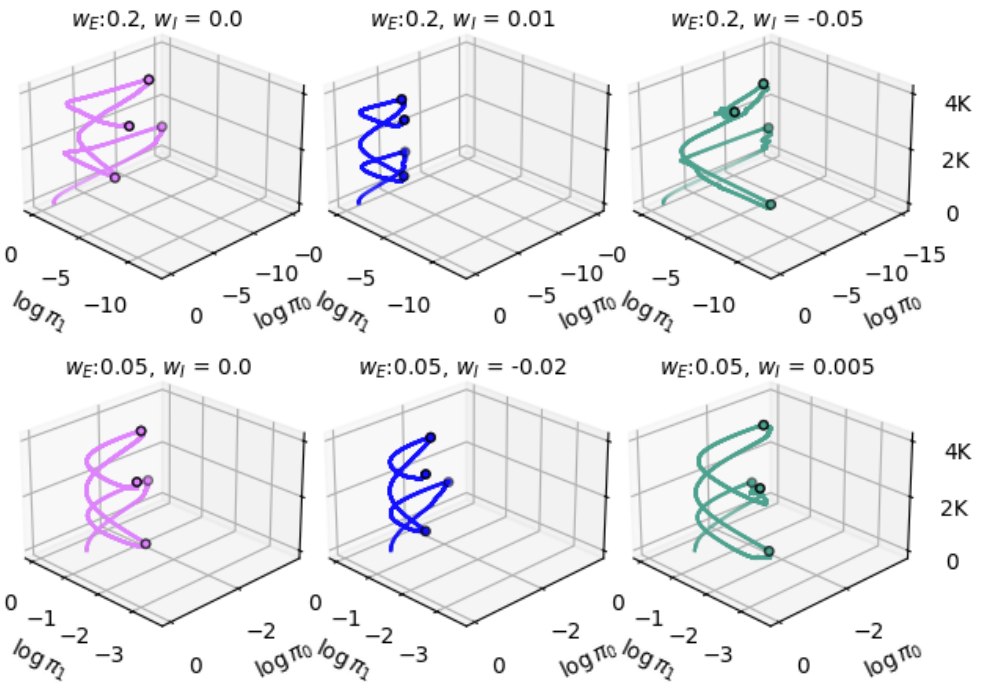


Figure 6.8: Inhibitory modulation of the policy in adaptive decision-making. **(Top)** High precision (large w_{E}. **(Bottom)** Low precision (small w_{E}). **(Left)** **no-inhibition baseline** $w_{\mathrm{I}} = 0$. **(Center)** **Ipsi-specific inhibition** ($w_{\mathrm{I}} > 0$) enhances flexibility by normalizing policy probabilities. **(Right)** **Contra-specific inhibition** ($w_{\mathrm{I}} < 0$) increases competition between decisions.

shifting task demands, resulting in suboptimal performance. This trade-off is crucial for effective continual learning. Our results show that inhibition provides a mechanism to modulate this trade-off, enabling an agent to adjust behaviour flexibly in non-stationary environments.

3.3 E/I in Recurrent Neural Networks

While earlier sections focused on analytical and algorithmic levels, we now examine how E/I constraints operate in scalable models of decision-making with learned weight connectivity. Recurrent neural networks (RNNs) are widely used in modelling partially observable decision-making, yet standard architectures lack biological structure. To explore whether E/I principles generalize to such models, we compare standard RNNs with architectures that enforce E/I separation during learning. Our goal is to assess whether these biological constraints can coexist with continual learning and adaptive decision-

making. Additional experimental details, pseudo-code and parameters used in numerical simulations are provided in Appendix 3.

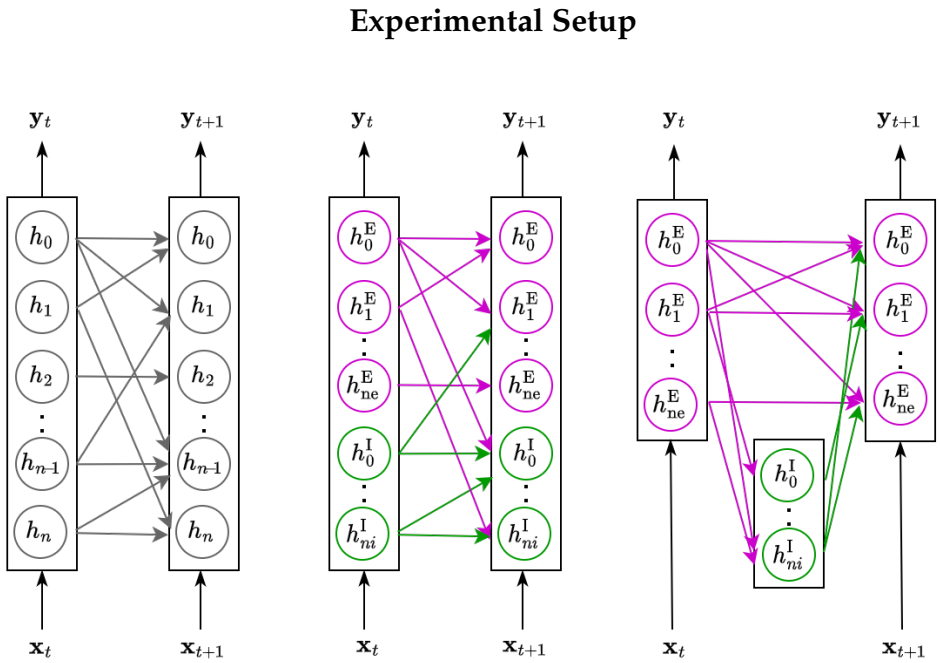


Figure 6.9: RNN architectures. (Left) **Vanilla RNN**. **(Center) ColEI** separates within layer neurons into E and I. **(Right) DANN** places I neurons between E layers.

We evaluate and compare three types of RNNs (Fig.6.9) to assess their ability to learn adaptively in decision-making tasks:

- **Standard/vanilla RNNs**: recurrent networks where synaptic weights are unconstrained by sign. The hidden layer activations of a vanilla RNN at time t, $\mathbf{h}_t$ are defined as

$$\mathbf{h}_t \doteq f_{\mathrm{rec}}(\mathbf{W}_{\mathrm{in}}\mathbf{x}_t + \mathbf{W}_{\mathrm{rec}}\mathbf{h}_{t-1} + \mathbf{b}_{\mathrm{rec}})$$

  where f_{rec} is the activation function, $\mathbf{x}$ is the input, $\mathbf{W}_{\mathrm{in}}$ and $\mathbf{W}_{\mathrm{rec}}$ are the input and recurrent synaptic weight matrices, and $\mathbf{b}$ is the bias. The output $\mathbf{y}_t$ is defined as

$$\mathbf{y}_t \doteq f_{\mathrm{out}}(\mathbf{W}_{\mathrm{out}}\mathbf{h}_t + \mathbf{b}_{\mathrm{out}})$$

where $\mathbf{W}_{\text{out}}$ is the output weight matrix, $\mathbf{b}_{\text{out}}$ is the output bias, and f_{out} is an optional output non-linearity. We set f_{out} based on the task.

- **Column Excitation-Inhibition (ColEI) RNNs**: naively partitioned E/I networks, i.e. networks where neurons are strictly excitatory or inhibitory, with sign-constrained weight matrices (cf. Song et al. [2016]). Specifically, the input weights $\mathbf{W}_{\text{in}}$ are positive and all other weight matrices are constrained to have columns whose entries are the same sign, either all positive, corresponding to the output weights of presynaptic excitatory neurons, or all negative for inhibitory neurons.
- **Dale's Artificial Neural Networks (DANNs)**: an E/I network architecture inspired by feed-forward inhibition in the brain [Cornford et al., 2021, Pouille et al., 2009], which keeps E and I neuron populations separate, but reparameterizes all weight matrices $\mathbf{W}$ in the network so learning remains unconstrained: $\mathbf{W} = \mathbf{W}_{\text{EE}} - \mathbf{W}_{\text{EI}}\mathbf{W}_{\text{IE}}$, where $\mathbf{W}_{\mathbf{ee}}$, $\mathbf{W}_{\text{EI}}$, and $\mathbf{W}_{\text{IE}}$ denote the $\text{E} \to \text{E}$, $\text{E} \to \text{I}$, $\text{I} \to \text{E}$ projections, and are sign-constrained by clipping weights to 0 after each update. Empirically, this preserves performance on challenging supervised learning tasks [Cornford et al., 2021, Li et al., 2023].

We explore how different E/I partitioning schemes affect learning. We test the RNNs on two settings. All experiments were run with PyTorch. Unless otherwise stated, all presented results were averaged over 50 seeds and shaded error bars denote standard deviation. We use $f_{\text{rec}}(\mathbf{x}) = \max(\mathbf{x}, \mathbf{0})$ (ReLU) throughout.

Setting (i): Decision-Making Task & Supervised Learning

First, we explore how different E/I partitioning schemes affect learning efficiency in a *decision-making* task (similar to the one used in Sec. 3.1, and inspired by the methodology of Roach et al. [2023]). The RNNs are trained to remain in a low activity state until stimulated and elevate the correct output in response to input stimuli.

Learning protocol An RNN-based agent is trained to integrate two parallel input streams of stimulus evidence

$$\mathbf{x}_t \doteq \begin{cases} \bar{x} + [(1 - x \cdot c), (1 + x \cdot c)]^\top + \eta_t, & \text{if } t \in [t_{\text{on}}, t_{\text{off}}] \\ \bar{x} + \eta_t, & \text{if } t \notin [t_{\text{on}}, t_{\text{off}}] \end{cases}$$

where $\bar{x}$ is a base background signal, x is the stimulus magnitude, $c \in [-1, 1]$ is the stimulus coherence, t_{on}, t_{off} represent the time of stimulus onset and offset, respectively, and $\eta_t \sim \mathcal{N}(\mu, \sigma)$ is a noise term sampled from a Gaussian distribution with mean μ and standard deviation σ. The network is trained to elevate the output activation corresponding to the input stream with a higher mean value. We use identity as the activation function f_{out} over the output units $\mathbf{y}$.

RNNs are trained by *backpropagation through time (BTT)* [Rumelhart et al., 1986] with *supervised learning* and *mean-square error* between the network outputs and supervised targets. The targets $\mathbf{y}^*$ are set to a constant baseline value of $\bar{\mathbf{y}}$ and, following the stimulus onset, are elevated to a value of $\mathbf{y}_{\max}$ for the output corresponding to the correct choice. For half of training trials no stimulus is presented and target values remain $\bar{\mathbf{y}}$ throughout the trial. A training batch of 200 trials is randomly generated every training epoch. Within the batch, for trials that have a stimulus input, the strength of the stimulus (and thus the difficulty of the trial) varies uniformly random with coherence sampled within the interval $c \sim \mathcal{U}([-0.2, 0.2])$. The losses incurred during the simulated period are masked to encourage the networks to perform evidence integration. We use Adam [Kingma and Ba, 2015] as the optimizer.

Evaluation protocol We report the accuracy over the test set, measured as the fraction of correct choices out of 100 trials. Trials are considered valid if (i) at least 75% of the pre-stimulus period was below a fixed decision threshold δ (i.e. $|y_0 - y_1| < \delta$, for $\mathbf{y} \doteq [y_0, y_1]^\top$), and (ii) at least 50% of the post-stimulus period is above the decision threshold ($|y_0 - y_1| > \delta$).

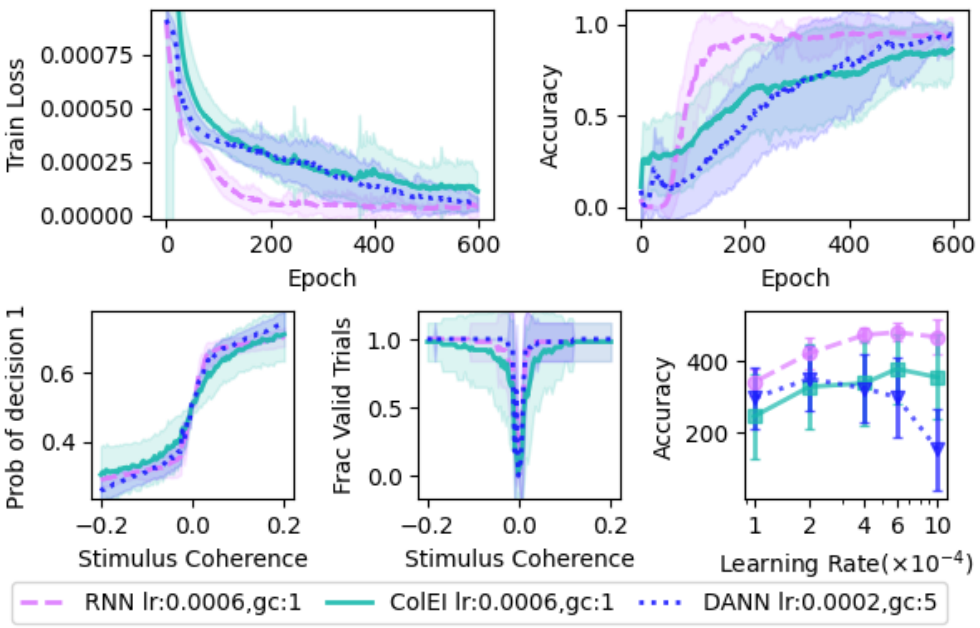


Figure 6.10: Performance of RNNs on Setting (i): We use shorthand notation lr for learning rate and gc for gradient clipping value. **(Top-Left) Training loss** per epoch. **(Top-Right) Test accuracy** measured as the fraction of correct choices out of all test trials. **(Bottom-Left) Psychometric function** evaluated over the test set, representing the probability of taking decision 1, is defined using a sigmoid activation function over the activity gap between the two decisions $\pi_0 \doteq {}^1/_{(1+\exp(-(y_0-y_1)))}$. **(Bottom-Center) Probability of trial completion** evaluated over the test set. Legend corresponds to these plots. Learning rates and gradient clipping values are searched over for each network. **(Bottom-Right) Sensitivity analysis** over the test set for lr. Values indicate area-under-the-curve (auc) and correspond to the best gc value for each lr computed with respect to the score = mean(auc) - std(auc). We keep the same colours and markers as in the general legend for RNN type.

Results & observations Fig.6.10 shows the training loss (Top-Left) and evaluation performance (Top-Right, Bottom) of the networks on the decision-making task. Detailed parameter sensitivity plots are included in Fig. D.1, D.2 and D.3, Appendix 4. We observe the two E/I models achieve comparable asymptotic performance, but learn slower compared to vanilla RNNs due to their constraints (Top-Right). Notably, the naive partitioning of ColEI presents challenges during training exhibiting larger variance, and struggles most with in making hard decisions, i.e. on trials with stimulus coherence c closer to 0 (Bottom-Center).

Fig.6.11 illustrates that output units increase activation following the onset of a stimulus corresponding to the chosen decision and maintain that activation after stimulus offset.

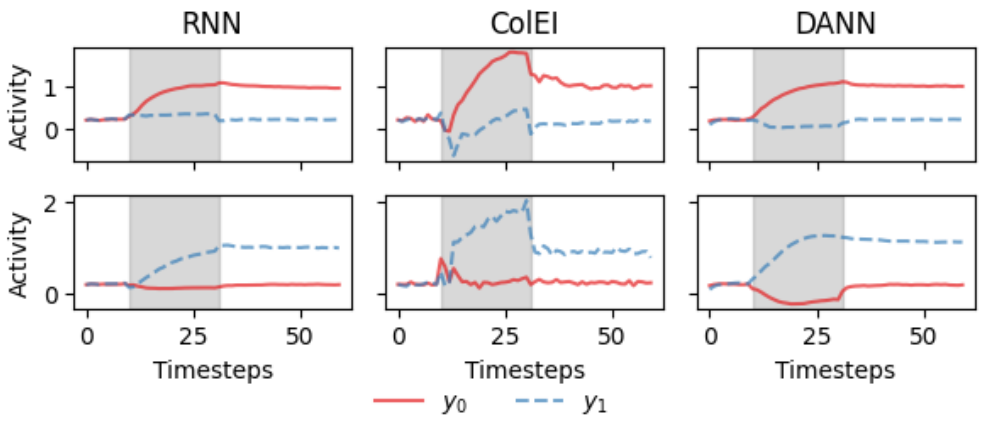


Figure 6.11: RNN Activity Traces: showing activity of output units in trained RNNs on two example trials with choice 1 (upper row, stimulus coherence $c = -0.2$) and choice 2 (lower row, stimulus coherence $c = 0.2$).

Setting (ii): Adaptive Decision-Making Task & CRL

Next, we define a *CRL* task for *adaptive decision-making*, in which an RNN-based RL agent is confronted with a series of interrelated tasks sampled from a task distribution $\mathcal{D}$. The tasks then differ from one another, but also share some underlying set of regularities. Specifically, we use a series of two-arm bandit problems varying only in their parameterization.

The arm distributions are Bernoulli distributions (rewards are 1 with probability p and 0 with probability $1 - p$). First, we consider task distributions $\mathcal{D}$ with correlated Bernoulli distributions, where the parameters (p_1, p_2) of the two arms are correlated in the sense that $p_1 = 1 - p_2$. We consider several such distributions over bandit environments: $\mathcal{D}_{\text{easy}}$ ($p1 \sim \mathcal{U}(\{\{0.1, 0.9\}\})$—correlated uniform distribution over those two possible values), $\mathcal{D}_{\text{uniform}}$ ($p1 \sim \mathcal{U}([0, 1])$—-correlated uniform distribution over the unit interval, $\mathcal{D}_{\text{medium}}$ ($p_1 \sim \mathcal{U}(\{0.25, 0.75\})$. We also test when the arm distributions are independent Bernoulli distributions $\mathcal{D}_{\text{indep}}$, where the parameters of each arm (p_1 and p_2) are sampled independently and uniformly over $[0, 1]$. Lastly, we denote with $\mathcal{D}_{\text{restless}}$ the bandit problem in which reward probabilities drift over the course of an episode, with each episode sampling a different rate of change (volatility) ν.

RL Agent The agent is an RNN-based Policy Gradient (PG) algorithm [Sutton et al., 1999b, Wang et al., 2016, Watkins and Dayan, 1992a]. Specifically, the output of the RNN $\mathbf{y}$, corresponding to the two arms, is passed through a softmax function, yielding a probability distribution over the action space $\mathcal{A}$, i.e. the agent's policy for acting in the environment $\pi(a) = {\exp \mathbf{y}[a]}/{\sum_{a' \in \mathcal{A}} \exp \mathbf{y}[a']}$, such that $\sum_{a \in \mathcal{A}} \pi(a) = 1$.

Agent-environment interaction At the start of a new episode, a new task and an initial state for this task are sampled, and the internal state of the agent (i.e., the pattern of activation over its recurrent units) is reset. The agent then executes its action-selection strategy in this environment for a certain number of discrete timesteps.

Learning The network weights are trained to maximize the sum of observed rewards over all steps and episodes. Through the use of RNNs, at each step, an action is executed as a function of the whole history of agent interaction in the current task during the current episode. Specifically, the RNN-based RL agent receives a concatenated input of the timestep, the action output on the previous step in one-hot format, and the reward resulting from that action, in line with the methodology of Wang et al. [2016]. The same reward is used to learn the weights $\mathbf{W}$ of the RNN by BTT using Monte-Carlo (MC) returns of the rewards collected throughout an episode of interaction with the bandit. In particular, we use approximate gradient ascent on the objective function

$$\mathcal{L}(\mathbf{W}) \doteq \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r(A_t) | \tau = (A_0, A_1 \dots), A_t \sim \pi_{\mathbf{W}}\right]$$

where r is a bounded reward function, T is the number of timesteps in an episode, $\gamma = 0.8$ is a discount factor, and τ denotes the whole trajectory. Here we use the standard REINFORCE algorithm [Williams, 1992] which updates the policy parameters in the direction $\nabla_{\mathbf{W}} \log \pi(a_t; \mathbf{W}) \sum_{k=t}^{T} \gamma^t r(a_k)$.

Evaluation protocol After training, the agent's policy is fixed (weights are frozen, but activations are changing due to input from the environment and the hidden state of the

recurrent layer), and it is evaluated on a set of tasks drawn from the same distribution $\mathcal{D}$. The internal state is reset at the beginning of each evaluation episode. Since the policy learned by the agent is history-dependent (as it makes uses of a recurrent network), when exposed to a new task, it should be able to adapt and deploy a strategy that optimizes rewards for it.

We evaluate the performance of the trained RNN-based RL agents using *cumulative regret*, a measure of the loss (in expected rewards) suffered when playing sub-optimal arms. Writing $r_k(a)$ the expected reward of arm a in bandit environment k, and $r_k^* = \max_a r_k(a) = r_k(a_k^*)$ (where a_k^* is the optimal arm) the optimal expected reward, we report the *performance* (average over bandit environments drawn from the test distribution)

$$E_{k\sim\mathcal{D}_{\text{test}}}[\mathcal{R}_T(k)], \text{ where } \mathcal{R}_T(k) = \sum_{t=1}^{T} r_k^* - r_k(a_t)$$

where a_t is the arm/action chosen at time t and $\mathcal{R}_T(k)$ is the cumulative regret (in environment k). We use $\mathcal{D}_{\text{test}} = \mathcal{D}_{\text{train}}$ in the numerical simulations.

Results & observations Fig.6.12 (Top) shows the performance of the three RNN networks on the $\mathcal{D}_{\text{easy}}$ distribution. Results and analysis for the other distributions are added to Appendix 4. Vanilla RNNs adapt quickly, but they lack biological plausibility. We find the naive partitioning of the RNN units into separate excitatory and inhibitory populations, as ColEI does, significantly impairs learning performance, supporting previous findings (cf. Li et al. [2023]). After running the network for 2K episodes, we observed performance improves much slower with considerable variance (Fig. D.5 (Top) in Appendix 4). DANNs strike a balance between biological realism and performance, demonstrating robust adaptability to non-stationary task distributions.

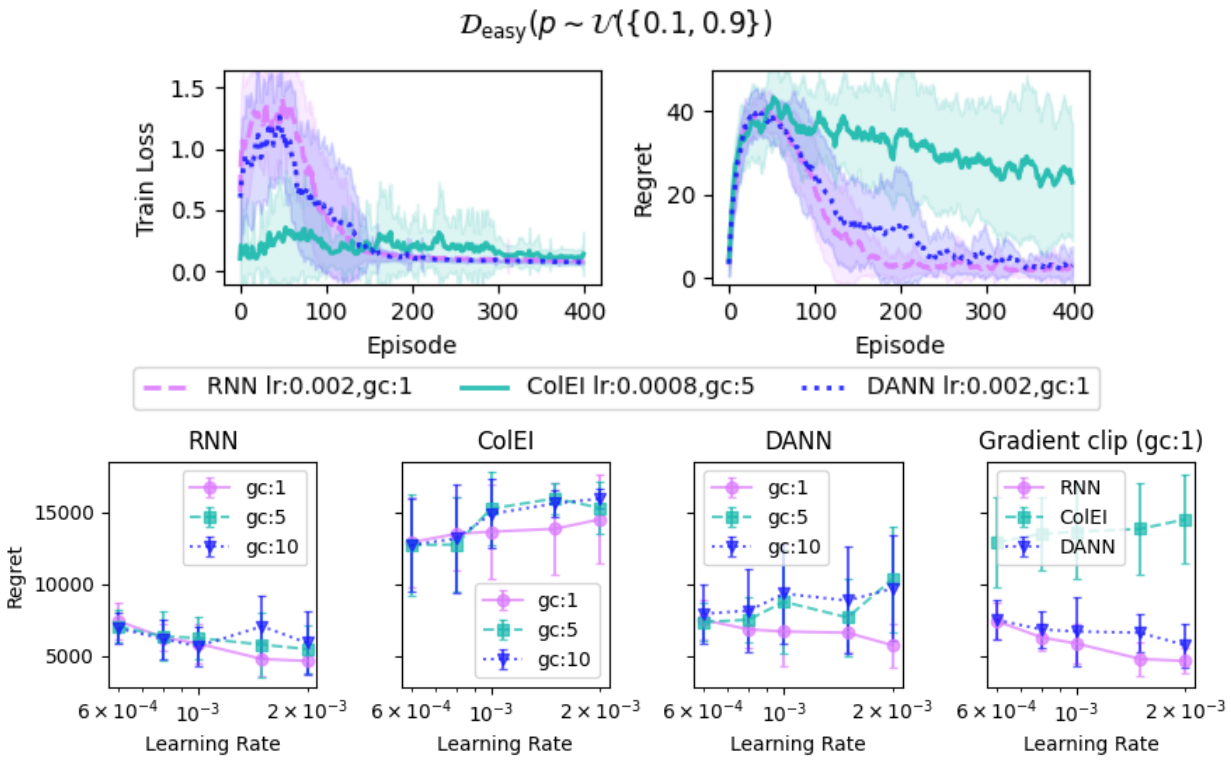


Figure 6.12: Performance of RNNs on Setting (ii): (Top-Left) Training loss per episode of experience. **(Top-Right) Evaluation performance** as average cumulative regret. Parameters are chosen based on score = mean(auc) - std(auc) cf. (Bottom). **(Bottom) Parameter sensitivity analysis** showing area-under-the-curve (auc). (Column 1-3) gc and lr per algorithm. (Column 4) lr for gc : 1.

4 Discussion

We investigated how selective excitatory–inhibitory (E/I) interactions may support adaptive learning and decision-making across three modelling levels of increasing complexity. Each level builds on insights from the previous, linking low-level dynamics to behavioural flexibility.

Our **main findings** are as follows:

(i) In a mean-field decision model, excitatory recurrence enhances responsiveness and amplifies decision signals, while inhibition stabilizes or sharpens competition between opposing choices. This suggests that E/I dynamics can tune the flexibility in decision-making.

(ii) Incorporating an E/I mechanism into the preference function of an RL agent offers a means of controlling the speed of adaptation in changing environments.

(iii) For RNN-based RL agents engaged in adaptive learning and decision-making, enforcing biological constraints (Dale's Law) in naive ways hinders learning, whereas a carefully constructed architecture, DANN, which enforces inhibition to be local to each layer, preserves sign-constrained connectivity while maintaining the ability to adapt in continual RL settings.

Taken together, our results suggest that structured E/I mechanisms may offer a computational means of adjusting behavioural flexibility. While the models we present are abstract and idealized, they provide a stepping stone toward understanding the role of E/I interactions in adaptive decision-making.

Future work should focus on further analyzing how E/I interactions in RNNs support behavioural flexibility in continual learning and adaptive-decision making tasks, by investigating phase-shifts. Comparing these models to neural data—e.g., inferring latent E/I circuit dynamics from rodent decision-making tasks—could help validate and refine the computational mechanisms proposed here.

7 Conclusion

"If the doors of perception were cleansed, everything would appear to man as it is, Infinite."

William Blake

The thesis weaves together threads drawn from multiple disciplines—reinforcement learning (RL) and decision-making, cognitive science, psychology, and computational neuroscience—united under a single, overarching question: **How can agents, whether biological or artificial, flexibly adapt their decision-making in complex or dynamic environments?** To address this central question, this thesis examines biological systems as a source of inspiration for building more adaptive artificial agents, and relies on computational models to generate hypotheses about how the brain maintains the flexibility and stability much needed for lifelong learning.

1 Summary of Contributions

This thesis has examined four interrelated themes that collectively advance our understanding of how agents, both artificial and biological, can navigate uncertainty, learn efficiently, and adapt in complex or dynamic environments:

Forethought and Hindsight in Credit Assignment Chapter 3 focused on model-based planning and reasoning. We showed how internal models enable anticipatory planning

and retrospective reasoning, allowing agents to simulate future outcomes and credit alternative actions for past outcomes. These complementary processes enhance learning efficiency, bridging model-free and model-based approaches.

Salience Maps for Selective Credit Assignment In Chapter 4, inspired by salience/attentional maps in biological systems, we proposed methods that prioritize learning updates. Focusing computational resources on impactful updates improves stability and convergence in noisy, sparse, or delayed reward environments, mirroring cognitive processes.

Acceleration in Policy Optimization Chapter 5 focused on integrating extrapolation techniques from optimization, particularly momentum, into RL to accelerate behaviour optimization. Harnessing recent experience to guide future adjustments accelerates learning, particularly in sparsely-connected MDPs and long-horizon settings.

Excitatory-Inhibitory Dynamics in Adaptive Learning and Decision-Making Chapter 6 investigated the computational role of excitation-inhibition (E/I) in shaping the neural dynamics of decision-making and adaptive learning. Using an E/I mean-field model we illustrated how recurrent excitation and selective inhibition shape attractor dynamics. Embedding these principles into an RL agent in a continual learning setting, we found that inhibition can mediate flexibility in dynamic environments. Scaling up these mechanistic principles to RL agents using E/I-constrained RNNs reveals naïve E/I separation impairs learning, whereas a structured architecture (DANN) preserves both biological plausibility and adaptive performance. Our results show E/I interactions may provide a computational mechanism for modulating flexibility in dynamic environments.

2 Discussion

Throughout these four chapters, we presented interrelated themes that illuminate ideas at the interface of these fields, offering insights for improving AI systems while deepening our understanding of biological cognition.

2.1 Interdisciplinary Perspectives: Biological Inspiration and Computational Frameworks

Collectively, our contributions strengthen the bridge between biological inspiration and artificial intelligence design, underscoring the value of an interdisciplinary approach.

A key motivation for this work is that insights from biological systems can inform the development of more robust computational agents. Biological organisms thrive in intrinsically non-stationary and uncertain environments, excelling at adapting to new conditions. They integrate novel information and make effective decisions despite incomplete or noisy sensory input. From an engineering perspective, artificial agents have historically struggled with analogous challenges: retaining old skills while incorporating new ones, learning new information without erasing previously acquired knowledge, adapting to changing task contingencies, and maintaining flexibility when faced with shifting reward structures and evolving contexts. By adopting an interdisciplinary framework, we can develop computational models that better reflect the complexity and adaptability observed in natural intelligence. The goal is not just to mimic nature but to draw inspiration from it, uncovering fundamental computational principles that can guide both scientific inquiry and technological innovation. In the words of Antoni Gaudí, "nothing is invented, for it is written in nature first".

This reciprocal relationship also means that computational models can propose formal hypotheses about neural mechanisms, guiding empirical research in neuroscience. Foundational concepts in reinforcement learning are deeply connected to neuroscience and

behavioural psychology, such as dopamine-mediated reward prediction errors (RPEs) and hippocampal-based cognitive maps. Model-free RL aligns with habitual, stimulus-driven responding supported by dopaminergic RPE signals and striatal circuits, while model-based RL corresponds to goal-directed, planning-oriented strategies involving prefrontal and hippocampal computations. Each concept can be enriched and validated by insights from different fields. Neuroscience provides a tapestry of cellular mechanisms, experimental evidence, and behavioural phenomena. Machine learning and RL offer mathematical rigour and formal frameworks. Cognitive science and psychology supply a conceptual vocabulary for linking subjective experiences—such as emotion, salience, motivation—to measurable effects on decision-making. Computational models can clarify and test precise hypotheses about brain function.

2.2 Continual Learning Agents: Necessity and Challenges

A second overarching theme addresses the necessity and difficulty of building agents that can learn continuously across their lifespans. Biological agents—from simple organisms to complex mammals—face changing food supplies, fluctuating social contexts, seasonal variations, and evolving threats. This persistent update of internal models and strategies, commonly referred to as lifelong learning, ensures that organisms remain viable despite external volatility.

By contrast, traditional artificial systems often assume a fixed distribution of tasks or a stationary environment. Many RL agents optimize a single policy until it converges, then struggle when confronted with new tasks or altered conditions [Hadsell et al., 2020]. Such agents often require costly retraining or explicit task segmentation [Dohare et al., 2021]. Real-world environments are often dynamic, requiring artificial agents, deployed in applications such as household robots, autonomous vehicles, conversational agents, and adaptive recommendation systems to cope with evolving conditions. Instead of converging to a single optimal policy, they must retain previously learned skills while incorporating

new ones, dynamically adjust their exploration-exploitation strategies [Luketina et al., 2022], remember information as tasks and contexts change [Kirkpatrick et al., 2017], and remain robust in the face of environmental volatility without "losing plasticity" [Dohare et al., 2021] or behavioural flexibility.

Continual reinforcement learning (CRL) [Abel et al., 2023, Khetarpal et al., 2020] seeks to overcome these limitations, pushing toward algorithms capable of lifelong adaptation. Biological analogues suggest solutions such as neural replay [Liu et al., 2019], selective memory consolidation [Cowan et al., 2021], and hierarchical or compositional representations that modularize knowledge [El-Gaby et al., 2024]. Understanding how the brain maintains stable yet flexible representations and how it can control learning speed and modulate plasticity can inform the design of artificial agents that do not need periodic resets or explicit task segmentation. This vision brings artificial agents closer to the adaptive intelligence displayed by living organisms.

In computational terms, continual RL must manage interference between old and new knowledge, addressing challenges related to "catastrophic forgetting" [Kirkpatrick et al., 2017] and "loss of plasticity" [Dohare et al., 2021]. Biological systems provide several candidate mechanisms for mitigating these problems: replay and preplay consolidate experience [Mattar and Daw, 2018], hierarchical and compositional representations modularize knowledge [El-Gaby et al., 2024], and neuromodulatory signals and E/I dynamics can control the learning process [Marder, 2012, Nadim and Bucher, 2014, Roach et al., 2023]. These insights can inform architectures that remain robust and flexible in open-ended settings.

2.3 Adaptive Learning Through Integrated Model-Based and Model-Free Approaches

Another central theme is integrating model-free and model-based learning strategies as a requirement for adaptive decision-making. Dopamine's role as a carrier of reward

prediction errors (RPEs) is anchored in the model-free tradition, where TD errors guide incremental updates to value functions [Schultz et al., 1997]. Dopamine encourages repeated engagement in rewarding actions and is integral to habit formation [Dolan and Dayan, 2013].

Model-free RL excels in stable environments where habitual responses can be incrementally tuned by RPE-driven updates. However, solely relying on model-free methods is suboptimal in dynamic environments where tasks change rapidly. Model-based systems excel in such conditions due to their flexibility and ability to update internal representations quickly, enabling accurate responses to new contingencies without requiring extensive retraining. Conversely, model-free systems rely on cached values, which makes them slower to adapt to shifting task demands and thus less effective in dynamic settings. This trade-off between speed and accuracy underscores the strengths and limitations of each system: while model-free strategies allow for quick, habitual responses, they falter when tasks change frequently [Kool et al., 2018].

Model-based methods use internal models of the environment to plan ahead, simulating future outcomes to guide behaviour proactively [Sutton and Barto, 2018]. Agents can also reason in hindsight, through counterfactual credit assignment as we have shown in Chapter 3. Therein, we highlighted how integrating forethought (prospective planning) with hindsight (retrospective reasoning) merges the complementary properties that each brings, with prospective planning instrumental when the future is predictable, and backward planning useful when hypotheses are causal in nature.

The combination of model-based and model-free mirrors the brain's dual decision-making systems: habitual circuits in the dorsolateral striatum (DLS) and model-based planning arising from prefrontal and hippocampal computations with the dorsomedial striatum (DMS) [Daw et al., 2005, 2011, Huang et al., 2020, Schreiner et al., 2020]. Robust adaptability emerges from fluid transitions between these strategies, selected as a function of environmental demands, complexity, uncertainty, and resource constraints [Kool et al.,

2018]. In Chapter 3, we highlighted that artificial agents also benefit from integrating model-based and model-free modules.

2.4 Cognitive Constructs for Efficiency in Resource-Limited Agents

Another key idea is that biological cognition operates under practical constraints [Dong et al., 2021, Gershman et al., 2015]. Natural intelligence does not have unlimited computational bandwidth, memory, or time. Instead, organisms use heuristics, selective attention, and processes shaped by cognitive biases, emotion, contextual salience, to optimize decision-making within real-world constraints [Dunsmoor et al., 2022]. The *bounded rationality* framework recognizes that agents seek good-enough solutions subject to limitations in energy, computation, and time.

Our long-term memory systems face the challenge of deciding which information to retain and which to discard amidst the constant influx of sensory input in daily life. Humans excel at preserving memories of significant events, whether positive, negative, or unexpected. These vivid and confidently remembered experiences stand out compared to mundane ones. Such memory enhancements result from a cascade of neurochemical processes that prioritize meaningful events at every stage of memory processing—encoding, consolidation, and retrieval. In essence, our memory systems focus on salient events marked by emotion, novelty, or surprise, selectively retaining and stabilizing information that aligns with our sense of well-being [Dunsmoor et al., 2022].

Emotional states profoundly influence actions, perception, and learning, and are believed to have been pivotal in enhancing human survival throughout evolution [Emanuel and Eldar, 2023]. Emotional tagging is a process where the amygdala marks emotionally significant events, enhancing their consolidation into long-term memory by modulating synaptic plasticity through neuromodulatory signals like noradrenaline [Richter-Levin and Akirav, 2003]. It selectively enhances associative memory for neutral events, offering

insights into adaptive memory processes and maladaptive generalization in conditions like PTSD [Zhu et al., 2022].

In Chapter 4, we proposed salience maps for selective credit assignment. By focusing on salient features and events, agents can allocate learning resources more efficiently. Biological analogues of this selective updating process are evident in selective attention, emotional salience tagging, and memory consolidation in the brain. These mechanisms may help agents identify what to remember, where to invest exploration, and when to prioritize updating value estimates. The result is improved sample efficiency, particularly in tasks with sparse or delayed rewards.

2.5 Extrapolation in Learning Behaviour

In a biological system, components of brain circuitry with slow dynamics over time may play a role in shaping behaviour in addition to the standard reinforcement signal. For instance, emotion, mood especially. Diffuse and subjective, distinct from emotions tied to specific events, mood may reflect integrated information about recent performance, environmental volatility, or fluctuating reward rates. Empirical evidence indicates that mood biases outcome interpretations, influences risk sensitivity, and modulates learning [Bakic et al., 2014, Bennett et al., 2022, Eldar et al., 2016]. Chapter 5, taking inspiration from these biological processes, proposes a momentum-based policy update that can leverage prior performance trends to shape behaviour.

2.6 E/I Dynamics and the Neural Substrate of Continual Reinforcement Learning

While classical RL models often assume a stable representational substrate, the brain is an evolving dynamical system [Gerstner et al., 2014]. The activity of a neural network constantly evolves in time, and for this reason, neurons can be modelled as dynamical systems. Rather than focusing on individual neurons in isolation, the dynamical systems

framework emphasizes their collective behaviour, where the interactions within the group define the computations being performed [Khona and Fiete, 2022, Langdon et al., 2023]. This approach views neural population activity as trajectories moving through a complex, high-dimensional space, shaped by the system's inherent dynamics [Sussillo, 2014]. Rather than modelling and simulating individual neurons (e.g. Linear Leaky-Integrate-and-Fire model (LIF) [Lapicque, 1907]), we can represent them as a uniform population and describe their dynamics with a single equation that tracks the time evolution of their average firing rate.

It remains unknown how exactly the dynamics of neuronal networks affect the representation and processing of information in the brain. Signatures of altered brain dynamics are present in all psychiatric conditions. In this context, the brain's capacity for continual adaptation is not only a function of learned representations. Neuronal networks consist of both excitatory and inhibitory neuronal populations [Wilson and Cowan, 1972]. Analysis of cortical models and cortical data have led to the understanding that the operating point of the cortex is a balanced state in which synaptic excitatory and inhibitory currents across the cell membranes of cortical neurons are approximately equal [Isaacson and Scanziani, 2011, Shapley and Xing, 2013, van Vreeswijk and Sompolinsky, 1996]. Disruptions in the E/I balance are implicated in several psychiatric disorders, including anxiety disorders, depression, and schizophrenia [Tsuboi et al., 2024]. Anhedonia, although misleading from its naming, has little to do with pleasure or hedonic function, and all to do with reduced responsiveness to reward or feedback, characterized by deficits in reinforcement learning [Robinson and Chase, 2017, Shankman et al., 2014]. Anhedonia is a component of most (if not all) psychiatric conditions: depressive disorders, substance-related disorders, psychotic disorders, and personality disorders [Huys et al., 2013, Rømer Thomsen et al., 2015].

In Chapter 6 we investigate the computational role of excitation-inhibition (E/I) interactions in continual adaptation. Our findings suggest that selective excitation and inhibition can stabilize or amplify competition between alternative decisions, learning to a computa-

tional mechanism for modulating flexibility in continual learning. Functionally, inhibition acts as an automatic gain control of the decision-making policy, which is consistent with experimental studies [Shapley and Xing, 2013]. This kind of adaptation might prevent "loss of plasticity" [Dohare et al., 2021] and enhance an agent's capacity to adapt to changing tasks over extended periods. Neuromodulators such as dopamine and serotonin further tune these processes. The activation of neuromodulator receptors alters overall information transmission [Nadim and Bucher, 2014].

3 Limitations

- **Scalability and Engineering Challenges:** While the frameworks improve learning efficiency in controlled environments, their application to high-dimensional, real-world systems remains nontrivial, and requires careful engineering, experimentation and tuning.
- **Context-Dependence:** Internal models, salience functions, and momentum parameters may need customization, raising questions about generalizability.
- **Empirical Validation** Direct empirical validation is incomplete. Collaborations with neuroscientists and cognitive scientists, plus neural and behavioural data, are needed to test and confirm the biological plausibility of these mechanisms.
- **Continual and Non-Stationary Settings:** Performance in fully continual or non-stationary scenarios needs much more exploration.

Addressing these limitations requires a combination of theoretical refinement, empirical testing, and methodological innovation.

4 Directions for Future Work

This thesis lays the groundwork for numerous avenues of further inquiry:

4.1 Computational Modelling

- **Selectivity Analysis of E/I-constrained RNNs in Adaptive Decision-Making:** In Chapter 6, we used E/I-constrained RNNs to investigate how the interplay of excitation and inhibition (E/I) shapes neural dynamics during decision-making. Future work should analyze the phase space of the RNNs to understand the role of selectivity in the emergence and stability of attractor states. This analysis can inform the development of scalable models with more behavioural flexibility in dynamic environments.

- **Mechanisms of Unlearning and Extinction:** While much of the literature focuses on how agents acquire new knowledge, relatively little attention is given to how they *unlearn*. Biological systems leverage inhibitory circuits to suppress outdated or maladaptive associations, a process known as *extinction learning* [Hartley and Phelps, 2012, Quirk and Mueller, 2008]. Future research should model these substrates of *unlearning*, investigating how inhibitory modulation removes or attenuates no-longer-relevant information [Floris et al., 2024]. Insights gleaned here may inform algorithms that enable artificial agents to selectively discard obsolete knowledge, fostering robustness and flexibility in ever-changing environments.

- **Plasticity and Structural Adaptation:** Beyond synaptic weight changes, neural adaptation involves structural modifications such as synaptogenesis, dendritic branching, and spinogenesis [Hesselgrave et al., 2021, Ly et al., 2018, Shao et al., 2021]. These processes, combined with synaptic pruning, sculpt the connectivity landscape, balancing the creation of novel pathways with the elimination of redundant ones. Future computational models that incorporate such structural adaptations stand to improve our understanding of how the brain achieves both efficiency and flexibility. Such models could inspire scalable and biologically grounded strategies for plasticity-related challenges in artificial systems.

- **Mechanisms of Persistence and Dynamic Encoding of Experience:** Continual learning hinges on how experiences are encoded, integrated, and updated over time. Future studies should consider how hippocampal-prefrontal cortex (HPC-PFC) interactions guide schema formation, supporting the assimilation of congruent experiences while directing novel or disjoint events into orthogonal representational subspaces [Duncker et al., 2020, Guerreiro and Clopath, 2024]. Modelling how persistent encoding mechanisms facilitate pattern separation, memory linking, and selective suppression of redundant activity [Langdon and Engel, 2022, Langdon et al., 2023] can inform strategies to maintain coherence across tasks, reduce interference, and improve long-term knowledge retention.

4.2 Information Gathering

- **Integrating Neural Recordings, Pharmacological Interventions, and Modelling:** A crucial next step is to collaborate with behavioural neuroscientists to design experiments that record large-scale neuronal activity and behaviour during tasks that challenge continual learning. Such experiments could involve both control conditions and targeted manipulations of E/I balance through pharmacological interventions (e.g., psychedelics or other neuromodulatory drugs) or optogenetic perturbations that selectively alter excitation/inhibition through neuromodulation. By applying latent variable models to these datasets, we could track, in the learned underlying latent circuits, how changes in neural dynamics correspond to shifts in network plasticity and behaviour, potentially revealing phase transitions due to altered connectivity. These studies can inform how changes in E/I balance reshape the neural substrate for continual learning, providing empirically grounded insights into biologically inspired algorithms and architectures for artificial agents.

4.3 Evaluation

- **Empirical Studies in More Complex Neuroscience Tasks:** In this thesis we used probabilistic reversal learning tasks in Chapter 6. Future work should expand to more complex paradigms that probe flexibility, exploration, adaptation, and diverse agent strategies. Tasks requiring sustained attention and working memory would yield richer data and deeper insights, bridging the gap between controlled computational models and the complexity of real-world cognitive processes.

Collectively, these directions highlight a rich landscape of future research. By integrating computational models with empirical data, we can move closer to understanding the fundamental principles that underlie adaptive, continual learning.

5 Closing Remarks

This research highlights the intersection of biological and artificial systems in adaptive learning and decision-making. Although limited in scope, it raises key questions and suggests future directions for deeper integration of biological insights into computational models. Emphasizing the mechanisms governing neuroplasticity and continual learning may lead to more robust, flexible artificial agents.

While this thesis marks the conclusion of one line of inquiry, it underscores the need for continued interdisciplinary efforts to unite biological insights and computational frameworks.

A Forethought & Hindsight in Credit Assignment

1 Theoretical Background and Derivations

In the following, we use $\overleftarrow{\mathcal{P}}_\pi(s_{t-1}, a_{t-1}|s_t)$ as a shorthand for $\overleftarrow{\mathcal{P}}_\pi(S_{t-1} = s_{t-1}, A_{t-1} = a_{t-1}|S_t = s_t)$ to unclutter notation and drop the * superscript, where it is obvious we refer to the true transition model.

1.1 Backward Models are Policy Dependent

Using the stationarity assumption, we write $d_\pi(s)$ as the stationary distribution of π at s. The joint posterior of the previous state and action can be expressed using Bayes rule as

$$\begin{aligned}\overleftarrow{\mathcal{P}}_\pi(s_{t-1}, a_{t-1}|s_t) &= \frac{\mathcal{P}(s_t|s_{t-1}, a_{t-1}) d_\pi(s_{t-1}, a_{t-1})}{\sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} \mathcal{P}(s_t|s, a) d_\pi(x, a)} \\ &= \frac{\mathcal{P}(s_t|x_{t-1}, a_{t-1})\pi(a_{t-1}|s_{t-1}) d_\pi(s_{t-1})}{d_\pi(s_t)}\end{aligned}$$

where the last expression follows by expressing the prior as $\mathcal{P}_\pi(s_{t-1}, a_{t-1}) = \pi(a_{t-1}|s_{t-1}) d_\pi(s_{t-1})$, and folding the denominator into the stationary distribution of the policy at the next state $\sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} \mathcal{P}(s_t|s, a)\mathcal{P}_\pi(s, a) = d_\pi(s_t)$.

1.2 Action Conditioned Backward Models

Backward models mirror their counterparts – forward models. As such, one may define action-condition backward models, yet note that these too are dependent on a policy.

According to Bayes, the probability of a previous state $\overleftarrow{\mathcal{P}}(s_{t-1}|s_t, a_{t-1})$, conditioned on a *previous* action a_{t-1} and policy π, leading into the current state s_t is:

$$\overleftarrow{\mathcal{P}}_\pi(s_{t-1}|s_t, a_{t-1}) = \frac{\mathcal{P}(s_t|s_{t-1}, a_{t-1})\mathcal{P}_\pi(s_{t-1}|a_{t-1})}{\mathcal{P}_\pi(s_t|a_{t-1})} \tag{A.1}$$

Applying Bayes rule again for the probability of a previous state $\mathcal{P}_\pi(s_{t-1}|a_{t-1})$ conditioned on a previous action a_{t-1} and policy π

$$\mathcal{P}_\pi(s_{t-1}|a_{t-1}) = \frac{\pi(a_{t-1}|s_{t-1})d_\pi(s_{t-1})}{\pi(a_{t-1})} \propto d_\pi(s_{t-1}, a_{t-1}) \tag{A.2}$$

The denominator can be rewritten using Bayes as

$$\mathcal{P}_\pi(x_t|a_{t-1}) = \frac{\overleftarrow{\pi}(a_{t-1}|s_t)d_\pi(x_t)}{\pi(a_{t-1})} \propto d_\pi(s_t, a_{t-1}) \tag{A.3}$$

The posterior probability, i.e. the action-conditioned backward model for policy π becomes

$$\overleftarrow{\mathcal{P}}_\pi(s_{t-1}|s_t, a_{t-1}) = \frac{\mathcal{P}(s_t|s_{t-1}, a_{t-1})d_\pi(s_{t-1}, a_{t-1})}{d_\pi(s_t, a_{t-1})} \tag{A.4}$$

$$= \frac{\pi(a_{t-1}|s_{t-1})}{\overleftarrow{\pi}(a_{t-1}|s_t)} d_\pi(s_t)^{-1} d_\pi(s_{t-1})\mathcal{P}(s_t|s_{t-1}, a_{t-1}), \tag{A.5}$$

where $\overleftarrow{\pi}(a_{t-1}|s_t)$ gives the marginal probability of an action, conditioned on the future state s_t.

1.3 Multi-step models

Policy-conditioned models hold many shapes and have the potential to be useful in reasoning over larger timescales. To start, given a backward transition model $\overleftarrow{\mathcal{P}}_\pi$, we define $\overleftarrow{\mathcal{P}}_\pi^{(n)}(\cdot|s) = (\overleftarrow{\mathcal{P}}_\pi)^n(\cdot|s)$ as the predecessor state distribution of having followed policy π for n steps, arriving at state s. Similarly, we denote the associated n steps reward model as: $\overleftrightarrow{r}_\pi^{(n)}(\tilde{s}, s) = \mathbb{E}\left[\sum_{i=0}^{n-1} \gamma^i R_{i+1})|S_0 = \tilde{s}, S_n = x, A_{i+1} \sim \pi(\cdot|S_i)\right]$. For a *single-step* model

we can marginalize the actions from the joint predecessor and action model as: $\overleftarrow{\mathcal{P}}_\pi(\tilde{s}|s) = \sum_{\tilde{a}} \overleftarrow{\mathcal{P}}_\pi(\tilde{s}, \tilde{a}|s)$. An *n-step model* is one that predicts the probability of of a predecessor state n-steps in the past:

$$\overleftarrow{\mathcal{P}}_\pi^{(n)}(\tilde{s}|s) = \mathbb{E}_{\mathcal{F}(\pi)}\left[\overleftarrow{\mathcal{P}}_\pi(S_{t-1}, A_{t-1}, \ldots S_{t-n}, A_{t-n}|S_t)|S_t = s, S_{t-n} = \tilde{s}\right], \tag{A.6}$$

where we denote with $\mathcal{F}(\pi)$ – the forward Markov chain induced by policy π. An important property of n-step models is that they can seamlessly be substituted into Bellman evaluation equations.

A multi-time model is a combination of n-step models:

$$\overleftarrow{\mathcal{P}}_\pi^{(1:\infty)}(\tilde{s}|s) = \sum_{n=1}^{\infty} w_n \overleftarrow{\mathcal{P}}_\pi^{(n)}(\tilde{s}|s), \tag{A.7}$$

$$\overleftarrow{r}_\pi^{(1:\infty)}(\tilde{s}, s) = \sum_{n=1}^{\infty} w_n \overleftarrow{r}_\pi^{(n)}(\tilde{s}, s), \tag{A.8}$$

where $\sum_{n=1}^{\infty} w_n = 1$. In the general case, the number of steps we "remember" the past can be a random variable. We can augment the state space with an event indicating the decision to consider a state as predecessor (and to re-evaluate predictions from there on) or to look further back in time. If the probability associated with the event is given by a function $\lambda : \mathcal{S} \to [0, 1]$, we can write the model with the semantics of continuing the backward processing as $\overleftarrow{\mathcal{P}}_{\pi,\lambda}(\tilde{s}|s) = \overleftarrow{\mathcal{P}}_\pi(\tilde{s}|s)\lambda(\tilde{s})$, and denote initiation with $1 - \lambda(\tilde{s})$ at state $\tilde{s}$ and re-evaluating predictions.

N-step models involve a hard cutoff contingent on the number of steps. If λ is a smooth function the resulting models allow for a spectrum of time-extended models ranging from single-step models to n-step models and a wide variety in between. Having a λ independent of state defaults to interpolating between n-step predecessors with a mixture:

$$\overleftarrow{\mathcal{P}}_{\pi,\lambda}(\tilde{s}|s) = (1 - \lambda) \sum_{n=0}^{\infty} \lambda^n \overleftarrow{\mathcal{P}}_\pi^{(n)}(\tilde{s}|s), \tag{A.9}$$

λ here has the interpretation of a *trace decay function*. Likewise, a state dependent λ-*model* is instrumental when the scale of prediction varies. Therefore, a backward model conditioned on the smooth state-dependent λ function can choose to ignore inconsequential chaos lying in-between cause and effect [1]:

$$\overleftarrow{\mathcal{P}}_{\pi,\lambda}(\tilde{s}|s) = \sum_{n=0}^{\infty} \left(\prod_{i=0}^{n} \lambda(s_i) \right) \overleftarrow{\mathcal{P}}_{\pi}^{(n)}(\tilde{s}|s) \tag{A.10}$$

We can obtain recursive formulations for these models (similarly to Bellman equations):

$$\overleftarrow{\mathcal{P}}_{\pi,\lambda}(\tilde{s}|s) = (1 - \lambda(\tilde{s}))\, \overleftarrow{\mathcal{P}}_{\pi}(\tilde{s}|s) + \sum_{\tilde{\tilde{s}}} \lambda(\tilde{\tilde{s}}) \overleftarrow{\mathcal{P}}_{\pi}(\tilde{\tilde{s}}|s) \overleftarrow{\mathcal{P}}_{\pi,\lambda}(\tilde{s}|\tilde{\tilde{s}}). \tag{A.11}$$

Subsuming all other formulations, *backward option models* may also be defined as a special type of backward value functions tethered, not just to a policy, but to a triple $(\pi_o, \iota_o, \beta_o)$, where π_o is the decision policy, ι_o and β_o determine the initiation and termination of the option. A *backward option model* is defined as:

$$\overleftarrow{\mathcal{P}}_{\pi_{\mathcal{O}}}(\tilde{s}|s, o) = \iota_o(\tilde{s}) \overleftarrow{\mathcal{P}}_{\pi_o}(\tilde{s}|s) + \sum_{\tilde{\tilde{s}}} \left(1 - \iota_o(\tilde{\tilde{s}})\right) \overleftarrow{\mathcal{P}}_{\pi_o}(\tilde{\tilde{s}}|s) \overleftarrow{\mathcal{P}}_{\pi_{\mathcal{O}}}(\tilde{s}|\tilde{\tilde{s}}, o) \tag{A.12}$$

where $\iota(\tilde{s})$ denotes initiation at the predecessor state $\tilde{s}$ and $\pi_{\mathcal{O}}$ denotes a policy over options.

Extended time models can be learned on-policy by Monte-Carlo methods by taking samples of trajectories from the forward Markov chain $\mathcal{F}(\pi)$ induced by policy π:

$$\overleftarrow{\mathcal{P}}_{\pi}^{(n)}(\tilde{s}|s) = \mathbb{E}_{\mathcal{F}(\pi)} \left[\frac{\mathcal{P}_\pi(S_t, A_{t-1}|S_{t-1}) \ldots \mathcal{P}_\pi(S_{t-n+1}, A_{t-n}|S_{t-n}) d_\pi(S_{t-k})}{d_\pi(S_t)} | S_t = s_t, S_{t-n} = s_{t-n} \right] \tag{A.13}$$

$$= \mathbb{E}_{\mathcal{F}(\pi)} \left[\frac{\mathcal{P}_\pi(S_t|S_{t-n}) d_\pi(S_{t-n})}{d_\pi(S_t)} | S_t = x, S_{t-n} = \tilde{s} \right], \tag{A.14}$$

[1] Learning the λ function is part of the *'discovery problem'* and the best strategy still unclear, although a meta-learning strategy is perhaps effective [Xu et al., 2018].

or by TD-learning using a recursive formulation. For instance, λ-models could be updated with the rule:

$$\overleftarrow{\mathcal{P}}_{\pi,\lambda}(\tilde{s}|s_t) = \overleftarrow{\mathcal{P}}_{\pi,\lambda}(\tilde{s}|s_t) + \alpha \left((1-\lambda(s_t))\mathbf{1}_{s_{t-1}=\tilde{s}} + \lambda(s_t)\overleftarrow{\mathcal{P}}_{\pi,\lambda}(\tilde{s}|s_{t-1}) - \overleftarrow{\mathcal{P}}_{\pi,\lambda}(\tilde{s}|s_t) \right) \quad \text{(A.15)}$$

One can use any of these models in computing value functions, yet some types of models will have different properties. For instance a single-step model can be iterated or composed to obtain a trajectory. This possibility is available for full-distribution and sample models, except for expectation models, for which iteration is not possible since their output is not a real state. If we would have estimators for n-step models, $\forall n$, or a λ-model, notice that we can update backward all trajectories at once, in contrast to a single trajectory as is the case for eligibility traces [Sutton, 1988].

1.4 Off-policy Backward Models

In the general setting, we may be interested in learning *counterfactually* about many policies at the same time from a single stream of experience [Sutton et al., 2011]. We briefly note that this is possible using an *excursion formulation* [Gelada and Bellemare, 2019, Hallak and Mannor, 2017, Mahmood et al., 2015, Sutton et al., 2016a, Zhang et al., 2020a,b]. We defer investigation of the off-policy regime to future work.

2 Forward-Backward Equivalences

In this appendix we review some equivalences of forward and backward views and include for completeness some properties of forward and reverse Markov chains [cf. Morimura et al., 2010], along with other forward-backward equivalences [cf. Sutton, 1988].

2.1 Forward-Backward Views – Equivalence of Temporal Difference Learning

The temporal difference learning over an episode of experience can be equivalently interpreted in the off-line regime looking forward or backward [cf. Sutton, 1988]:

$$\mathbf{w} = \mathbf{w} + \alpha \sum_{t=0}^{\infty} \left[\sum_{k=t}^{\infty} \gamma^{k-t} R_{k+1} - V_{\mathbf{w}}(S_t) \right] \nabla_{\mathbf{w}} V_{\mathbf{w}}(S_t) \qquad \text{(forward view)} \quad \text{(A.16)}$$

$$= \mathbf{w} + \alpha \sum_{t=0}^{\infty} \sum_{k=t}^{\infty} \gamma^{k-t} \left[R_{k+1} + \gamma V_{\mathbf{w}}(S_{k+1}) - V_{\mathbf{w}}(S_k) \right] \nabla_{\mathbf{w}} V_{\mathbf{w}}(S_t) \quad \text{(A.17)}$$

$$= \mathbf{w} + \alpha \sum_{k=0}^{\infty} \sum_{t=0}^{k} \gamma^{k-t} \delta_k \nabla_{\mathbf{w}} V_{\mathbf{w}}(S_t) \quad \text{(A.18)}$$

$$= \mathbf{w} + \alpha \sum_{t=0}^{\infty} \delta_t \sum_{k=0}^{t} \gamma^{t-k} \nabla_{\mathbf{w}} V_{\mathbf{w}}(S_k) \qquad \text{(backward view).}$$

$$\text{(A.19)}$$

2.2 Relationship Forward-Backward Markov Chains

Detailed balance The relationship between the forward and the backward model can be written as

$$\overleftarrow{\mathcal{P}}_{\pi}(s_{t-1}, a_{t-1} | s_t) = \frac{\mathcal{P}(s_t | s_{t-1}, a_{t-1}) \pi(a_{t-1} | s_{t-1}) d_{\pi}(s_{t-1})}{d_{\pi}(s_t)} \quad \text{(A.20)}$$

$$= \frac{\mathcal{P}_{\pi}(s_t, a_{t-1} | s_{t-1}) d_{\pi}(s_{t-1})}{d_{\pi}(s_t)}. \quad \text{(A.21)}$$

Multiplying by $d_{\pi}(s_t)$ and summing over all actions a_{t-1} in the previous equation gives the **detailed balance equation** [MacKay, 2002]:

$$d_{\pi}(s_t) \overleftarrow{\mathcal{P}}_{\pi}(x_{t-1} | x_t) = \mathcal{P}_{\pi}(s_t | s_{t-1}) d_{\pi}(s_{t-1}) \quad \text{(A.22)}$$

Stationary distribution equivalence Summing over all states s_t

$$\sum_{s_t \in \mathcal{S}} \overleftarrow{\mathcal{P}}_\pi(s_{t-1}|s_t) d_\pi(x_t) = d_\pi(s_{t-1}) \tag{A.23}$$

we have d_π also as the stationary distribution under the backward Markov chain, and consequently that the forward and backward chains have the same stationary distribution.

Backward transition matrix The backward transition dynamics matrix can be written as

$$\overleftarrow{\mathcal{P}}_\pi = \operatorname{diag}(d_\pi)^{-1} \mathbf{P}_\pi^\top \operatorname{diag}(d_\pi) \tag{A.24}$$

2.3 Equivalence between Forward and Backward Planning

The expected parameter corrections for n-step learning updates are defined over the forward Markov chain $\mathcal{F}$ induced by a policy π and such can be redefined in terms of the backward Markov chain $\mathcal{B}$ as:

$$\mathbb{E}_{\mathcal{F}(\pi)} \left[\left(\overleftarrow{r}^{(n)}(S_t, S_{t+n}) + \gamma V_{\mathbf{w}}(S_{t+n}) - V_{\mathbf{w}}(S_t) \right) \nabla_{\mathbf{w}} V_{\mathbf{w}}(S_t) | S_{t+n}, d_\pi(S_t) \right] = \tag{A.25}$$

$$= \mathbb{E}_{\mathcal{F}(\pi)} \left[\left(\overleftarrow{r}^{(n)}(S_{t-n}, S_t) + \gamma V_{\mathbf{w}}(S_t) - V_{\mathbf{w}}(S_{t-n}) \right) \nabla_{\mathbf{w}} V_{\mathbf{w}}(S_{t-n}) | S_t, d_\pi(S_{t-n}) \right] \tag{A.26}$$

$$= \mathbb{E}_{\mathcal{B}(\pi)} \left[\left(\overleftarrow{r}^{(n)}(S_{t-n}, S_t) + \gamma V_{\mathbf{w}}(S_t) - V_{\mathbf{w}}(S_{t-n}) \right) \nabla_{\mathbf{w}} V_{\mathbf{w}}(S_{t-n}) | S_t \right], \tag{A.27}$$

since

$$\overleftarrow{\mathcal{P}}_\pi(s_{t-1}, a_{t-1}, \ldots, s_{t-n}, a_{t-n}|s_t) = \overleftarrow{\mathcal{P}}_\pi(s_{t-1}, a_{t-1}|s_t) \ldots \overleftarrow{\mathcal{P}}_\pi(s_{t-n}, a_{t-n}|s_{t-n+1}) \tag{A.28}$$

$$= \frac{\mathcal{P}_\pi(s_t, a_{t-1}|s_{t-1}) \ldots \mathcal{P}_\pi(s_{t-n+1}, a_{t-n}|s_{t-n}) d_\pi(s_{t-n})}{d_\pi(s_t)} \tag{A.29}$$

$$\propto \mathcal{P}_\pi(s_t, a_{t-1}|s_{t-1}) \ldots \mathcal{P}_\pi(s_{t-n+1}, a_{t-n}|s_{t-n}) d_\pi(s_{t-n}) \tag{A.30}$$

The backward parameter corrections are be defined over the backward distribution as:

$$\overleftarrow{\Delta}(s) = \mathbb{E}\left[\left(\overleftarrow{r}^{(n)}(S_{t-n}, S_t) + \gamma V_{\mathbf{w}}(S_t) - V_{\mathbf{w}}(S_{t-n})\right) \nabla_{\mathbf{w}} V_{\mathbf{w}}(S_{t-n}) | S_t = s, S_{t-n} \sim \overleftarrow{\mathcal{P}}_{\pi}^{(n)}(\cdot|S_t = s)\right] \tag{A.31}$$

3 Model-Learning Objectives

3.1 Expected Linear Backward Models

We assume the linear representation $V_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^\top \mathbf{w}$ for the value function, and we use a bilinear model for the reward: $\tilde{r} = \tilde{\mathbf{x}}^\top \theta_r \mathbf{x}$, where $\theta_r \in \mathbb{R}^{m \times m}$ is a matrix of reward parameters. This form allows us to express rewards over transitions from features $\tilde{\mathbf{x}}$ to successor features $\mathbf{x}$ in a structured way, and it allows computing the expected reward over all $\tilde{\mathbf{x}}$ given $\mathbf{x}$ as $\mathbb{E}[\tilde{r}|\mathbf{x}] = \sum_{\tilde{\mathbf{x}}} \Pr(\tilde{\mathbf{x}}|\mathbf{x}) \tilde{\mathbf{x}}^\top \theta_r \mathbf{x} = \mathbb{E}[\tilde{\mathbf{x}}^\top|\mathbf{x}] \theta_r \mathbf{x}$

The goal is to express the expected parameter update starting from arbitrary features $\mathbf{x}$, based on expected rewards and value function corrections. The full expression is:

$$\sum_{\tilde{\mathbf{x}}} Pr(\tilde{\mathbf{x}}|\mathbf{x}) \left(\sum_{\tilde{r}} Pr(\tilde{r}|\tilde{\mathbf{x}}, \mathbf{x}) \tilde{r} + \gamma V_{\mathbf{w}}(\mathbf{x}) - V_{\mathbf{w}}(\tilde{\mathbf{x}}) \right) \tilde{\mathbf{x}} = \tag{A.32}$$

(apply linearity and insert the definitions of $V_{\mathbf{w}}$ and $\tilde{r}$) (A.33)

$$= \sum_{\tilde{\mathbf{x}}} Pr(\tilde{\mathbf{x}}|\mathbf{x}) \tilde{\mathbf{x}} \tilde{\mathbf{x}}^\top \theta_r \mathbf{x} + \gamma \sum_{\tilde{\mathbf{x}}} Pr(\tilde{\mathbf{x}}|\mathbf{x}) \tilde{\mathbf{x}} \mathbf{x}^\top \mathbf{w} - \sum_{\tilde{\mathbf{x}}} Pr(\tilde{\mathbf{x}}|\mathbf{x}) \tilde{\mathbf{x}} \tilde{\mathbf{x}}^\top \mathbf{w} \tag{A.34}$$

(separate expectations and combine terms) (A.35)

$$= \mathbb{E}\left[\tilde{\mathbf{x}} \tilde{\mathbf{x}}^\top | \mathbf{x}\right] \theta_r \mathbf{x} + \gamma \mathbb{E}\left[\tilde{\mathbf{x}}^\top | \mathbf{x}\right] \mathbf{x}^\top \mathbf{w} - \mathbb{E}\left[\tilde{\mathbf{x}} \tilde{\mathbf{x}}^\top | \mathbf{x}\right] \mathbf{w} \tag{A.36}$$

(factorize the final term) (A.37)

$$= \mathbb{E}\left[\tilde{\mathbf{x}} \tilde{\mathbf{x}}^\top | \mathbf{x}\right] \theta_r \mathbf{x} + \left(\gamma \mathbb{E}\left[\tilde{\mathbf{x}}|\mathbf{x}\right] \mathbf{x}^\top - \mathbb{E}\left[\tilde{\mathbf{x}} \tilde{\mathbf{x}}^\top | \mathbf{x}\right]\right) \mathbf{w} \tag{A.38}$$

$$= \overleftarrow{r}_{\mathbf{x}}(\mathbf{x}) + \left(\gamma \overleftarrow{p}_{\mathbf{x}}(\mathbf{x}) \mathbf{x}^\top - \overleftarrow{p}_{\mathbf{x}^2}(\mathbf{x})\right) \mathbf{w}, \tag{A.39}$$

where $\overleftarrow{p}_{\mathbf{x}}(\mathbf{x}) = \mathbb{E}\left[\tilde{\mathbf{x}}|\mathbf{x}\right]$ is a backward expectation model, $\overleftarrow{p}_{\mathbf{x}^2}(\mathbf{x}) = \mathbb{E}\left[\tilde{\mathbf{x}}\tilde{\mathbf{x}}^\top|\mathbf{x}\right]$ is a predecessor features covariance matrix and $\overleftrightarrow{r}_{\mathbf{x}}(\mathbf{x}) = \mathbb{E}\left[\tilde{\mathbf{x}}\tilde{\mathbf{x}}^\top|\mathbf{x}\right]\theta_r\mathbf{x}$ is a vector reward model (θ_r are the parameters of the expected linear reward model).

As defined, $\tilde{r}$ is the expected reward for a transition from $\tilde{\mathbf{x}} \to \mathbf{x}$, conditioned on both the origin and destination features. It denotes a scalar reward that the agent receives for that transition. This form expresses the reward in terms of feature pair interactions. In contrast, $\overleftrightarrow{r}_{\mathbf{x}}(\mathbf{x})$ is a linear reward model in feature space. It is derived from the assumption $\tilde{r} = \tilde{\mathbf{x}}^\top\theta_r\mathbf{x}$. We no longer need to compute or store $\tilde{r}$ for each pair; instead, we propagate its expected influence forward using the model structure.

Note that this formulation only conditions on the current features $\mathbf{x}$, and not directly on both $\mathbf{x}$ and $\tilde{\mathbf{x}}$. This is possible because we assumed a bilinear reward model and we then take the full expectation over $\tilde{\mathbf{x}} \sim Pr(\cdot|\mathbf{x})$. We use this structure to reduce the number of unknowns in reward estimation, and to express everything in terms of expectations under the transition dynamics.

In contrast, the forward view update is:

$$\sum_{\mathbf{x}'}\left(Pr(\mathbf{x}'|\mathbf{x})\left(\sum_r Pr(r|\mathbf{x},\mathbf{x}')r + \gamma V_{\mathbf{w}}(\mathbf{x}')\right) - V_{\mathbf{w}}(\mathbf{x})\right)\mathbf{x} = \tag{A.40}$$

$$\left(\sum_{\mathbf{x}'} Pr(\mathbf{x}'|\mathbf{x})(\mathbf{x}')^\top\theta_r\mathbf{x} + \gamma\sum_{\mathbf{x}'} Pr(\mathbf{x}'|\mathbf{x})(\mathbf{x}')^\top\mathbf{w} - \mathbf{x}^\top\mathbf{w}\right)\mathbf{x} \tag{A.41}$$

$$= \mathbb{E}\left[\mathbf{x}'|\mathbf{x}\right]\theta_r\mathbf{x}\mathbf{x} + \left(\gamma\mathbf{x}\mathbb{E}\left[(\mathbf{x}')^\top|\mathbf{x}\right] - \mathbf{x}\mathbf{x}^\top\right)\mathbf{w} \tag{A.42}$$

$$= \mathbf{x}\left(\overleftrightarrow{r}_{\mathbf{x}}(\mathbf{x},\mathbf{x}') + \gamma\mathcal{P}_{\mathbf{x}}(\mathbf{x})^\top\mathbf{w} - \mathbf{x}^\top\mathbf{w}\right), \tag{A.43}$$

where $\mathcal{P}_{\mathbf{x}} = \mathbb{E}\left[(\mathbf{x}')^\top|\mathbf{x}\right]$ is the forward expectation model and $\overleftrightarrow{r}_{\mathbf{x}}(\mathbf{x},\mathbf{x}') = \mathbb{E}\left[\mathbf{x}'|\mathbf{x}\right]\theta_r\mathbf{x}\mathbf{x}$ is the corresponding reward expectation model.

3.2 Maximum Likelihood Estimation for Expectation Models

The expectation model approach trains a model predictor to output a point estimate as a function of the input. When trained with mean-squared error, the probabilistic interpretation is that the point estimate corresponds to the mean of a Gaussian distribution with fixed input-independent variance σ^2: $\mathcal{P}(\tilde{s}|s) = \mathcal{N}(\mu(s); \sigma^2)$. Minimizing the negative log likelihood in this case leads to least-squares regression. Concretely, the empirical loss for the MLE objective of backward model learning is

$$l_{\text{MLE}}(\overleftarrow{p}) = \frac{1}{n} \sum_{(\mathbf{x}_i, A_i, \mathbf{x}'_i) \in \mathcal{D}_n} \left| \overleftarrow{p}(\mathbf{x}'_i) - \mathbf{x}_i \right|^2 \approx \mathbb{E} \left[\left| \overleftarrow{p}(\mathbf{x}') - \mathbf{x} \right|^2 \right], \tag{A.44}$$

where $\overleftarrow{p} \leftarrow \operatorname{argmin}_{\overleftarrow{p}^\dagger \in \Delta_{\mathcal{S}}} l_{\text{MLE}}(\overleftarrow{p}^\dagger)$, $\Delta_{\mathcal{S}}$ is the model space and $\mathcal{D}_n = \{(\mathbf{x}_i, A_i, R_i, \mathbf{x}'_i)\}_{i=1}^n$ represents collected data from interaction.

4 Empirical Studies Details

4.1 Direct effect of backward planning on prediction

The first setting in which we illustrate the antithesis between forward and backward planning is a Random Markov Chain consisting of a levelled state space with one (or more) sets of states (or levels): $\{x_i\}_{i\in[0:n_x]}$ and $\{y_j\}_{j\in[0:n_y]}$, where we vary n_x and n_y in our experiments (Fig. A.1). We additionally experiment with adding intermediary levels: $\{z_k^l\}_{k\in[0:n_z^l],l\in[1:L]}$, where L is the number of levels and n_z^l is the size of level l. The states from a particular level transition only to states in the next level, thus establishing a particular flow and stationary structure of the Markov Chain under study. The transition probabilities between the leveled sets of states are sampled randomly from a uniform distribution $\mathcal{U}(0,1)$ and normalized accordingly. The states $\{y_j\}_{j\in[0:n_y]}$ are terminal; their rewards are dependent on both ends of the transition and are sampled from a normal distribution $\mathcal{N}(10,10)$. The criteria we use to evaluate the quality of the estimated models is the root mean squared value error (RMSVE): $\sqrt{(|V_\pi - V|_2^2)}$, which shows how close $V_\mathcal{P}$ and $V_{\tilde{p}}$ are to V_π. The experiments are ablation studies of the effects of varying the *fan-in* (n_x), the *fan-out* (n_y) and the number of levels l with their corresponding sizes n_z^l.

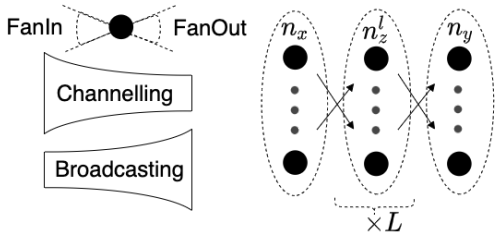


Figure A.1: Random Chain: Illustration of the Markov Reward Process used in the prediction experiment. The chain flows from left to right.

For this investigation we performed two types of experiments: (i) on 3-level bipartite graphs as illustrated in figure 3.1 (ii) on 2-level bipartite graphs as shown in figure 3.3. All experiments start with learning rates of 1.0 for model-free learning, planning and model-learning, which are linearly decayed over the course of learning. For (i) for the channelling structure we use: $L = 1, n_x = 500, n_z^1 = 50, n_y = 5$ and the opposite for the broadcasting structure $L = 1, n_x = 5, n_z^1 = 50, n_y = 500$. For (ii) the results reported are for

$(n_x, n_y) \in \{(500, 5), (50, 5), (5, 5), (5, 50), (5, 500)\}$, labelling the x-axis with the simplified ratio. Results shown in the main text are averaged over 20 seeds and show the standard error over runs.

All experiments use, for any transition $s \xrightarrow{a} s'$, the following reference frame for planning: background models – the current state s' of a transition, forward models – the previous state s of a transition. The exact definition of reference frames is given in the main text and below, in the next experiment. To make things more concrete, we include a complete pseudo-code for the backward planning algorithm used for this experiment in Algorithm 10 (forward planning is done similarly, see Alg. 6 and 7 for more details on the differences).

Algorithm 10 Online Backward-Dyna: Learning & Backward Planning

Input policy π, n
$s \sim \text{env}()$
for each interaction $\{1, 2 \dots T\}$ **do**
  $a \sim \pi(x)$
  $r, \gamma, s' \sim \text{env}(a)$
  $\overleftarrow{p}, \overleftrightarrow{r} \leftarrow \text{model_learning_update}(s, a, s')$
  $V \leftarrow \text{learning_update}(s, a, r, \gamma, s')$
  $s_{\text{ref}} \leftarrow \text{planning_reference_state}(s, s')$
  for each $\tilde{s} \in \mathcal{S}$ **do**
    $Y = \overleftrightarrow{r}(\tilde{s}, s_{\text{ref}}) + \gamma V(s_{\text{ref}})$
    $\overleftarrow{\Delta}(\tilde{s}) = \overleftarrow{p}(\tilde{s}|s_{\text{ref}})\,(y - V(\tilde{s}))$
    $V(\tilde{s}) \leftarrow V(\tilde{s}) + \alpha \overleftarrow{\Delta}(\tilde{s})$
  $s \leftarrow s'$

4.2 Indirect effect of backward planning on control

In the following experiments, we perform ablation studies on the discrete navigation task from [Sutton and Barto, 2018] illustrated in Fig. A.2. "G" marks the position of the goal and the end of an episode. "S" denotes the starting state to which the agent is reset at the end of the episode. The maximum size of episodes is set to 400 to allow the agent sufficient exploration. The state space size is 48, $\gamma = 0.99$. There are 4 actions that can transition the agent to each one of the adjacent states.

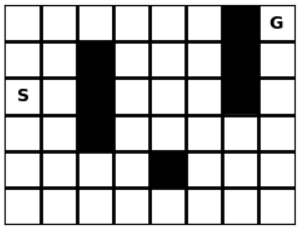


Figure A.2: Maze Navigation.

The following two experiments use algorithms that perform *background planning* with forward and backward mechanisms. Specifically, for the forward case we use algorithm 6 and for the backward case algorithm 7. The control algorithms interlace model-free learning of the optimal action-value function Q (line 7 – forward & backward) with model-learning (of forward and backward models respectively – line 6) and additional planning steps (forward and backward – lines 8:10). The model-free ``learning_update'' performs q-learning updates of the form:

$$Q(s,a) = Q(s,a) + \alpha \left(r\left(s,a,s'\right) + \gamma \max_{a'} Q(s',a') - Q\left(s,a\right) \right). \tag{A.45}$$

The models are estimated using MLE – for the transition dynamics (and for the termination function $\bar{\gamma}$ in the case of forward models) and regression - for the reward models. The planning processes add model-based forward and backward updates using the estimated models. Forward planning updates perform an expected planning-update using the forward models $(\mathcal{P}, \overrightarrow{r}, \bar{\gamma})$, $\forall s \in \mathcal{S}, \forall a \in \mathcal{A}$:

$$Q(s_{\text{ref}},a) = Q(s_{\text{ref}},a) + \alpha \left(\sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s_{\text{ref}},a) \left(\overrightarrow{r}(s') + \bar{\gamma}(s') \max_{a'} Q(s',a') \right) - Q(s_{\text{ref}},a) \right) \tag{A.46}$$

and the backward planning process adds the following updates according to the backward models $(\overleftarrow{p}, \overleftarrow{r}, \forall \tilde{s} \in \mathcal{S}, \forall a \in \mathcal{A}$:

$$q(\tilde{s},\tilde{a}) = q(\tilde{s},\tilde{a}) + \alpha \overleftarrow{p}\left(\tilde{s},\tilde{a}|s_{\text{ref}}\right) \left(\overleftarrow{r}\left(s_{\text{ref}}\right) + \gamma \max_{\tilde{a}} q\left(s_{\text{ref}},\tilde{a}\right) - q\left(\tilde{s},\tilde{a}\right) \right). \tag{A.47}$$

The procedure specified in the pseudocode with ``planning_reference_state'' is the subject of the first experiment we performed for this control setting.

Planning frame of reference

We first ask ourselves whether the frame of reference from which the agent starts planning matters. When it encounters a transition $s \xrightarrow{a} s'$, it could plan from either s or s'. This

choice represents the output of the procedure ``planning_reference_state'' in the pseudocode. In our experiments, we perform ablation studies for pure planning, i.e. without the additional model-free procedure ``learning_update'', and for the full learning framework described in the pseudocode. The experiments are performed with deterministic dynamics. The learning rates used for q-learning, model learning and planning are started at 1.0 and linearly decayed over the course of training. The policy used for acting is ϵ-greedy with ϵ decayed linearly over the course of training from 0.5 to 0. Results shown in the main text are averaged over 20 seeds and show the standard error over runs.

Changing the level of stochasticity

We next investigate how stochasticity affects performance. The four settings we investigate are

(i) deterministic dynamics – when the environment transitions the agent to the intended direction with probability 1; similarly reward is $+1$ with probability 1;

(ii) stochastic dynamics – reward is still deterministic, yet the environment transitions the agent to a random adjacent state with probability 0.5;

(iii) stochastic rewards – transition dynamics are deterministic, yet the rewards are $+1$ with probability 0.5, otherwise 0;

(iv) stochastic rewards identical to the previous setting; rewards are $+1$ with probability 0.1.

We apply the same algorithms described in the previous section taking s to be the reference frame for backward planning and s' for forward planning. Table A.1 specifies the learning rates used for q-learning, model learning and planning for each of the four settings we described above. These are linearly decayed over the course of training. The policy used for acting is ϵ-greedy with ϵ decayed linearly over the course of training from 0.5 to 0.

Results shown in the main text are averaged over 20 seeds and show the standard error over runs. Hyperparameters have been chosen from $\{1.0, 0.5, 0.1, 0.05, 0.01\}$.

Table A.1: Hyperparameters for tabular control on the Maze Gridworld

Environment Setting	α **(learning&planning)**	α_m **(model learning)**
Deterministic rewards and transitions	1.0	1.0
Deterministic transitions/stochastic rewards (p=0.5)	0.1	0.5
Deterministic transitions/stochastic reward (p=0.1)	0.05	0.05
Stochastic transitions (p=0.5)/deterministic rewards	0.1	0.5

B Salience Maps for Selective Credit Assignment

1 Background and preliminaries (details)

1.1 Expected eligibility traces

Expected eligibility trace (ET) algorithms [van Hasselt et al., 2020] have been introduced for *off-trajectory*, *on-policy* value learning, replacing the instantaneous trace $\mathbf{e}_t$ with an estimated expectation:

$$\mathbf{z}_{\boldsymbol{\Theta}}(s) \approx \mathbb{E}_{\pi}\left[\mathbf{e}_t \mid S_t = s\right] \text{ , with}$$
$$\mathbf{e}_t = \gamma_t \lambda_t \mathbf{e}^{\eta}_{t-1} + \nabla V_{\mathbf{w}_t}(S_t)\,.$$

We can approximate the expected traces by regressing on the instantaneous eligibility trace $\mathbf{e}_t$, or by a mechanism similar to value bootstrapping using the *mixture trace* $\mathbf{e}^{\eta}_t$:

$$\mathbf{e}^{\eta}_t = (1-\eta)\mathbf{z}_{\boldsymbol{\Theta}_t}(S_t) + \eta[\gamma_t \lambda_t \mathbf{e}^{\eta}_{t-1} + \nabla V_{\mathbf{w}_t}(S_t)]\,.$$

The trace-bootstrapping parameter η specifies the credit assignment mechanism with $\mathbf{e}^{\eta=0}_t = \mathbf{z}_{\boldsymbol{\Theta}_t}(S_t)$ resulting in *counterfactual* credit assignment (fully relying on the expected traces), while $\mathbf{e}^{\eta=1}_t = \gamma_t \lambda_t \mathbf{e}^{\eta=1}_{t-1} + \nabla V_{\mathbf{w}_t}(S_t)$ uses *factual* or *trajectory-based* learning (reverting fully to instantaneous traces).

The expected trace algorithm ET(λ, η) is then defined by:

$$\begin{aligned}\mathbf{w}_{t+1} &= \mathbf{w}_t + \alpha_t^{\mathbf{w}_t}\Delta_t^V \text{ , with}\\ \Delta_t^V &= \delta_t \mathbf{e}_t^\eta \,,\end{aligned}$$

and smoothly interpolates between these two end points (using expected or instantaneous traces).

The expected trace $\mathbf{z}_{\boldsymbol{\Theta}}(S_t)$ can be learned by regressing on $\gamma_t\lambda_t\mathbf{e}_{t-1}^\eta + \nabla V_{\mathbf{w}_t}(S_t)$, s.t.:

$$\begin{aligned}\Delta_t^{\mathbf{z}} &\equiv \frac{\partial \mathbf{z}_{\boldsymbol{\Theta}_t}(S_t)}{\partial \boldsymbol{\Theta}_t}\Big(\gamma_t\lambda_t\mathbf{e}_{t-1}^\eta + \nabla V_{\mathbf{w}_t}(S_t) - \mathbf{z}_{\boldsymbol{\Theta}_t}(S_t)\Big)\,,\\ \boldsymbol{\Theta}_{t+1} &= \boldsymbol{\Theta}_t + \alpha_t^{\boldsymbol{\Theta}}\Delta_t^{\mathbf{z}}\,,\end{aligned} \tag{B.1}$$

or using a different mixture parameter $\tilde{\eta}$ for learning the trace:

$$\mathbf{e}_t^{\tilde{\eta}} = (1-\tilde{\eta})\mathbf{z}_{\boldsymbol{\Theta}_t}(S_t) + \tilde{\eta}[\gamma_t\lambda_t\mathbf{e}_{t-1}^{\tilde{\eta}} + \nabla V_{\mathbf{w}_t}(S_t)]\,,$$

keeping η just for determining when to use the model. So updating $\boldsymbol{\Theta}$ with

$$\begin{aligned}\Delta_t^{\mathbf{z}} &\equiv \frac{\partial \mathbf{z}_{\boldsymbol{\Theta}_t}(S_t)}{\partial \boldsymbol{\Theta}_t}\Big(\gamma_t\lambda_t\mathbf{e}_{t-1}^{\tilde{\eta}} + \nabla V_{\mathbf{w}_t}(S_t) - \mathbf{z}_{\boldsymbol{\Theta}_t}(S_t)\Big)\,,\\ \boldsymbol{\Theta}_{t+1} &= \boldsymbol{\Theta}_t + \alpha_t^{\boldsymbol{\Theta}}\Delta_t^{\mathbf{z}}\,,\end{aligned}$$

where $\alpha_t^{\boldsymbol{\Theta}} \in (0,1)$ is a (possibly time-varying) step-size parameter and $\tilde{\eta}$ is analogous to λ, but for learning traces rather than values, using TD methods when $\tilde{\eta} = 0$, and Monte-Carlo regression when $\tilde{\eta} = 1$. Note that, like TD(0), we always include at least one sampled transition, even if $\tilde{\eta} = 0$.

2 Proofs and derivations

2.1 New derivation of off-policy Q(λ)

Let $\rho(s,a) = \frac{\pi(a|s)}{\mu(a|s)}$ be the importance sampling ratio between the target policy π and the behaviour μ. In the trajectory context, we may write for any k and t: $\rho_k^t = \prod_{j=k}^{t} \rho_j,$ with $\rho_j = \rho(S_j, A_j)$, with the convention that $\rho_k^t = 1$ for $t < k$.

With this notation, the off-policy importance sampled λ-return for state-dependent λ and γ can be written as:

$$
\begin{aligned}
G_t^{\lambda,\pi} &= (1-\lambda_{t+1})G_{t:t+1}^{\pi} + \lambda_{t+1}(1-\lambda_{t+2})G_{t:t+2}^{\pi} + \dots \\
&= (1-\lambda_{t+1})\rho_t[R_{t+1} + \gamma_{t+1}V(S_{t+1})] + (1-\lambda_{t+2})\lambda_{t+1}\rho_t[R_{t+1} + \gamma_{t+1}\rho_{t+1}R_{t+2} + \\
&\quad \gamma_{t+1}\rho_{t+1}\gamma_{t+2}V(S_{t+1}) \dots \\
&= \rho_t \left(R_{t+1} + (1-\lambda_{t+1})\gamma_{t+1}V(S_{t+1})\right) + \lambda_{t+1}\gamma_{t+1}\rho_t\rho_{t+1}[R_{t+2} + (1-\lambda_{t+2})\gamma_{t+2}V(S_{t+2})] + \dots \\
&= \sum_{k=0}^{\infty} \left(\prod_{j=0}^{k-1} \gamma_{t+j+1}\lambda_{t+j+1}\rho_{t+j} \right) \rho_{t+k} \left[R_{t+k+1} + (1-\lambda_{t+k+1})\gamma_{t+k+1}V(S_{t+k+1}) \right]
\end{aligned}
$$

This is the return from some state S_t onward. For the return from some state S_t and action A_t, we can just drop the final importance sampling ratio, yielding:

$$
G_t^{\lambda,\pi} = \sum_{k=0}^{\infty} \left(\prod_{j=0}^{k-1} \gamma_{t+j+1}\lambda_{t+j+1}\rho_{t+j} \right) \left[R_{t+k+1} + (1-\lambda_{t+k+1})\gamma_{t+k+1}V(S_{t+k+1}) \right].
$$

For Q-learning the bootstrapped target is: $V(S_{t+k+1}) \equiv \max_a Q(S_{t+k+1}, a)$, whereas for SARSA we have: $V(S_{t+k+1}) \equiv \mathbb{E}_\pi[Q(S_{t+k+1}, A)]$.

Let

$$
\begin{aligned}
\Delta_t^Q &= (G_t^{\lambda,\pi} - Q_{\mathbf{w}}(S_t, A_t))\nabla Q_{\mathbf{w}}(S_t, A_t) \\
&= G_t^{\lambda,\pi}\nabla Q_{\mathbf{w}}(S_t, A_t) - Q_{\mathbf{w}}(S_t, A_t)\nabla Q_{\mathbf{w}}(S_t, A_t)\,,
\end{aligned}
$$

be the update to the parameters $\mathbf{w}$ at time step t.

We look at the first part of the update, which can be rewritten as:

$$\begin{aligned}
\sum_{t=0}^{\infty} G_t^{\lambda,\pi} \nabla Q_{\mathbf{w}}(S_t, A_t) &= \sum_{t=0}^{\infty} \sum_{k=0}^{\infty} \left(\prod_{j=0}^{k-1} \gamma_{t+j+1} \lambda_{t+j+1} \rho_{t+j} \right) \left[R_{t+k+1} + (1 - \lambda_{t+k+1}) \gamma_{t+k+1} V(S_{t+k+1}) \right] \\
&\quad \cdot \nabla Q_{\mathbf{w}}(S_t, A_t) \\
&= \sum_{t=0}^{\infty} \sum_{k=t}^{\infty} \left(\prod_{j=t}^{k-1} \gamma_{j+1} \lambda_{j+1} \rho_j \right) \left[R_{k+1} + (1 - \lambda_{k+1}) \gamma_{k+1} V(S_{k+1}) \right] \nabla Q_{\mathbf{w}}(S_t, A_t) \\
&= \sum_{k=0}^{\infty} \sum_{t=0}^{k} \left(\prod_{j=t}^{k-1} \gamma_{j+1} \lambda_{j+1} \rho_j \right) \left[R_{k+1} + (1 - \lambda_{k+1}) \gamma_{k+1} V(S_{k+1}) \right] \nabla Q_{\mathbf{w}}(S_t, A_t) \\
&= \sum_{t=0}^{\infty} \sum_{k=0}^{t} \left(\prod_{j=k}^{t-1} \gamma_{j+1} \lambda_{j+1} \rho_j \right) \left[R_{t+1} + \gamma_{t+1} (1 - \lambda_{t+1}) V(S_{t+1}) \right] \nabla Q_{\mathbf{w}}(S_k, A_k) \\
&= \sum_{t=0}^{\infty} \left[R_{t+1} + \gamma_{t+1} (1 - \lambda_{t+1}) V(S_{t+1}) \right] \underbrace{\sum_{k=0}^{t} \left(\prod_{j=k}^{t-1} \gamma_{j+1} \lambda_{j+1} \rho_j \right) \nabla Q_{\mathbf{w}}(S_k, A_k)}_{\mathbf{e}_t},
\end{aligned}$$

with

$$\begin{aligned}
\mathbf{e}_t &= \sum_{k=0}^{t} \left(\prod_{j=k}^{t-1} \gamma_{j+1} \lambda_{j+1} \rho_j \right) \nabla Q_{\mathbf{w}}(S_k, A_k) \\
&= \nabla Q_{\mathbf{w}}(S_t, A_t) + \gamma_t \lambda_t \rho_{t-1} \nabla Q_{\mathbf{w}}(S_{t-1}, A_{t-1}) + \dots \\
&= \gamma_t \lambda_t \rho_{t-1} \mathbf{e}_{t-1} + \rho_t \nabla Q_{\mathbf{w}}(S_t, A_t)
\end{aligned}$$

Then, the full update is:

$$\sum_{t=0}^{\infty} \Delta_t^Q = \sum_{t=0}^{\infty} \underbrace{\left[R_{t+1} + \gamma_{t+1} (1 - \lambda_{t+1}) V(S_{t+1}) \right]}_{R_t^\lambda} \mathbf{e}_t - Q_{\mathbf{w}}(S_t, A_t) \nabla Q_{\mathbf{w}}(S_t, A_t) \,,$$

which is what we use in the new versions of Q(λ) and QET(λ, η) algorithms.

When using this definition with the QET algorithm, one important difference to the prior algorithm introduced by van Hasselt et al. [2020] is that we estimate the decayed

previous expected trace, so approximating $\mathbf{z}_{\boldsymbol{\Theta}}(s) \approx \mathbb{E}_\pi[\gamma_t \lambda_t \mathbf{e}_{t-1} \mid S_t = s]$, and then using $\mathbf{z}_{\boldsymbol{\Theta}}(s) + \nabla Q_{\mathbf{w}}(s, a)$ as the trace for action a. This avoids having to condition the expected trace on the action, which can significantly reduce computation in some implementations.

2.2 Sparse expected eligibility traces

Proposition 1. *Sparse accumulating traces with binary state weighting functions are equivalent to temporally extended backward models.*

Proof. From the definition of the selective expected trace, we have

$$\begin{aligned}
\tilde{\mathbf{z}}(s) &= \mathbb{E}_\pi[\tilde{\mathbf{e}}_t | S_t = s] \\
&= \mathbb{E}[\gamma_t \lambda_t \tilde{\mathbf{e}}_{t-1} + \nabla V_{\mathbf{w}_t}(S_t) \omega_t | S_t = s] \\
&= \nabla V_{\mathbf{w}_t}(s) \omega(s) + \gamma(s) \lambda(s) \sum_{\tilde{s}} P_\pi(s|\tilde{s}) \tilde{\mathbf{z}}(\tilde{s})
\end{aligned}$$

If $\omega(s) = 1 - \gamma(s)\lambda(s)$ (cf. Eq. (4.12)), then

$$\tilde{\mathbf{z}}(s) = \omega(s) \nabla V_{\mathbf{w}_t}(s) + (1 - \omega(s)) \sum_{\tilde{s}} P_\pi(s|\tilde{s}) \tilde{\mathbf{z}}(\tilde{s}) \tag{B.2}$$

In Eq. (B.2), considering $c(s) \equiv \nabla V_{\mathbf{w}_t}(s)$ a multi-dimensional cumulant and $\beta(s) \equiv 1 - \omega(s)$ the probability of termination at state s, then $\tilde{\mathbf{z}}(s)$ can be interpreted as a temporally-extended option model for an option defined as: $o = (\pi, \beta)$, where π is the option's policy and β is the corresponding binary termination function.

□

2.3 Weightings for distribution correction: off-policy expected emphasis

We consider methods that learn the expectation of the follow-on weighting $f(s) = \mathbb{E}_\mu[F_t | S_t = s]$, resulting in a state-dependent selectivity function to be used in place of the history-dependent weighting ω_t. We call the emphatic algorithm, analogous to λ-discounted

TD, resulting from the aforementioned approach, X-ETD(λ) (where "X" comes from "expected"):

$$\omega_t = \rho_t m_t \,, \text{with}$$
$$m_t = \lambda_t i_t + \gamma_t (1 - \lambda_t) f(s) \,,$$

The expected follow-on can be estimated using a function $f_\varphi \approx f$, with learnable parameters φ. Learning methods, typically used to learn value functions, can be applied, by reversing the direction of time in the learning update, similarly to expected eligibility traces, e.g., Monte-Carlo regression on the instantaneous follow-on trace

$$\varphi_{t+1} = \varphi_t + \alpha_t^\varphi (F_t - f_{\varphi_t}(S_t)) \nabla_{\varphi_t} f_{\varphi_t}(S_t) \,, \tag{B.3}$$

or backward TD:

$$\varphi_{t+1} = \varphi_t + \alpha_t^\varphi (f_\varphi(S_{t-1}) + i_t - f_{\varphi_t}(S_t)) \nabla_{\varphi_t} f_{\varphi_t}(S_t) \,, \tag{B.4}$$

with α_t^f a (possibly time-varying) step-size.

Unfortunately, both methods can be problematic. The Monte-Carlo regression problem has targets with infinite variance, so it is not guaranteed to converge, whereas the backward TD method suffers from "off-policiness", same as value learning.

Similarly to the case of expected eligibility traces, an interpolation between the instantaneous follow-on trace and its estimated expectation is possible using a mixture trace:

$$F_t^\eta = (1 - \eta^F) f_\varphi(S_t) + \eta^F \gamma_t \rho_{t-1} F_{t-1} + i_t \,,$$

with η^F the mixing parameter, interpolating between using the expected follow-on $f(s)$ or the history-dependent instantaneous follow-on trace F_t. The mixture trace F^η is then used

in the selectivity function ω as:

$$\omega_t = \rho_t(\underbrace{\lambda_t i_t + (1-\lambda_t)F_t^{\eta}}_{m_t}),$$

with m_t the mixed emphasis. We would then use ω to learn the value parameters $\mathbf{w}$.

A second mixture trace e_t^f can be used as target in the estimation of the expected follow-on f_φ (same as for expected traces), one that uses a different mixture parameter η^f (possibly different from η^F, distinguishing the way the trace is learned from how it is used:

$$e_t^f = (1-\eta^f) f_\varphi(S_t) + \eta^f \gamma_t \rho_{t-1} F_{t-1} + i_t\,, \tag{B.5}$$

The expected mixture trace is then used as target for an estimated model s.t. $f_\varphi \approx \mathbb{E}[\gamma_t \rho_{t-1} e_{t-1}^f | S_t = s]$, with η^f interpolating between Monte-Carlo regression on the instantaneous trace ($\eta^f = 1$), and backward TD by bootstrapping on the expected trace ($\eta^f = 0$). The follow-on trace learning process updates the trace parameters φ with:

$$\varphi_{t+1} = \varphi_t + \alpha_t^f \left(\gamma_t \rho_{t-1} e_{t-1}^f - f_{\varphi_t}(S_t)\right) \nabla_{\varphi_t} f_{\varphi_t}(S_t)\,,$$

with α_t^f—the step size. We use X(η^f)-ETD(λ) to explicitly denote the mechanism used to learn the expected follow-on trace, with $\eta^f = 1$ for Monte-Carlo regression, and $\eta^f = 0$ for backward TD(0).

2.4 Weightings for on-policy learning: on-policy expected emphasis

Emphatic TD uses a state weighting function of the form:

$$\begin{aligned}
\omega_t &= M_t \rho_t \\
&= \rho_t \lambda_t i_t + \rho_t (1-\lambda_t) F_t, \text{ with } F_t = \gamma_t \rho_{t-1} F_{t-1} + i_t \\
&= \rho_t i_t + \rho_t \gamma_t (1-\lambda_t) \sum_{k=1}^{t} \left(\prod_{j=k}^{t} \rho_{t-j} \gamma_{t-j} \right) i_{t-k} \\
&= \rho_t i_t + \rho_t \gamma_t (1-\lambda_t) \rho_{t-1} i_{t-1} + \rho_t \gamma_t (1-\lambda_t) \rho_{t-1} \gamma_{t-1} \rho_{t-2} i_{t-2} \dots,
\end{aligned}$$

For the on-policy learning, we have $\rho_t = 1, \forall t$. For on-policy learning and uniform interest $i_t = 1, \forall t$, then, the state weighting becomes:

$$\begin{aligned}
\omega_t &= M_t \\
&= \lambda_t + (1-\lambda_t) F_t, \text{ with } F_t = \gamma_t F_{t-1} + 1 \\
&= 1 + \gamma_t (1-\lambda_t) \sum_{k=1}^{t} \left(\prod_{j=k}^{t} \gamma_{t-j} \right)
\end{aligned}$$

The expected emphasis in this setting is:

$$\begin{aligned}
f(s) &= \lim_{t \to \infty} \mathbb{E}_\pi [F_t | S_t = s] \\
&= \mathbb{E}_\pi \left[1 + \sum_{k=1}^{t} \prod_{j=k}^{t} \gamma_{t-j} \mid S_t = s \right]
\end{aligned}$$

Constant γ Furthermore, for constant γ we have:

$$f(s) = 1 + \gamma + \gamma^2 + \dots = \frac{1}{1-\gamma}$$

Replacing the expected emphasis in the definition of the weighting ω:

$$\begin{aligned}
\omega_t &= \lambda_t + (1 - \lambda_t)/(1 - \gamma) \\
&= (\lambda_t - \lambda_t\gamma + 1 - \lambda_t)/(1 - \gamma) \\
&= (1 - \lambda_t\gamma)/(1 - \gamma) \\
\implies \lambda_t &= (\gamma\omega_t + (1 - \omega_t))/\gamma
\end{aligned}$$

For constant γ, we can omit the denominator, since it would be just re-scaling the update by a constant factor which can be folded into the learning rate, yielding:

$$\omega_t = 1 - \gamma\lambda_t \implies \lambda_t = \gamma\omega_t + (1 - \omega_t)$$

Adaptive γ Let $\mathcal{P}_\pi$ be the transition matrix induced by following policy π, with $[\mathcal{P}_\pi]_{s,s'} = P(s'|s,a)\pi(a|s)$, and $[\mathcal{P}_\pi^\top]_{:,s} = P(\cdot|s,a)\pi(a|s)$ the vector corresponding to all entries of the successor states of s, a. Let $\mathbf{\Gamma}$, and $\mathbf{\Lambda}$ be diagonal matrices, the former representing the discount matrix—with diagonal entries $\gamma(s)$, and the latter the trace-decay matrix—using $\lambda(s)$ on its diagonal. Let $\mathbf{d}_\pi$ denote the vector, with entries coresponding to the stationary distributions $d_\pi(s)$. Then, in matrix notation, the following hold:

$$\begin{aligned}
\mathcal{P}_\pi^\top \mathbf{d}_\pi = \sum_s d_\pi(s)[\mathcal{P}_\pi^\top]_{:,s} = \mathbf{d}_\pi \implies \mathbf{\Gamma}\mathcal{P}_\pi^\top \mathbf{d}_\pi = \mathbf{\Gamma}\mathbf{d}_\pi \\
\mathcal{P}_\pi^\top \mathbf{\Gamma}\mathbf{d}_\pi = \sum_s \gamma(s) d_\pi(s)[\mathcal{P}_\pi^\top]_{:,s}\,.
\end{aligned}$$

Furthermore, we can write the stationary distribution under π reweighted by the follow-on weighting, in matrix notation:

$$\begin{aligned}\mathbf{d}_\pi^f &= (\mathbf{I} - \mathbf{\Gamma}\mathcal{P}_\pi^\top)^{-1}\mathbf{d}_\pi \\ &= \mathbf{d}_\pi + \mathbf{\Gamma}\mathcal{P}_\pi^\top\mathbf{d}_\pi + ((\mathbf{\Gamma}\mathcal{P}_\pi)^2)^\top\mathbf{d}_\pi + \dots \\ &= \mathbf{d}_\pi + \mathbf{\Gamma}\mathcal{P}_\pi^\top\mathbf{d}_\pi + \mathbf{\Gamma}\mathcal{P}_\pi^\top\mathbf{\Gamma}\mathcal{P}_\pi^\top\mathbf{d}_\pi + \dots\end{aligned}$$

Assuming we can bound the discount factor $\gamma(s)$ with β_λ, $\forall s \in \mathcal{S}$, s.t. $\gamma(s) \leq \beta_\lambda$, then:

$$\mathcal{P}_\pi^\top\mathbf{\Gamma}\mathbf{d}_\pi \leq \beta_\lambda \sum_s d_\pi(s)[\mathcal{P}_\pi^\top]_{:,s} = \beta_\lambda \mathbf{d}_\pi \implies \mathbf{\Gamma}\mathcal{P}_\pi^\top\mathbf{\Gamma}\mathbf{d}_\pi < \beta_\lambda^2\mathbf{d}_\pi \tag{B.6}$$

Using the assumption in Eq. B.6, we obtain the expected follow-on trace:

$$\begin{aligned}\mathbf{d}_\pi^f &= \mathbf{d}_\pi + \mathbf{\Gamma}\mathbf{d}_\pi + \mathbf{\Gamma}\mathcal{P}_\pi^\top\mathbf{\Gamma}\mathbf{d}_\pi + \dots \\ &= (1 + \beta_\lambda + \beta_\lambda^2 + \dots)\mathbf{d}_\pi \\ &= (1 - \beta_\lambda)^{-1}\mathbf{d}_\pi\end{aligned} \tag{B.7}$$

Let h be a weighting matrix, subject to the constraint:

$$\begin{aligned}h &= \mathbf{\Lambda} + (\mathbf{I} - \mathbf{\Lambda})(\mathbf{I} - \mathbf{\Gamma}\mathcal{P}_\pi^\top)^{-1} \\ &= (\mathbf{I} - \mathbf{\Lambda}\mathbf{\Gamma}\mathcal{P}_\pi^\top)(\mathbf{I} - \mathbf{\Gamma}\mathcal{P}_\pi^\top)^{-1} \\ \implies \mathbf{\Lambda} &= \mathbf{I} - h(\mathbf{I} - \mathbf{\Gamma}\mathcal{P}_\pi^\top)(\mathbf{\Gamma}\mathcal{P}_\pi^\top)^{-1} \\ &= (h\mathbf{\Gamma}\mathcal{P}_\pi^\top + (\mathbf{I} - h))(\mathbf{\Gamma}\mathcal{P}_\pi^\top)^{-1}\end{aligned} \tag{B.8}$$

Inserting the result from Eq. (B.7) in Eq. (B.8), we obtain:

$$\begin{aligned}\omega_t &= \frac{1 - \gamma_t\lambda_t}{1 - \beta_\lambda} \\ \implies \lambda_t &= \frac{\beta_\lambda\omega_t + (1 - \omega_t)}{\gamma_t} = \frac{1 - \omega_t(1 - \beta_\lambda)}{\gamma_t} = \frac{1 - \omega_t + \omega_t\beta_\lambda}{\gamma_t}\end{aligned}$$

The constant discount weighting can be recovered by making $\beta_\lambda = \gamma$ for constant γ.

2.4.1 Stability & convergence

For the on-policy case we can show that this coupling between the weighting and the trace decay parameter is sufficient for stability. Moreover, because the traces are on-policy and they do not have importance sampling ratios, the variance is always finite, so the coupling also ensures convergence, not just stability of the value learning process.

Let the value function $V_{\mathbf{w}}$ be a linear function of the form: $V_{\mathbf{w}}(s) = \mathbf{w}^\top \mathbf{x}(s)$ with $\mathbf{w} \in \mathbb{R}^n$ learnable parameters, where $\mathbf{x} : \mathcal{S} \to \mathbb{R}^n$ is a feature mapping. Let $\mathbf{X} \in \mathbb{R}^{|\mathcal{S}| \times n}$ be the feature matrix whose rows are the vectors $\mathbf{x}(s)$ for different states s.

Consider the semi-gradient learning update for the selective TD(λ, ω) algorithm:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha_t \tilde{\mathbf{e}}_t \underbrace{(R_{t+1} + \gamma_t V_{\mathbf{w}}(S_{t+1}) - V_{\mathbf{w}}(S_t))}_{\delta_t}, \tag{B.9}$$

with $\tilde{\mathbf{e}}_t = \gamma_t \lambda_t \tilde{\mathbf{e}}_{t-1} + \omega_t \nabla V_{\mathbf{w}}(S_t)$, and δ_t shorthand for the TD error, $\tilde{\mathbf{e}}_t$ the selective instantaneous eligibility trace, α_t is the step size parameter, $\omega : \mathcal{S} \to [0, 1]$ is the weighting function, $\lambda : \mathcal{S} \to [0, 1]$ is the trace-decay function, $\gamma : \mathcal{S} \to [0, 1]$ is the temporal discounting function. Additionally, we assume the following mild conditions:

1. the state space is finite
2. the feature function $\mathbf{x} : \mathcal{S} \to \mathbb{R}^n$ s.t. the $\mathbf{X} \in \mathbb{R}^{|S| \times n}$ has linearly independent columns, with bounded variance;
3. the rewards are bounded;
4. the step-size sequence satisfies the Robbins-Monro conditions [Robbins and Monro, 1951]: $\sum_{t=0}^{\infty} \alpha_t = \infty$ and $\sum_{t=0}^{\infty} \alpha_t^2 < \infty$;
5. $\gamma : \mathcal{S} \to [0, 1]$ s.t. $\prod_{k=1}^{\infty} \gamma(S_{t+k}) = 0$, w.p. 1, $\forall t > 0$;
6. $\omega : \mathcal{S} \to [0, 1]$ s.t. $\prod_{k=1}^{\infty} \omega(S_{t+k}) \neq 0$, w.p. 1, $\forall t > 0$;

7. experience is sampled on-policy from the Markov chain

$$(S_t, A_t, R_{t+1}, S_{t+1}) \sim d_\pi(S_t)\pi(A_t|S_t)P(S_{t+1}|S_t, A_t) \text{ with stationary distribution } d_\pi.$$

Proposition 2. *For $V_{\mathbf{w}}(s) = \mathbf{w}^\top \mathbf{x}(s)$, the semi-gradient update of selective TD(λ, ω) in Eq. (B.9) with $\omega(s) = 1 - \gamma(s)\lambda(s)$ converges to the fixed point:*

$$\mathbf{w}^* = \mathbb{E}\left[\left(\sum_{k=0}^{t} \gamma_{t-k}^{(k)} \lambda_{t-k}^{(k)} \omega_{t-k} \mathbf{x}_{t-k}\right) \left(\gamma_{t+1}\mathbf{x}_{t+1} - \mathbf{x}_t\right)^\top\right]^{-1} \mathbb{E}\left[\left(\sum_{k=0}^{t} \gamma_{t-k}^{(k)} \lambda_{t-k}^{(k)} \omega_{t-k} \mathbf{x}_{t-k}\right) R_{t+1}\right], \tag{B.10}$$

with $\gamma_t^{(k)} = \prod_{j=t+1}^{t+k} \gamma_j$, $\lambda_t^{(k)} = \prod_{j=t+1}^{t+k} \lambda_j$.

Proof. Firstly, under the assumptions 1-4 above, the stochastic algorithm TD(λ, ω) (Eq. (B.9)) behaves like the corresponding expected update equation under the on-policy stationary distribution d_π:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha_t \left(\mathbb{E}_{d_\pi}[\tilde{\mathbf{e}}_t R_{t+1}] - \mathbb{E}_{d_\pi}[\tilde{\mathbf{e}}_t (\mathbf{x}_t - \gamma_t \mathbf{x}_{t+1})^\top]\mathbf{w}_t\right), \tag{B.11}$$

and $\mathbf{w}_t$ converges with probability 1 to $\mathbf{w}^*$, the solution to the expected update equation (B.11).

Stability We begin by showing stability of TD(λ, ω). Let $Z_t = (S_t, A_t, \tilde{\mathbf{e}}_t)$ for $t \geq 0$ be the Markov chain resulting from adding $\tilde{\mathbf{e}}$ to the stationary Markov chain $\{(S_t, A_t)\}_{t=0}^{\infty}$ with transition probabilities given by P_π, s.t.:

$$\tilde{\mathbf{e}}_t = \sum_{k=0}^{t} \left(\prod_{j=0}^{k-1} \gamma_{t-j}\lambda_{t-j}\right) \omega_{t-k}\mathbf{x}_{t-k}\,.$$

Since $\tilde{\mathbf{e}}_t$ and $\mathbf{x}_{t+1}$ are deterministic functions of (S_t, A_t) and the distribution of s_{t+1} only depends of s_t, the resulting chain Z_t is Markov. Let $\mathbb{E}_{d_\pi}[\cdot]$ denote the expectation with respect to the steady state distribution d_π.

Let $\mathbf{A}(Z_t) = \tilde{\mathbf{e}}_t(\mathbf{x}_t - \gamma_t \mathbf{x}_{t+1})^\top$, $\mathbf{b}(Z_t) = \tilde{\mathbf{e}}_t R_{t+1}$ and $\mathbf{A} = \mathbb{E}_{d_\pi}[\tilde{\mathbf{e}}_t(\mathbf{x}_t - \gamma_t \mathbf{x}_{t+1})^\top]$, $\mathbf{b} = \mathbb{E}_{d_\pi}[\tilde{\mathbf{e}}_t R_{t+1}]$. The fixed point equation of the deterministic system is:

$$\mathbf{w}_{t+1} = (\mathbf{I} - \alpha_t \mathbf{A})\mathbf{w}_t + \alpha_t \mathbf{b}$$

Since these methods are not true gradient methods, the asymptotic behaviour of any TD algorithm generally depends on a stability criteria that requires the eigenvalues of the iteration matrix $\mathbf{A}$ have positive real components [Sutton et al., 2016b]. The stochastic algorithm converges if and only if the deterministic algorithm converges [Sutton, 1988] and if both algorithms converge, they converge to the same fixed point.

Let $\mathbf{\Lambda}$ and $\mathbf{\Gamma}$ be matrices with diagonal entries corresponding to the functions λ and γ. Let

$$\mathcal{P}_\pi^\lambda = (\mathbf{I} - \mathcal{P}_\pi \mathbf{\Gamma} \mathbf{\Lambda})^{-1} \mathbf{\Gamma}(\mathbf{I} - \mathbf{\Lambda}) \mathcal{P}_\pi \,,$$

such that:

$$\begin{aligned}
(\mathbf{I} - \mathcal{P}_\pi \mathbf{\Gamma} \mathbf{\Lambda})^{-1}(\mathbf{I} - \mathcal{P}_\pi \mathbf{\Gamma}) = & && \text{(B.12)} \\
= (\mathbf{I} - \mathcal{P}_\pi \mathbf{\Gamma} \mathbf{\Lambda})^{-1}(\mathbf{I} - \mathcal{P}_\pi \mathbf{\Gamma} \mathbf{\Lambda} + \mathcal{P}_\pi \mathbf{\Gamma} \mathbf{\Lambda} - \mathcal{P}_\pi \mathbf{\Gamma}) & && \text{(B.13)} \\
= (\mathbf{I} - \mathcal{P}_\pi \mathbf{\Gamma})^{-1}(\mathbf{I} - \mathcal{P}_\pi \mathbf{\Gamma} \mathbf{\Lambda} + \mathcal{P}_\pi \mathbf{\Gamma}(\mathbf{\Lambda} - \mathbf{I})) & && \text{(B.14)} \\
= \mathbf{I} - (\mathbf{I} - \mathcal{P}_\pi \mathbf{\Gamma} \mathbf{\Lambda})^{-1} \mathbf{\Gamma}(\mathbf{I} - \mathbf{\Lambda}) \mathcal{P}_\pi & && \text{(B.15)} \\
= \mathbf{I} - \mathcal{P}_\pi^\lambda \,. & && \text{(B.16)}
\end{aligned}$$

Expanding the $\mathbf{A}$ and $\mathbf{b}$ matrices we have:

$$\mathbf{A} = \mathbf{X}^\top \underbrace{\widetilde{\mathbf{D}}_\pi (\mathbf{I} - \mathcal{P}_\pi \mathbf{\Gamma} \mathbf{\Lambda})^{-1} (\mathbf{I} - \mathcal{P}_\pi \mathbf{\Gamma})}_{\mathbf{K}} \mathbf{X},$$

$$\mathbf{b} = \mathbf{X}^\top \widetilde{\mathbf{D}}_\pi (\mathbf{I} - \mathcal{P}_\pi \mathbf{\Gamma} \mathbf{\Lambda})^{-1} \mathbf{r},$$

where $\tilde{\mathbf{D}}_\pi$ is a diagonal matrix with elements $d_\pi(s)\omega(s)$ on the diagonal. Following Sutton et al. [2016b], we refer to $\mathbf{K}$ as the *"key matrix"*.

To ensure convergence regardless of the representation function, we use the assumption that $\mathbf{X}$ is full rank and require cf. Sutton et al., 2016b that:

1. the diagonal entries of $\mathbf{K}$ are non-negative
2. the off-diagonal entries are non-positive
3. the row sums are non negative
4. the columns sums are positive

Conditions (1-3) follow from Lemma 4 of White [2017] and the fact that $\tilde{\mathbf{D}}_\pi$ is a non-negative diagonal weighting matrix.

For the last condition, if we assume $\omega(s) > 0, \forall s \in \mathcal{S}$, then, similarly to [Sutton et al., 2016b], we have:

$$
\begin{aligned}
\mathbf{1}^\top \mathbf{K} &= \mathbf{1}^\top \tilde{\mathbf{D}}_\pi \left(\mathbf{I} - \mathcal{P}_\pi \mathbf{\Gamma} \mathbf{\Lambda}\right)^{-1} \left(\mathbf{I} - \mathcal{P}_\pi \mathbf{\Gamma}\right) \\
&\quad \text{(using } \mathbf{h} = \mathbf{I} - \mathbf{\Lambda}\mathbf{\Gamma}\text{)} \\
&= \mathbf{1}^\top \mathbf{D}_\pi (\mathbf{I} - \mathbf{\Gamma}\mathbf{\Lambda}) \left(\mathbf{I} - \mathcal{P}_\pi \mathbf{\Gamma} \mathbf{\Lambda}\right)^{-1} \left(\mathbf{I} - \mathcal{P}_\pi \mathbf{\Gamma}\right) \\
&= (\mathbf{d}_\pi^\top - \mathbf{d}_\pi^\top \mathbf{\Gamma}\mathbf{\Lambda}) \left(\mathbf{I} - \mathcal{P}_\pi \mathbf{\Gamma} \mathbf{\Lambda}\right)^{-1} \left(\mathbf{I} - \mathcal{P}_\pi \mathbf{\Gamma}\right) \\
&\quad \text{(using } \mathbf{d}_\pi^\top = \mathbf{d}_\pi^\top \mathcal{P}_\pi\text{)} \\
&= (\mathbf{d}_\pi^\top - \mathbf{d}_\pi^\top \mathcal{P}_\pi \mathbf{\Gamma}\mathbf{\Lambda}) \left(\mathbf{I} - \mathcal{P}_\pi \mathbf{\Gamma} \mathbf{\Lambda}\right)^{-1} \left(\mathbf{I} - \mathcal{P}_\pi \mathbf{\Gamma}\right) \\
&= \mathbf{d}_\pi^\top (\mathbf{I} - \mathcal{P}_\pi \mathbf{\Gamma}\mathbf{\Lambda}) \left(\mathbf{I} - \mathcal{P}_\pi \mathbf{\Gamma} \mathbf{\Lambda}\right)^{-1} \left(\mathbf{I} - \mathcal{P}_\pi \mathbf{\Gamma}\right) \\
&= \mathbf{d}_\pi^\top \left(\mathbf{I} - \mathcal{P}_\pi \mathbf{\Gamma}\right) \\
&\quad \text{(using } \mathbf{d}_\pi^\top = \mathbf{d}_\pi^\top \mathcal{P}_\pi\text{)} \\
&= \mathbf{d}_\pi^\top (\mathbf{I} - \mathbf{\Gamma}) > 0
\end{aligned}
$$

Using $\omega(s) = 1 - \gamma(s)\lambda(s)$, all components of the column sums become positive. Thus, the key matrix $\mathbf{A}$ is positive definite and the selective TD(λ, ω) algorithm is stable.

If $\exists s \in \mathcal{S}$ s.t. $\omega(s) = 0$, then we can set $\gamma(s) = \lambda(s) = 1$, which induces a new super-imposed MDP, with temporally-extended dynamics comprising of multi-step transition dynamics and multi-step cumulated rewards of the original MDP. The new MDP is then stable by invoking the argument proved above.

Convergence of TD(λ_t, ω_t). Stability is a prerequisite for full convergence of the stochastic algorithm. For full convergence, we can apply Theorem 2 from Tsitsiklis and Van Roy [1996], adapted and restated below.

Theorem 2 from Tsitsiklis and Van Roy [1996] Consider an iterative algorithm of the form:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha_t(-A(Z_t)\mathbf{w}_t + b(Z_t)),$$

where:

1. the (predetermined) step-size sequence α_t is positive, non-increasing, and satisfies $\sum_{t=0}^{\infty} \alpha_t = \infty$ and $\sum_{t=0}^{\infty} < \infty$;
2. Z_t is a Markov process with a unique invariant distribution, and there exists a mapping h from the states of the Markov process to the positive reals, satisfying the remaining conditions. Let $\mathbb{E}_{d_\pi}[\cdot]$ stand for the expectation with respect to this invariant distribution;
3. $\mathbf{A}(\cdot)$ and $\mathbf{b}(\cdot)$ are matrix and vector valued functions, respectivly, for which $\mathbf{A} = \mathbb{E}_{d_\pi}[\mathbf{A}(Z_t)]$ and $\mathbf{b} = \mathbb{E}_{d_\pi}[\mathbf{b}(Z_t)]$ are well-defined and finite;
4. the matrix $\mathbf{A}$ is positive definite;
5. there exist constants C and q such that for all Z:

$$\sum_{t=0}^{\infty} \|\mathbb{E}[\mathbf{A}(Z_t)|Z_0 = Z] - \mathbf{A}\| \leq C(1 + h^q(Z)) \text{ and}$$
$$\sum_{t=0}^{\infty} \|\mathbb{E}[\mathbf{b}(Z_t)|Z_0 = Z] - b\| \leq C(1 + h^q(Z))\,;$$

6. for any $q > 1$ there exists a constant μ_q such that for all Z, t

$$\mathbb{E}[h^q(Z_t)|Z_0 = Z] \leq \mu_q(1 + h^q(Z)).$$

Then, $\mathbf{w}_t$ converges to $\mathbf{w}^*$, with probability 1, where $\mathbf{w}^*$ is the unique vector that satisfies $\mathbf{A}\mathbf{w}^* = \mathbf{b}$.

The assumptions of Theorem 2 hold in our case since the last two remaining conditions (v) and (vi), stating that the dependence of $\mathbf{A}(Z_t)$ and $\mathbf{b}(Z_t)$ on $Z_k, \forall k \leq t$ is exponentially decreasing, are satisfied by the fact that the trace iterates have bounded variance and the fact that $\{Z_t\}_{t=0}^{\infty}$ is Markov, cf. Assumption 3, Tsitsiklis and Van Roy, 1996. The first follows from the fact that the range of ω and λ is $[0, 1]$, and the second by definition of the trace.

Fixed point. We now examine the fixed point of the system:

$$\mathbb{E}[\delta_t \tilde{\mathbf{e}}_t] = \mathbb{E}[\tilde{\mathbf{e}}_t(R_{t+1} + \gamma_t \mathbf{x}_{t+1}^\top \mathbf{w} - \mathbf{x}_t^\top \mathbf{w})] = 0\,,$$

Unfolding the trace, we have:

$$\begin{aligned}
\tilde{\mathbf{e}}_t &= \gamma_t \lambda_t \tilde{\mathbf{e}}_{t-1} + \omega_t \mathbf{x}_t \\
&= \omega_t \mathbf{x}_t + \gamma_t \lambda_t (\gamma_{t-1} \lambda_{t-1} \tilde{\mathbf{e}}_{t-2} + \omega_{t-1} \mathbf{x}_{t-1}) \\
&= \sum_{k=0}^{t} \left(\prod_{j=0}^{k-1} \gamma_{t-j} \lambda_{t-j} \right) \omega_{t-k} \mathbf{x}_{t-k}
\end{aligned}$$

which results in:

$$\mathbf{w}^* = \mathbb{E}\left[\left(\sum_{k=0}^{t} \gamma_{t-k}^{(k)} \lambda_{t-k}^{(k)} \omega_{t-k} \mathbf{x}_{t-k} \right) (\gamma_{t+1} \mathbf{x}_{t+1} - \mathbf{x}_t)^\top \right]^{-1} \mathbb{E}\left[\left(\sum_{k=0}^{t} \gamma_{t-k}^{(k)} \lambda_{t-k}^{(k)} \omega_{t-k} \mathbf{x}_{t-k} \right) R_{t+1} \right],$$

with $\gamma_t^{(k)} = \prod_{j=t+1}^{t+k} \gamma_j$, $\lambda_t^{(k)} = \prod_{j=t+1}^{t+k} \lambda_j$. □

3 Details on empirical illustrations

3.1 Weightings for off-policy distribution correction

Experimental setup We used the 5-state MRP denoted in Fig. B.1 to illustrate the following policy-evaluation algorithms: Off-policy TD, ETD, X(1)-ETD, and X(0)-ETD, described next.

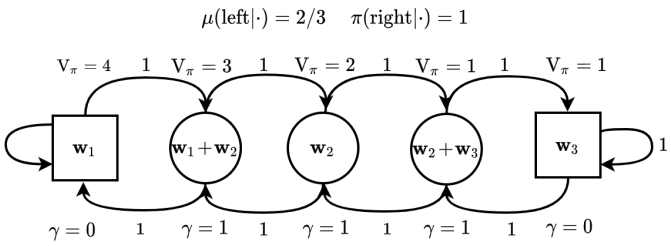


Figure B.1: The 5-state chain MDP (cf. Sutton et al. [2016b]): The states shown as square cause soft termination, with $\gamma(s) = 0$, but they do not restart the agent. There are two actions, "left" and "right", which deterministically cause transitions to the left or right except at the ends of the chain, where there may be a self-transition. The reward on all transitions is $+1$. The behaviour policy selects "left" 2/3rds of the time in all states, which causes more time to be spent in states on the left than on the right. The true value function V_π is depicted above each state. The notation $\mathbf{w}_i$ describes state aliasing in the observations, denoting that the i-th component of the current parameter vector is $\mathbf{w}_i$. Since there are five states and only three parameters, it is impossible, to represent the true V_π exactly.

Algorithms & hyperparameters The Off-policy TD algorithm is the canonical off-policy evaluation algorithm [Precup et al., 2001], without any prior distribution corrections in the form of stationary distribution ratios between the target policy and the behaviour policy; it only uses importance sampling ratios ρ.

The ETD algorithm is the one introduced in [Sutton et al., 2016b].

For the two emphatic algorithms that use expected traces, we differentiate with η^f (cf. 2.3) the following instances of the generic algorithm X(η_f)-ETD:

- X(1)-ETD – learns the follow-on trace with Monte-Carlo regression on the follow-on trace, cf. Eq. (B.5) with $\eta_f = 1$;
- X(0)-ETD – learns the follow-on by backward TD, i.e. the follow-on trace model bootstraps on itself, cf. Eq. (B.5) with $\eta_f = 0$.

All the algorithms are compared for $\lambda = 0$. We use no other internal discount factor beside the chain's own termination function γ. The step-sizes for the value function, for all algorithms, are decayed with $1/t^d$, where t is the time-step, and d is chosen from $\{0.4, 0.5, 0.8, 0.9\}$, with the best values obtained through hyperparameter search: (i) Off-policy TD: $d = 0.5$, (ii) ETD: $d = 0.9$, (iii) X(1)-ETD: $d = 0.9$, (iv) X(0)-ETD: $d = 0.9$.

For the two algorithms that use expected emphasis, learning rates for the model f_φ that estimates the follow-on trace F, use the same step-size decay schedule $1/t^{d_f}$, with values for d_f chosen from the same interval, and the best value obtained for both algorithms $d_f = 0.5$. The model is a linear function approximator on the observations, followed by a ReLU non-linearity to keep the output positive (the latter is not important, as similar results can be obtained without it).

3.2 Weightings for on-policy distribution correction – Atari Ms.Pac-Man experiments

For our deep reinforcement learning experiments on Atari games, we used the canonical Ms.Pac-Man to illustrate the importance of the connection between selectivity ω, trace-decay λ and trace-bootstrapping η. We start by describing the experimental setup, followed by the online selective Q(λ, ω) algorithms, after which we move on to the expected traces versions, namely QET(λ, η, ω).

Experimental setup All the Atari experiments were run with the ALE [Bellemare et al., 2012], similarly to van Hasselt et al. [2020], including using action repeats (4x), but without downsampling (or framestacking, or pooling) the observation stream. With probability ϵ, the agent's observation is replaced with random standard Gaussian noise, to mimic a noisy observation sensor. To simulate access to a module that detects such noisy observations, we provide access to a time-dependent interest i_t, capturing whether an observation is noisy or not, s.t. $i_t = 0$ if the observation at time step t is noisy, and $i_t = 1$ otherwise. We use no other corrections, e.g. corrections to the discrepancy between the behaviour ϵ-greedy and the target greedy policy, as we found those to not be useful in this setting, resulting in $\omega_t \equiv i_t$.

We keep the discount factor $\gamma = 0.99$ constant. We report the mean return while training on 20M frames.

Algorithm 11 Selective Credit Assignment - QET(λ, η, ω)

Initialize: policy $\pi, \mathbf{w}, \mathbf{\Theta}, \omega, \gamma, \lambda, \eta$
$S \sim \text{env}()$
for each interaction $\{1, 2 \dots T\}$ **do**
  $A \sim \pi(S), R, \gamma, S' \sim \text{env}(A)$
  $\mathbf{\Theta} \leftarrow \mathbf{\Theta} + \alpha^{\mathbf{\Theta}} \frac{\partial \tilde{\mathbf{z}}_{\mathbf{\Theta}}(S)}{\partial \mathbf{\Theta}} (\gamma \lambda(S) \tilde{\mathbf{e}} - \tilde{\mathbf{z}}_{\mathbf{\Theta}}(S))$
  $\tilde{\mathbf{e}} \leftarrow \eta_t \gamma \lambda(S) \tilde{\mathbf{e}} + (1 - \eta_t) \tilde{\mathbf{z}}_{\mathbf{\Theta}}(S) + \omega(S) \nabla Q_{\mathbf{w}}(S, A)$
  $R^\lambda = R + \gamma(1 - \lambda(S')) Q_{\mathbf{w}}(S', a')$
  $\Delta^Q = R^\lambda \tilde{\mathbf{e}} - Q_{\mathbf{w}}(S, A) \nabla Q_{\mathbf{w}}(S, A) \omega(S)$
  $a' = \arg\max_a Q_{\mathbf{w}}(S', a)$
  $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \Delta^Q$
  if S is terminal: $S \sim \text{env}()$; $\tilde{\mathbf{e}} \leftarrow 0$

In all cases, we used ϵ-greedy exploration (cf. Sutton and Barto [2018]), with an $\epsilon = 1e-2$, which we do not decay.

We use a similar setting to van Hasselt et al. [2020], with the following modifications. We apply a different feature extraction network, cf. Hessel et al. [2019]; particularly, we use 2 convolutional layers, with 5×5 kernels, stride 5, and 64 channels, followed by ReLU nonlinearities. The output is then passed through a 512 linear layer, with ReLU activation.

These experiments were conducted using Jax Bradbury et al. [2018], Haiku Hennigan et al. [2020] and Optax Hessel et al. [2020].

Algorithms evaluated We start by describing the baseline algorithm Q(λ), followed by the algorithms that specifically apply selectivity.

Q(λ) In the algorithm 11, we have $\eta_t = 1 \forall t$, i.e. we only use accumulating traces, so we omit line 5, since this algorithm does not use the expected traces $\tilde{\mathbf{z}}_{\mathbf{\Theta}}$. For each transition, we first decay the trace $\tilde{\mathbf{e}}$ and then update it using line 6. We further compute the finite-horizon one-step return R^λ (line 7), where $\gamma = 0$ on termination (and then S' is the first observation of the next episode). Instead of the usual SGD algorithm illustrated for simplicity in line 10, for training, we use ADAM [Kingma and Ba, 2015]; we learn the value function with momentum$= 0.9$, which simulates soft-batching; we set the other parameters from ADAM to $b_1 = 0.99$ (the exponential decay rate to track the first moment of past gradients) and

$b_2 = 0.9999$ (the exponential decay rate to track the second moment of past gradients), $\epsilon = 1e-4$ (the small constant applied to denominator outside the square root–as in [Kingma and Ba, 2015], to avoid dividing by zero when rescaling). We use a step-size of $\alpha^{\mathbf{w}} = 1e-5$ for learning the value function.

Variations of Q(λ, ω) The algorithms used in the experiments are all variations of Q(λ, ω). We label Q(λ), the default baseline algorithm that uses $\lambda = 0.9$ and uniform weightings over the state space: $\omega = \omega_t = 1, \forall t$. We use Q($\lambda, \omega_t$) (with the "$t$" subscript denoting state or time-dependence) for the algorithm that uses $\lambda = 0.9$, but uses the ground-truth interest to set the weighting ω_t (1 for non-noisy states, and 0 otherwise). Lastly, Q(λ_t, ω_t) denotes the algorithm using, in addition to the ground truth interest in setting ω_t, also Eq. (4.12) to set λ_t.

QET(λ, η, ω) The expected-traces algorithm is similar to Q(λ, ω), except we now use the expected traces $\tilde{\mathbf{z}}_{\boldsymbol{\Theta}}$ in place of the instantaneous traces, so we update the parameters $\boldsymbol{\Theta}$ as well, in addition to $\mathbf{w}$, using ADAM, with the same hyperparameters as for the value function: momentum, b_1, b_2, and ϵ. We use the step size $\alpha^{\boldsymbol{\Theta}} = 1e-2$ for learning the expected traces.

Similarly to van Hasselt et al. [2020], we also split the computation of $Q(s, a)$ into two separate parts, such that $Q(\mathbf{w}, \zeta)(s, a) = \mathbf{w}_a^\top \mathbf{x}_\zeta(s)$. This separation is just so that we can keep labeled separate subsets of parameters as $(\mathbf{w}, \zeta)$ rather than merging all of them into a single vector $\mathbf{w}$, using $\mathbf{x}(s)$ to denote the last hidden layer of the feature extraction part of the network, on top of which the last linear layer of the q-function is applied. We keep separate traces for these subsets, and we just apply accumulating instantaneous traces to the feature extraction network, similarly to van Hasselt et al. [2020]. This separation is equivalent to keeping one big trace for the combined set. We refer the reader to van Hasselt et al. [2020] for more details on this particularity. The motivation for this split in parameters is to avoid learning an expected trace for the full trace, which has millions of

elements. Instead, in practice, we only learn expectations for traces corresponding to the last layer.

The difference from how this algorithm is presented in van Hasselt et al. [2020] is that we only condition the function $z_{\boldsymbol{\Theta}}(s)$ on the state, and not the action, due to our new derivation of the algorithm in 2.1.

As customary, we do not backpropagate the gradient coming from the expected traces' loss further into the feature representation.

Variations of QET(λ, η, ω) We now describe the variations of QET compared in the experiment. For the first versions of the algorithms, we use $\eta = 0, \forall t$, i.e. using expected traces everywhere, instead of the instantaneous counterparts. The algorithms QET(λ, η)-baseline, QET(η, λ, ω_t), QET($\eta, \lambda_t, \omega_t$) are analogous to Q($\lambda$)-baseline, Q($\lambda, \omega_t$), and QET($\lambda_t, \omega_t$), respectively, which were described in the previous section. The algorithm QET($\eta_t, \lambda_t, \omega_t$), in addition to using Eq. (4.12) for setting λ_t, and the ground truth interest for setting ω_t, also uses Eq. (4.16) to set η_t, i.e. it uses the expected trace more in states where the selectivity weighting ω is higher, and the instantaneous traces more when the weighting is lower. Lastly, QET($\eta_t, \lambda_t, \omega_t, \tilde{\Delta}_t^{\mathbf{z}}$), in addition, uses the coupling Eq. (4.17) to learn expected traces constrained by selectivity.

3.3 Off-policy counterfactual evaluation

Discussion on learning with function approximation With function approximation, learning expected traces off-policy can be problematic. Particularly, one can learn selective expected eligbility traces $\tilde{\mathbf{z}}_{\boldsymbol{\Theta}}$ using Monte-Carlo methods, by regressing on $\gamma_t \rho_t \lambda_t \tilde{\mathbf{e}}_{t-1} + \omega_t \nabla V_{\mathbf{w}_t}(S_t)$, but the traces can have very high variance resulting from the product of importance sampling ratios. On the other hand, learning the traces with backward TD can easily diverge due to "off-policiness", from the same reasons the value learning process can diverge. We can interpolate between Monte-Carlo methods and backward TD using *selective mixture traces*, similarly to regular mixture traces, using a different mixture

parameter $\tilde{\eta}$, s.t.:

$$\tilde{\mathbf{e}}_t^{\tilde{\eta}} = (1-\tilde{\eta})\tilde{\mathbf{z}}_{\boldsymbol{\Theta}_t}(S_t) + \tilde{\eta}(\rho_t\gamma_t\lambda_t\tilde{\mathbf{e}}_{t-1}^{\tilde{\eta}} + \omega_t\nabla V_{\mathbf{w}_t}(S_t))\,, \tag{B.17}$$

and updating $\boldsymbol{\Theta}$ with

$$\begin{aligned}\Delta_t^{\mathbf{z}} &\equiv \frac{\partial\tilde{\mathbf{z}}_{\boldsymbol{\Theta}_t}(S_t)}{\partial\boldsymbol{\Theta}_t}\Big(\rho_t\gamma_t\lambda_t\tilde{\mathbf{e}}_{t-1}^{\tilde{\eta}} + \omega_t\nabla V_{\mathbf{w}_t}(S_t) - \tilde{\mathbf{z}}_{\boldsymbol{\Theta}_t}(S_t)\Big)\,,\\ \boldsymbol{\Theta}_{t+1} &= \boldsymbol{\Theta}_t + \alpha_t^{\boldsymbol{\Theta}}\Delta_t^{\mathbf{z}}\,,\end{aligned}$$

where $\alpha_t^{\boldsymbol{\Theta}} \in (0,1)$ is a step-size parameter and $\tilde{\eta}$ is analogous to λ, but for learning selective traces here, rather than values, using TD methods when $\tilde{\eta}=0$, and Monte-Carlo regression when $\tilde{\eta}=1$.

To stabilize learning, we could choose an intermediary value for $\tilde{\eta}$, that achieves an optimal balance between bias and variance. Moreover, we can also stabilize the learning process by instantiating ω using emphatic weightings, which can guarantee convergence for linear function approximation.

We now discuss the details regarding the empirical illustration in the Open world grid-world domain we used. This illustration is meant to illustrate the learning process, so we do not use function approximation, therefore we omit any non-uniform weightings.

Experimental setup In Fig. 4.5-Left, there are two goals depicted with "G", giving the same reward of $10/\epsilon_r$, with probability ϵ_r, otherwise 0. All other rewards are 0. When the agent reaches the goal, the episode restarts with a random initial location sampled from the state space.

The agent learns different expected traces for the two policies transitioning it to each of the two corners of the world: top-right and bottom-left (we may call those loosely options), while following a random behaviour policy μ.

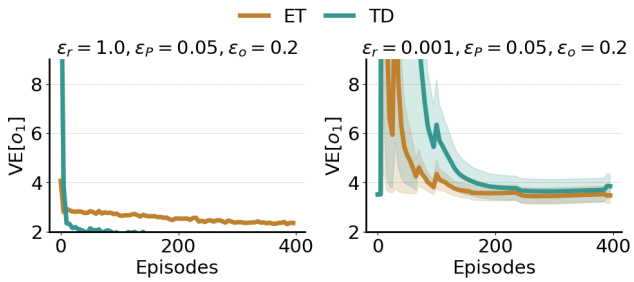


Figure B.2: Open World: Value error for policy evaluation using TD($\lambda = 0.98$) and ET($\lambda = 0.98$), for option o_2. Increasing the sparsity of the reward signal (left vs right plot), we observe policy-conditioned expected traces coped better with sparse rewards. Shaded areas show standard errors over 20 seeds.

To increase the difficulty in learning, we let each option's policy take a random action with probability $\epsilon_o = 0.2$, and let the environment transition the agent randomly in a cardinal direction with probability $\epsilon_p = 0.05$. We use one-hot state representations, so we omit emphatic weightings, and just use importance sampling ratios between each option's policy π_o and the behaviour policy μ, when learning the expected trace of each option.

Algorithms & hyperparameters We compare the off-policy TD(λ) and ET(λ, η) with uniform weightings. We clip the importance sampling ratios to 1. We learn the traces with Monte-Carlo regression, using $\tilde{\eta} = 1$ in Eq. (B.17). The discount is $\gamma = 0.99$, except at the goals, where it is 0. We set $\lambda = 0.98$. We decay step-sizes with $1/t^d$, with $d \in \{0.5, 0.7, 0.9, 1.\}$, the best values obtained through hyperparameter search: $d = 0.5$, for $\epsilon_r = 1.$, and $d = 0.9$, for $\epsilon_r = 0.001$. For the expected traces algorithm, we similarly decay the step size for the trace learning process, using $d_z = 0.001$, searched over values $\{0.1, 0.01, 0.001\}$. We learn both the value function and the traces with SGD, starting from a step size of 1.

Additional results Fig. B.2 illustrates the performance for the option with policy transitioning toward the bootom corner—o_2 (with similar results illustrated for the other policy—o_1, in the main text).

3.4 Sparse expected eligibility traces

To illustrate the idea of sparse expected traces that can flow credit over the state space using temporally-extended actions, we use the following experimental setup.

Experimental setup We use the Four Rooms domain from figure Fig. 4.5-Right. We pre-learn the option policies using pre-defined interest and sub-goal functions, such that the options learn policies as illustrated in Fig. 4.5-Right. The agent receives a reward of $r = 10./\epsilon_r$ with probabiltiy ϵ_r at the goal, depicted with "G" in Fig. 4.5-Right. The discount is $\gamma = 0.98$ everywhere, except at the goal where it is 0, after which the episode restarts and the first observation of the new episode is sampled from one of the hallways.

Algorithms & hyperparameters The baselines we compare against are the default $\text{Q}_A(\lambda = 0.9)$ and $\text{QET}_A(\lambda = 0.9, \eta = 0)$, which are defined over the primitive action-space.

For the sparse learning algorithms, we assume the policy over actions is defined using a pre-specified weighting function ω, s.t. $\omega(s) = 1$, if s is a hallway, and 0 otherwise. The policy over options uses only the pre-learned option space, without any primitive actions, which is sufficient for this illustration since the start states are in hallway locations, and so is the goal.

For $\text{Q}_O(\lambda_t, \omega_t)$ and $\text{QET}_A(\lambda_t, \eta_t, \omega_t)$ (with the "t" subscript indicating state or time-dependence), we use Eq. (4.12) for setting λ_t, with $\beta_\lambda = 0.9$, and similarly Eq. (4.16) for setting η_t, with $\beta_\eta = 0.9$. We also learn the traces themselves using Eq. (4.17). We decay all learning rates for the value function using $1/t^d$, searching over $d \in \{0.1, 0.2, 0.5, 0.7, 0.9, 1.\}$. We find the best parameters are $d = 0.2$ for all algorithms. For the expected traces algorithms, we use the same scehdule $1/t^{d_z}$, with $d_z \in \{1., 0.1, 0.01, 0.01\}$, finding the best value is $d_z = 0.1$, for both algorithms. We learn both the value function and the traces with SGD, starting from a step size of 1. For exploration we use ϵ-greedy policies with $\epsilon = 0.1$. The options' policies are learned with intra-option discount factor $\gamma_O = 0.9$ and without randomness $\epsilon_O = 0.$,

i.e. greedy policies. The trace learning process uses Monte-Carlo regression, i.e. $\tilde{\eta} = 1$ in Eq. (B.17). We do not use importance sampling ratios, or any other corrections.

C Acceleration in Policy Optimization

1 Proofs and derivations

1.1 Proofs and Derivations for Sec.1: Background & Preliminaries

1.1.1 Functional Policy Gradient

The Performance Difference Lemma (PDL) is a property that relates the difference in values of policies to the policies themselves.

Lemma 1. ***(Performance Difference Lemma from Kakade and Langford [2002])*** *For any policies π_{t+1} and π_t, and an initial distribution ρ*

$$\begin{aligned} V^{\rho}_{t+1} - V^{\rho}_{t} &= {}^{1}\!/_{1-\gamma} \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} d^{\rho}_{t+1}(s)(\pi_{t+1}(s,a) - \pi_t(s,a), Q_t(s,a)) \\ &= {}^{1}\!/_{1-\gamma} \mathbb{E}_{s\sim d^{\rho}_{t+1}} \left[\langle Q_t(s), \pi_{t+1}(s) - \pi_t(s)\rangle\right] \end{aligned}$$

Proof. According to the definition of the value function

$$\begin{aligned} V_{t+1}(s) - V_t(s) &= \langle Q_{t+1}(s), \pi_{t+1}(s)\rangle - \langle Q_t(s), \pi_t(s)\rangle \\ &= \langle Q_t(s), \pi_{t+1}(s) - \pi_t(s)\rangle + \langle Q_{t+1}(s) - Q_t(s), \pi_{t+1}(s)\rangle \\ &= \langle Q_t(s), \pi_{t+1}(s) - \pi_t(s)\rangle + \gamma \sum_{s'} \sum_{a} P(s'|s,a)\pi_{t+1}(s,a)[V_{t+1}(s') - V_t(s')] \\ &= {}^{1}\!/_{1-\gamma} \sum_{s'} d_{t+1}(s \to s')\langle Q_t(s'), \pi_{t+1}(s') - \pi_t(s')\rangle \end{aligned}$$

□

The following lemma is a version of the policy gradient theorem [Sutton et al., 1999b] applied to the direct policy representation—the functional representation of the policy probabilities, and has appeared in various forms in Agarwal et al. [2019], Bhandari and Russo [2019, 2021], Russo [2022].

Lemma 2. ***(Policy Gradient Theorem for Directional Derivatives)*** *For any two policies* π_{t+1}, $\pi_t \in \Pi$,

$$\begin{aligned}\langle \nabla V_t^\rho, \pi_{t+1} - \pi_t \rangle &= \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} d_t(s) Q_t(s,a)(\pi_{t+1}(s,a) - \pi_t(s,a)) \\ &= \mathbb{E}_{s\sim d_t}\left[\langle Q_t(s), \pi_{t+1}(s) - \pi_t(s)\rangle\right]\end{aligned}$$

Proof. A Taylor expansion using the Performance Difference Lemma 1 reveals

$$\begin{aligned}V_{t+1}^\rho - V_t^\rho &= {}^1\!/_{1-\gamma} \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} d_{t+1}^\rho(s)(\pi_{t+1}(s,a) - \pi_t(s,a), Q_t(s,a)) \\ &= \mathbb{E}_{s\sim d_{t+1}^\rho}\left[\langle Q_t(s), \pi_{t+1}(s) - \pi_t(s)\rangle\right] \\ &= {}^1\!/_{1-\gamma}\mathbb{E}_{s\sim d_t^\rho}\left[\langle Q_t(s), \pi_{t+1}(s) - \pi_t(s)\rangle\right] \\ &\quad + {}^1\!/_{1-\gamma} \underbrace{\sum_{s\in\mathcal{S}} (d_{t+1}^\rho(s) - d_t^\rho(s))\langle Q_t(s), \pi_{t+1}(s) - \pi_t(s)\rangle}_{=O(\|\pi_{t+1}(s)-\pi_t(s)\|^2)}\end{aligned}$$

The last error term is second-order since P_π is linear in π and then d_π^ρ is differentiable in π.

□

The next lemma states that the MD method minimizes the local linearization of a function while not moving too far away from the previous point, with distances measured via the Bregman divergence of the mirror map.

Lemma 3. ***(Proximal perspective on mirror descent)*** *The MD update for* $x \in \mathcal{X} \cap \mathcal{C}$*, with mirror map* $h : \mathcal{X} \to \mathbb{R}$ *for the minimization problem* $\min_{x\in\mathcal{X}\cap\mathcal{C}} f(x)$*, with* $f : \mathcal{X} \to \mathbb{R}$ *can be*

rewritten in the following ways, for step-size $\eta \geq 0$ and $t \geq 0$

$$\begin{aligned} x_{t+1} &= \operatorname{argmin}_{x \in \mathcal{X} \cap \mathcal{C}} D_h(x, \nabla h^*(\nabla h(x_t) + \eta \nabla f(x_t))) && \textit{(PGD)} \\ &= \operatorname{argmin}_{x \in \mathcal{X} \cap \mathcal{C}} \eta \langle \nabla f(x_t), x \rangle + D_h(x, x_t) && \textit{(proximal perspective)} \end{aligned}$$

Proof.

$$\begin{aligned} x_{t+1} &= \operatorname{argmin}_{x \in \mathcal{X} \cap \mathcal{C}} D_h(x, \nabla h^*(\nabla h(x_t) + \eta \nabla f(x_t))) && \text{(generalized GD)} \\ &= \operatorname{argmin}_{x \in \mathcal{X} \cap \mathcal{C}} h(x) - \langle \nabla h(\nabla h^*(\nabla h(x_t) + \eta \nabla f(x_t))), x \rangle \\ &= \operatorname{argmin}_{x \in \mathcal{X} \cap \mathcal{C}} h(x) - \langle \nabla h(x_t) + \eta \nabla f(x_t), x \rangle \\ &= \operatorname{argmin}_{x \in \mathcal{X} \cap \mathcal{C}} \eta \langle \nabla f(x_t), x \rangle + D_h(x, x_t) && \text{(proximal perspective)} \end{aligned}$$

□

1.1.2 Helpful Lemmas for Policy Mirror Descent

Key to the analysis of Xiao [2022] and Johnson et al. [2023] is the Three-Point Descent Lemma, that relates the improvement of the proximal gradient update compared to an arbitrary point. It originally comes from Chen and Teboulle [1993] (Lemma 3.2).

Lemma 4. ***(Three-Point Descent Lemma, Lemma 6 in Xiao [2022]).*** *Suppose that $\mathcal{X} \subset \mathbb{R}^n$ is a closed convex set, $\psi : \mathcal{X} \to \mathbb{R}$ is a proper, closed convex function, $D_h(\cdot, \cdot)$ is the Bregman divergence generated by a function h of Legendre type and $\operatorname{rint} \operatorname{dom} h \cap \mathcal{X} \neq \emptyset$. For any $x_t \in \operatorname{rint} \operatorname{dom} h$, let*

$$x_{t+1} = \operatorname{argmin}_{x \in \mathcal{X}} \psi(x) + D_h(x, x_t)$$

Then $x_{t+1} \in \operatorname{rint} \operatorname{dom} h \cap \mathcal{X}$ and $\forall x \in \mathcal{X}$,

$$\psi(x_{t+1}) + D_h(x_{t+1}, x_t) \leq \psi(x) + D_h(x, x_t) - D_h(x, x_{t+1})$$

The PMD update is an instance of the proximal minimisation with $\mathcal{X} = \Delta(\mathcal{A}), x_t = \pi_t(s)$ *and* $\psi(x) = -\eta_{\pi_{t+1}}\langle Q_t(s), x\rangle$*. Plugging these in, the Three-Point Descent Lemma relates the decrease in the proximal objective of* $\pi_{t+1}(s)$ *to any other policy, i.e.* $\forall \pi(s) \in \Delta(\mathcal{A})$*,*

$$-\eta_{\pi_{t+1}}\langle Q_t(s), \pi_{t+1}(s) - \pi(s)\rangle \leq D_h(\pi(s), \pi_t(s)) - D_h(\pi_{t+1}(s), \pi_t(s)) - D_h(\pi(s), \pi_{t+1}(s))$$

This equation is key to the analysis of convergence of exact PMD, leading to Lemma 6 regarding the monotonic improvement in Q-functions of PMD iterates.

Lemma 5. ***(Descent Property of PMD for Q-functions, Lemma 7 in Xiao [2022]*** *Consider the policies produced by the iterative updates of exact PMD. For any* $t \geq 0$

$$\langle Q_t(s), \pi_{t+1}(s) - \pi_t(s)\rangle \geq 0, \quad \forall s \in \mathcal{S},$$

Proof. From the Three-Point Descent Lemma 4 of Xiao [2022] with $\pi = \pi_t$,

$$\eta_{\pi_{t+1}}\langle Q_t(s), \pi_{t+1}(s) - \pi_t(s)\rangle \geq D_h(\pi_t(s), \pi_{t+1}(s)) + D_h(\pi_{t+1}(s), \pi_t(s))$$

since the Bregman divergences are non-negative and $\eta_{\pi_{t+1}} > 0$,

$$\langle Q_t(s), \pi_{t+1}(s) - \pi_t(s)\rangle \geq 0$$

□

Lemma 6. ***(Descent Property of PMD for Value Functions, Lemma A.2. from Johnson et al. [2023])*** *Consider the policies produced by the iterative updates of exact PMD. Then for any* $t \geq 0$*,*

$$Q_{t+1}(s) \geq Q_t(s), \quad \forall s \in \mathcal{S}$$

$$V^{\rho}_{t+1} \geq V^{\rho}_t, \quad \forall \rho \in \Delta(\mathcal{S})$$

Proof. Follows from Lemma 5 by an application of the Performance Difference Lemma 1, for an initial state distribution ρ

$$V_{t+1}^{\rho} - V_t^{\rho} = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_{t+1}^{\rho}}\left[\langle Q_t(s), \pi_{t+1}(s) - \pi_t(s)\rangle\right] \geq 0$$

□

1.1.3 Derivation of the Suboptimality Decomposition and Convergence of PMD

Suboptimality decomposition Fix a state s. For any $\pi(s), \tilde{\pi}(s)$, let $D_{-V}(\pi(s), \tilde{\pi}(s))$ be analogous to a standard Bregman divergence with mirror map $-V$, capturing the curvature of $-V$ at $\pi(s)$

$$\begin{aligned}
D_{-V}(\pi(s), \tilde{\pi}(s)) &\doteq -V_{\pi}(s) - (-V_{\tilde{\pi}}(s)) - \langle -Q_{\tilde{\pi}}(s), \pi(s) - \tilde{\pi}(s)\rangle \\
&\doteq -V_{\pi}(s) + V_{\tilde{\pi}}(s) + \langle Q_{\tilde{\pi}}(s)(s), \pi(s) - \tilde{\pi}(s)\rangle \\
&= -\langle Q_{\pi}(s) - Q_{\tilde{\pi}}(s), \pi(s)\rangle && \text{(C.1)} \\
&\text{(using Holder's inequality)} && \text{(C.2)} \\
&\geq -\|Q_{\pi}(s) - Q_{\tilde{\pi}}(s)\|_{\infty}\|\pi(s)\|_1 \\
&\geq -\gamma\|V_{\pi} - V_{\tilde{\pi}}\|_{\infty} && \text{(C.3)}
\end{aligned}$$

For the general case, using the approximation $\widehat{Q}_t(s) \approx Q_t(s)$, the per-iteration suboptimality is

$$\begin{aligned}
V^*(s) - V_t(s) = &-\langle \widehat{Q}_t(s), \pi_{t+1}(s) - \pi^*(s)\rangle - \langle \widehat{Q}_t(s), \pi_t(s) - \pi_{t+1}(s)\rangle - D_{-V}(\pi^*(s), \pi_t(s)) \\
&- \langle Q_t(s) - \widehat{Q}_t(s), \pi_t(s) - \pi^*(s)\rangle && \text{(C.4)}
\end{aligned}$$

The first term, $-\langle \widehat{Q}_t(s), \pi_{t+1}(s) - \pi^*(s)\rangle$, is the forward regret (cf. Joulani et al. [2020a]), defined as the regret of a "cheating" algorithm that uses the π_{t+1} at time t, and depends only on the choices of the algorithm and the feedback it receives. This quantity can be can

be upper-bounded using an idealized lookahead policy, $\bar{\pi}_{t+1}(s)$—greedy with respect to $\widehat{Q}_t(s)$ (cf. Johnson et al. [2023]).

If $\pi_{t+1}(s)$ is the result of a PMD update, then Johnson et al. [2023] show that using the Three-Point Descent Lemma (Lemma 6, Xiao [2022], included in Appendix 1.1, Lemma 4), denoting the step sizes $\eta_{\pi_{t+1}} \geq 0$, the forward regret is further upper-bounded by

$$\begin{aligned}
-\langle \widehat{Q}_t(s), \pi_{t+1}(s) - \pi^*(s) \rangle &\leq -\langle \widehat{Q}_t(s), \pi_{t+1}(s) - \bar{\pi}_{t+1}(s) \rangle \\
&\leq \langle \widehat{Q}_t(s), \bar{\pi}_{t+1} - \pi_{t+1} \rangle \\
&\leq \eta_{\pi_{t+1}}^{-1} D_h(\bar{\pi}_{t+1}(s), \pi_t(s)) - \eta_{\pi_{t+1}}^{-1} D_h(\bar{\pi}_{t+1}(s), \pi_{t+1}(s)) \\
&\quad - \eta_{\pi_{t+1}}^{-1} D_h(\pi_{t+1}(s), \pi_t(s)) \\
&\leq \eta_{\pi_{t+1}}^{-1} D_h(\bar{\pi}_{t+1}(s), \pi_t(s)) \qquad \text{(C.5)}
\end{aligned}$$

The second term in Eq. C.4 is

$$\begin{aligned}
-\langle \widehat{Q}_t(s), \pi_t(s) - \pi_{t+1}(s) \rangle = V^{t+1}(s) - V_t(s) + D_{-V}(\pi_{t+1}(s), \pi_t(s)) \\
+\langle Q_t(s) - \widehat{Q}_t(s), \pi_t(s) - \pi_{t+1}(s) \rangle \qquad \text{(C.6)}
\end{aligned}$$

The third term $-D_{-V}(\pi^*(s), \pi_t(s))$ can be bounded by applying the upper-approximation from Eq.C.3, resulting in

$$-D_{-V}(\pi^*(s), \pi_t(s)) \leq \gamma \|V^* - V_t\|_\infty \qquad \text{(C.7)}$$

Plugging Eq. C.5, C.6, C.7 back in the suboptimality decomposition from Eq. C.4, we obtain

$$V^*(s) - V_{t+1}(s) \leq \gamma \|V^* - V_t\|_\infty + \underbrace{\eta_{\pi_{t+1}}^{-1} D_h(\bar{\pi}_{t+1}(s), \pi_t(s)) + \langle Q_t(s) - \widehat{Q}_t(s), \pi^*(s) - \pi_{t+1}(s) \rangle + D_{-V}(\pi_{t+1}(s), \pi_t(s))}_{\xi_{\pi_{t+1}} \text{(iteration error)}} \tag{C.8}$$

With $\xi_{\pi_{t+1}}$—the iteration error, recursing Eq. C.8

$$\|V^* - V_t\|_\infty \leq \gamma^t \sum_{i \leq t} \gamma^t \|V^* - V_0\|_\infty + \xi_i / \gamma^i$$

Convergence of Exact PMD at γ-rate If the PMD update is exact, then $\widehat{Q}_t(s) = Q_t(s), \forall s \in \mathcal{S}$. The Three-Point Descent Lemma 4 guarantees policy improvement for an Exact PMD update, and yields Lemma 6 stating $V_{t+1}(s) \geq V_t(s)$, and $\langle Q_{t+1}(s) - Q_t(s), \pi_{t+1}(s) \rangle \geq 0$. Consequently

$$D_{-V}(\pi_{t+1}(s), \pi_t(s)) = -\langle Q_{t+1}(s) - Q_t(s), \pi_{t+1}(s) \rangle \leq 0$$

There remains only one term in the suboptimality from Eq. C.4, namely

$$\xi_{\pi_{t+1}} \leq \eta_{\pi_{t+1}}^{-1} D_h(\bar{\pi}_{t+1}(s), \pi_t(s))$$

An optimal step-size $\eta_{\pi_{t+1}}$ can be derived by upper-bounding it $\eta_{\pi_{t+1}}^{-1} D_h(\bar{\pi}_{t+1}(s), \pi_t(s)) \leq \epsilon_{\pi_{t+1}}$, for any arbitrary constant $\epsilon_{\pi_{t+1}}$. Setting $\epsilon_{\pi_{t+1}} = \gamma^{2(t+1)} \epsilon_0$ for some $\epsilon_0 > 0$, gives the optimal step-size with a geometrically increasing component, which guarantees linear convergence at the γ-rate

$$\|V^* - V_t\|_\infty \leq \gamma^t \left(\|V^* - V_0\|_\infty + \epsilon_0 / 1 - \gamma \right)$$

matching the bounds of PI and VI as ϵ_0 goes to 0 (cf. Theorem 4.1., Johnson et al. [2023]).

1.2 Proofs for Sec. 2: Functional Acceleration for PMD

Definition 7. (Functional gradient of the Bregman divergence) Fix a state s. For any policies π_1, π_0, we denote the gradient of the Bregman divergence with respect to the first argument

$$\nabla D_h(\pi_1(s), \pi_0(s)) \doteq \nabla h(\pi_1(s)) - \nabla h(\pi_0(s))$$

The following lemma can be also interpreted as a definition for the difference of differences of Bregman divergences.

Lemma 8. ***(Four-Point Identity Lemma of Bregman divergences)*** *For any four policies $\pi_3, \pi_2, \pi_1, \pi_0$, we have*

$$\begin{aligned}\langle \nabla D_h(\pi_1(s), \pi_0(s)), \pi_3(s) - \pi_2(s) \rangle = D_h(\pi_3(s), \pi_0(s)) - D_h(\pi_3(s), \pi_1(s)) \\ -[D_h(\pi_2(s), \pi_0(s)) - D_h(\pi_2(s), \pi_1(s))]\end{aligned}$$

Proof. Immediate from the definition. □

An immediate consequence is the Three-Point Identity Lemma of Bregman divergences (cf. Bubeck [2015], Eq. 4.1, Beck and Teboulle [2003], Lemma 4.1).

Lemma 9. ***(Three-Point Identity Lemma of Bregman divergences)*** *For any three policies π_2, π_1, π_0,*

$$\langle \nabla D_h(\pi_1(s), \pi_0(s)), \pi_2(s) - \pi_1(s) \rangle = D_h(\pi_2(s), \pi_0(s)) - D_h(\pi_2(s), \pi_1(s)) - D_h(\pi_1(s), \pi_0(s))$$

Proof. Apply Lemma 8 with $\pi_3 = \pi_2$, $\pi_2 = \pi_1$. □

Corollary 10. ***(Three-Point Descent Corollary for PMD)*** *Consider the policies produced by the iterative updates of exact PMDin Eq.C.1. For any policy π, timestep $t \geq 0$, and state s*

$$\langle \eta_{\pi_{t+1}} Q_t(s), \pi(s) - \pi_{t+1}(s) \rangle \leq \langle \nabla h(\pi_{t+1}(s)) - \nabla h(\pi_t(s)), \pi(s) - \pi_{t+1}(s) \rangle$$

Proof. From the Three-Point descent Lemma 4 (Eq.C.1)

$$-\eta_{\pi_{t+1}} \langle Q_t(s), \pi_{t+1}(s) - \pi(s) \rangle \leq D_h(\pi(s), \pi_t(s)) - D_h(\pi(s), \pi_{t+1}(s)) - D_h(\pi_{t+1}(s), \pi_t(s))$$

apply Lemma 9 on the right hand side, with $\pi_0 = \pi_t, \pi_1 = \pi_{t+1}, \pi_2 = \pi$, which yields the claimed inequality. □

Lemma 11. ***(Extrapolation from the future: Equivalence of updates)*** *For any state s and timestep $t > 0$, the update of PMD(+ext), $\pi_{t+1} = \operatorname{argmin}_{\pi(s) \in \Delta(\mathcal{A})} -\langle (\mathcal{T}_{\mu_t} Q_t)(s) \pi(s) \rangle + \eta_{\pi_{t+1}}^{-1} D_h(\pi(s), \pi_t(s))$, can be rewritten as a correction to the update $\mu_t = \operatorname{argmin}_{\pi(s) \in \Delta(\mathcal{A})} -\langle Q_t(s), \pi(s) \rangle + \eta_{\mu_t}^{-1} D_h(\pi(s), \pi_t(s))$*

$$\pi_{t+1}(s) \doteq \operatorname{argmin}_{\pi(s) \in \Delta(\mathcal{A})} -\langle (\mathcal{T}_{\mu_t} Q_t)(s) - \eta_{\pi_{t+1}}^{-1} \eta_{\mu_t} Q_t(s), \pi(s) \rangle + \eta_{\pi_{t+1}}^{-1} D_h(\pi(s), \mu_t(s))$$

Proof. For any timestep $t > 0$, and state s, from the definition of PMD(+ext)

$$\mu_t(s) = \operatorname{argmin}_{\pi(s) \in \Delta(\mathcal{A})} -\langle Q_t(s), \pi(s) \rangle + \eta_{\mu_t}^{-1} D_h(\pi(s), \pi_t(s))$$

Applying Corollary 10 for $\pi_{t+1} = \mu_t$, and for any policy π, we have

$$\langle \eta_{\mu_t} Q_t(s) - \nabla h(\mu_t)(s)), \pi(s) - \mu_t(s) \rangle \leq -\langle \nabla h(\pi_t(s)), \pi(s) - \mu_t(s) \rangle \tag{C.9}$$

Plugging Eq. C.9 in the proposed update, we recover the formulation of PMD(+ext)

$$
\begin{aligned}
\pi_{t+1}(s) &\doteq \operatorname{argmin}_{\pi(s)\in\Delta(\mathcal{A})} -\langle(\mathcal{T}_{\mu_t}Q_t)(s) - \eta_{\pi_{t+1}}^{-1}\eta_{\mu_t}Q_t(s), \pi(s)\rangle + \eta_{\pi_{t+1}}^{-1}D_h(\pi(s), \mu_t(s)) \\
&= \operatorname{argmin}_{\pi(s)\in\Delta(\mathcal{A})} -\langle(\mathcal{T}_{\mu_t}Q_t)(s) - \eta_{\pi_{t+1}}^{-1}\eta_{\mu_t}Q_t(s), \pi(s)\rangle \\
&\qquad + \eta_{\pi_{t+1}}^{-1}(h(\pi(s)) - h(\mu_t(s)) - \langle\nabla h(\mu_t(s)), \pi(s) - \mu_t(s)\rangle) \\
&= \operatorname{argmin}_{\pi(s)\in\Delta(\mathcal{A})} -\langle(\mathcal{T}_{\mu_t}Q_t)(s), \pi(s)\rangle + \eta_{\pi_{t+1}}^{-1}(h(\pi(s)) - h(\mu_t(s))) \\
&\qquad + \eta_{\pi_{t+1}}^{-1}\eta_{\mu_t}\langle Q_t(s), \pi(s)\rangle - \langle\nabla h(\mu_t(s)), \pi(s) - \mu_t(s)\rangle) \\
&\text{(adding the constant term } -\eta_{\pi_{t+1}}^{-1}\eta_{\mu_t}\langle Q_t(s), \mu_t(s)\rangle\text{)} \\
&= \operatorname{argmin}_{\pi(s)\in\Delta(\mathcal{A})} -\langle(\mathcal{T}_{\mu_t}Q_t)(s), \pi(s)\rangle\eta_{\pi_{t+1}}^{-1}(h(\pi(s)) - h(\mu_t(s))) \\
&\qquad + \eta_{\pi_{t+1}}^{-1}\eta_{\mu_t}\langle Q_t(s) - \nabla h(\mu_t(s)), \pi(s) - \mu_t(s)\rangle \\
&\text{(plugging in Eq. C.9)} \\
&= \operatorname{argmin}_{\pi(s)\in\Delta(\mathcal{A})} -\langle(\mathcal{T}_{\mu_t}Q_t)(s), \pi(s)\rangle + \eta_{\pi_{t+1}}^{-1}(h(\pi(s)) - h(\mu_t(s)) - \langle\nabla h(\pi_t(s)), \pi(s) - \mu_t(s)\rangle) \\
&\text{(swapping constant terms } h(\mu_t(s)) \text{ with } h(\pi_t(s)) \text{ and } \langle\nabla h(\pi_t(s)), \mu_t(s)\rangle \text{ with } \langle\nabla h(\pi_t(s)), \pi_t(s)\rangle\text{)} \\
&= \operatorname{argmin}_{\pi(s)\in\Delta(\mathcal{A})} -\langle(\mathcal{T}_{\mu_t}Q_t)(s)\pi(s)\rangle + \eta_{\pi_{t+1}}^{-1}(h(\pi(s)) - h(\pi_t(s)) - \langle\nabla h(\pi_t(s)), \pi(s) - \pi_t(s)\rangle) \\
&= \operatorname{argmin}_{\pi(s)\in\Delta(\mathcal{A})} -\langle(\mathcal{T}_{\mu_t}Q_t)(s)\pi(s)\rangle + \eta_{\pi_{t+1}}^{-1}D_h(\pi(s), \pi_t(s))
\end{aligned}
$$

□

1.2.1 Extrapolation from the past: derivation of the lazy momentum-based PMD algorithm

We take the following steps to arrive at the proposed algorithm.

First, we rely on the equivalence in Lemma 11 to rewrite the update to π_{t+1} in PMD(+ext)

$$
\pi_{t+1}(s) = \operatorname{argmin}_{\pi(s)\in\Delta(\mathcal{A})} -\langle(\mathcal{T}_{\mu_t}Q_t)(s) - \eta_{\pi_{t+1}}^{-1}\eta_{\mu_t}Q_t(s), \pi(s)\rangle + \eta_{\pi_{t+1}}^{-1}D_h(\pi(s), \mu_t(s)) \quad \text{(C.10)}
$$

Next, to remedy the need for two evaluation procedures, we recycle evaluation updates from previous steps. Specifically, we replace Q_t with the closest past evaluation, i.e.

$\mathcal{T}_{\mu_{t-1}}Q_{t-1}$, twice, in Eq. C.10 (the equivalent update to π_{t+1} from PMD(+ext)) and in the update to μ_t of PMD(+ext). This leads to

$$\mu_t(s) \doteq \operatorname{argmin}_{\pi(s)\in\Delta(\mathcal{A})} -\langle (\mathcal{T}_{\mu_{t-1}}Q_{t-1})(s), \pi(s)\rangle + \eta^{-1}_{\pi_{t+1}} D_h(\pi(s), \pi_t(s)) \tag{C.11}$$

$$\pi_{t+1}(s) = \operatorname{argmin}_{\pi(s)\in\Delta(\mathcal{A})} -\langle (\mathcal{T}_{\mu_t}Q_t)(s) - \eta^{-1}_{\pi_{t+1}}\eta_{\mu_t}(\mathcal{T}_{\mu_{t-1}}Q_{t-1})(s), \pi(s)\rangle + \eta^{-1}_{\mu_t} D_h(\pi(s), \mu_t(s)) \tag{C.12}$$

We now apply the equivalence shown in Lemma 12 (which is analogous to Lemma 11, but time-reversed), yielding an update written in terms of one policy variable μ and $\mathcal{T}_{\mu_t}Q_t$

$$\mu_{t+1}(s) = \operatorname{argmin}_{\pi(s)\in\Delta(\mathcal{A})} -\langle (\mathcal{T}_{\mu_t}Q_t)(s) + \eta^{-1}_{\mu_{t+1}}\eta_{\pi_{t+1}}((\mathcal{T}_{\mu_t}Q_t)(s) - (\mathcal{T}_{\mu_{t-1}}Q_{t-1})(s)), \pi(s)\rangle + \eta^{-1}_{\mu_{t+1}} D_h(\pi(s), \mu_t(s))$$

For consistency with PMD(+loo) and PMD(+ext), we henceforth replace these variables (μ and $\mathcal{T}_\mu Q$) with π and Q. In this case, $Q = Q_\pi$, and instead of $\eta^{-1}_{\mu_{t+1}}\eta_{\pi_{t+1}}$, we then use two consecutive step-sizes $\eta^{-1}_{\pi_{t+1}}\eta_{\pi_t}$, yielding

$$\pi_{t+1}(s) = \operatorname{argmin}_{\pi(s)\in\Delta(\mathcal{A})} -\langle Q_t(s) + \eta^{-1}_{\pi_{t+1}}\eta_{\pi_t}(Q_t(s) - Q_{t-1}(s)), \pi(s)\rangle + \eta^{-1}_{\pi_{t+1}} D_h(\pi(s), \pi_t(s)) \tag{C.13}$$

Lemma 12. ***(Extrapolation from the past: equivalence of updates)*** *For any state s and timestep $t > 0$, the updates*

$$\mu_t(s) \doteq \operatorname{argmin}_{\pi(s)\in\Delta(\mathcal{A})} -\langle (\mathcal{T}_{\mu_{t-1}}Q_{t-1})(s), \pi(s)\rangle + \eta^{-1}_{\mu_t} D_h(\pi(s), \pi_t(s)) \tag{C.14}$$

$$\pi_{t+1}(s) = \operatorname{argmin}_{\pi(s)\in\Delta(\mathcal{A})} -\langle (\mathcal{T}_{\mu_t}Q_t)(s) - \eta^{-1}_{\pi_{t+1}}\eta_{\mu_t}(\mathcal{T}_{\mu_{t-1}}Q_{t-1})(s), \pi(s)\rangle + \eta^{-1}_{\pi_{t+1}} D_h(\pi(s), \mu_t(s)) \tag{C.15}$$

can be rewritten as:

$$\mu_{t+1}(s) = \operatorname{argmin}_{\pi(s)\in\Delta(\mathcal{A})} -\langle (\mathcal{T}_{\mu_t}Q_t)(s) + \eta^{-1}_{\mu_{t+1}}\eta_{\pi_{t+1}}\delta_t(s), \pi(s)\rangle + \eta^{-1}_{\mu_{t+1}} D_h(\pi(s), h(\mu_t(s)))$$

Proof. First, we increment the index t in equation Eq. C.14

$$\mu_{t+1}(s) \doteq \text{argmin}_{\pi(s)\in\Delta(\mathcal{A})} -\langle (\mathcal{T}_{\mu_t} Q_t)(s), \pi(s)\rangle + \eta^{-1}_{\mu_{t+1}} D_h(\pi(s), \pi_{t+1}(s)) \tag{C.16}$$

Then, we apply Corollary 10 to π_{t+1}, and for any policy π, with the shorthand notation $\delta_t(s) \doteq \mathcal{T}_{\mu_t} Q_t)(s) - \eta^{-1}_{\pi_{t+1}} \eta_{\mu_t} (\mathcal{T}_{\mu_{t-1}} Q_{t-1})(s)$, this yields

$$\langle \eta_{\pi_{t+1}} \delta_t(s) - \nabla h(\pi_{t+1})(s)), \pi(s) - \pi_{t+1}(s)\rangle \leq -\langle \nabla h(\mu_t(s)), \pi(s) - \pi_{t+1}(s)\rangle \tag{C.17}$$

Plugging Eq. C.17 in back in Eq. C.16

$$\begin{aligned}
\mu_{t+1}(s) &= \text{argmin}_{\pi(s)\in\Delta(\mathcal{A})} -\langle (\mathcal{T}_{\mu_t} Q_t)(s), \pi(s)\rangle \\
&\qquad + \eta^{-1}_{\mu_{t+1}} (h(\pi(s)) - h(\pi_{t+1}(s)) - \langle \nabla h(\pi_{t+1}(s)), \pi(s) - \pi_{t+1}(s)\rangle \\
&= \text{argmin}_{\pi(s)\in\Delta(\mathcal{A})} -\langle (\mathcal{T}_{\mu_t} Q_t)(s) + \eta^{-1}_{\mu_{t+1}} \eta_{\pi_{t+1}} \delta_t(s), \pi(s)\rangle + \eta^{-1}_{\mu_{t+1}} (h(\pi(s)) - h(\pi_{t+1}(s))) \\
&\qquad + \eta^{-1}_{\mu_{t+1}} \eta_{\pi_{t+1}} \langle \delta_t(s), \pi(s)\rangle - \eta^{-1}_{\mu_{t+1}} \langle \nabla h(\pi_{t+1}(s)), \pi(s) - \pi_{t+1}(s)\rangle \\
&\text{(adding the constant term } -\eta^{-1}_{\mu_{t+1}} \eta_{\pi_{t+1}} \langle \delta_t(s), \pi_{t+1}(s)\rangle) \\
&= \text{argmin}_{\pi(s)\in\Delta(\mathcal{A})} -\langle (\mathcal{T}_{\mu_t} Q_t)(s) + \eta^{-1}_{\mu_{t+1}} \eta_{\pi_{t+1}} \delta_t(s), \pi(s)\rangle + \eta^{-1}_{\mu_{t+1}} (h(\pi(s)) - h(\pi_{t+1}(s))) \\
&\qquad + \eta^{-1}_{\mu_{t+1}} \eta_{\pi_{t+1}} \langle \delta_t(s) - \eta^{-1}_{\mu_{t+1}} \nabla h(\pi_{t+1}(s)), \pi(s) - \pi_{t+1}(s)\rangle \\
&\text{(applying Eq.C.17)} \\
&= \text{argmin}_{\pi(s)\in\Delta(\mathcal{A})} -\langle (\mathcal{T}_{\mu_t} Q_t)(s) + \eta^{-1}_{\mu_{t+1}} \eta_{\pi_{t+1}} \delta_t(s), \pi(s)\rangle + \eta^{-1}_{\mu_{t+1}} (h(\pi(s)) - h(\pi_{t+1}(s))) \\
&\qquad - \eta^{-1}_{\mu_{t+1}} \langle \nabla h(\mu_t(s)), \pi(s) - \pi_{t+1}(s)\rangle \\
&\text{(swapping constant terms)} \\
&= \text{argmin}_{\pi(s)\in\Delta(\mathcal{A})} -\langle (\mathcal{T}_{\mu_t} Q_t)(s) + \eta^{-1}_{\mu_{t+1}} \eta_{\pi_{t+1}} \delta_t(s), \pi(s)\rangle + \eta^{-1}_{\mu_{t+1}} (h(\pi(s)) - h(\mu_t(s))) \\
&\qquad - \eta^{-1}_{\mu_{t+1}} \langle \nabla h(\mu_t(s)), \pi(s) - \mu_t(s)\rangle \\
&= \text{argmin}_{\pi(s)\in\Delta(\mathcal{A})} -\langle (\mathcal{T}_{\mu_t} Q_t)(s) + \eta^{-1}_{\mu_{t+1}} \eta_{\pi_{t+1}} \delta_t(s), \pi(s)\rangle + \eta^{-1}_{\mu_{t+1}} D_h(\pi(s), h(\mu_t(s)))
\end{aligned}$$

□

Lemma 13. ***(Descent Property of exact PMD(+loo))*** *Consider the policies produced by the iterative updates of PMD(+loo)*

$$\pi_{t+1}(s) = \text{argmin}_{\pi(s)\in\Delta(\mathcal{A})}\langle \mathcal{T}_{\mu_t} Q_t(s), \pi(s)\rangle + \eta_{\pi_{t+1}}^{-1} D_h(\pi(s), \pi_t(s))$$

where $(\mathcal{T}_{\mu_t} Q_t)(s) \doteq \mathbb{E}[r(s) + \gamma\langle Q_t(s'), \mu_t(s')\rangle]$ *is the lookahead,* $\mu_t(s)$ *is an intermediary policy greedy with respect to* $Q_t(s)$*, and* $\eta_{\pi_{t+1}} \geq 0$ *is a step-size. Then, for any timestep* $t \geq 0$

$$\langle (\mathcal{T}_{\mu_t} Q_t)(s), \pi_{t+1}(s) - \mu_t(s)\rangle \geq 0, \forall s \in \mathcal{S} \tag{C.18}$$

$$\langle Q_{t+1}(s) - (\mathcal{T}_{\mu_t} Q_t)(s), \pi_{t+1}(s)\rangle \geq 0, \forall s \in \mathcal{S} \tag{C.19}$$

Proof. Consider first the descent property of μ_t

$$\begin{aligned}\langle Q_t(s) - (\mathcal{T}_{\mu_t} Q_t)(s), \mu_t(s)\rangle &= \sum_{a\in\mathcal{A}} (Q_t(s,a) - (\mathcal{T}_{\mu_t} Q_t)(s,a))\mu_t(a|s) \\ &= \gamma \sum_{s'\in\mathcal{S}} \sum_{a\in\mathcal{A}} P(s'|s,a)\mu_t(a|s)\left[\langle Q_t(s'), \pi_t(s')\rangle - \langle Q_t(s'), \mu_t(s')\rangle\right] \leq 0\end{aligned} \tag{C.20}$$

where the last inequality follows from the definition of $\mu_t(s)$ as greedy with respect to $Q_t(s)$, which implies $\langle Q_t(s'), \mu_t(s')\rangle \geq \langle Q_t(s'), \pi_t(s')\rangle$.

Then, for the descent property of $\pi_{t+1}(s)$, we have

$$
\begin{aligned}
&\langle Q_{t+1}(s) - (\mathcal{T}_{\mu_t} Q_t)(s), \pi_{t+1}(s)\rangle \qquad \text{(C.21)}\\
&= \sum_{a\in\mathcal{A}} (Q_{t+1}(s,a) - (\mathcal{T}_{\mu_t} Q_t)(s,a))\pi_{t+1}(a|s)\\
&= \gamma \sum_{s'\in\mathcal{S}} \sum_{a\in\mathcal{A}} P(s'|s,a)\pi_{t+1}(a|s)\left[\langle Q_{t+1}(s'), \pi_{t+1}(s')\rangle - \langle Q_t(s'), \mu_t(s')\rangle\right]\\
&= \gamma \sum_{s'\in\mathcal{S}} \sum_{a\in\mathcal{A}} P(s'|s,a)\pi_{t+1}(a|s)\left[\langle Q_{t+1}(s'), \pi_{t+1}(s')\rangle - \langle (\mathcal{T}_{\mu_t} Q_t)(s'), \mu_t(s')\rangle - \langle Q_t(s') - (\mathcal{T}_{\mu_t} Q_t)(s'), \mu_t(s')\rangle\right]\\
&\overset{(Eq.\ C.20)}{\geq} \gamma \sum_{s'\in\mathcal{S}} \sum_{a\in\mathcal{A}} P(s'|s,a)\pi_{t+1}(a|s)\left[\langle Q_{t+1}(s'), \pi_{t+1}(s')\rangle - \langle (\mathcal{T}_{\mu_t} Q_t)(s'), \mu_t(s')\rangle\right]\\
&= \gamma \sum_{s'} \sum_{a} P(s'|s,a)\pi_{t+1}(a|s)\left[\langle (\mathcal{T}_{\mu_t} Q_t)(s'), \pi_{t+1}(s') - \mu_t(s')\rangle + \langle Q_{t+1}(s') - (\mathcal{T}_{\mu_t} Q_t)(s'), \pi_{t+1}(s')\rangle\right]
\end{aligned}
$$

Recursing, yields

$$
\langle Q_{t+1}(s) - (\mathcal{T}_{\mu_t} Q_t)(s), \pi_{t+1}(s)\rangle = \gamma/_{1-\gamma} \sum_{s'} d_{t+1}(s' \to s')\langle (\mathcal{T}_{\mu_t} Q_t)(s'), \pi_{t+1}(s') - \mu_t(s')\rangle \qquad \text{(C.22)}
$$

From Lemma 4 with $\pi = \mu_t$ and $\eta_{\pi_{t+1}} \geq 0$

$$
\langle (\mathcal{T}_{\mu_t} Q_t)_s, \pi_{t+1}(s) - \mu_t(s)\rangle \geq \eta^{-1}_{\pi_{t+1}} \left(D_h(\mu_t(s), \pi_{t+1}(s)) + D_h(\pi_{t+1}(s), \mu_t(s))\right) \geq 0 \qquad \text{(C.23)}
$$

This proves the first claim in Eq. C.18. Plugging Eq. C.23 back in Eq. C.22 yields the second claim in Eq. C.19, $\langle Q_{t+1}(s) - (\mathcal{T}_{\mu_t} Q_t)(s), \pi_{t+1}(s)\rangle \geq 0$ □

Theorem 14. ***(Functional acceleration with PMD(+loo))*** *Consider the policies produced by the iterative updates of PMD(+loo)*

$$
\pi_{t+1}(s) = \text{argmin}_{\pi(s)\in\Delta(\mathcal{A})} -\langle (\mathcal{T}_{\mu_t} Q_t)(s), \pi(s)\rangle + \eta^{-1}_{\pi_{t+1}} D_h(\pi(s), \pi_t(s)) \qquad \text{(C.24)}
$$

where $(\mathcal{T}_{\mu_t} Q_t)(s) \doteq \mathbb{E}[r(s) + \gamma \langle Q_t(s'), \mu_t(s') \rangle]$, $\mu_t(s)$ *is greedy with respect to* $Q_t(s)$, $\eta_{\pi_{t+1}} \geq 0$ *are adaptive step sizes, such that* $\forall \epsilon_{\pi_{t+1}}$ *arbitrarily small,* $\eta_{\pi_{t+1}}^{-1} D_h(\text{greedy}((\mathcal{T}_{\mu_t} Q_t)(s)), \pi_t(s)) \leq \epsilon_{\pi_{t+1}}$. *Then,*

$$V^*(s) - V_{t+1}(s) \leq \gamma^2 \|V^* - V_t\|_\infty + \epsilon_{\pi_{t+1}} \tag{C.25}$$

and recursing yields

$$\|V^* - V_t\|_\infty \leq (\gamma^2)^t (\|V^* - V_0\|_\infty + \sum_{i \leq t} \epsilon_{\pi_{i+1}} / (\gamma^2)^i) \tag{C.26}$$

Proof. If π_{t+1} is the result of a PMD update which uses $(\mathcal{T}_{\mu_t} Q_t)(s)$, and step-sizes $\eta_{\pi_{t+1}}$, then applying Lemma 4 for $\pi(s) = \bar{\pi}_{t+1}(s)$ greedy with respect to $(\mathcal{T}_{\mu_t} Q_t)(s)$

$$\begin{aligned} -\langle (\mathcal{T}_{\mu_t} Q_t)(s), \pi_{t+1}(s) - \bar{\pi}_{t+1}(s) \rangle &\leq \eta_{\pi_{t+1}}^{-1} \big(D_h(\bar{\pi}_{t+1}(s), \pi_t(s)) - D_h(\pi_{t+1}(s), \pi_t(s)) \\ &\quad - D_h(\pi_{t+1}(s), \bar{\pi}_{t+1}(s)) \big) \\ &\leq \eta_{\pi_{t+1}}^{-1} D_h(\bar{\pi}_{t+1}(s), \pi_t(s)) \end{aligned} \tag{C.27}$$

Further, the suboptimality is

$$\begin{aligned} V^*(s) - V_{t+1}(s) = &-\langle (\mathcal{T}_{\mu_t} Q_t)(s), \pi_{t+1}(s) - \pi^*(s) \rangle - \langle Q_{t+1}(s) - (\mathcal{T}_{\mu_t} Q_t)(s), \pi_{t+1}(s) \rangle \\ &+ \langle Q^*(s) - (\mathcal{T}_{\mu_t} Q_t)(s), \pi^*(s) \rangle \end{aligned} \tag{C.28}$$

Since $\langle (\mathcal{T}_{\mu_t} Q_t)(s), \pi^*(s) \rangle \leq \langle (\mathcal{T}_{\mu_t} Q_t)(s), \bar{\pi}_{t+1}(s) \rangle$ if $\bar{\pi}_{t+1}(s)$ is greedy with respect to $(\mathcal{T}_{\mu_t} Q_t)(s)$, then plugging Eq C.27 in Eq C.28

$$\begin{aligned} V^*(s) - V_{t+1}(s) \leq {}^1\!/\!_{\eta^t} D_h(\bar{\pi}_{t+1}(s), \pi_t(s)) &- \langle Q_{t+1}(s) - (\mathcal{T}_{\mu_t} Q_t)(s), \pi_{t+1}(s) \rangle \\ &+ \langle Q^*(s) - (\mathcal{T}_{\mu_t} Q_t)(s), \pi^*(s) \rangle \end{aligned} \tag{C.29}$$

Next, cf. Lemma 13, $\langle Q_{t+1}(s) - (\mathcal{T}_{\mu_t} Q_t)(s), \pi_{t+1}(s) \rangle \geq 0$. Plugging this back into Eq C.28

$$\begin{aligned} V^*(s) - V_{t+1}(s) &\leq \eta_{\pi_{t+1}}^{-1} D_h(\bar{\pi}_{t+1}(s), \pi_t(s)) + \langle Q^*(s) - (\mathcal{T}_{\mu_t} Q_t)(s), \pi^*(s) \rangle \\ &\leq \langle Q^*(s) - (\mathcal{T}_{\mu_t} Q_t)(s), \pi^*(s) \rangle + \epsilon_{\pi_{t+1}} \end{aligned} \tag{C.30}$$

where the last step follows from step-size adaptation condition $\eta_{\pi_{t+1}}^{-1} D_h(\bar{\pi}_{t+1}(s), \pi_t(s)) \leq \epsilon_{\pi_{t+1}}$.

Decomposing the remaining term

$$\begin{aligned} \langle Q^*(s) - (\mathcal{T}_{\mu_t} Q_t)(s), \pi^*(s) \rangle &= \sum_{a \in \mathcal{A}} (Q^*(s,a) - (\mathcal{T}_{\mu_t} Q_t)(s,a)) \pi^*(a|s) \\ &= \gamma \sum_{s' \in \mathcal{S}} \sum_{a \in \mathcal{A}} P(s'|s,a) \pi^*(a|s) \left[\langle Q^*(s'), \pi^*(s') \rangle - \langle Q_t(s'), \mu_t(s') \rangle \right] \\ &= \gamma \sum_{s' \in \mathcal{S}} \sum_{a \in \mathcal{A}} P(s'|s,a) \pi^*(a|s) \\ &\qquad \left[\langle Q^*(s') - Q_t(s'), \pi^*(s') \rangle - \langle Q_t(s'), \mu_t(s') - \pi^*(s') \rangle \right] \\ &\leq \gamma \sum_{s' \in \mathcal{S}} \sum_{a \in \mathcal{A}} P(s'|s,a) \pi^*(a|s) \left[\langle Q^*(s') - Q_t(s'), \pi^*(s') \rangle \right] \\ &= \gamma^2 \sum_{s' \in \mathcal{S}} \sum_{a \in \mathcal{A}} \pi^*(a|s) P(s'|s,a) \sum_{s'' \in \mathcal{S}} \sum_{a' \in \mathcal{A}} \pi^*(s',a') P(s''|s',a') \left[V^*(s'') - V_t(s'') \right] \end{aligned}$$

where the inequality follows due to $\mu_t(s)$ being greedy with respect to $Q_t(s)$, $\forall s \in \mathcal{S}$ by definition.

Taking the max norm and applying the triangle inequality and the contraction property

$$\| \langle Q^* - (\mathcal{T}_{\mu_t} Q_t), \pi^* \rangle \|_\infty \leq \gamma^2 \| V^* - V_t \|_\infty$$

and then plugging this back in Eq. C.30

$$\begin{aligned} V^*(s) - V_{t+1}(s) &\leq \| \langle Q^*(s) - (\mathcal{T}_{\mu_t} Q_t), \pi^* \rangle \|_\infty + \epsilon_{\pi_{t+1}} \\ &\leq \gamma^2 \| V^* - V_t \|_\infty + \epsilon_{\pi_{t+1}} \end{aligned}$$

which is the first claim in Eq. C.25. Then recursing yields the second claim in Eq. C.26. □

Theorem 15. ***(Functional acceleration with PMD(+ext))*** *Consider the policies produced by the iterative updates of PMD(+ext)*

$$\mu_t(s) = \text{argmin}_{\pi(s)\in\Delta(\mathcal{A})} -\langle Q_t(s), \pi(s)\rangle + \eta_{\mu_t}^{-1} D_h(\pi(s), \pi_t(s)) \tag{C.31}$$

$$\pi_{t+1}(s) = \text{argmin}_{\pi(s)\in\Delta(\mathcal{A})} -\langle (\mathcal{T}_{\mu_t} Q_t)(s), \pi(s)\rangle + \eta_{\pi_{t+1}}^{-1} D_h(\pi(s), \pi_t(s)) \tag{C.32}$$

where $(\mathcal{T}_{\mu_t} Q_t)(s) \doteq \mathbb{E}[r(s) + \gamma\langle Q_t(s'), \mu_t(s')\rangle]$ *is the lookahead,* $\eta_{\pi_{t+1}}, \eta_{\mu_t} \geq 0$ *are adaptive step sizes, such that* $\forall \epsilon_{\pi_{t+1}}, \epsilon_{\mu_t}$ *arbitrarily small,* $\eta_{\pi_{t+1}}^{-1} D_h(\text{greedy}((\mathcal{T}_{\mu_t} Q_t)(s)), \pi_t(s)) \leq \epsilon_{\pi_{t+1}}$ *and* $\eta_{\mu_t}^{-1} D_h(\text{greedy}(Q_t), \pi_t(s)) \leq \epsilon_{\mu_t}$. *Then*

$$V^*(s) - V_{t+1}(s) \leq \gamma^2 \|V^* - V_t\|_\infty + \gamma\epsilon_{\mu_t} + \epsilon_t \tag{C.33}$$

and recursing yields

$$\|V^*(s) - V_t(s)\|_\infty \leq (\gamma^2)^t (\|V^* - V_0\|_\infty + \sum_{i\leq t} {}^{(\epsilon_{\pi_{i+1}} + \gamma\epsilon_{\mu_i})}/_{(\gamma^2)^i}) \tag{C.34}$$

Proof. If π_{t+1} is the result of a PMD update with $(\mathcal{T}_{\mu_t} Q_t)(s)$, applying Lemma 4 for $\bar{\pi}_{t+1}(s)$ greedy with respect to $(\mathcal{T}_{\mu_t} Q_t)(s)$

$$\begin{aligned} -\langle (\mathcal{T}_{\mu_t} Q_t)(s), \pi_{t+1}(s) - \bar{\pi}_{t+1}(s)\rangle &\leq {}^1/_{\eta_{\pi_{t+1}}} \big(D_h(\bar{\pi}_{t+1}(s), \pi_t(s)) - D_h(\pi_{t+1}(s), \pi_t(s)) \\ &\quad - D_h(\pi_{t+1}(s), \bar{\pi}_{t+1}(s)) \big) \\ &\leq \eta_{\pi_{t+1}}^{-1} D_h(\bar{\pi}_{t+1}(s), \pi_t(s)) \end{aligned} \tag{C.35}$$

Further, the suboptimality is

$$\begin{aligned} V^*(s) - V_{t+1}(s) = &-\langle (\mathcal{T}_{\mu_t} Q_t)(s), \pi_{t+1}(s) - \pi^*(s)\rangle - \langle Q_{t+1}(s) - (\mathcal{T}_{\mu_t} Q_t)(s), \pi_{t+1}(s)\rangle \\ &+ \langle Q^*(s) - (\mathcal{T}_{\mu_t} Q_t)(s), \pi^*(s)\rangle \end{aligned} \tag{C.36}$$

Since $\langle (\mathcal{T}_{\mu_t} Q_t)(s), \pi^*(s) \rangle \leq \langle (\mathcal{T}_{\mu_t} Q_t)(s), \bar{\pi}_{t+1}(s) \rangle$ if $\bar{\pi}_{t+1}(s)$ is greedy with respect to $(\mathcal{T}_{\mu_t} Q_t)(s)$, then plugging Eq C.35 in Eq C.36 we have

$$V^*(s) - V_{t+1}(s) \leq \eta_{\pi_{t+1}}^{-1} D_h(\bar{\pi}_{t+1}(s), \pi_t(s)) - \langle Q_{t+1}(s) - (\mathcal{T}_{\mu_t} Q_t)(s), \pi_{t+1}(s) \rangle + \langle Q^*(s) - (\mathcal{T}_{\mu_t} Q_t)(s), \pi^*(s) \rangle \tag{C.37}$$

Next, cf. Lemma 13, $\langle Q_{t+1}(s) - (\mathcal{T}_{\mu_t} Q_t)(s), \pi_{t+1}(s) \rangle \geq 0$. Plugging back into Eq C.36, we obtain

$$\begin{aligned} V^*(s) - V_{t+1}(s) &\leq \eta_{\pi_{t+1}}^{-1} D_h(\bar{\pi}_{t+1}(s), \pi_t(s)) + \langle Q^*(s) - (\mathcal{T}_{\mu_t} Q_t)(s), \pi^*(s) \rangle \\ &\leq \langle Q^*(s) - (\mathcal{T}_{\mu_t} Q_t)(s), \pi^*(s) \rangle + \epsilon_{\pi_{t+1}} \end{aligned} \tag{C.38}$$

where the last step follows from step-size adaptation condition.

Applying Lemma 4 for $\tilde{\pi}_{t+1}(s)$ greedy with respect to Q_t,

$$\begin{aligned} -\langle Q_t(s), \mu_t(s) - \pi^*(s) \rangle \leq -\langle Q_t(s), \mu_t(s) - \tilde{\pi}_{t+1}(s) \rangle &\leq \eta_{\mu_t}^{-1} \big(D_h(\tilde{\pi}_{t+1}(s), \pi_t(s)) \\ &\quad - D_h(\pi_{t+1}(s), \pi_t(s)) \\ &\quad - D_h(\pi_{t+1}(s), \tilde{\pi}_{t+1}(s)) \big) \\ &\leq \eta_{\mu_t}^{-1} D_h(\tilde{\pi}_{t+1}(s), \pi_t(s)) \end{aligned} \tag{C.39}$$

Further,

$$
\begin{aligned}
&\langle Q^*(s) - (\mathcal{T}_{\mu_t} Q_t)(s), \pi^*(s)\rangle \\
&= \sum_{a\in\mathcal{A}} (Q^*(s,a) - (\mathcal{T}_{\mu_t} Q_t)(s,a))\pi^*(a|s) \\
&= \gamma \sum_{s'\in\mathcal{S}} \sum_{a\in\mathcal{A}} P(s'|s,a)\pi^*(a|s)\Big[\langle Q^*(s'), \pi^*(s')\rangle - \langle Q_t(s'), \mu_t(s')\rangle\Big] \\
&= \gamma \sum_{s'\in\mathcal{S}} \sum_{a\in\mathcal{A}} P(s'|s,a)\pi^*(a|s)\Big[\langle Q^*(s') - Q_t(s'), \pi^*(s')\rangle - \langle Q_t(s'), \mu_t(s') - \pi^*(s')\rangle\Big] \\
&\overset{\text{Eq. C.39}}{\leq} \gamma \sum_{s'\in\mathcal{S}} \sum_{a\in\mathcal{A}} P(s'|s,a)\pi^*(a|s)\Big[\langle Q^*(s') - Q_t(s'), \pi^*(s')\rangle + \eta_{\mu_t}^{-1} D_h(\tilde{\pi}_{t+1}(s), \pi_t(s))\Big] \\
&\overset{\text{cf. premise}}{\leq} \gamma \sum_{s'\in\mathcal{S}} \sum_{a\in\mathcal{A}} P(s'|s,a)\pi^*(a|s)\Big[\langle Q^*(s') - Q_t(s'), \pi^*(s')\rangle + \epsilon_{\mu_t}\Big] \\
&= \gamma^2 \sum_{s'\in\mathcal{S}} \sum_{a\in\mathcal{A}} \pi^*(a|s)P(s'|s,a)\Big[\sum_{s''\in\mathcal{S}} \sum_{a'\in\mathcal{A}} \pi^*(s',a')P(s''|s',a')\Big[V^*(s'') - V_t(s'')\Big] + \epsilon_{\mu_t}\Big]
\end{aligned}
$$

Taking the max norm, using the triangle inequality and contraction property

$$
\|\langle Q^* - (\mathcal{T}_{\mu_t} Q_t), \pi^*\rangle\|_\infty \leq \gamma^2 \|V^* - V_t\|_\infty + \gamma\epsilon_{\mu_t}
$$

Plugging back in Eq. C.38

$$
\begin{aligned}
V^*(s) - V_{t+1}(s) &\leq \langle Q^*(s) - (\mathcal{T}_{\mu_t} Q_t)(s), \pi^*(s)\rangle + \epsilon_{\pi_{t+1}} \\
&\leq \|\langle Q^* - (\mathcal{T}_{\mu_t} Q_t), \pi^*\rangle\|_\infty + \epsilon_{\pi_{t+1}} \\
&\leq \gamma^2 \|V^* - V_t\|_\infty + \gamma\epsilon_{\mu_t} + \epsilon_{\pi_{t+1}}
\end{aligned}
$$

which is the first claim in Eq.C.33. Recursing yields the second claim in Eq.C.34. □

2 Details on PMD updates for Sec. 2: Functional Acceleration for PMD

Algorithm 12 PMD(++)

1: Input: $\mu_0, \pi_0 \in \operatorname{rint} \Pi$, adaptive $\{\eta_{\pi_{t+1}}, \eta_{\mu_t}\}_{t\geq 0}$

2: **for** $t = 1, 2 \dots T$ **do**

3: `PMD (+lookahead)`

$$\mu_t(s) = \operatorname{argmin}_{\pi(s)\in\Delta(\mathcal{A})} -\langle Q_t(s), \pi(s)\rangle, \quad (\mathcal{T}_{\mu_t} Q_t)(s) \doteq \mathbb{E}[r(s) + \gamma\langle Q_t(s'), \mu^t(s')\rangle]$$
$$\pi_{t+1}(s) = \operatorname{argmin}_{\pi(s)\in\Delta(\mathcal{A})} -\langle (\mathcal{T}_{\mu_t} Q_t)(s), \pi(s)\rangle + \eta^{-1}_{\pi_{t+1}} D_h(\pi(s), \pi_t(s))$$

4: `PMD (+extragradient)`

$$\mu_t(s) = \operatorname{argmin}_{\pi(s)\in\Delta(\mathcal{A})} -\langle Q_t(s), \pi(s)\rangle + \eta^{-1}_{\mu_t} D_h(\pi(s), \pi_t(s)), \quad (\mathcal{T}_{\mu_t} Q_t)(s) \doteq \mathbb{E}[r(s) + \gamma\langle Q_t(s'), \mu^t(s')\rangle]$$
$$\pi_{t+1}(s) = \operatorname{argmin}_{\pi(s)\in\Delta(\mathcal{A})} -\langle (\mathcal{T}_{\mu_t} Q_t)(s), \pi(s)\rangle + \eta^{-1}_{\pi_{t+1}} D_h(\pi(s), \pi_t(s))$$

5: `Lazy PMD (+momentum)`

$$\pi_{t+1}(s) = \operatorname*{argmin}_{\pi(s)\in\Delta(\mathcal{A})} -\langle Q_t(s) + \eta^{-1}_{\pi_{t+1}} \eta_{\pi_t} (Q_t(s) - Q_{t-1}(s)), \pi(s)\rangle + \eta^{-1}_{\pi_{t+1}} D_h(\pi(s), \pi_t(s))$$

3 Details on Algorithmic Implementation for Sec. 2.1: Approximate Functional Acceleration for Parametric Policies

There are two ways of updating the parameter vector θ (cf. Lemma 3 [Bubeck, 2015]): (i) the PGD perspective of MD [Abdolmaleki et al., 2018, Alfano et al., 2024, Haarnoja et al., 2018] (see Appendix 5), or (ii) the proximal perspective [Tomar et al., 2020, Vaswani et al., 2021, 2023]. The latter is used and described in Alg. 9.

We execute the parameter optimization in Alg. 9 in expectation over the state-action space—in full-batch (computing d^π_ρ exactly and in expectation for all actions) to showcase the higher-level optimization that is the spotlight of this work and remove any other collateral artifacts or confounding effects from exploration of the state space or too early committal to a strategy [Mei et al., 2021]. Practical large-scale algorithms apply mini-batches sampled from a replay buffer, with the updates somewhere between full-batch and online. In making this simplification, we inevitably leave complementary investigations on the influence of stochasticity and variance of the policy gradient for future work.

Policy approximation We parametrize the policy iterates using a Bregman policy class $\{\pi_\theta : \pi_\theta(s) = \text{proj}^h_{\Delta(\mathcal{A})}(\nabla h^*(f_\theta(s))), s \in \mathcal{S}\}$ with a tabular parametrization θ. For the updates requiring two policies, we keep them parametrized separately with $\mu_{\mathbf{w}}$ and π_θ. We formulate the policy optimization problem using the extension proposed by Tomar et al. [2020]. Each iteration, in an "inner-loop" optimization procedure, we update θ and $\mathbf{w}$ using k updates with standard GD on the composite PMD surrogate model, denoted $\ell : \Theta \to \mathbb{R}$ (with $\Theta \doteq \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}$ cf. the tabular parametrization) associated with the policy represented by those parameters π_θ, or $\mu_{\mathbf{w}}$, respectively. We execute the parameter optimization in expectation over the state-action space. Concretely, for PMD we use the surrogate

$$\ell(\theta) \doteq \mathbb{E}_{s\sim d^\rho_t}\left[-\langle Q_{\pi_t}(s), \pi_\theta\rangle + \eta^{-1}_{\pi_{t+1}}(s) D_h(\pi_\theta(s), \pi_t(s))\right] \tag{C.40}$$

and update in an "inner-loop" optimization procedure

$$\textit{(init)}\ \theta^{(0)} \doteq \theta_t \qquad \textit{(for } i \in [0..k]\textit{)}\ \theta^{(i+1)} = \theta^{(i)} - \beta\nabla_{\theta^{(i)}}\ell(\theta^{(i)}) \qquad \textit{(final)}\ \theta_{t+1} \doteq \theta^{(k)}$$

with β—a small learning rate. The optimization procedure for $\mathbf{w}$ is analogous. The rest of the algorithms use the surrogate objectives as described in Sec. 2.

Value function Everywhere in the numerical studies, we used Q_{μ_t} instead of $(\mathcal{T}Q_{\pi_t})(s) \doteq \mathbb{E}[r(s) + \gamma\langle Q_{\pi_t}(s'), \mu_t\rangle]$. In the inexact setting, we estimate $\widehat{Q}_{\pi_t}$ and use it in place of Q_{π_t}. Similarly, $\widehat{Q}_{\mu_t}$ in place of Q_{μ_t}.

Step-size adaptation We use state dependent step-sizes, and for step-size adaptation, we compute $\eta_{\mu_t}(s) = \gamma^{-t}\epsilon_0^{-1} D_h(\text{greedy}(Q_{\pi_t}(s)), \pi_t(s))$, with $\epsilon_0 = 10^{-4}$ a small constant according to the optimal adaptive schedule for PMD, cf. Johnson et al. [2023]. The value for $\eta_{\pi_{t+1}}(s)$ is chosen analogously.

Objectives We now describe in detail the objectives for each algorithm.

PMD—We approximate the optimization of the objective in Eq. C.40 with respect to the parameters θ of π_θ with k GD updates using a per-state step-size $\eta_{\mu_t}(s) = \gamma^{-t}\epsilon_0^{-1} D_h(\text{greedy}(Q_{\pi_t}(s)), \pi_t(s))$.

PMD(+loo)—We keep two policies μ, π_θ—the former is the non-parametric greedy policy

$$\mu_t(s) = \max_{a \in \mathcal{A}} Q_{\pi_t}(s)(a)$$

The latter is parametrized and its parameters θ optimize the following lookahead-based surrogate with k GD updates using step-size adaptation $\eta_{\pi_{t+1}}(s) = \gamma^{-t}\epsilon_0^{-1} D_h(\text{greedy}((\mathcal{T}_{\mu_t}Q_{\pi_t})(s)), \pi_t(s))$.

$$\ell(\theta) \doteq \mathbb{E}_{s\sim d_t^\rho}\left[-\langle(\mathcal{T}_{\mu_t}Q_{\pi_t})(s), \pi_\theta(s)\rangle + \eta_{\pi_{t+1}}^{-1}(s) D_h(\pi_\theta(s), \pi_t(s))\right]$$

In the results for the experimental section, we used Q_{μ_t} in place of $\mathcal{T}_{\mu_t}Q_{\pi_t}$.

PMD(+ext)—The update to π_θ is identical to PMD(+loo). In contrast to PMD(+loo), $\mu_{\mathbf{w}}$ is parametrized with parameter vector $\mathbf{w}$. The update to $\mu_{\mathbf{w}}$ uses, at each iteration, k GD updates on the surrogate objective

$$\ell(\mathbf{w}) \doteq \mathbb{E}_{s\sim d_t^\rho}\left[-\langle Q_{\pi_t}(s), \mu_{\mathbf{w}}(s)\rangle + \eta_{\mu_t}^{-1} D_h(\mu_{\mathbf{w}}(s), \pi_t(s))\right]$$

The step-sizes η_{μ_t} are adapted using $\eta_{\mu_t}(s) = \gamma^{-t}\epsilon_0^{-1} D_h(\text{greedy}(Q_{\pi_t}(s)), \pi_t(s))$.

PMD(+mom)—A single set of parameters θ are learned by updating, at each iteration, using k GD updates on the objective

$$\ell(\theta) \doteq \mathbb{E}_{s\sim d_t^\rho}\left[-\langle Q_{\pi_t}(s) + \eta_{\pi_{t+1}}^{-1}\eta_{\pi_t}[Q_{\pi_t}(s) - Q_{\pi_{t-1}}(s)], \pi_\theta(s)\rangle + \eta_{\pi_{t+1}}^{-1}(s) D_h(\pi_\theta(s), \pi_t(s))\right]$$

4 Acceleration Methods in Optimization

We now summarize two generalizations of the well-known core algorithms of online optimization for predictable sequences, cf. Joulani et al. [2020b]: **(i) Optimistic Mirror Descent (OMD)** [Chiang et al., 2012, Rakhlin and Sridharan, 2013a,b], including Extragradient Descent (Korpelevich [1976], and Mirror-Prox [Juditsky et al., 2011, Nemirovski, 2004], and **(ii) Adaptive Optimistic Follow-the-Regularized-Leader (AO-FTRL)** [Mohri and Yang, 2016, Rakhlin and Sridharan, 2013a].

(i) Optimistic Mirror Descent (OMD) Starting with some previous iterate $x_t \in \mathcal{X}$, an OMD [Joulani et al., 2020b] learner x uses a prediction $\tilde{g}_{t+1} \in \mathcal{X}^*$, with $\mathcal{X}^*$ the dual space of $\mathcal{X}$, to minimize the regret on its convex loss function $f : \mathcal{X} \to \mathbb{R}$ against an optimal comparator $x^* \in \mathcal{X}$ with

$$x_{t+1} = \arg\min_{x\in\mathcal{X}} \langle \eta_t g_t + \eta_{t+1}\tilde{g}_{t+1} - \eta_t \tilde{g}_t, x\rangle + D_h(x, x_t) \qquad \text{(OMD update)}$$

with $\tilde{g}_{t+1} \approx \nabla f(x_{t+1})$ an optimistic gradient prediction, and $g_t \doteq \nabla f(x_t)$ gradient feedback, D_h a Bregman divergence with mirror map h.

Extragradient (extrapolated gradient) methods consider two-step update rules for the same objective using an intermediary sequence $\tilde{x}$

$$\tilde{x}_t = \arg\min_{x\in\mathcal{X}} \langle g_t, x\rangle + D_h(x, x_t) \qquad \text{(gradient update)}$$

$$x_{t+1} = \arg\min_{\pi\in\Pi} \langle \tilde{g}_{t+1}, x\rangle + D_h(x, x_t) \qquad \text{(Extragradient update)}$$

with $g_t \doteq \nabla f(x_t)$ the gradient feedback for x_t, and $\tilde{g}_{t+1} \equiv \nabla f(\tilde{x}_t)$ the gradient feedback for the intermediary iterate $\tilde{x}_t$.

(ii) Adaptive Optimistic Follow-The-Regularized-Leader (AO-FTRL) A learner using AO-FTRL updates x with

$$x_{t+1} = \arg\min_{x \in \mathcal{X}} \langle g_{0:t} + \tilde{g}_{t+1}, x \rangle + h_{1:t-1}(x) \qquad \text{(AO-FTRL update)}$$

where $g_{0:t} = \sum_{j=0}^{t} g_j$ is the gradient feedback, $\tilde{g}_{t+1}$ is the optimistic part of the update, and $h_{0:t}(x) = \sum_{j=0}^{t} h_j(x)$ represent the "proximal" part of the adaptive regularization (cf. Joulani et al. [2020b]), counterparts of the Bregman divergence in the MD update that regularizes iterates to maintain proximity.

5 Approximate Policy Mirror Descent as Projected Gradient Descent (PGD)

In this section we provide an alternative perspective on PMD—cf. Lemma 3, stating that the MD update can be rewritten in the following ways

$$\begin{aligned} x_{t+1} &= \operatorname{argmin}_{x \in \mathcal{X} \cap \mathcal{C}} D_h(x, \nabla h^*(\nabla h(x_t) + \eta \nabla f(x_t))) && \text{(PGD)} \\ &= \operatorname{argmin}_{x \in \mathcal{X} \cap \mathcal{C}} \eta \langle \nabla f(x_t), x \rangle + D_h(x, x_t) && \text{(proximal perspective)} \end{aligned}$$

Alfano et al. [2024] introduces the concept of Bregman policy class

$$\{\pi_\theta : \pi_\theta(s) = \operatorname{proj}^h_{\Delta(\mathcal{A})}(\nabla h^*(f_\theta(s))), s \in \mathcal{S}\}$$

and uses a parametrized function f_θ to approximate the dual update of MD $f_{t+1}(s) \doteq \nabla h(\pi_t(s)) - \eta_{\pi_{t+1}} \widehat{Q}_t(s)$. To satisfy the simplex constraint, Alfano et al. [2024] uses the L2 norm to measure function approximation errors, whereas Xiong et al. [2024] extends this

to a Bregman projection on the dual approximation mapped back to the policy space $\pi_\theta(s) = \text{proj}^h_{\Delta(\mathcal{A})}(\nabla h^*(f_{t+1}(s)))$, equivalent to $\theta_{t+1} = \text{argmin}_{\theta\in\Theta} D_h(\pi_\theta(s), \nabla h^*(f_{t+1}(s)))$ (cf. Amari [2016], the divergences derived from two convex functions are substantially the same, except for the order of the arguments $D_h(x, \nabla h^*(\nabla h(y))) = D_{h^*}(\nabla h(y), \nabla h^*(x))$).

Using the negative Boltzmann-Shannon entropy, yields the softmax policy class $\pi_\theta(s,a) \doteq {\exp f_\theta(s,a)}/{\| \exp f_\theta(s)\|_1}, \forall s, a \in \mathcal{S} \times \mathcal{A}$.

For the approximate setting, in the main text, we rely on the formulation introduced by Tomar et al. [2020]. Here, we provide additional details on the formulation by Alfano et al. [2024] and Xiong et al. [2024].

Algorithm 13 Approximate PMD(+mom) (PGD perspective)

1: Initialize policy parameter $\theta_0 \in \Theta$, mirror map h, small constant ϵ_0, learning rate β
2: **for** $t = 1, 2 \dots T$ **do**
3: Approximate Q_{π_t} with $\widehat{Q}_t$ (critic update)
4: Compute adaptive step-size $\eta_{\pi_{t+1}}(s) = {D_h(\text{greedy}(\widehat{Q}_t(s))}/{\gamma^{2(t+1)}\epsilon_0}$
5: Find $\pi_{t+1} \doteq \pi_{\theta_{t+1}} = \nabla h^*(f_{\theta_{t+1}})$ by (approximately) solving the surrogate problem (with k GD updates)
6: $\min_{\theta\in\Theta} \ell(\theta)$ $\quad \ell(\theta) \doteq -\mathbb{E}_{s\sim d^\rho_t}[D_h(\pi_\theta(s), \nabla h^*(\nabla h(\pi_t(s)) - \eta_{\pi_{t+1}}(s)\widehat{Q}_t(s) - \eta_{\pi_t}(s)(\widehat{Q}_t(s) - \widehat{Q}_{t-1}(s)))]$
7: *(init)* $\theta^{(0)} \doteq \theta_t$ $\quad$ *(for* $i \in [0..k-1]$*)* $\theta^{(i+1)} = \theta^{(i)} - \beta\nabla_{\theta^{(i)}}\ell(\theta^{(i)})$ $\quad$ *(final)* $\theta_{t+1} \doteq \theta^{(k)}$

Alg. 13 describes a PGD perspective on PMD(+mom), following Abdolmaleki et al. [2018], Alfano et al. [2024], Haarnoja et al. [2018], Xiong et al. [2024]. At each iteration, after taking a gradient step in dual space, a Bregman projection is used on the dual approximation mapped back to the policy space, to satisfy the simplex constraint.

6 Newton's method

The Newton-Kantorovich theorem generalizes Newton's method for solving nonlinear equations to infinite-dimensional Banach spaces. It provides conditions under which

Newton's method converges and gives an estimate of the convergence rate. Newton-Kantorovich theorem deals with the convergence of Newton's method for a nonlinear operator $F : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$ and $\mathcal{Y}$ are Banach spaces. The method iteratively solves $F(x) = 0$ using

$$x_{t+1} = x_t - (\nabla F)^{-1} F(x_t)$$

where ∇F is a generalization of the Jacobian of F, provided F is differentiable. Intuitively, at each iteration, the method performs a linearization of $F(x) = 0$ close to x, using a first order Taylor expansion: $F(x + \Delta x) \approx F(x) + \nabla F(x)\Delta x$, where $F(x) + \nabla F(x)\Delta x = 0 \iff \Delta x = -(\nabla F)^{-1} F(x)$. For x close to x^*, $x - (\nabla F)^{-1} F(x)$ is a good approximation of x^*. The iterative sequence $\{x_t\}_{t \geq 0}$ converges to x^*, assuming the Jacobian matrix exists, is invertible, and Lipschitz continuous.

Quasi-Newton methods Any method that replaces the exact computation of the Jacobian matrices in the Newton's method (or their inverses) with an approximation, is a quasi-Newton method. A quasi-Newton method constructs a sequence of iterates $\{x_t\}_{t \geq 0}$ and a sequence of matrices $\{J_t\}_{t \geq 0}$ such that J_t is an approximation of the Jacobian $\nabla F(x_t)$ for any $t \geq 0$ and

$$x_{t+1} = x_t - (J_t)^{-1} F(x_t)$$

In Anderson's acceleration [Anderson, 1965], information about the last iterates is used to update the approximation of J_t.

Policy iteration as Newton's method In the context of Markov Decision Processes (MDPs), policy iteration may be interpreted as Newton's method with the following notations and analogies. First, using the Bellman optimality operator $\mathcal{T}V(s) \doteq \max_a [r(s, a) + \gamma \sum_{s' \in \mathbb{S}} P(s'|s, a)V(s')]$, the aim is to find V such that $V = \mathcal{T}V$, which is akin to finding

the roots V, such that $F(V) = V - \mathcal{T}V = (I - \mathcal{T})(V) = 0$. We interpret $F = \nabla f$ as the gradient of an unknown function $f : \mathbb{R}^n \to \mathbb{R}^n$, despite the Bellman operator being non-differentiable in general due to the max. Where the greedy policy π_t attains the max in $\mathcal{T}V$, we obtain $J_t = \mathbf{I} - \gamma \mathbf{P}_{\pi_t}$, which is invertible for $\gamma \in [0, 1)$. Expanding the Bellman operator, we have

$$V_{\pi_{t+1}} = r_{\pi_{t+1}} + \gamma P_{\pi_{t+1}} V_{\pi_{t+1}} \implies V_{\pi_{t+1}} = (J_t)^{-1} r_{\pi_{t+1}}$$
$$J_t V_{\pi_{t+1}} = r_{\pi_{t+1}}$$

The values corresponding to two successive PI steps can be related in the following way by manipulating the equations [Grand-Clément, 2021, Puterman and Brumelle, 1979]

$$\begin{aligned}
V_{\pi_{t+1}} &= (J_t)^{-1} r_{\pi_{t+1}} \\
&= V_{\pi_t} - V_{\pi_t} + (J_t)^{-1} r_{\pi_{t+1}} \\
&= V_{\pi_t} - (J_t)^{-1} J^t V_{\pi_t} + (J_t)^{-1} r_{\pi_{t+1}} \\
&= V_{\pi_t} - (J_t)^{-1} (-r_{\pi_{t+1}} + J^t V_{\pi_t}) \\
&= V_{\pi_t} - (J_t)^{-1} (-r_{\pi_{t+1}} + (I - \gamma P_{\pi_{t+1}}) V_{\pi_t}) \\
&= V_{\pi_t} - (J_t)^{-1} (V_{\pi_t} - r_{\pi_{t+1}} - \gamma P_{\pi_{t+1}} V_{\pi_t}) \\
&= V_{\pi_t} - (J_t)^{-1} (V_{\pi_t} - \mathcal{T} V_{\pi_t})
\end{aligned}$$

In the main text we used the notation $\Psi_t \doteq (I - \gamma P_{\pi_t})^{-1} = (J_t)^{-1}$, and applied the definition $\nabla f(V_{\pi_t}) \doteq F(V_{\pi_t}) = (I - \mathcal{T})(V_{\pi_t}) = V_{\pi_t} - \mathcal{T} V_{\pi_t}$ which yielded the expression

$$V_{\pi_{t+1}} = V_{\pi_t} - \Psi \nabla f(V_{\pi_t})$$

7 Experimental details for Sec. 3: Numerical Studies

7.1 Details of two-state Markov Decision Processes

In this section we give the specifics of the two-state MDPs presented in this work. We make use of the notation

$$P(s_k|s_i, a_j) = \mathbf{P}[i \times |\mathcal{A}| + j][k], \text{ with } \mathbf{P} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|\times|\mathcal{S}|}$$
$$r(s_i, a_j) = \mathbf{r}[i \times |\mathcal{A}| + j], \text{ with } \mathbf{r} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$$

In the main text, we used example (i), and in Appendix 8.2, additionally examples (ii), (iii), and (iv)

(i) $|\mathcal{A}| = 2, \gamma = 0.9, \mathbf{r} = [-0.45, -0.1, 0.5, 0.5],$
$\mathbf{P} = [[-0.45, 0.3], [0.99, 0.01], [0.2, 0.8], [0.99, 0.01]]$

(ii) $|\mathcal{A}| = 2, \gamma = 0.9, \mathbf{r} = [0.06, 0.38, -0.13, 0.64],$
$\mathbf{P} = [[0.01, 0.99], [0.92, 0.08], [0.08, 0.92], [0.70, 0.30]]$

(iii) $|\mathcal{A}| = 2, \gamma = 0.9, \mathbf{r} = [0.88, -0.02, -0.98, 0.42],$
$\mathbf{P} = [[0.96, 0.04], [0.19, 0.81], [0.43, 0.57], [0.72, 0.28]]$

(iv) $|\mathcal{A}| = 3, \gamma = 0.8, \mathbf{r} = [-0.1, -1., 0.1, 0.4, 1.5, 0.1],$
$\mathbf{P} = [[0.9, 0.1], [0.2, 0.8], [0.7, 0.3], [0.05, 0.95], [0.25, 0.75], [0.3, 0.7]]$

7.2 Details of Random Markov Decision Processes

We consider randomly constructed finite MDPs—**Random MDP** problems (a.k.a. Garnet MDPs: Generalized Average Reward Non-stationary Environment Test-bench) [Archibald et al., 1995, Bhatnagar et al., 2009], abstract, yet representative of the kind of MDP encountered in practice, which serve as a test-bench for RL algorithms [Goyal and Grand-Clement, 2021, Scherrer and Geist, 2014, Vieillard et al., 2019]. A **Random MDP** generator $\mathcal{M} \doteq (|\mathcal{S}|, |\mathcal{A}|, b, \gamma)$ is parameterized by 4 parameters: number of states $|\mathcal{S}|$, number of actions $|\mathcal{A}|$, branching factor b specifying how many possible next states are possible for each state-action pair.

The transition probabilities $P(s_0|s, a)$ are then computed as follows. First, b states $(s_1, \dots s_b)$ are chosen uniformly at random and transition probabilities are set by sampling uniform random b–1 numbers (cut points) between 0 and 1 and sorted as $(p_0 = 0, p_1, \dots p_{b-1}, p_b = 1)$. Then, the transition probabilities are assigned as $P(s_i|s, a) = p_i - p_{i-1}$ for each $1 \leq i \leq b$. The reward is state-dependent, and for each MDP, the per-state reward $r(s)$ is uniformly sampled between 0 and $R_{\max}$, such that $r \sim (0, R_{\max})^{|\mathcal{S}|}$. The illustrations shown use $|\mathcal{S}| = 100$ and $R_{\max} = 100$. Other choices yield similar results.

7.3 Details of Experimental Setup for Sec. 3.1

We use $\beta = 0.5$—the learning rate of the parameter "inner-loop" optimization problem, π_0 : *center* for all experiments of this section. We use vary one parameter of the problem while keeping all others fixed cf. Table C.1. We use 50 randomly generated MDPs for each configuration and compute the mean and standard deviation shown in the plots.

Table C.1: The parameters used for the optimization in Sec. 3.1.

Experiment	Alg/MDP param	Values
k sweep Fig. 5.2 *(Left)*	k	$\{1, 5, 10, 20, 30\}$
	b	5
	γ	0.95
	$\lvert\mathcal{A}\rvert$	10
	T	10
b sweep Fig. 5.2 *(Center-Left)*	k	100
	b	$\{5, 10, 20, 30, 40\}$
	γ	0.95
	$\lvert\mathcal{A}\rvert$	10
	T	10
γ *sweep* Fig. 5.2 *(Center-Right)*	k	30
	b	5
	γ	$\{0.98, 0.95, 0.9, 0.85\}$
	$\lvert\mathcal{A}\rvert$	10
	T	10
$\lvert\mathcal{A}\rvert$ *sweep* Fig. 5.2 *(Right)*	k	100
	b	5
	γ	0.95
	$\lvert\mathcal{A}\rvert$	$\{2, 5, 10, 15\}$
	T	20

8 Supplementary Results for Sec. 3: Numerical Studies

8.1 Supplementary results for Sec. 3.1

This section presents additional results to those in Sec. 3.1. Fig. C.1 shows the final performance and is analogous to Fig.5.2 in the main text. Fig. C.2 shows the optimality gap while learning for T iterations.

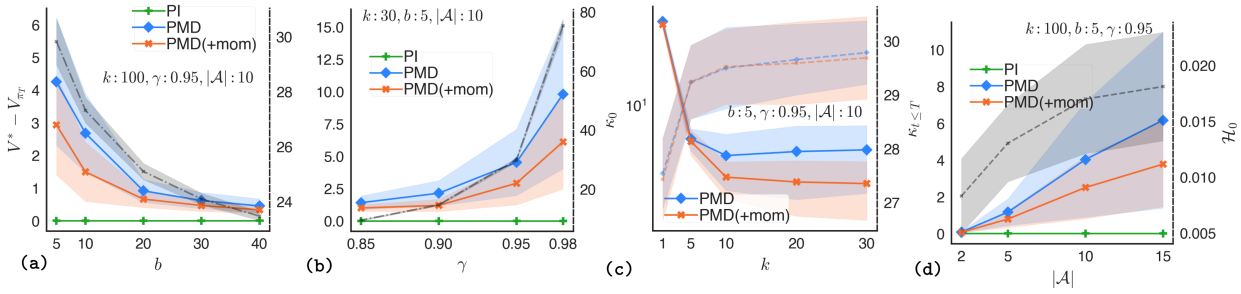


Figure C.1: The *left y-axis* shows the final optimality gap (regret) at timestep T (cf. Table C.1) of the updates: PI, PMD and PMD(+mom), after T iterations ($T = 10$ *(a-c)*, $T = 20$ *(d)*) relative to changing the hyperparameters: (a) b—the branching factor of the *Random MDP*, (b) γ—the discount factor, (c) k—the number of parameter updates, (d) $|\mathcal{A}|$—the number of actions. Shades denote standard deviation over 50 sampled MDPs. The *right y-axis* and dotted curves measure: *(a-b)*—the condition number κ_0, *(c)* the average condition number $\kappa_{t\leq T}$, *(d)* the entropy $\mathcal{H}_0$.

8.2 Supplementary results for Sec. 3.2

In this section we provide supplementary results that were omitted in the main body, related to the policy optimization dynamics of the functional acceleration methods introduced.

PMD In Fig. C.3, we compare the optimization dynamics of PMD and PI for different MDPs. We observe the policy tends to move in a straight line between semi-deterministic policies (cf. Dadashi et al. [2019]) on the boundary of the polytope, and when it passes over an attractor point it can get delayed slowing down convergence. Fig. C.4 shows the speed of convergence is governed by k which reflects the inner-loop optimizationn procedure.

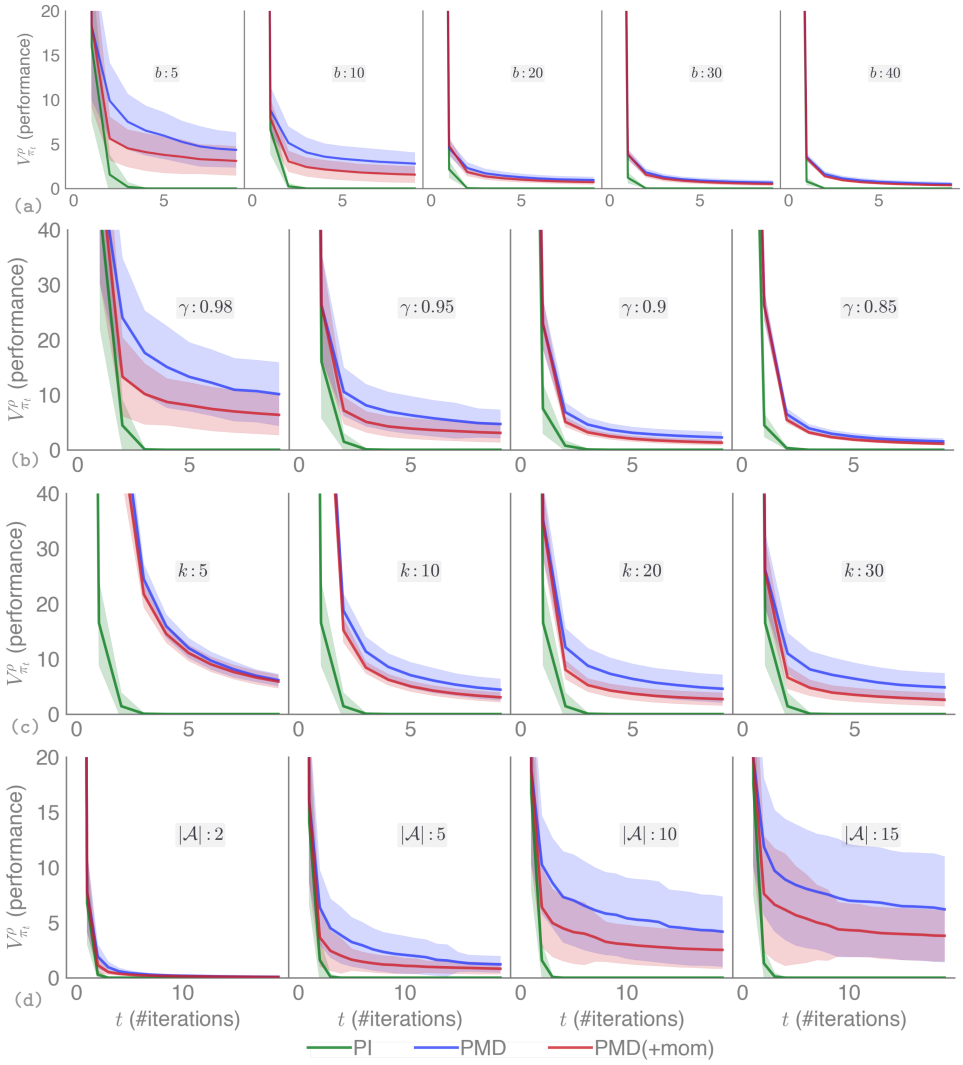


Figure C.2: The *left y-axis* shows the optimality gap (regret) of the updates: PI, PMD and PMD(+mom), for T iterations ($T = 20$ final column, $T = 10$ rest of the columns) relative to changing the hyperparameters: (a) b—the branching factor of the *Random MDP*, (b) γ—the discount factor, (c) $|\mathcal{A}|$—the number of actions, (d) k—the number of parameter updates. Shades denote standard deviation over 50 sampled MDPs.

We again observe in Fig. C.4 the accummulation points and long-escape attractor points of the optimization procedure.

PMD(+loo) In Fig. C.5 we observe the dynamics of PMD(+loo) sometimes follow a different path through the polytope compared to PMD or PI (Fig. C.3), as they are following a different ascent direction, which may be more direct compared to that of PI. Compared

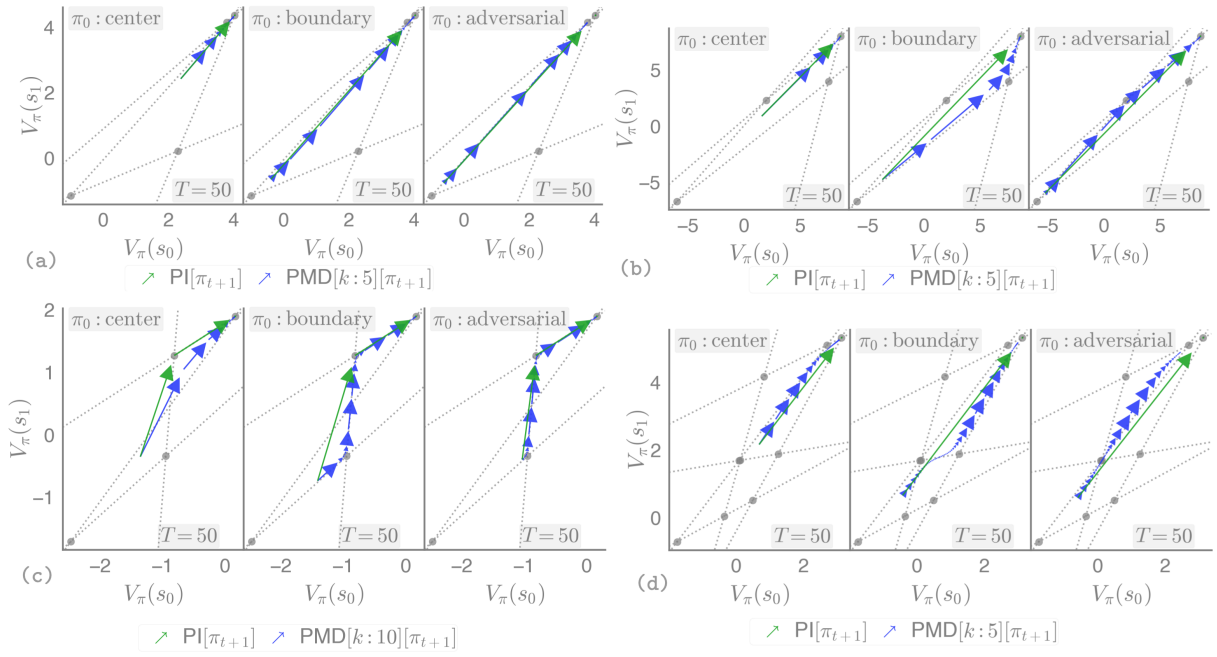


Figure C.3: Compares the policy optimization dynamics of PMD and PI on the value polytope of the different example MDPs in Sec. 7.1: *(a)* example (ii), *(b)* example (iii), *(c)* example (i), *(d)* example (iv).

to Fig. C.4, in Fig. C.6, we see less accumulation points, and more jumps, i.e. the policy improvement step returns policies at further distance apart.

PMD(+mom) Fig. C.7 compares the policy dynamics of PMD and PMD(+mom) and shows acceleration of the latter in those directions of ascent that align over consecutive steps, and have ill-conditioned optimization surfaces.

8.3 Supplementary results for Sec. 3.3

Fig. C.8, C.9, C.10, C.11 illustrate the variance over 50 optimization trajectories initialized from a random uniform distribution with mean 0 and standard deviation 1, for each of the example two-state MDPs described in Appendix 7.1. We use the same *(controlled)* setting as in Sec.3.3, and vary the critic's error τ, and the policy approximation via k.

The most illustrative example is Fig. C.8, since this setting presents the most ill-conditioned surface on which we can observe the impact of acceleration. We observe in *(a)* the instability

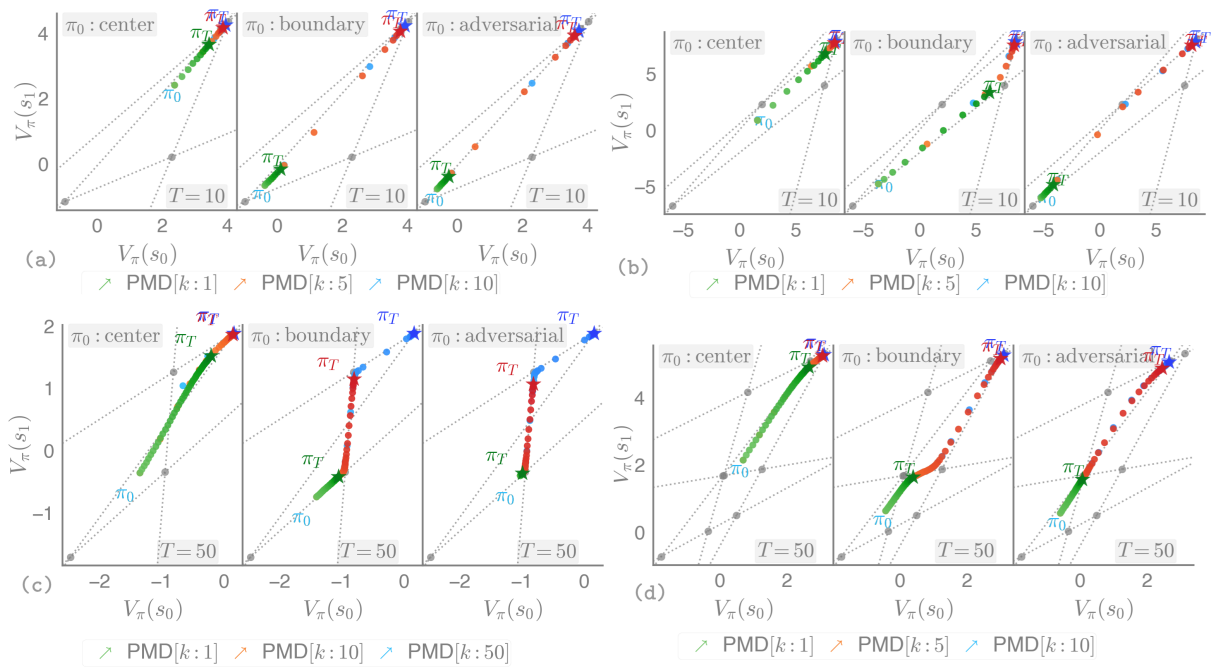


Figure C.4: Shows the policy optimization dynamics of PMD for different values of k on the value polytope of the different example MDPs in Sec. 7.1: *(a)* example (ii), *(b)* example (iii), *(c)* example (i), *(d)* example (iv).

of policy iteration with an inexact critic. As the critic's error grows the policy iterates start to oscillate between the corners of the polytope.

PMD is better behaved for low k values and starts to exhibit behaviour similar to PI at larger k values *(c,d)*. We observe PMD(+mom) is more unstable than PMD when presented with high level of errors in the inexact critic *(f)*, and tends to stay more on the boundary of the polytope *(e)*, which is consistent with having larger values of the gradient due to added momentum.

In *(g,h)* the learning curves show the variance over trajectories stemming from the random initialization, the instability of PI, the relative improvement of PMD(+mom) over PMD, particularly striking for larger k consistent with the theory. We observe in *(h)* PMD(+mom) has more variance in the beginning, which may actually be desirable in terms of exploration, and that is achieves a similar performance to PMD at the end of the optimization.

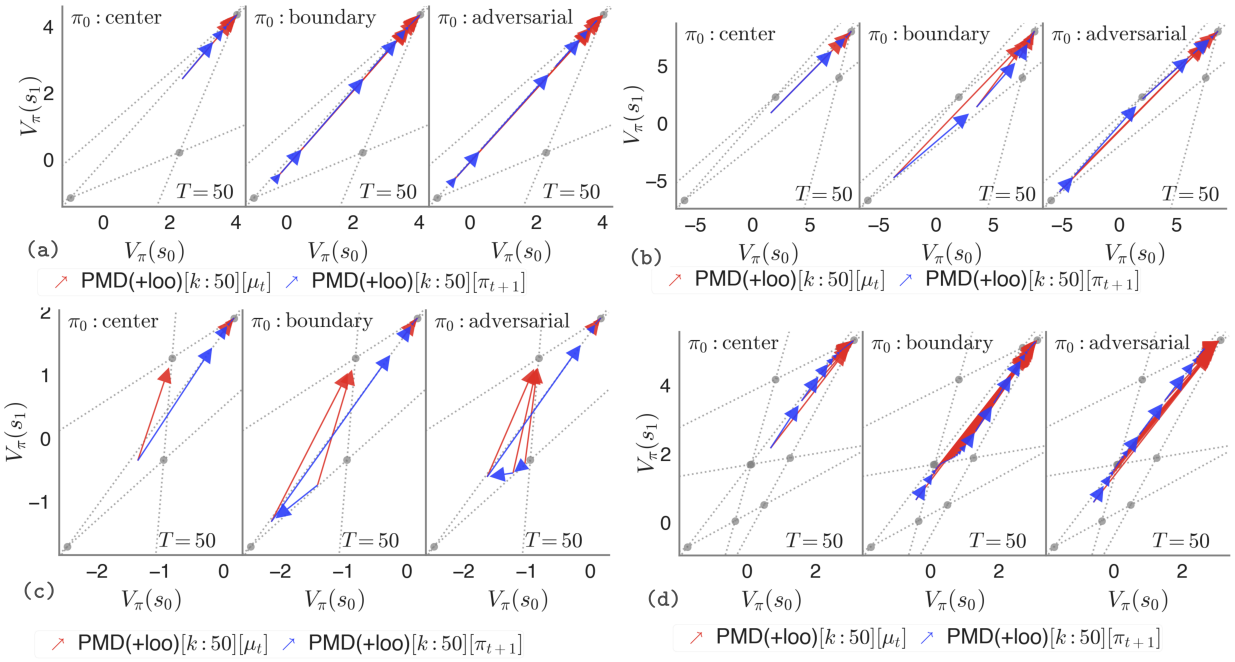


Figure C.5: Shows the policy optimization dynamics of PMD(+loo) on the value polytope of the different example MDPs in Sec. 7.1: *(a)* example (ii), *(b)* example (iii), *(c)* example (i), *(d)* example (iv).

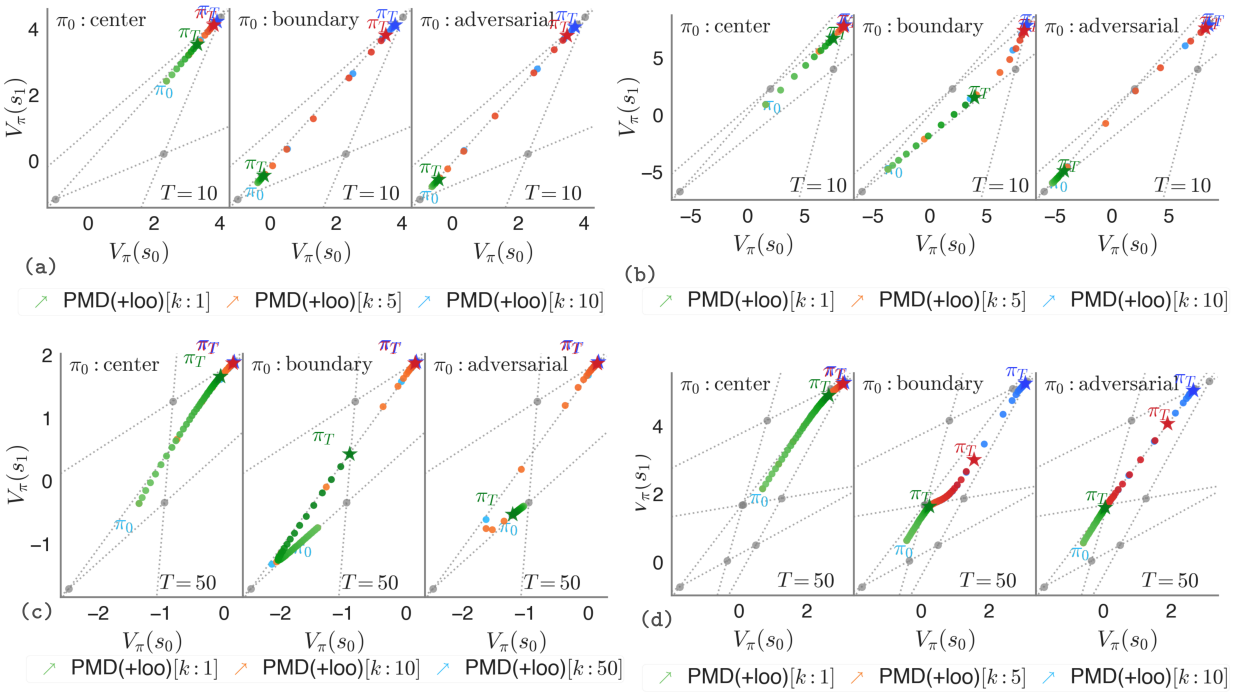


Figure C.6: Shows the policy optimization dynamics of PMD(+loo)for different values of k on the value polytope of the different example MDPs in Sec. 7.1: *(a)* example (ii), *(b)* example (iii), *(c)* example (i), *(d)* example (iv).

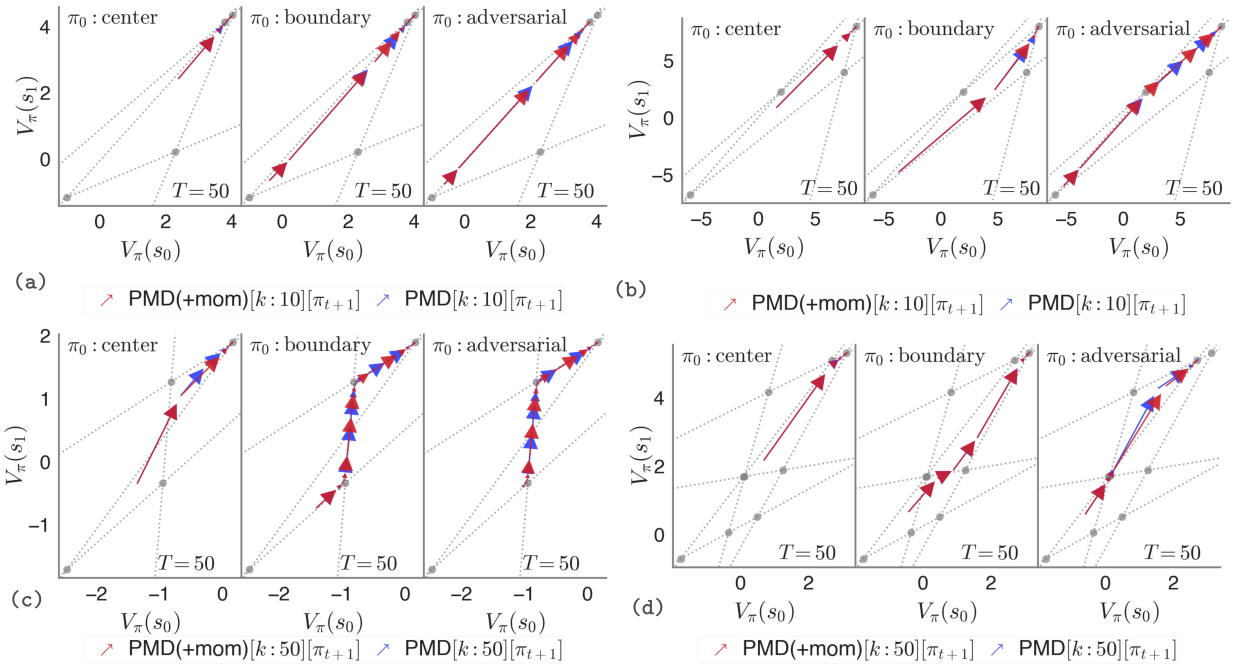


Figure C.7: Compares the policy optimization dynamics of PMD and PMD(+mom) on the value polytope of the different example MDPs in Sec. 7.1: *(a)* example (ii), *(b)* example (iii), *(c)* example (i), *(d)* example (iv).

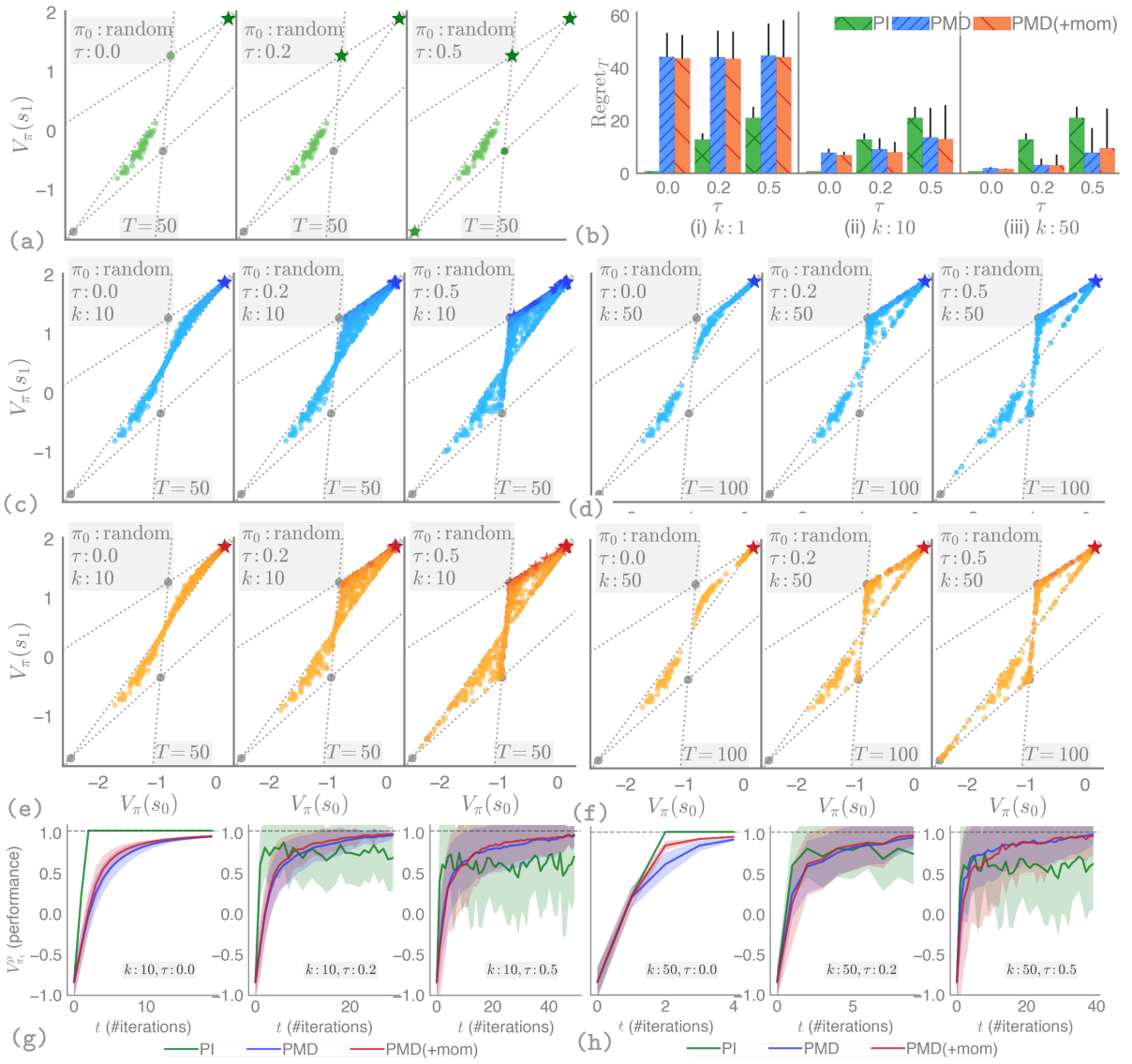


Figure C.8: Compares the statistics of the policy optimization dynamics of PI, PMD and PMD(+mom) subject to variance from random initialization (π_0 : *random_uniform(0,1)*, relative to the error in the inexact critic (τ), and over different levels of policy approximation (k). Results correspond to example (i) from Sec. 7.1.

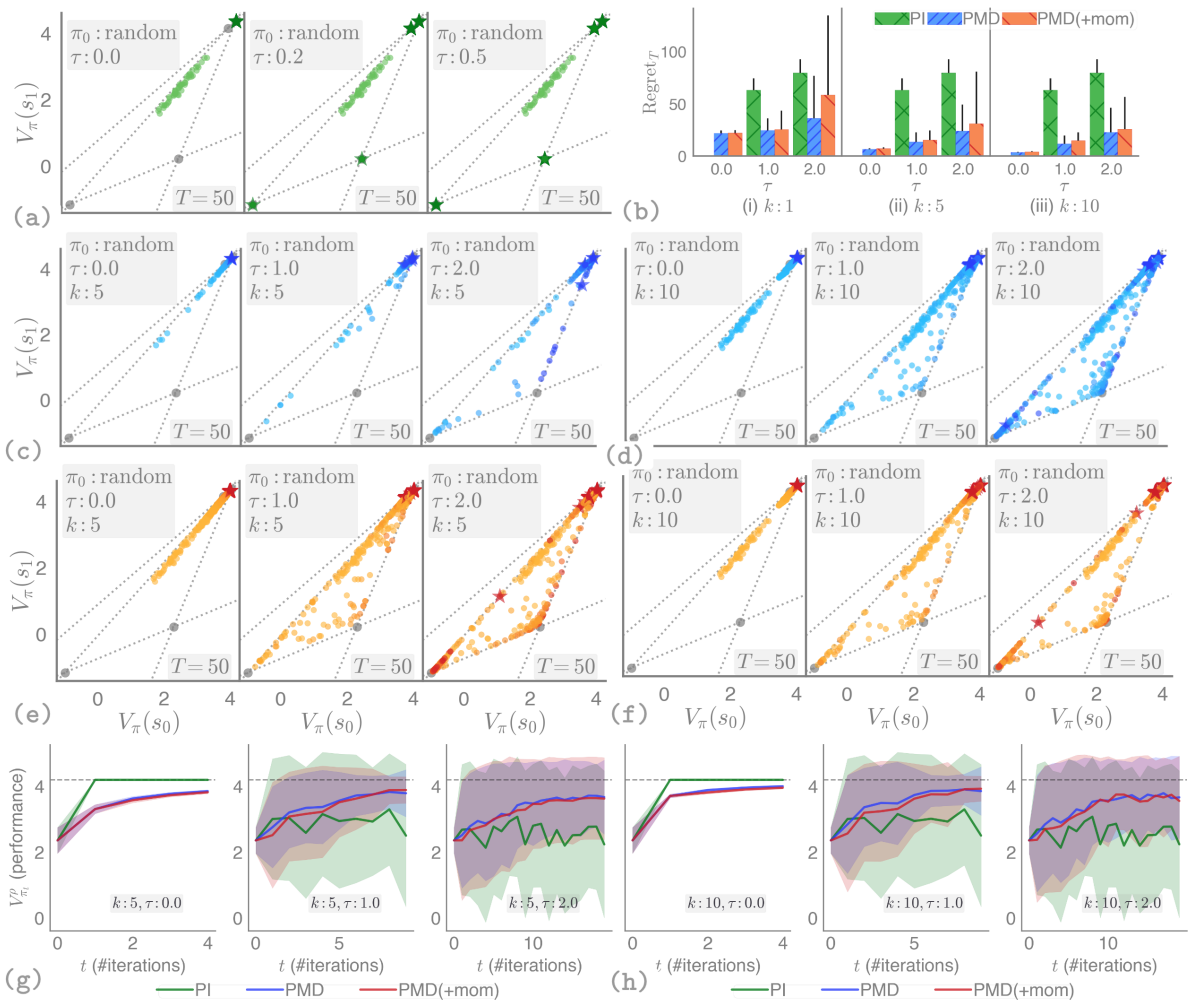


Figure C.9: Compares the statistics of the policy optimization dynamics of PI, PMD and PMD(+mom) subject to variance from random initialization (π_0 : *random_uniform(0,1)*, relative to the error in the inexact critic (τ), and over different levels of policy approximation (k). Results correspond to example (ii) from Sec. 7.1.

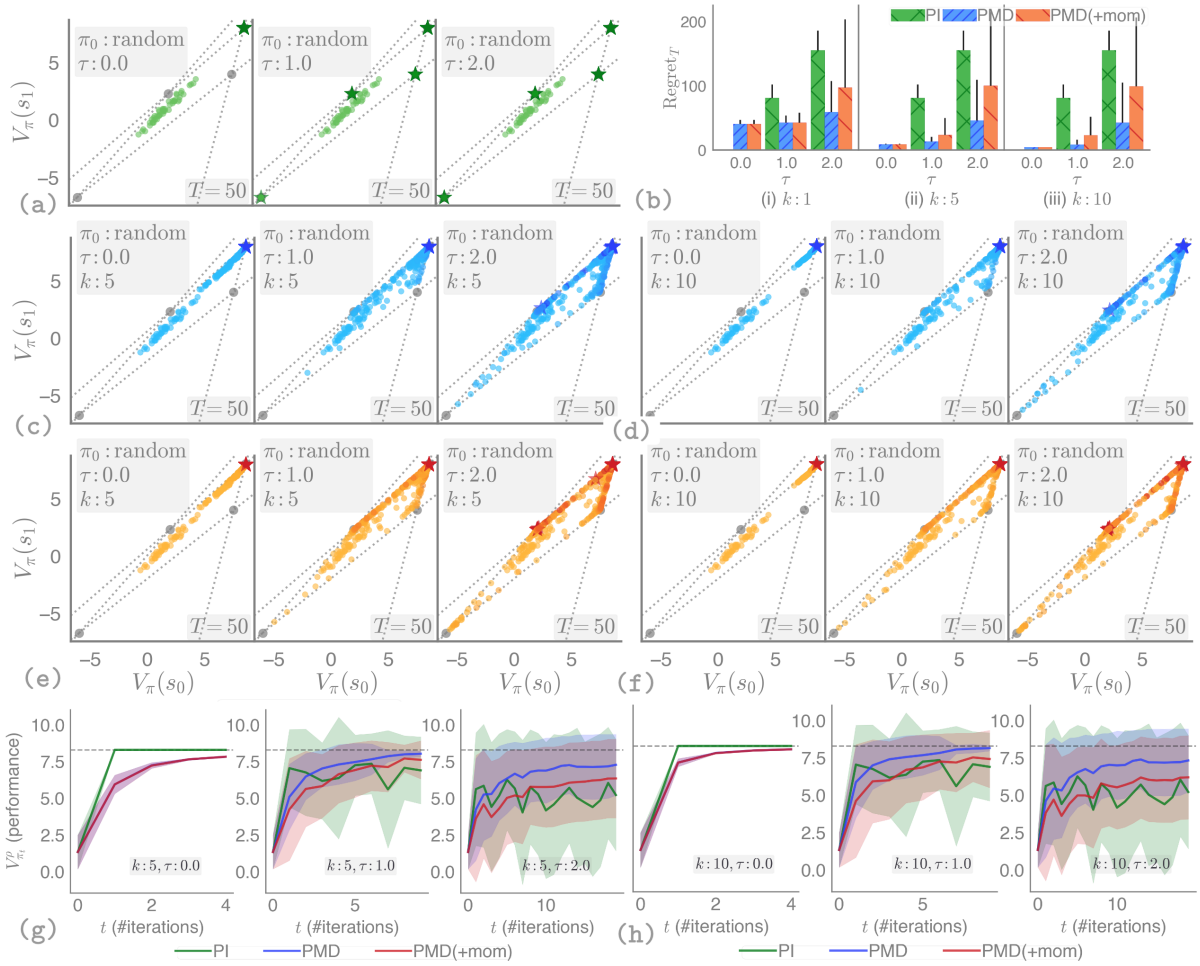


Figure C.10: Compares the statistics of the policy optimization dynamics of PI, PMD and PMD(+mom) subject to variance from random initialization (π_0 : *random_uniform(0,1)*, relative to the error in the inexact critic (τ), and over different levels of policy approximation (k). Results correspond to example (iii) from Sec. 7.1.

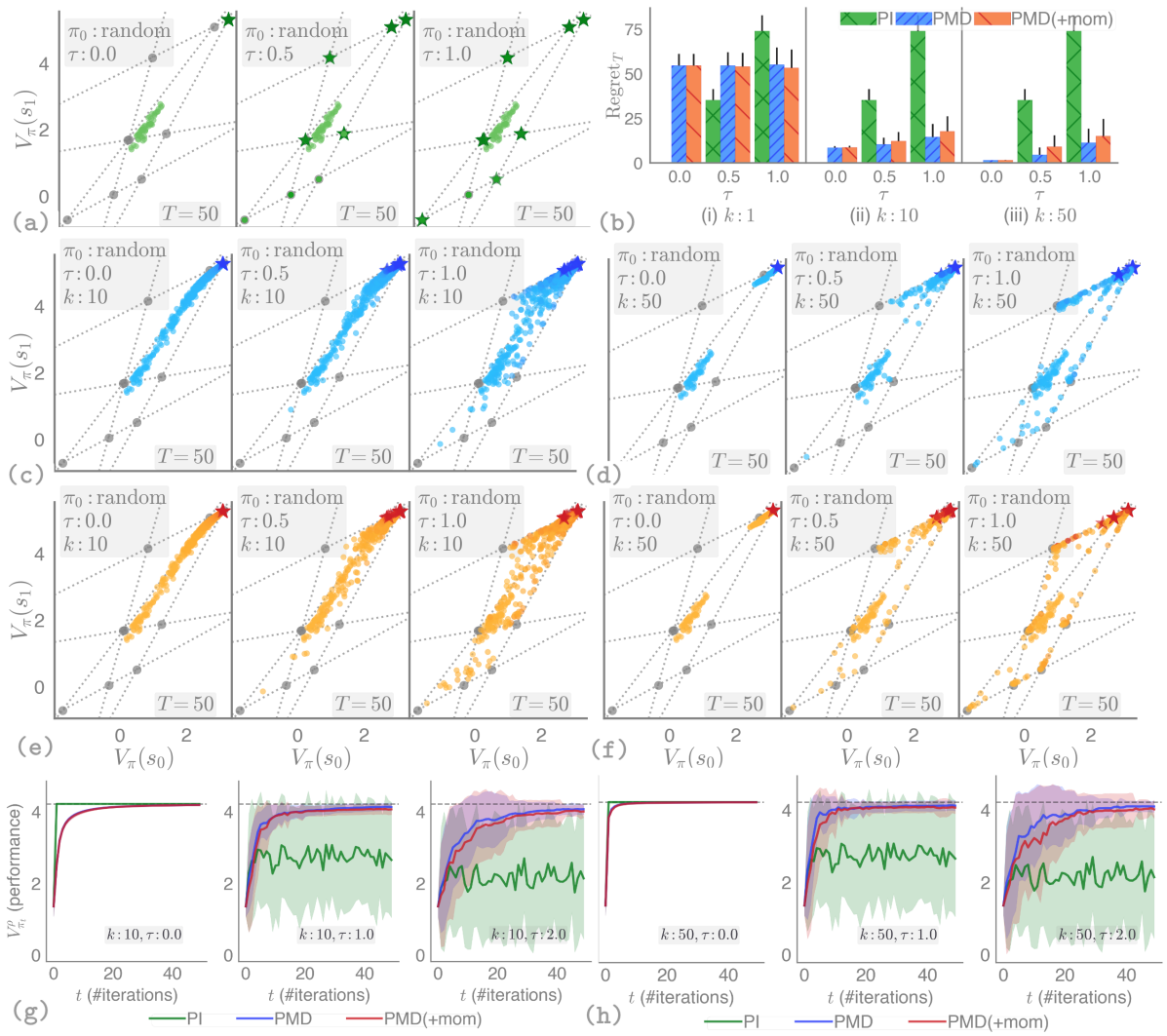


Figure C.11: Compares the statistics of the policy optimization dynamics of PI, PMD and PMD(+mom) subject to variance from random initialization (π_0 : *random_uniform(0,1)*, relative to the error in the inexact critic (τ), and over different levels of policy approximation (k). Results correspond to example (iv) from Sec. 7.1.

D Excitatory-Inhibitory Dynamics in Adaptive Learning and Decision-Making

1 Details for Sec. 3.1

1.1 Nullcline Derivation

The circuit dynamics are given by the system

$$\alpha_{\mathrm{E}}\frac{d\mathbf{r}_{\mathrm{E}}}{dt} \doteq -\mathbf{r}_{\mathrm{E}} + f_{\mathrm{E}}(\mathbf{W}_{\mathrm{EE}}\mathbf{r}_{\mathrm{E}} - \mathbf{W}_{\mathrm{IE}}\mathbf{r}_{\mathrm{I}} + \mathbf{I}_{\mathrm{ext}} + \mathbf{I}_{\mathrm{AMPA}})$$
$$\alpha_{\mathrm{I}}\frac{d\mathbf{r}_{\mathrm{I}}}{dt} \doteq -\mathbf{r}_{\mathrm{I}} + f_{\mathrm{I}}(\mathbf{W}_{\mathrm{EI}}\mathbf{r}_{\mathrm{E}})$$

By definition, the nullcline of the I population is obtained for $\alpha_{\mathrm{I}}\frac{d\mathbf{r}_{\mathrm{I}}}{dt} = \mathbf{0}$, in which case $\mathbf{r}_{\mathrm{I}} = f_{\mathrm{I}}(\mathbf{W}_{\mathrm{EI}}\mathbf{r}_{\mathrm{E}})$. Similiarly, for the E population, if $\alpha_{\mathrm{E}}\frac{d\mathbf{r}_{\mathrm{E}}}{dt} = \mathbf{0}$, then considering that the expected value of $\mathbf{I}_{\mathrm{AMPA}}$ is 0, we have

$$\mathbf{r}_{\mathrm{E}} = f_{\mathrm{E}}(\mathbf{W}_{\mathrm{EE}}\mathbf{r}_{\mathrm{E}} - \mathbf{W}_{\mathrm{IE}}\mathbf{r}_{\mathrm{I}} + \mathbf{I}_{\mathrm{ext}})$$
$$f_{\mathrm{E}}^{-1}(\mathbf{r}_{\mathrm{E}}) = \mathbf{W}_{\mathrm{EE}}\mathbf{r}_{\mathrm{E}} - \mathbf{W}_{\mathrm{IE}}\mathbf{r}_{\mathrm{I}} + \mathbf{I}_{\mathrm{ext}}$$

Substituting $\mathbf{r}_{\mathrm{I}}$

$$f_{\mathrm{E}}^{-1}(\mathbf{r}_{\mathrm{E}}) = \mathbf{W}_{\mathrm{EE}}\mathbf{r}_{\mathrm{E}} - \mathbf{W}_{\mathrm{IE}}f_{\mathrm{I}}(\mathbf{W}_{\mathrm{EI}}\mathbf{r}_{\mathrm{E}}) + \mathbf{I}_{\mathrm{ext}}$$

Denoting $\mathbf{I}_{\text{ext}} = [I_1, I_2]^\top$

$$\begin{bmatrix} f_{\text{E}}^{-1}(r_{E_1}) \\ f_{\text{E}}^{-1}(r_{E_2}) \end{bmatrix} = \begin{bmatrix} 1+w_{\text{EE}} & 1-w_{\text{EE}} \\ 1-w_{\text{EE}} & 1+w_{\text{EE}} \end{bmatrix} \begin{bmatrix} r_{E_1} \\ r_{E_2} \end{bmatrix} - \begin{bmatrix} 1+w_{\text{IE}} & 1-w_{\text{IE}} \\ 1-w_{\text{IE}} & 1+w_{\text{IE}} \end{bmatrix} f_{\text{I}} \left(\begin{bmatrix} 1+w_{\text{EI}} & 1-w_{\text{EI}} \\ 1-w_{\text{EI}} & 1+w_{\text{EI}} \end{bmatrix} \begin{bmatrix} r_{E_1} \\ r_{E_2} \end{bmatrix} \right) + \begin{bmatrix} I_1 \\ I_2 \end{bmatrix}$$

If $f_{\text{I}}(r) \doteq a_{\text{I}} r - b_{\text{I}}$ is the linear activation function of the I population, then

$$\begin{aligned}
\begin{bmatrix} f_{\text{E}}^{-1}(r_{E_1}) \\ f_{\text{E}}^{-1}(r_{E_2}) \end{bmatrix} &= \begin{bmatrix} 1+w_{\text{EE}} & 1-w_{\text{EE}} \\ 1-w_{\text{EE}} & 1+w_{\text{EE}} \end{bmatrix} \begin{bmatrix} r_{E_1} \\ r_{E_2} \end{bmatrix} - a_{\text{I}} \begin{bmatrix} 1+w_{\text{IE}} & 1-w_{\text{IE}} \\ 1-w_{\text{IE}} & 1+w_{\text{IE}} \end{bmatrix} \begin{bmatrix} 1+w_{\text{EI}} & 1-w_{\text{EI}} \\ 1-w_{\text{EI}} & 1+w_{\text{EI}} \end{bmatrix} \begin{bmatrix} r_{E_1} \\ r_{E_2} \end{bmatrix} + \begin{bmatrix} I_1 \\ I_2 \end{bmatrix} \\
&\quad + b_{\text{I}} \begin{bmatrix} 1+w_{\text{IE}} & 1-w_{\text{IE}} \\ 1-w_{\text{IE}} & 1+w_{\text{IE}} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\
&= \left[\begin{bmatrix} 1+w_{\text{EE}} & 1-w_{\text{EE}} \\ 1-w_{\text{EE}} & 1+w_{\text{EE}} \end{bmatrix} - 2a_{\text{I}} \begin{bmatrix} 1+w_{\text{IE}}w_{\text{EI}} & 1-w_{\text{IE}}w_{\text{EI}} \\ 1-w_{\text{IE}}w_{\text{EI}} & 1+w_{\text{IE}}w_{\text{EI}} \end{bmatrix} \right] \begin{bmatrix} r_{E_1} \\ r_{E_2} \end{bmatrix} + \begin{bmatrix} I_1 + 2b_{\text{I}} \\ I_2 + 2b_{\text{I}} \end{bmatrix} \\
&= \left[\begin{bmatrix} 1+w_{\text{EE}} \\ 1-w_{\text{EE}} \end{bmatrix} - 2a_{\text{I}} \begin{bmatrix} 1+w_{\text{IE}}w_{\text{EI}} \\ 1-w_{\text{IE}}w_{\text{EI}} \end{bmatrix} \right] r_{E_1} + \left[\begin{bmatrix} 1-w_{\text{EE}} \\ 1+w_{\text{EE}} \end{bmatrix} - 2a_{\text{I}} \begin{bmatrix} 1-w_{\text{IE}}w_{\text{EI}} \\ 1+w_{\text{IE}}w_{\text{EI}} \end{bmatrix} \right] r_{E_2} \\
&\quad + \begin{bmatrix} I_1 + 2b_{\text{I}} \\ I_2 + 2b_{\text{I}} \end{bmatrix} \\
&= \begin{bmatrix} (1+w_{\text{EE}}) - 2a_{\text{I}}(1+w_{\text{IE}}w_{\text{EI}}) \\ (1-w_{\text{EE}}) - 2a_{\text{I}}(1-w_{\text{IE}}w_{\text{EI}}) \end{bmatrix} r_{E_1} + \begin{bmatrix} (1-w_{\text{EE}}) - 2a_{\text{I}}(1-w_{\text{IE}}w_{\text{EI}}) \\ (1+w_{\text{EE}}) - 2a_{\text{I}}(1+w_{\text{IE}}w_{\text{EI}}) \end{bmatrix} r_{E_2} + \begin{bmatrix} I_1 + 2b_{\text{I}} \\ I_2 + 2b_{\text{I}} \end{bmatrix}
\end{aligned}$$

This yields two equations

$$\begin{aligned}
f_{\text{E}}^{-1}(r_{E_1}) &= [(1+w_{\text{EE}}) - 2a_{\text{I}}(1+w_{\text{IE}}w_{\text{EI}})]\, r_{E_1} + [(1-w_{\text{EE}}) - 2a_{\text{I}}(1-w_{\text{IE}}w_{\text{EI}})]\, r_{E_2} + I_1 + 2b_{\text{I}} \\
f_{\text{E}}^{-1}(r_{E_2}) &= [(1-w_{\text{EE}}) - 2a_{\text{I}}(1-w_{\text{IE}}w_{\text{EI}})]\, r_{E_1} + [(1+w_{\text{EE}}) - 2a_{\text{I}}(1+w_{\text{IE}}w_{\text{EI}})]\, r_{E_2} + I_2 + 2b_{\text{I}}
\end{aligned}$$

To obtain the nullclines, using the first equation, representing the nullcline of r_{E_1}, we express r_{E_2} as a function of r_{E_1}, and using the second equation, representing the nullcline

of r_{E_2}, we express r_{E_1} as a function of r_{E_2}

$$r_{E_2} = \frac{f_{\mathrm{E}}^{-1}(r_{E_1}) - \left[(1+w_{\mathrm{EE}}) - 2a_{\mathrm{I}}(1+w_{\mathrm{IE}}w_{\mathrm{EI}})\right] r_{E_1} - I_1 - 2b_{\mathrm{I}}}{(1-w_{\mathrm{EE}}) - 2a_{\mathrm{I}}(1-w_{\mathrm{IE}}w_{\mathrm{EI}})}$$
$$r_{E_1} = \frac{f_{\mathrm{E}}^{-1}(r_{E_2}) - \left[(1+w_{\mathrm{EE}}) - 2a_{\mathrm{I}}(1+w_{\mathrm{IE}}w_{\mathrm{EI}})\right] r_{E_2} - I_2 - 2b_{\mathrm{I}}}{(1-w_{\mathrm{EE}}) - 2a_{\mathrm{I}}(1-w_{\mathrm{IE}}w_{EI})}$$

1.2 Fixed Point and Stability Analysis

The intersection points of the E nullcline curves are the fixed points of the Wilson-Cowan model used. At the fixed point $^{\mathrm{d}\mathbf{r}}/_{\mathrm{dt}} = \mathbf{0}$. Away from the fixed point, the trajectory of neural responses might be attracted back to, or diverge from the fixed point. These two possibilities define the type of fixed point, i.e., stable or unstable. The stability of a fixed point can be determined by analyzing the gradient vector fields $\mathbf{g}_{\mathrm{E}}(\cdot) \doteq [g_{\mathrm{E}_1}(\cdot), g_{\mathrm{E}_2}(\cdot)]^\top$, $\mathbf{g}_{\mathrm{I}}(\cdot) \doteq [g_{\mathrm{I}_1}(\cdot), g_{\mathrm{I}_2}(\cdot)]^\top$

$$\mathbf{g}_{\mathrm{E}}(\mathbf{r}_{\mathrm{E}}, \mathbf{r}_{\mathrm{I}}) \doteq \left(-\mathbf{r}_{\mathrm{E}} + f_{\mathrm{E}}(\mathbf{W}_{\mathrm{EE}}\mathbf{r}_{\mathrm{E}} - \mathbf{W}_{\mathrm{IE}}\mathbf{r}_{\mathrm{I}} + \mathbf{I}_{\mathrm{ext}})\right)/\alpha_{\mathrm{E}}$$
$$\mathbf{g}_{\mathrm{I}}(\mathbf{r}_{\mathrm{E}}, \mathbf{r}_{\mathrm{I}}) \doteq \left(-\mathbf{r}_{\mathrm{I}} + f_{\mathrm{I}}(\mathbf{W}_{\mathrm{EI}}\mathbf{r}_{\mathrm{E}})\right)/\alpha_{\mathrm{I}}$$

At the fixed point, the I population is given by

$$\mathbf{g}_{\mathrm{I}}(\mathbf{r}_{\mathrm{E}}^*, \mathbf{r}_{\mathrm{I}}^*) = 0 \implies \mathbf{r}_{\mathrm{I}}^* = f_{\mathrm{I}}(\mathbf{W}_{\mathrm{EI}}\mathbf{r}_{\mathrm{E}}^*)$$

Plugging back into the gradient vector field of the E population, we obtain

$$\mathbf{g}_{\mathrm{E}}(\mathbf{r}_{\mathrm{E}}^*) \doteq \left(-\mathbf{r}_{\mathrm{E}}^* + f_{\mathrm{E}}(\mathbf{W}_{\mathrm{EE}}\mathbf{r}_{\mathrm{E}}^* - \mathbf{W}_{\mathrm{IE}}f_{\mathrm{I}}(\mathbf{W}_{\mathrm{EI}}\mathbf{r}_{\mathrm{E}}^*) + \mathbf{I}_{\mathrm{ext}})\right)/\alpha_{\mathrm{E}}$$

The stability of a fixed point can be determined by linearizing the dynamics of the system at the fixed point. The associated Jacobian $\mathbf{J}_{\mathrm{E}}$ of the gradient vector field $\mathbf{g}_{\mathrm{E}}$ is

$$\mathbf{J}_{\mathrm{E}}(\mathbf{r}_{\mathrm{E}}^{*}) \doteq \begin{bmatrix} \frac{\partial g_{\mathrm{E}_1}(\mathbf{r}_{\mathrm{E}}^{*})}{\partial r_{\mathrm{E}_1}} & \frac{\partial g_{\mathrm{E}_1}(\mathbf{r}_{\mathrm{E}}^{*})}{\partial r_{\mathrm{E}_2}} \\ \frac{\partial g_{\mathrm{E}_2}(\mathbf{r}_{\mathrm{E}}^{*})}{\partial r_{\mathrm{E}_1}} & \frac{\partial g_{\mathrm{E}_2}(\mathbf{r}_{\mathrm{E}}^{*})}{\partial r_{\mathrm{E}_2}} \end{bmatrix}$$

where the elements are defined

$$\begin{aligned}
\frac{\partial g_{\mathrm{E}_1}(\mathbf{r}_{\mathrm{E}}^{*})}{\partial r_{\mathrm{E}_1}} &= \Big(-1 + \big((1+w_{\mathrm{EE}}) - (1+w_{\mathrm{IE}})(1+w_{\mathrm{EI}}) f'_{\mathrm{I}_1} - (1-w_{\mathrm{IE}})(1-w_{\mathrm{EI}}) f'_{\mathrm{I}_2} \big) f'_{\mathrm{E}_1} \Big) / \alpha_{\mathrm{E}} \\
\frac{\partial g_{\mathrm{E}_1}(\mathbf{r}_{\mathrm{E}}^{*})}{\partial r_{\mathrm{E}_2}} &= \big((1-w_{\mathrm{EE}}) - (1+w_{\mathrm{IE}})(1-w_{\mathrm{EI}}) f'_{\mathrm{I}_1} - (1-w_{\mathrm{IE}})(1+w_{\mathrm{EI}}) f'_{\mathrm{I}_2} \big) f'_{\mathrm{E}_1} / \alpha_{\mathrm{E}} \\
\frac{\partial g_{\mathrm{E}_2}(\mathbf{r}_{\mathrm{E}}^{*})}{\partial r_{\mathrm{E}_1}} &= \big((1-w_{\mathrm{EE}}) - (1-w_{\mathrm{IE}})(1+w_{\mathrm{EI}}) f'_{\mathrm{I}_1} - (1+w_{\mathrm{IE}})(1-w_{\mathrm{EI}}) f'_{\mathrm{I}_2} \big) f'_{\mathrm{E}_2} / \alpha_{\mathrm{E}} \\
\frac{\partial g_{\mathrm{E}_2}(\mathbf{r}_{\mathrm{E}}^{*})}{\partial r_{\mathrm{E}_2}} &= \Big(-1 + \big((1+w_{\mathrm{EE}}) - (1-w_{\mathrm{IE}})(1-w_{\mathrm{EI}}) f'_{\mathrm{I}_1} - (1+w_{\mathrm{IE}})(1+w_{\mathrm{EI}}) f'_{\mathrm{I}_2} \big) f'_{\mathrm{E}_2} \Big) / \alpha_{\mathrm{E}}
\end{aligned}$$

Here, the terms $f'_{\mathrm{E}_1}, f'_{\mathrm{E}_2}, f'_{\mathrm{I}_1}, f'_{\mathrm{I}_2}$ represent derivatives of the activation functions f_{E} and f_{I} at the respective points.

The stable fixed points correspond to the negative eigenvalues, while unstable points correspond to at least one positive eigenvalue. The sign of the eigenvalues is determined by the interaction between the E and I populations.

The stability of the fixed points is determined by connectivity weights $w_{\mathrm{EE}}, w_{\mathrm{EI}}, w_{\mathrm{IE}}$. Specifically, the fixed point is stable if:

$$\frac{\partial g_{\mathrm{E}_1}(\mathbf{r}_{\mathrm{E}}^{*})}{\partial r_{\mathrm{E}_1}} \frac{\partial g_{\mathrm{E}_2}(\mathbf{r}_{\mathrm{E}}^{*})}{\partial r_{\mathrm{E}_2}} > \frac{\partial g_{\mathrm{E}_1}(\mathbf{r}_{\mathrm{E}}^{*})}{\partial r_{\mathrm{E}_2}} \frac{\partial g_{\mathrm{E}_2}(\mathbf{r}_{\mathrm{E}}^{*})}{\partial r_{\mathrm{E}_1}}$$

Let

$$\Delta_{11} \doteq (1 + w_{\mathrm{EE}}) - (1+w_{\mathrm{IE}})(1+w_{\mathrm{EI}}) f'_{\mathrm{I}_1} - (1-w_{\mathrm{IE}})(1-w_{\mathrm{EI}}) f'_{\mathrm{I}_2}$$

$$\Delta_{12} \doteq (1 - w_{\mathrm{EE}}) - (1+w_{\mathrm{IE}})(1-w_{\mathrm{EI}}) f'_{\mathrm{I}_1} - (1-w_{\mathrm{IE}})(1+w_{\mathrm{EI}}) f'_{\mathrm{I}_2}$$

$$\Delta_{21} \doteq (1 - w_{\mathrm{EE}}) - (1-w_{\mathrm{IE}})(1+w_{\mathrm{EI}}) f'_{\mathrm{I}_1} - (1+w_{\mathrm{IE}})(1-w_{\mathrm{EI}}) f'_{\mathrm{I}_2}$$

$$\Delta_{22} \doteq (1 + w_{\mathrm{EE}}) - (1-w_{\mathrm{IE}})(1-w_{\mathrm{EI}}) f'_{\mathrm{I}_1} - (1+w_{\mathrm{IE}})(1+w_{\mathrm{EI}}) f'_{\mathrm{I}_2}$$

Using this notation

$$\frac{\partial g_{\mathrm{E}_1}(\mathbf{r}^*_{\mathrm{E}})}{\partial r_{\mathrm{E}_1}} = \frac{\Delta_{11} f'_{\mathrm{E}_1} - 1}{\alpha_{\mathrm{E}}} \qquad \frac{\partial g_{\mathrm{E}_1}(\mathbf{r}^*_{\mathrm{E}})}{\partial r_{\mathrm{E}_2}} = \frac{\Delta_{12} f'_{\mathrm{E}_1}}{\alpha_{\mathrm{E}}}$$

$$\frac{\partial g_{\mathrm{E}_2}(\mathbf{r}^*_{\mathrm{E}})}{\partial r_{\mathrm{E}_1}} = \frac{\Delta_{21} f'_{\mathrm{E}_2}}{\alpha_{\mathrm{E}}} \qquad \frac{\partial g_{\mathrm{E}_2}(\mathbf{r}^*_{\mathrm{E}})}{\partial r_{\mathrm{E}_2}} = \frac{\Delta_{22} f'_{\mathrm{E}_2} - 1}{\alpha_{\mathrm{E}}}$$

the **condition for stability** is

$$\begin{aligned} (f'_{\mathrm{E}_1}\Delta_{11} - 1)(f'_{\mathrm{E}_2}\Delta_{22} - 1) &> f'_{\mathrm{E}_1}\Delta_{12} f'_{\mathrm{E}_2}\Delta_{21} \\ \implies \frac{(f'_{\mathrm{E}_1}\Delta_{11} - 1)(f'_{\mathrm{E}_2}\Delta_{22} - 1)}{f'_{\mathrm{E}_1}\Delta_{12} f'_{\mathrm{E}_2}\Delta_{21}} &> 1 \\ \implies \frac{(f'_{\mathrm{E}_1}\Delta_{11} - 1)}{f'_{\mathrm{E}_1}\Delta_{12}} \frac{(f'_{\mathrm{E}_2}\Delta_{22} - 1)}{f'_{\mathrm{E}_2}\Delta_{21}} &> 1 \end{aligned} \tag{D.1}$$

Nullcline Analysis From the nullcline equations

$$\mathbf{r}_{\mathrm{E}} = f_{\mathrm{E}}(\mathbf{W}_{\mathrm{EE}}\mathbf{r}_{\mathrm{E}} - \mathbf{W}_{\mathrm{IE}} f_{\mathrm{I}}(\mathbf{W}_{\mathrm{EI}}\mathbf{r}_{\mathrm{E}}) + \mathbf{I})$$

using the E_1 nullcline, we can take the derivative of r_{E_1} with respect to r_{E_2}

$$\begin{aligned}
\left(\frac{\mathrm{d}r_{\mathrm{E}_1}}{\mathrm{d}r_{\mathrm{E}_2}}\right)_{\mathrm{E}_1\text{-nullcline}} &\doteq f'_{\mathrm{E}_1}(\Delta_{11}\frac{\mathrm{d}r_{\mathrm{E}_1}}{\mathrm{d}r_{\mathrm{E}_2}} + \Delta_{12}) \\
(1 - f'_{\mathrm{E}_1}\Delta_{11})\left(\frac{\mathrm{d}r_{\mathrm{E}_1}}{\mathrm{d}r_{\mathrm{E}_2}}\right)_{\mathrm{E}_1\text{-nullcline}} &\doteq f'_{\mathrm{E}_1}\Delta_{12} \\
\left(\frac{\mathrm{d}r_{\mathrm{E}_1}}{\mathrm{d}r_{\mathrm{E}_2}}\right)_{\mathrm{E}_1\text{-nullcline}} &\doteq \frac{f'_{\mathrm{E}_1}\Delta_{12}}{(1 - f'_{\mathrm{E}_1}\Delta_{11})}
\end{aligned}$$

In the phase plane of the E populations we can obtain the slope along the E_1 nullcline as

$$\left(\frac{\mathrm{d}r_{\mathrm{E}_2}}{\mathrm{d}r_{\mathrm{E}_1}}\right)_{\mathrm{E}_1\text{-nullcline}} \doteq \frac{(1 - f'_{\mathrm{E}_1}\Delta_{11})}{f'_{\mathrm{E}_1}\Delta_{12}} \tag{D.2}$$

Similarly, we can obtain the slope along the E_2 nullcline. Taking the derivative of r_{E_2} with respect to r_{E_1}

$$\begin{aligned}
\left(\frac{\mathrm{d}r_{\mathrm{E}_2}}{\mathrm{d}r_{\mathrm{E}_1}}\right)_{\mathrm{E}_2\text{-nullcline}} &\doteq f'_{\mathrm{E}_2}(\Delta_{22}\frac{\mathrm{d}r_{\mathrm{E}_2}}{\mathrm{d}r_{\mathrm{E}_1}} + \Delta_{21}) \\
(1 - f'_{\mathrm{E}_2}\Delta_{22})\left(\frac{\mathrm{d}r_{\mathrm{E}_2}}{\mathrm{d}r_{\mathrm{E}_1}}\right)_{\mathrm{E}_2\text{-nullcline}} &\doteq f'_{\mathrm{E}_2}\Delta_{21} \\
\left(\frac{\mathrm{d}r_{\mathrm{E}_2}}{\mathrm{d}r_{\mathrm{E}_1}}\right)_{\mathrm{E}_2\text{-nullcline}} &\doteq \frac{f'_{\mathrm{E}_2}\Delta_{21}}{(1 - f'_{\mathrm{E}_2}\Delta_{22})}
\end{aligned} \tag{D.3}$$

Then, we find that

$$\frac{\left(\frac{\mathrm{d}r_{\mathrm{E}_2}}{\mathrm{d}r_{\mathrm{E}_1}}\right)_{\mathrm{E}_1\text{-nullcline}}}{\left(\frac{\mathrm{d}r_{\mathrm{E}_2}}{\mathrm{d}r_{\mathrm{E}_1}}\right)_{\mathrm{E}_2\text{-nullcline}}} = \frac{(1 - f'_{\mathrm{E}_1}\Delta_{11})}{f'_{\mathrm{E}_1}\Delta_{12}}\frac{(1 - f'_{\mathrm{E}_2}\Delta_{22})}{f'_{\mathrm{E}_2}\Delta_{21}} = \frac{(f'_{\mathrm{E}_1}\Delta_{11} - 1)}{f'_{\mathrm{E}_1}\Delta_{12}}\frac{(f'_{\mathrm{E}_2}\Delta_{22} - 1)}{f'_{\mathrm{E}_2}\Delta_{21}}$$

Similarly, we can obtain the slope along the E_2 nullcline from Eq.D.3 as

$$\left(\frac{\mathrm{d}r_{\mathrm{E}_1}}{\mathrm{d}r_{\mathrm{E}_2}}\right)_{\mathrm{E}_2\text{-nullcline}} \doteq \frac{(1 - f'_{\mathrm{E}_2}\Delta_{22})}{f'_{\mathrm{E}_1}\Delta_{21}}$$

Then, together with Eq.D.2 we find that

$$\frac{\left(\frac{\mathrm{d}r_{\mathrm{E}_1}}{\mathrm{d}r_{\mathrm{E}_2}}\right)_{\mathrm{E}_2\text{-nullcline}}}{\left(\frac{\mathrm{d}r_{\mathrm{E}_1}}{\mathrm{d}r_{\mathrm{E}_2}}\right)_{\mathrm{E}_1\text{-nullcline}}} = \frac{(1-f'_{\mathrm{E}_2}\Delta_{22})}{f'_{\mathrm{E}_1}\Delta_{21}}\frac{(1-f'_{\mathrm{E}_1}\Delta_{11})}{f'_{\mathrm{E}_1}\Delta_{12}} = \frac{(f'_{\mathrm{E}_2}\Delta_{22}-1)}{f'_{\mathrm{E}_2}\Delta_{21}}\frac{(f'_{\mathrm{E}_1}\Delta_{11}-1)}{f'_{\mathrm{E}_1}\Delta_{12}}$$

where the right-hand side is the left-hand side of the stability condition in Eq. D.1, thus yielding

$$\frac{\left(\frac{\mathrm{d}r_{\mathrm{E}_1}}{\mathrm{d}r_{\mathrm{E}_2}}\right)_{\mathrm{E}_2\text{-nullcline}}}{\left(\frac{\mathrm{d}r_{\mathrm{E}_1}}{\mathrm{d}r_{\mathrm{E}_2}}\right)_{\mathrm{E}_1\text{-nullcline}}} = \frac{\left(\frac{\mathrm{d}r_{\mathrm{E}_2}}{\mathrm{d}r_{\mathrm{E}_1}}\right)_{\mathrm{E}_1\text{-nullcline}}}{\left(\frac{\mathrm{d}r_{\mathrm{E}_2}}{\mathrm{d}r_{\mathrm{E}_1}}\right)_{\mathrm{E}_2\text{-nullcline}}} > 1$$

Therefore, at the stable fixed point, the equation describe the **relative alignment of the nullcline slopes**.

1.3 Noise generation

The noise process $\mathbf{I}_{\mathrm{AMPA}}$ follows an Ornstein-Uhlenbeck (OU)

$$\alpha_{\mathrm{AMPA}}\frac{\mathrm{d}\mathbf{I}_{\mathrm{AMPA}}}{\mathrm{dt}} \doteq -\mathbf{I}_{\mathrm{AMPA}} + \mathcal{N}(\mu=0,\sigma=0.02)$$

where $\alpha_{\mathrm{AMPA}} = 1$ is the characteristic timescale of decay, and $\mathcal{N}(\mu = 0, \sigma = 0.02)$ is a standard Gaussian random variable, where μ is the noise mean, and σ is the noise amplitude.

1.4 Evidence Integration

The trajectories in Fig. 6.1 are generated using the evidence integration simulation loop in Alg. 14 for each trial. Numerical simulations use the Euler method with a 1 ms time step.

Algorithm 14 Evidence Integration in the Wilson-Cowan Mean-Field Model

Input: stimuli $\mathbf{I}_{\text{ext}}$, noise $\eta \sim \mathcal{N}(\mu = 0, \sigma = 0.02)$, activation functions $f_{\text{E}}, f_{\text{I}}$, connectivity weight matrices $\mathbf{W}_{\text{EE}}, \mathbf{W}_{\text{IE}}, \mathbf{W}_{\text{EI}}$

Initialize: activity $\mathbf{r}_{\text{E}}, \mathbf{r}_{\text{I}}, \mathbf{I}_{\text{AMPA}}$ to $\mathbf{0}$.

for each timestep $t = 1$ to $T - 1$ **do**

  $\mathbf{I}_{\text{ext}}$

  $\eta \sim \mathcal{N}(\mu = 0, \sigma = 0.02)$ {sample noise}

  $\Delta\mathbf{I}_{\text{AMPA}} \leftarrow (-\mathbf{I}_{\text{AMPA}} + \eta)/\alpha_{\text{AMPA}}$ {update noise process update}

  $\mathbf{r}_{\text{E}}\mathbf{W}_{\text{EE}}, \mathbf{r}_{\text{E}}\mathbf{W}_{\text{EI}}, \mathbf{r}_{\text{I}}\mathbf{W}_{\text{IE}}$ {evaluate synaptic inputs}

  $\Delta\mathbf{r}_{\text{E}} \leftarrow (-\mathbf{r}_{\text{E}} + f_{\text{E}}(\mathbf{r}_{\text{E}}\mathbf{W}_{\text{EE}} - \mathbf{r}_{\text{I}}\mathbf{W}_{\text{IE}} + \mathbf{I}_{\text{ext}} + \mathbf{I}_{\text{AMPA}}))/\alpha_{\text{E}}$ {compute rate updates}

  $\Delta\mathbf{r}_{\text{I}} \leftarrow (-\mathbf{r}_{\text{I}} + f_{\text{I}}(\mathbf{r}_{\text{E}}\mathbf{W}_{\text{EI}}))/\alpha_{\text{I}}$

  $\mathbf{r}_{\text{E}} \leftarrow \mathbf{r}_{\text{E}} + \Delta\mathbf{r}_{\text{E}} \cdot \Delta_t$ {update activity using Euler's method}

  $\mathbf{r}_{\text{I}} \leftarrow \mathbf{r}_{\text{I}} + \Delta\mathbf{r}_{\text{I}} \cdot \Delta_t$

  $\mathbf{I}_{\text{AMPA}} \leftarrow \mathbf{I}_{\text{AMPA}} + \Delta\mathbf{I}_{\text{AMPA}} \cdot \Delta_t$ {update noise process}

return $\mathbf{r}_{\text{E}}, \mathbf{r}_{\text{I}}$ (activity of excitatory and inhibitory populations)

1.5 Parameters

Table. D.1 provides the parameters that were used in the derivations and implementation of the numerical simulations.

Table D.1: Parameters for the Wilson-Cowan mean-field model

Parameter	Value	Description
$f_{\text{E}}(x)$	$\doteq \frac{(ax-b)}{(1-\exp(-d(ax-b)))}$	exponential linear unit (ELU) activation function; $a=1, b=0, d=1$
$f_{\text{I}}(x)$	$\doteq ax-b$	linear activation function, where $a=1, b=0, d=1$
c	$\in [-1, 1]$	stimulus coherence
$\mathbf{I}_{\text{ext}}$	1	baseline signal strength
$t_{\textbf{on}}$	20	stimulus onset
$t_{\textbf{off}}$	30	stimulus offset
time	50	total duration of simulation [ms]
Δ_t	1	simulation time step [ms]
seeds	10	no. of runs

2 Details for Sec. 3.2

2.1 Agent-Environment Interaction

Alg. 15 presents the agent-environment interaction with the task distribution of bandit environments.

Algorithm 15 Learning for E/I in Value-Based Control: Agent-Environment Interaction

Input: precision w_{E}, entropy strength τ, value learning rate α, σ reward variance
Initialize: bandit environment with arm probabilities $\{p_1^0, p_2^0\} = \{0.2, 0.8\}$, policy π^0, value estimates $Q_{\mathrm{E}}^0, Q_{\mathrm{I}}^0$ to 0
for each step t in 1 to T
    $A_t \sim \pi^t$ {select action}
    $R_t \sim \mathrm{TwoArmBandit}(\{p_1^t, p_2^t\}, \sigma)$ {receive reward from environment}
    $Q_{\mathrm{E}}^{t+1}[A_t] = Q_{\mathrm{E}}^t[A_t] + \alpha(R_t - Q_{\mathrm{E}}^t[A_t]) - w_{\mathrm{I}} Q_{\mathrm{I}}^t[A_t])$ {update $Q_{\mathrm{E}}, Q_{\mathrm{I}}$ using credit assignment with R_t, A_t}
    $Q_{\mathrm{I}}^{t+1}[A_t] = Q_{\mathrm{I}}^t[A_t] + \alpha((R_t - Q_{\mathrm{E}}^t[A_t]) - Q_{\mathrm{I}}^t[A_t])$
    $\pi^{t+1}(a) \propto \pi^t(a)^{1-\tau} \exp\big(w_{\mathrm{E}}\, Q_{\mathrm{E}}^{t+1}(a)\big), \forall a \in \mathcal{A}$ {Update policy with closed-form solution of PMD with entropy regularization}
    if not $t \% k$ **then**
        $\{p_1^{t+1}, p_2^{t+1}\} \doteq \{p_2^t, p_1^t\}$ {reverse bandit arm reward probabilities every $k = T/K$ timesteps (i.e. total timesteps/no.tasks)}
return

2.2 Parameters

Table. D.2 provides the parameters that were used in the implementation of the numerical simulations.

Table D.2: Parameters for Value-based Control

Parameter	Value	Description
T	4000	total no. of timesteps in experiment
K	4	no. of tasks in the experiment
task distribution	Bernoulli($p_1 = 1 - p_2$, $p_1 \sim \mathcal{U}(\{0.2, 0.8\})$	
no. trials/seeds	50	no. of runs for averaging results
w_{I}	$\in [0.0, 0.9, 1, 1.1, -1, -5, -10] \times 10^{-2}$	inhibitory connectivity parameter
w_{E}	$\in [0.2, 0.05]$	precision parameter
α	0.01	value function learning rate
τ	0.01	policy exploration strength
σ	0.8	reward variance

3 Details for Sec. 3.3

3.1 Initialization details

The parameters of the RNN networks were initialized following Li et al. [2023]. We include a brief summary for convenience.

- **Standard/vanilla RNNs**: The weights of are sampled from a uniform distribution $\mathcal{U}([1/\sqrt{n}, 1/\sqrt{n}])$, where $n = n_{\mathrm{E}} + n_{\mathrm{I}}$ is the total number of hidden units. Biases were initialized to 0.

- **Column Excitation-Inhibition (ColEI) RNNs**: The weights in each column are sampled from the (positive(excitatory)/negative(inhibitory)) exponential distribution. The distribution means were set so that the sum of excitatory and inhibitory inputs to each neuron was balanced in expectation. Recurrent weights were adjusted to constrain their spectral radius, by setting $\mathbf{W}_{\mathrm{rec}} = \rho/\rho_0 \mathbf{W}_{\mathrm{rec}}$, where ρ_0 is the spectral radius of $\mathbf{W}_{\mathrm{rec}}$ before the adjustment, and ρ is tuned for each problem. Biases are initialized to 0, except for the first layer, where they are set to centre the activations.

- **Dale's Artificial Neural Networks (DANNs)**: all entries of W_{EE} matrices are sampled from an exponential distribution with variance $1/3(n-1)$, where $n = n_{\mathrm{E}} + n_{\mathrm{I}}$ is the total number of hidden units. Each row of $\mathbf{W}_{\mathrm{IE}}$ is initialized as the mean row of $\mathbf{W}_{\mathrm{EE}}$, and rows of $\mathbf{W}_{\mathrm{EI}}$ are sampled from an exponential distribution and normalized so each

row sums to 1. This is to balance excitatory and inhibitory inputs and so that the variance is the same as RNNs. Biases are initialized to 0.

3.2 Parameters

Tables D.3 and D.4 provide the parameters that were used in the implementation of the numerical simulations in Setting (i) and (ii), respectively.

Table D.3: Parameters for RNN Networks in Setting(i)

Parameter	Value	Description
n_{E}	100	no. of excitatory units
n_{I}	25	no. of inhibitory units
n_{out}	2	output size
n_{in}	2	input size
c	$\in [-0.2, 0.2]$	stimulus coherence
x	3.2	stimulus strength
$\bar{x}$	3.2	background signal
t_{on}	10	stimulus onset
t_{off}	21	stimulus offset
ρ	1.2	spectral radius for ColEI, best of $[0.8, 1, 1.2, 1.4, \mathrm{None}]$
$\eta_{t\geq 0}$	$\sim \mathcal{N}(\mu = 0, \sigma = 0.05)$	noise sampled from a Gaussian with mean μ and std. σ
$f_{\mathrm{out}}(x)$	x	identity function
batch size	200	
no. timesteps/trial	60	
$\bar{\mathrm{y}}$	0.2	baseline target value
$\mathrm{y}_{\mathrm{max}}$	1	target value for the correct choice post-stimulus
δ	0.25	decision threshold
gc	$\in [1, 5, 10]$	gradient clipping value
lr	$\in [1, 2, 4, 6, 10] \times 10^{-4}$	learning rate
seeds	50	no. of runs for averaging results

4 Additional Results for Sec. E/I in Recurrent Neural Networks

Table D.4: Parameters for RNN Networks in Setting(ii)

Parameter	Value	Description
n_{E}	48	number of excitatory units
n_{I}	9	number of inhibitory units
$n_{\text{out}} = \mathcal{A}$	2	output size, number of actions
n_{in}	4	input size
ρ	1	spectral radius for ColEI best of $[0.8, 1, 1.2, 1.4]$
$f_{\text{out}}(\mathbf{y})$	$\exp \mathbf{y}[a] / \sum_{a' \in \mathcal{A}} \exp \mathbf{y}[a']$	softmax function
γ	0.8	discount factor
batch size	256	
T	100	number of timesteps (trials) in an episode
episode length	100	
no. episodes	400	number of episodes
gc	$\in [1, 5, 10]$	gradient clipping value
lr	$\in [0.6, 0.8, 1, 1.5, 2] \times 10^{-3}$	learning rate
trials unrolled	100	number of trials the RNN is unrolled at test time
seeds	20	no. of runs for averaging results
ν	$\sim \mathcal{U}([0, 0.5])$	volatility for the rewarding probabilities drift $\sim \mathcal{D}_{\text{restless}}$ (sampled differently for each episode)

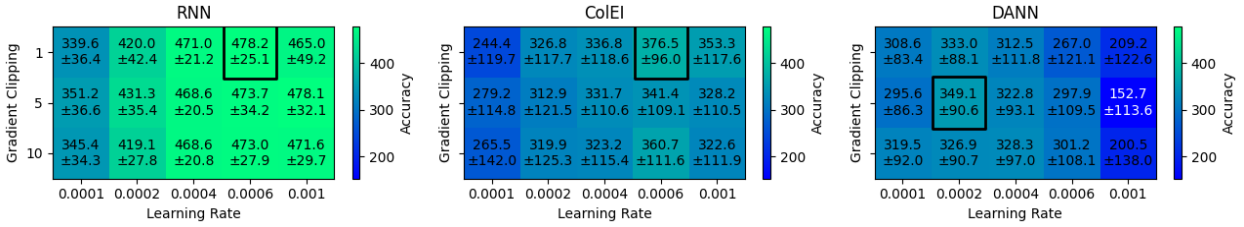


Figure D.1: Parameter Sensitivity Analysis: Heatmaps for Setting (i). Cells denote the mean and standard deviation (std.) of testing accuracy. Bordered cell represent the maximum of the score = mean - std.

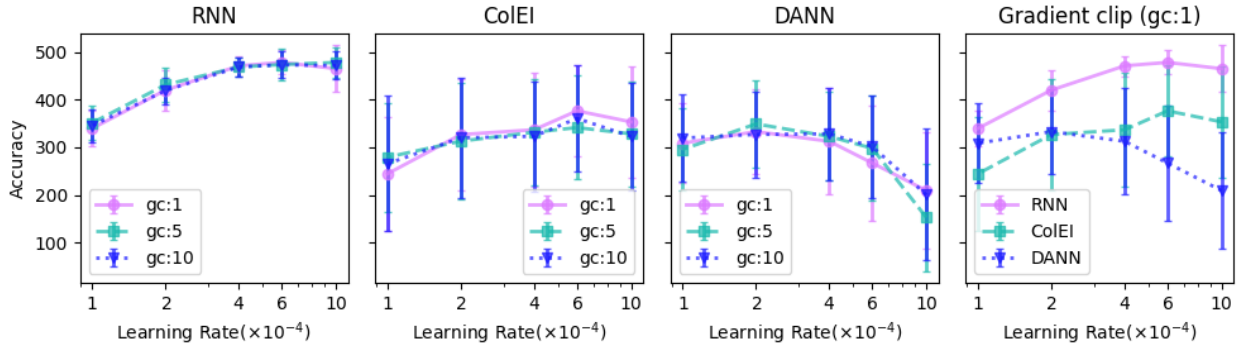


Figure D.2: Parameter Sensitivity Analysis: U-shaped curves for Setting (i). Gradient clipping value is denoted with gc. The first three subplots represent sensitivity plots for each RNN network architecture. The last subplot compares their U-shaped curves for gradient clipping value $\text{gc} : 1$

.

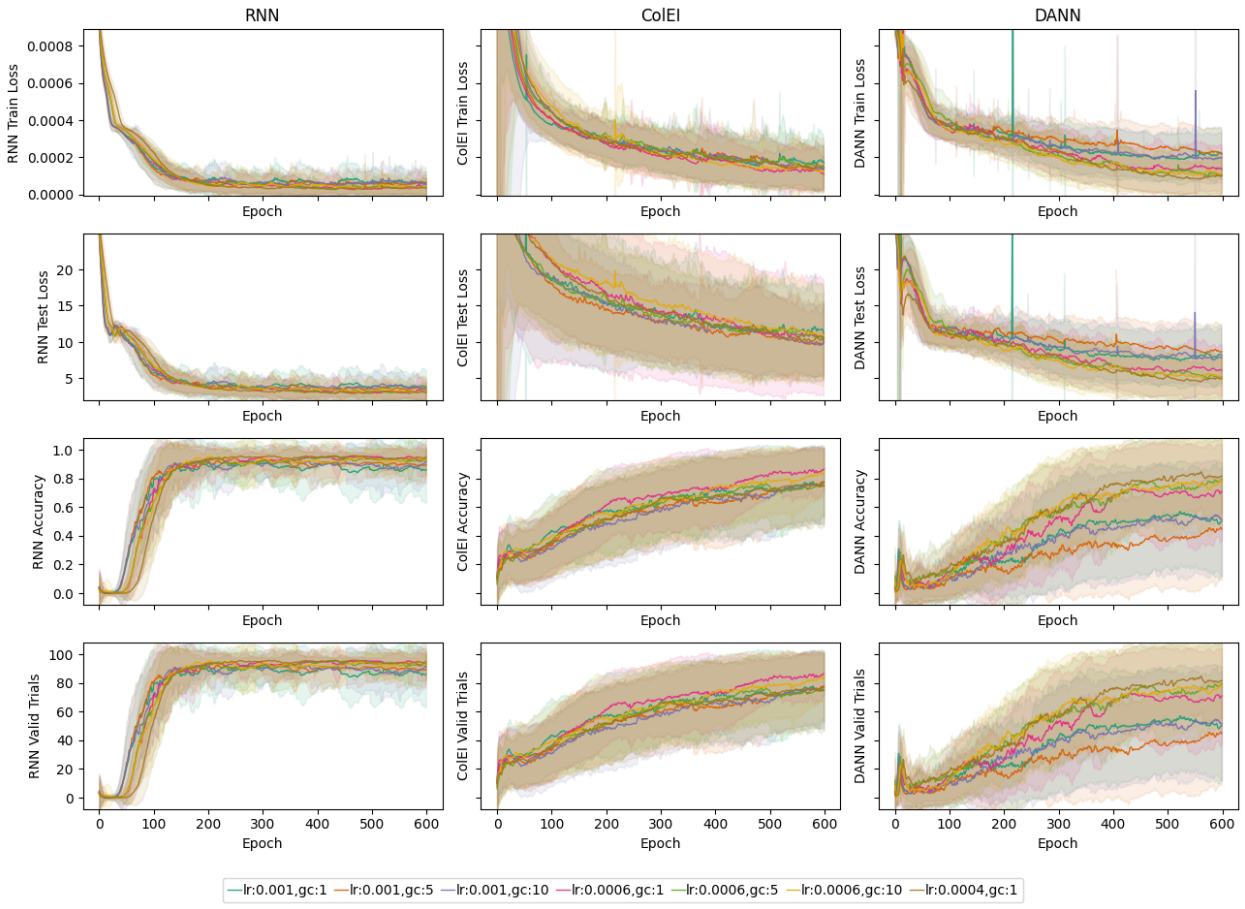


Figure D.3: Parameter Sensitivity Analysis: Learning curves for Setting (i). Columns correspond to RNN architectures. Rows denote training (top row) and testing (other rows) metrics.

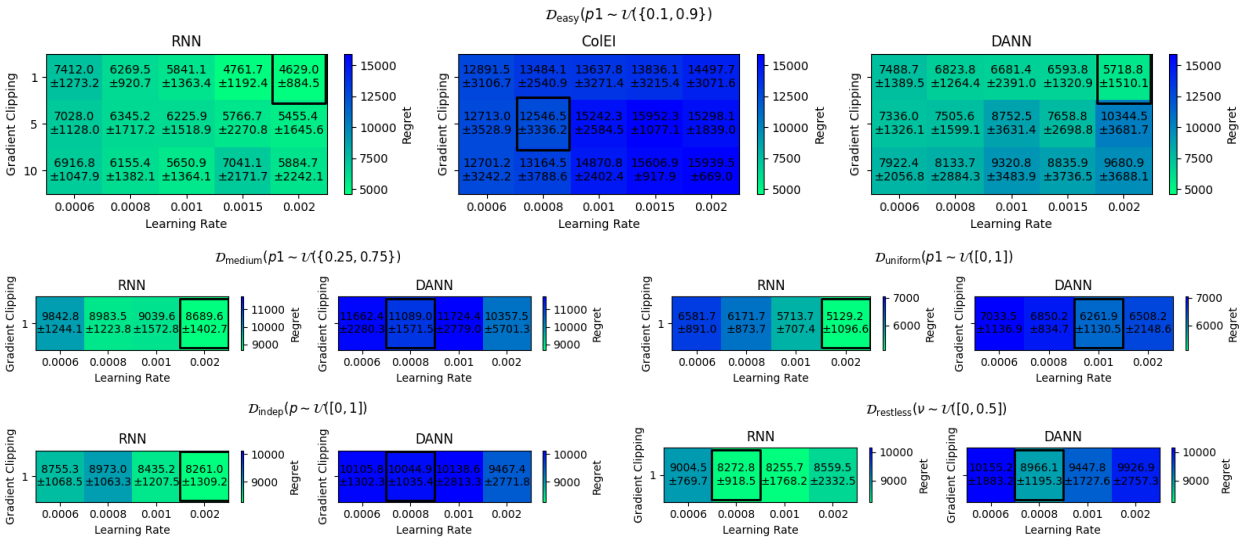


Figure D.4: Parameter Sensitivity Analysis for Setting (ii). Each plot represents a different task distribution. Cells denote the mean and standard deviation (std.) of the average cumulative regret. Bordered cell represent the minimum of the score = mean + std.

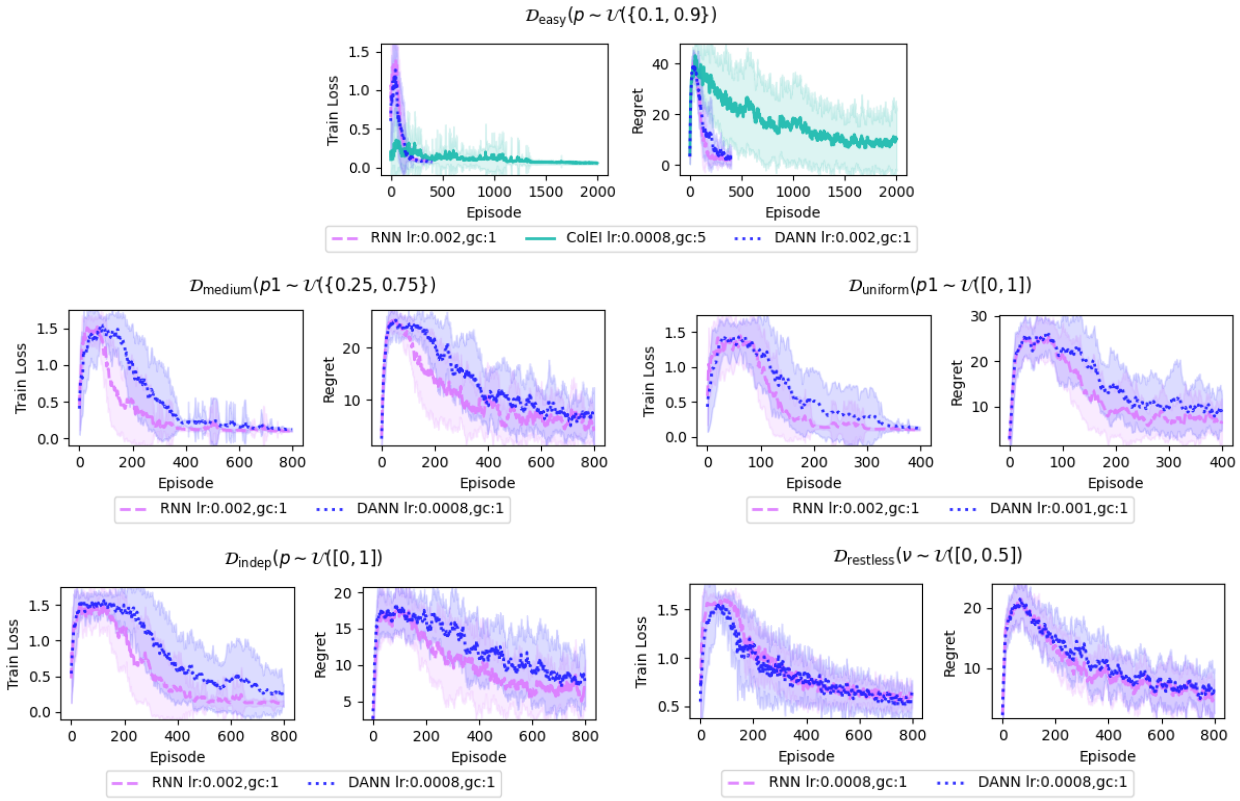


Figure D.5: Performance of RNNs on Setting (ii): Task distribution denoted on top of each subplot. The first subplot is run for $2K$ episodes to illustrate the performance of ColEI, which is then excluded from the other comparisons for the other task distributions. (Left) Training loss per episodes of experience. (Right) Evaluation performance as average cumulative regret.

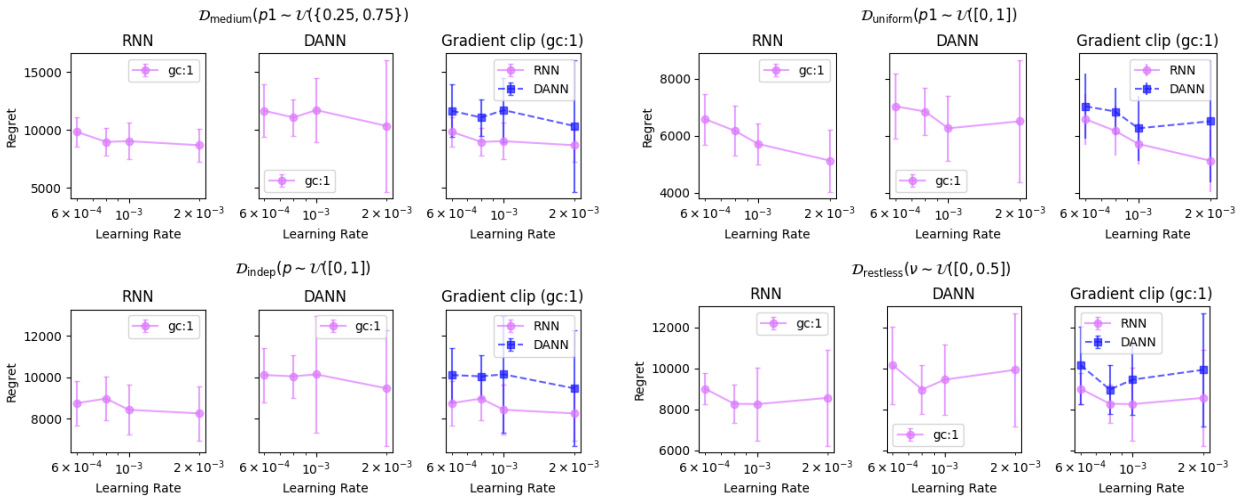


Figure D.6: Parameter Sensitivity Analysis: U-shaped curves for Setting (ii). Gradient clipping value is denoted with gc. Each plot represents a different task distribution. The first two subplots represent sensitivity plots for each RNN network architecture. The last subplot compares their U-shaped curves for gradient clipping value $\text{gc} : 1$

.

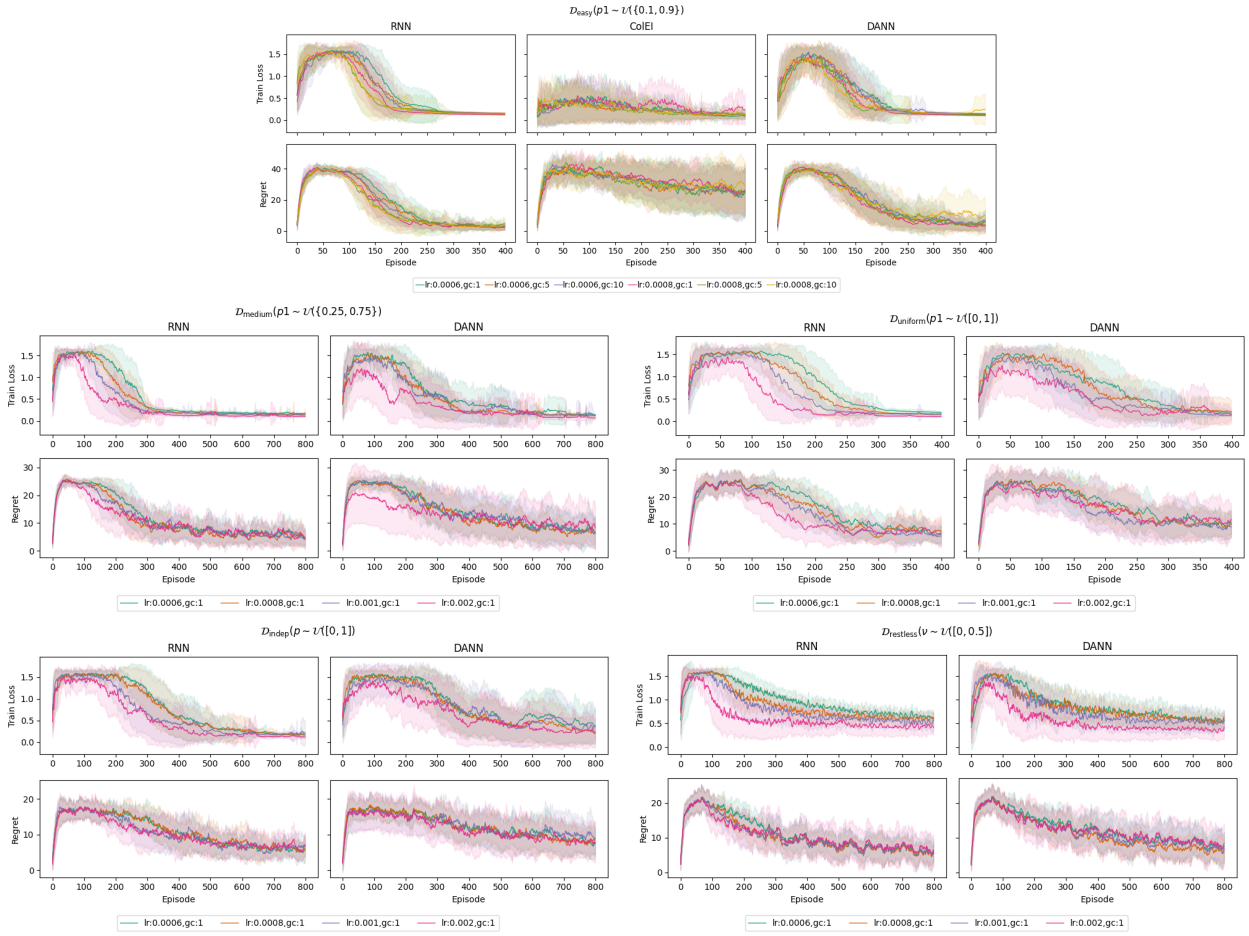


Figure D.7: Parameter Sensitivity Analysis: Learning Curves for Setting (ii): Each plot represents a different task distribution. Columns correspond to RNN architectures. Rows denote training (top row) and testing (Bottom row) metrics.

Bibliography

R. Abachi, M. Ghavamzadeh, and A. Farahmand. Policy-aware model learning for policy gradient methods. *CoRR*, abs/2003.00030, 2020. URL https://arxiv.org/abs/2003.00030.

A. Abdolmaleki, J. T. Springenberg, Y. Tassa, R. Munos, N. Heess, and M. A. Riedmiller. Maximum a posteriori policy optimisation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=S1ANxQW0b.

D. Abel, A. Barreto, B. V. Roy, D. Precup, H. van Hasselt, and S. Singh. A definition of continual reinforcement learning, 2023.

W. C. Abraham. Metaplasticity: tuning synapses and networks for plasticity. *Nature Reviews Neuroscience*, 9(5):387–387, 2008.

A. Agarwal, S. M. Kakade, J. D. Lee, and G. Mahajan. Optimality and approximation with policy gradient methods in markov decision processes. *CoRR*, abs/1908.00261, 2019. URL http://arxiv.org/abs/1908.00261.

C. Alfano, R. Yuan, and P. Rebeschini. A novel framework for policy mirror descent with general parameterization and linear convergence, 2024.

E. Altman. *Constrained Markov Decision Processes*. Chapman and Hall, 1999.

S.-i. Amari. Natural Gradient Works Efficiently in Learning. *Neural Computation*, 10(2): 251–276, 02 1998. ISSN 0899-7667. doi: 10.1162/089976698300017746. URL https://doi.org/10.1162/089976698300017746.

S.-i. Amari. *Information Geometry and Its Applications*. Springer Publishing Company, Incorporated, 1st edition, 2016. ISBN 4431559779.

N. Anand and D. Precup. Preferential temporal difference learning, 2021.

D. G. M. Anderson. Iterative procedures for nonlinear integral equations. *J. ACM*, 12 (4):547–560, 1965. doi: 10.1145/321296.321305. URL https://doi.org/10.1145/321296.321305.

T. W. Archibald, K. I. M. McKinnon, and L. C. Thomas. On the generation of markov decision processes. 46(3):354–361, 2024/04/28/ 1995.

J. A. Arjona-Medina, M. Gillhofer, M. Widrich, T. Unterthiner, and S. Hochreiter. Rudder: Return decomposition for delayed rewards. *ArXiv*, abs/1806.07857, 2019.

D. Aronov, R. Nevers, and D. W. Tank. Mapping of a non-spatial dimension by the hippocampal–entorhinal circuit. *Nature*, 543(7647):719–722, 2017.

P. Bacon, J. Harb, and D. Precup. The option-critic architecture. *ArXiv*, abs/1609.05140, 2017.

A. D. Baddeley and G. Hitch. *Working Memory*, volume 8, pages 47–89. Academic Press, 1974.

L. C. Baird. Residual algorithms: Reinforcement learning with function approximation. In *International Conference on Machine Learning*, 1995. URL https://api.semanticscholar.org/CorpusID:621595.

J. Bakic, M. Jepma, R. De Raedt, and G. Pourtois. Effects of positive mood on probabilistic learning: behavioral and electrophysiological correlates. *Biol Psychol*, 103:223–232, Dec 2014.

B. W. Balleine. Neural bases of food-seeking: affect, arousal and reward in corticostriatolimbic circuits. *Physiol Behav*, 86(5):717–730, Dec 2005.

S. Banach. Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fundamenta Mathematicae*, 3(1):133–181, 1922. URL http://eudml.org/doc/213289.

A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh. Clustering with bregman divergences. *J. Mach. Learn. Res.*, 6:1705–1749, dec 2005. ISSN 1532-4435. URL https://api.semanticscholar.org/CorpusID:8197416.

X. Bao, E. Gjorgieva, L. K. Shanahan, J. D. Howard, T. Kahnt, and J. A. Gottfried. Grid-like neural representations support olfactory navigation of a two-dimensional odor space. *Neuron*, 102(5):1066–1075.e5, 2019.

D. L. Barack and J. W. Krakauer. Two views on the cognitive brain. *Nature Reviews Neuroscience*, 22(6):359–371, 2021.

A. G. Barto. Adaptive critics and the basal ganglia. 1995.

A. G. Barto, R. S. Sutton, and C. Watkins. *Learning and sequential decision making*, volume 89. University of Massachusetts Amherst, MA, 1989.

M. Bear, B. Connors, and M. A. Paradiso. *Neuroscience: Exploring the Brain, Enhanced Edition: Exploring the Brain, Enhanced Edition*. Jones & Bartlett Learning, 2020.

A. Beck and M. Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Oper. Res. Lett.*, 31(3):167–175, 2003. doi: 10.1016/S0167-6377(02)00231-6. URL https://doi.org/10.1016/S0167-6377(02)00231-6.

T. E. J. Behrens, T. H. Muller, J. C. R. Whittington, S. Mark, A. B. Baram, K. L. Stachenfeld, and Z. Kurth-Nelson. What is a cognitive map? organizing knowledge for flexible behavior. *Neuron*, 100(2):490–509, 2018.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *CoRR*, abs/1207.4708, 2012. URL http://arxiv.org/abs/1207.4708.

M. G. Bellemare, W. Dabney, and R. Munos. A distributional perspective on reinforcement learning. *CoRR*, abs/1707.06887, 2017. URL http://arxiv.org/abs/1707.06887.

R. Bellman. *Dynamic Programming*. Dover Publications, 1957. ISBN 9780486428093.

D. Bennett, G. Davidson, and Y. Niv. A model of mood as integrated advantage. *Psychol Rev*, 129(3):513–541, Apr 2022.

D. P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 2nd edition, 2000. ISBN 1886529094.

D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-dynamic programming*. Number 3 in the Optimization and Neural Computation Series. Athena Scientific, Belmont, Mass, 1996.

J. Bhandari and D. Russo. Global optimality guarantees for policy gradient methods. *CoRR*, abs/1906.01786, 2019. URL http://arxiv.org/abs/1906.01786.

J. Bhandari and D. Russo. On the linear convergence of policy gradient methods for finite mdps. In A. Banerjee and K. Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 2386–2394. PMLR, 13–15 Apr 2021.

S. Bhatnagar, R. S. Sutton, M. Ghavamzadeh, and M. Lee. Natural actor–critic algorithms. *Automatica*, 45(11):2471–2482, 2009.

J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, et al. Jax: composable transformations of python+ numpy programs. 2018. URL http://github.com/google/jax.

S. Bubeck. Convex optimization: Algorithms and complexity. *Found. Trends Mach. Learn.*, 8(3-4):231–357, 2015. doi: 10.1561/2200000050. URL https://doi.org/10.1561/2200000050.

L. Buesing, T. Weber, Y. Zwols, S. Racanière, A. Guez, J.-B. Lespiau, and N. M. O. Heess. Woulda, coulda, shoulda: Counterfactually-guided policy search. *ArXiv*, abs/1811.06272, 2019. URL https://openreview.net/forum?id=BJG0voC9YQ.

N. Burgess. The 2014 nobel prize in physiology or medicine: a spatial model for cognitive neuroscience. *Neuron*, 84(6):1120–1125, Dec 2014.

V. Chelu and D. Precup. Functional acceleration for policy mirror descent, 2024a. URL https://arxiv.org/abs/2407.16602.

V. Chelu and D. Precup. Mood as an extrapolation engine for adaptive learning and decision-making. (in submission @ Computational and Systems Neuroscience (COSYNE) 2025) [pdf], 2024b. URL https://drive.google.com/file/d/1-yga6hoeucvl2IywFC5Vcy-XXguirEff/view?usp=sharing.

V. Chelu, D. Precup, and H. van Hasselt. Forethought and hindsight in credit assignment. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

V. Chelu, D. Borsa, D. Precup, and H. van Hasselt. Selective credit assignment, 2022. URL https://arxiv.org/abs/2202.09699.

V. Chelu, T. Zahavy, A. Guez, D. Precup, and S. Flennerhag. Acceleration in policy optimization, 2023. URL https://arxiv.org/abs/2306.10587.

G. Chen and M. Teboulle. Convergence analysis of a proximal-like minimization algorithm using bregman functions. *SIAM J. on Optimization*, 3(3):538–543, aug 1993. ISSN 1052-6234. doi: 10.1137/0803026. URL https://doi.org/10.1137/0803026.

C.-K. Chiang, T. Yang, C.-J. Lee, M. Mahdavi, C.-J. Lu, R. Jin, and S. Zhu. Online optimization with gradual variations. In S. Mannor, N. Srebro, and R. C. Williamson, editors, *Proceedings of the 25th Annual Conference on Learning Theory*, volume 23 of *Proceedings of Machine Learning Research*, pages 6.1–6.20, Edinburgh, Scotland, 25–27 Jun 2012. PMLR. URL https://proceedings.mlr.press/v23/chiang12.html.

S. Chung and L. F. Abbott. Neural population geometry: An approach for understanding biological and artificial neural networks. *Current Opinion in Neurobiology*, 70:137–144, 2021.

A. O. Constantinescu, J. X. O'Reilly, and T. E. J. Behrens. Organizing conceptual knowledge in humans with a gridlike code. *Science*, 352(6292):1464–1468, 2024/11/17 2016.

R. Cools. Neuropsychopharmacology of cognitive flexibility. 2015.

L. H. Corbit and B. W. Balleine. Double dissociation of basolateral and central amygdala lesions on the general and outcome-specific forms of pavlovian-instrumental transfer. *J Neurosci*, 25(4):962–970, Jan 2005.

J. Cornford, D. Kalajdzievski, M. Leite, A. Lamarquette, D. M. Kullmann, and B. A. Richards. Learning to live with dale's principle: {ANN}s with separate excitatory and inhibitory units. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=eU776ZYxEpz.

H. S. Courellis, J. Minxha, A. R. Cardenas, D. L. Kimmel, C. M. Reed, T. A. Valiante, C. D. Salzman, A. N. Mamelak, S. Fusi, and U. Rutishauser. Abstract representations emerge in human hippocampal neurons during inference. *Nature*, 632(8026):841–849, 2024.

E. T. Cowan, A. C. Schapiro, J. E. Dunsmoor, and V. P. Murty. Memory consolidation as an adaptive process. *Psychonomic Bulletin & Review*, 28(6):1796–1810, 2021.

R. Dadashi, A. A. Taïga, N. L. Roux, D. Schuurmans, and M. G. Bellemare. The value function polytope in reinforcement learning, 2019.

H. . .-. Dale. Pharmacology and nerve-endings, 1935.

N. D. Daw, Y. Niv, and P. Dayan. Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature Neuroscience*, 8(12):1704–1711, 2005.

N. D. Daw, S. J. Gershman, B. Seymour, P. Dayan, and R. J. Dolan. Model-based influences on humans' choices and striatal prediction errors. *Neuron*, 69(6):1204–1215, Mar 2011.

P. Dayan. Improving Generalization for Temporal Difference Learning: The Successor Representation. *Neural Computation*, 5(4):613–624, 07 1993. doi: 10.1162/neco.1993.5.4.613. URL https://doi.org/10.1162/neco.1993.5.4.613.

P. Dayan. Goal-directed control and its antipodes. *Neural Netw*, 22(3):213–219, Apr 2009.

N. de Lara, **Veronica Chelu**, and D. Precup. Recurrent policies are not enough for continual reinforcement learning. URL https://drive.google.com/file/d/1scdul85AzCyGUEh0xWQyWos-eAVqKq_8/view?usp=sharing.

M. P. Deisenroth, D. Fox, and C. E. Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(2):408–423, 2015. doi: 10.1109/TPAMI.2013.218. URL https://doi.org/10.1109/TPAMI.2013.218.

S. Dohare, R. S. Sutton, and A. R. Mahmood. Continual backprop: Stochastic gradient descent with persistent randomness. *arXiv preprint arXiv:2108.06325*, 2021.

S. Dohare, J. F. Hernandez-Garcia, Q. Lan, P. Rahman, A. R. Mahmood, and R. S. Sutton. Loss of plasticity in deep continual learning. *Nature*, 632(8026):768–774, 2024.

R. J. Dolan and P. Dayan. Goals and habits in the brain. *Neuron*, 80:312–325, 2013. URL https://api.semanticscholar.org/CorpusID:7199332.

S. Dong, B. V. Roy, and Z. Zhou. Simple agent, complex environment: Efficient reinforcement learning with agent states. *J. Mach. Learn. Res.*, 23:255:1–255:54, 2021. URL https://api.semanticscholar.org/CorpusID:231861555.

P. D'Oro, A. M. Metelli, A. Tirinzoni, M. Papini, and M. Restelli. Gradient-aware model-based policy search. *CoRR*, abs/1909.04115, 2019. URL http://arxiv.org/abs/1909.04115.

L. Duncker, L. Driscoll, K. V. Shenoy, M. Sahani, and D. Sussillo. Organizing recurrent network dynamics by task-computation to enable continual learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 14387–14397. Curran Associates, Inc., 2020.

J. E. Dunsmoor, V. P. Murty, D. Clewett, E. A. Phelps, and L. Davachi. Tag and capture: how salient experiences target and rescue nearby events in memory. *Trends Cogn Sci*, 26 (9):782–795, Sep 2022.

D. Dvorak, A. Chung, E. H. Park, and A. Fenton. Dentate spikes and external control of hippocampal function. *Cell Reports*, 36(5):109497, 2021.

J. C. Eccles, P. Fatt, and K. Koketsu. Cholinergic and inhibitory synapses in a pathway from motor-axon collaterals to motoneurones. *The Journal of physiology*, 126(3):524, 1954.

M. El-Gaby, A. L. Harris, J. C. R. Whittington, W. Dorrell, A. Bhomick, M. E. Walton, T. Akam, and T. E. J. Behrens. A cellular basis for mapping behavioural structure. *Nature*, 2024.

E. Eldar, R. B. Rutledge, R. J. Dolan, and Y. Niv. Mood as representation of momentum. *Trends Cogn Sci*, 20(1):15–24, Jan 2016.

A. Emanuel and E. Eldar. Emotions as computations. *Neurosci Biobehav Rev*, 144:104977, Jan 2023.

A. Farahmand. Iterative value-aware model learning. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, volume 31, pages

9090–9101. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/8121-iterative-value-aware-model-learning.

A.-M. Farahmand and M. Ghavamzadeh. Pid accelerated value iteration algorithm. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 3143–3153. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/farahmand21a.html.

A. M. Farahmand, A. Barreto, and D. Nikovski. Value-aware loss function for model-based reinforcement learning. In A. Singh and X. J. Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, volume 54 of *Proceedings of Machine Learning Research*, pages 1486–1494. PMLR, 2017. URL http://proceedings.mlr.press/v54/farahmand17a.html.

G. Farquhar, T. Rocktäschel, M. Igl, and S. Whiteson. Treeqn and atreec: Differentiable tree planning for deep reinforcement learning. *CoRR*, abs/1710.11417, 2017. URL http://arxiv.org/abs/1710.11417.

J. S. Farrell, E. Hwaun, B. Dudok, and I. Soltesz. Neural and behavioural state switching during hippocampal dentate spikes. *Nature*, 628(8008):590–595, 2024.

C. Findling, F. Hubert, L. Acerbi, B. Benson, J. Benson, D. Birman, N. Bonacchi, M. Carandini, J. A. Catarino, G. A. Chapuis, A. K. Churchland, Y. Dan, E. E. DeWitt, T. A. Engel, M. Fabbri, M. Faulkner, I. R. Fiete, L. Freitas-Silva, B. Gerçek, K. D. Harris, M. Häusser, S. B. Hofer, F. Hu, J. M. Huntenburg, A. Khanal, C. Krasniak, C. Langdon, P. E. Latham, P. Y. P. Lau, Z. Mainen, G. T. Meijer, N. J. Miska, T. D. Mrsic-Flogel, J.-P. Noel, K. Nylund, A. Pan-Vazquez, L. Paninski, J. Pillow, C. Rossant, N. Roth, R. Schaeffer, M. Schartner, Y. Shi, K. Z. Socha, N. A. Steinmetz, K. Svoboda, C. Tessereau, A. E. Urai, M. J. Wells, S. J. West, M. R. Whiteway, O. Winter, I. B. Witten, A. Zador, P. Dayan, and A. Pouget.

Brain-wide representations of prior information in mouse decision-making. *bioRxiv*, page 2023.07.04.547684, 01 2023.

G. Floris, K. R. Dabrowski, M. T. Zanda, and S. E. Daws. Psilocybin reduces heroin seeking behavior and modulates inflammatory gene expression in the nucleus accumbens and prefrontal cortex of male rats. *Molecular Psychiatry*, 2024.

C. Gelada and M. G. Bellemare. Off-policy deep reinforcement learning by bootstrapping the covariate shift. In *AAAI*, 2019.

S. J. Gershman, E. J. Horvitz, and J. B. Tenenbaum. Computational rationality: A converging paradigm for intelligence in brains, minds, and machines. *Science*, 349(6245):273–278, 2024/12/15 2015.

W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski. *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. Cambridge University Press, Cambridge, 2014.

V. Goudar, B. Peysakhovich, D. J. Freedman, E. A. Buffalo, and X.-J. Wang. Schema formation in a neural population subspace underlies learning-to-learn in flexible sensorimotor problem-solving. *Nature Neuroscience*, 26(5):879–890, 2023.

A. Goyal, P. Brakel, W. Fedus, S. Singhal, T. Lillicrap, S. Levine, H. Larochelle, and Y. Bengio. Recall traces: Backtracking models for efficient reinforcement learning. *ArXiv*, abs/1804.00379, 2019. URL https://openreview.net/forum?id=HygsfnR9Ym.

V. Goyal and J. Grand-Clement. A first-order approach to accelerated value iteration, 2021.

A. A. Grace. Gating of information flow within the limbic system and the pathophysiology of schizophrenia. *Brain Res Brain Res Rev*, 31(2-3):330–341, Mar 2000.

J. Grand-Clément. From convex optimization to mdps: A review of first-order, second-order and quasi-newton methods for mdps, 2021.

I. C. Guerreiro and C. Clopath. Memory's gatekeeper: The role of pfc in the encoding of congruent events. *Proceedings of the National Academy of Sciences*, 121(30):e2403648121, 2024/12/13 2024.

A. GX-Chen, V. Chelu, B. A. Richards, and J. Pineau. A generalized bootstrap target for value-learning, efficiently combining value and feature predictions. *CoRR*, abs/2201.01836, 2022.

D. Ha and J. Schmidhuber. Recurrent world models facilitate policy evolution. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pages 2455–2467, 2018. URL http://papers.nips.cc/paper/7512-recurrent-world-models-facilitate-policy-evolution.

T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 80:1856–1865, 2018. URL http://proceedings.mlr.press/v80/haarnoja18b.html.

R. Hadsell, D. Rao, A. A. Rusu, and R. Pascanu. Embracing change: Continual learning in deep neural networks. *Trends in Cognitive Sciences*, 24(12):1028–1040, 2024/09/17 2020.

A. Hallak and S. Mannor. Consistent on-line off-policy evaluation. *ArXiv*, abs/1702.07121, 2017.

A. Hallak, A. Tamar, R. Munos, and S. Mannor. Generalized emphatic temporal difference learning: Bias-variance analysis. *In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*, abs/1509.05172, 2016.

H. F. Harlow. The formation of learning sets. *Psychological Review*, 56(1):51–65, 1949.

C. A. Hartley and E. A. Phelps. *Extinction Learning*, pages 1252–1253. Springer US, Boston, MA, 2012.

A. Harutyunyan, W. Dabney, D. Borsa, N. Heess, R. Munos, and D. Precup. The termination critic. In K. Chaudhuri and M. Sugiyama, editors, *AISTATS*, volume 89 of *Proceedings of Machine Learning Research*, pages 2231–2240. PMLR, 2019a. URL http://proceedings.mlr.press/v89/harutyunyan19a.html.

A. Harutyunyan, W. Dabney, T. Mesnard, M. G. Azar, B. Piot, N. Heess, H. van Hasselt, G. Wayne, S. Singh, D. Precup, and R. Munos. Hindsight credit assignment. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, volume abs/1912.02503, pages 12467–12476, 2019b. URL http://papers.nips.cc/paper/9413-hindsight-credit-assignment.

A. Harutyunyan, P. Vrancx, P. Hamel, A. Nowe, and D. Precup. Per-decision option discounting. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2644–2652. PMLR, 09–15 Jun 2019c. URL https://proceedings.mlr.press/v97/harutyunyan19a.html.

H. V. Hasselt, M. Hessel, and J. Aslanides. When to use parametric models in reinforcement learning? *ArXiv*, abs/1906.05243, 2019.

R. Hattori, N. G. Hedrick, A. Jain, S. Chen, H. You, M. Hattori, J.-H. Choi, B. K. Lim, R. Yasuda, and T. Komiyama. Meta-reinforcement learning via orbitofrontal cortex. *Nature Neuroscience*, 26(12):2182–2191, 2023.

E. Hazan. *Introduction to Online Convex Optimization*. Foundations and Trends in Optimization. Now, Boston, 2017. ISBN 978-1-68083-170-2. doi: 10.1561/2400000013.

J. E. Heiss, Y. Katz, E. Ganmor, and I. Lampl. Shift in the balance between excitation and inhibition during sensory adaptation of s1 neurons. *The Journal of Neuroscience*, 28(49): 13320, 12 2008.

T. Hennigan, T. Cai, T. Norman, and I. Babuschkin. Haiku: Sonnet for JAX. In *Haiku: Sonnet for JAX*, 2020. URL http://github.com/deepmind/dm-haiku.

M. Hessel, J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. G. Azar, and D. Silver. Rainbow: Combining improvements in deep reinforcement learning. *CoRR*, abs/1710.02298, 2017. URL http://arxiv.org/abs/1710.02298.

M. Hessel, H. van Hasselt, J. Modayil, and D. Silver. On inductive biases in deep reinforcement learning. *CoRR*, abs/1907.02908, 2019. URL http://arxiv.org/abs/1907.02908.

M. Hessel, D. Budden, F. Viola, M. Rosca, E. Sezener, and T. Hennigan. Optax: composable gradient transformation and optimisation, in jax! In *Optax: composable gradient transformation and optimisation, in JAX!*, 2020. URL http://github.com/deepmind/optax.

N. Hesselgrave, T. A. Troppoli, A. B. Wulff, A. B. Cole, and S. M. Thompson. Harnessing psilocybin: antidepressant-like behavioral and synaptic actions of psilocybin are independent of 5-ht2r activation in mice. *Proceedings of the National Academy of Sciences*, 118 (17):e2022489118, 2024/09/17 2021.

T. Hester and P. Stone. TEXPLORE: real-time sample-efficient reinforcement learning for robots. *Mach. Learn.*, 90(3):385–429, 2013. doi: 10.1007/s10994-012-5322-7. URL https://doi.org/10.1007/s10994-012-5322-7.

G. Hinton, N. Srivastava, and K. Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *CSC321*, 2012. URL https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

Y. Huang, Z. A. Yaple, and R. Yu. Goal-oriented and habitual decisions: Neural signatures of model-based and model-free learning. *NeuroImage*, 215:116834, 2020.

C.-C. Hung, T. Lillicrap, J. Abramson, Y. Wu, M. Mirza, F. Carnevale, A. Ahuja, and G. Wayne. Optimizing agent behavior over long time scales by transporting value. *Nature Communications*, 10, 2019.

Q. J. Huys, D. A. Pizzagalli, R. Bogdan, and P. Dayan. Mapping anhedonia onto reinforcement learning: a behavioural meta-analysis. *Biology of Mood & Anxiety Disorders*, 3(1):12, 2013.

J. S. Isaacson and M. Scanziani. How inhibition shapes cortical activity. *Neuron*, 72(2): 231–243, Oct 2011.

E. S. Iyer, P. Vitaro, S. Wu, J. Muir, Y. C. Tse, V. Cvetkovska, and R. C. Bagot. Uncertainty gates redundancy in reward integration in prefrontal-cortical and ventral-hippocampal nucleus accumbens inputs to tune engagement. *bioRxiv*, page 2024.06.27.601063, 01 2024.

T. Jafferjee, E. Imani, E. Talvitie, M. White, and M. Bowling. Hallucinating value: A pitfall of dyna-style planning with imperfect environment models. *CoRR*, abs/2006.04363, 2020. URL https://arxiv.org/abs/2006.04363.

D. C. Jangraw, H. Keren, H. Sun, R. L. Bedder, R. B. Rutledge, F. Pereira, A. G. Thomas, D. S. Pine, C. Zheng, D. M. Nielson, and A. Stringaris. A highly replicable decline in mood during rest and simple tasks. *Nature Human Behaviour*, 7(4):596–610, 2023.

R. Jiang, S. Zhang, V. Chelu, A. White, and H. van Hasselt. Learning expected emphatic traces for deep RL. *CoRR*, abs/2107.05405, 2021. URL https://arxiv.org/abs/2107.05405.

E. Johnson, C. Pike-Burke, and P. Rebeschini. Optimal convergence rate for exact policy mirror descent in discounted markov decision processes, 2023.

J. M. Joseph, A. Geramifard, J. W. Roberts, J. P. How, and N. Roy. Reinforcement learning with misspecified model classes. In *2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, May 6-10, 2013*, pages 939–946. IEEE, 2013. doi: 10.1109/ICRA.2013.6630686. URL https://doi.org/10.1109/ICRA.2013.6630686.

P. Joulani, A. György, and C. Szepesvári. A modular analysis of adaptive (non-)convex optimization: Optimism, composite objectives, variance reduction, and variational bounds. *Theor. Comput. Sci.*, 808:108–138, 2020a. doi: 10.1016/j.tcs.2019.11.015. URL https://doi.org/10.1016/j.tcs.2019.11.015.

P. Joulani, A. Raj, A. Gyorgy, and C. Szepesvari. A simpler approach to accelerated optimization: iterative averaging meets optimism. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4984–4993. PMLR, 13–18 Jul 2020b. URL https://proceedings.mlr.press/v119/joulani20a.html.

A. Juditsky, A. S. Nemirovskii, and C. Tauvel. Solving variational inequalities with stochastic mirror-prox algorithm, 2011.

A. Juliani, V. Chelu, L. Graesser, and A. Safron. A dual-receptor model of serotonergic psychedelics. *bioRxiv*, page 2024.04.12.589282, 01 2024.

J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

D. Kahneman. A perspective on judgment and choice: Mapping bounded rationality. *Progress in Psychological Science around the World. Volume 1 Neural, Cognitive and Developmental Issues.*, pages 1–47, 2013.

S. Kakade and J. Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, page 267–274, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc. ISBN 1558608737.

S. M. Kakade. A natural policy gradient. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001. URL https://proceedings.neurips.cc/paper/2001/file/4b86abe48d358ecf194c56c69108433e-Paper.pdf.

N. Ke, A. Goyal, O. Bilaniuk, J. Binas, M. Mozer, C. Pal, and Y. Bengio. Sparse attentive backtracking: Temporal creditassignment through reminding. *ArXiv*, abs/1809.03702, 2018a.

N. R. Ke, A. Goyal, O. Bilaniuk, J. Binas, M. Mozer, C. Pal, and Y. Bengio. Sparse attentive backtracking: Temporal creditassignment through reminding. In *NeurIPS*, 2018b.

M. J. Kearns and S. P. Singh. Bias-variance error bounds for temporal difference updates. In *COLT*, pages 142–147. Citeseer, 2000.

J. Kemeny and J. . L. Snell. *Finite Markov Chains*. 1960.

K. Khetarpal, M. Riemer, I. Rish, and D. Precup. Towards continual reinforcement learning: A review and perspectives. *CoRR*, abs/2012.13490, 2020. URL https://arxiv.org/abs/2012.13490.

M. Khona and I. R. Fiete. Attractor and integrator networks in the brain. *Nature Reviews Neuroscience*, 23(12):744–766, 2022.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, Mar. 2017. ISSN 1091-6490. doi: 10.1073/pnas.1611835114. URL http://dx.doi.org/10.1073/pnas.1611835114.

E. B. Knudsen and J. D. Wallis. Hippocampal neurons construct a map of an abstract value space. *Cell*, 184(18):4640–4650.e10, 2021.

J. Z. Kolter. The fixed points of off-policy td. In *NIPS*, 2011.

V. R. Konda and V. S. Borkar. Actor-critic–type learning algorithms for markov decision processes. *SIAM Journal on Control and Optimization*, 38(1):94–123, 1999. doi: 10.1137/S036301299731669X. URL https://doi.org/10.1137/S036301299731669X.

W. Kool, F. A. Cushman, S. J. Gershman, R. Morris, A. Bornstein, and A. Shenhav. *Chapter 7 - Competition and Cooperation Between Multiple Reinforcement Learning Systems*, pages 153–178. Academic Press, 2018.

G. M. Korpelevich. The extragradient method for finding saddle points and other problems. 1976.

A. Kumar, A. Gupta, and S. Levine. Discor: Corrective feedback in reinforcement learning via distribution correction. *ArXiv*, abs/2003.07305, 2020.

S. Kumar, H. Marklund, A. S. Rao, Y. Zhu, H. J. Jeon, Y. Liu, and B. V. Roy. Continual learning as computationally constrained reinforcement learning. *ArXiv*, abs/2307.04345, 2023. URL https://api.semanticscholar.org/CorpusID:259501376.

I. B. Laboratory, B. Benson, J. Benson, D. Birman, N. Bonacchi, M. Carandini, J. A. Catarino, G. A. Chapuis, A. K. Churchland, Y. Dan, P. Dayan, E. E. DeWitt, T. A. Engel, M. Fabbri, M. Faulkner, I. R. Fiete, C. Findling, L. Freitas-Silva, B. Gerçek, K. D. Harris, M. Häusser, S. B. Hofer, F. Hu, F. Hubert, J. M. Huntenburg, A. Khanal, C. Krasniak, C. Langdon,

P. Y. P. Lau, Z. F. Mainen, G. T. Meijer, N. J. Miska, T. D. Mrsic-Flogel, J.-P. Noel, K. Nylund, A. Pan-Vazquez, A. Pouget, C. Rossant, N. Roth, R. Schaeffer, M. Schartner, Y. Shi, K. Z. Socha, N. A. Steinmetz, K. Svoboda, A. E. Urai, M. J. Wells, S. J. West, M. R. Whiteway, O. Winter, and I. B. Witten. A brain-wide map of neural activity during complex behaviour. *bioRxiv*, page 2023.07.04.547681, 01 2023.

C. Langdon and T. A. Engel. Latent circuit inference from heterogeneous neural responses during cognitive tasks. *bioRxiv*, page 2022.01.23.477431, 01 2022.

C. Langdon, M. Genkin, and T. A. Engel. A unifying perspective on neural manifolds and circuits for cognition. *Nature Reviews Neuroscience*, 24(6):363–377, 2023.

L. Lapicque. Quantitative investigations of electrical nerve excitation treated as polarization. *Biol Cybern*, 97(5-6):341–349, Dec 1907.

S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

M. Lengyel and P. Dayan. Hippocampal contributions to control: The third way. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, page 889–896, Red Hook, NY, USA, 2007. Curran Associates Inc. ISBN 9781605603520.

E. R. J. Levy, S. Carrillo-Segura, E. H. Park, W. T. Redman, J. Hurtado, S. Chung, and A. Fenton. A manifold neural population code for space in hippocampal coactivity dynamics independent of place fields. *Cell Reports*, 42(10):113142, 2023.

P. Li, J. Cornford, A. Ghosh, and B. Richards. Learning better with dale's law: a spectral perspective. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, 2023.

L.-J. Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8(3):293–321, 1992. doi: 10.1007/BF00992699. URL https://doi.org/10.1007/BF00992699.

Y. Liu, R. J. Dolan, Z. Kurth-Nelson, and T. E. J. Behrens. Human replay spontaneously reorganizes experience. *Cell*, 178(3):640–652, Jul 2019.

H. Lu, R. M. Freund, and Y. Nesterov. Relatively-smooth convex optimization by first-order methods, and applications, 2017.

J. Luketina, S. Flennerhag, Y. Schroecker, D. Abel, T. Zahavy, and S. Singh. Meta-gradients in non-stationary environments, 2022.

Y. Luo, H. Xu, Y. Li, Y. Tian, T. Darrell, and T. Ma. Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=BJe1E2R5KX.

C. Ly, A. C. Greb, L. P. Cameron, J. M. Wong, E. V. Barragan, P. C. Wilson, K. F. Burbach, S. Soltanzadeh Zarandi, A. Sood, M. R. Paddy, W. C. Duim, M. Y. Dennis, A. K. McAllister, K. M. Ori-McKenney, J. A. Gray, and D. E. Olson. Psychedelics promote structural and functional neural plasticity. *Cell Rep*, 23(11):3170–3182, Jun 2018.

D. J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, USA, 2002. ISBN 0521642981.

H. R. Maei. *Gradient temporal-difference learning algorithms*. PhD thesis, University of Alberta, 2011.

A. R. Mahmood, H. Yu, M. White, and R. S. Sutton. Emphatic temporal-difference learning. *ArXiv*, abs/1507.01569, 2015.

R. C. Malenka and M. F. Bear. Ltp and ltd: An embarrassment of riches. *Neuron*, 44(1):5–21, 2024/12/07 2004.

Y. Malitsky and M. K. Tam. A forward-backward splitting method for monotone inclusions without cocoercivity, 2020.

V. Mante, D. Sussillo, K. V. Shenoy, and W. T. Newsome. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature*, 503(7474):78–84, 2013.

E. Marder. Neuromodulation of neuronal circuits: back to the future. *Neuron*, 76(1):1–11, 2012.

M. G. Mattar and N. D. Daw. Prioritized memory access explains planning and hippocampal replay. *Nature Neuroscience*, 21(11):1609–1617, 2018. doi: 10.1038/s41593-018-0232-z. URL https://doi.org/10.1038/s41593-018-0232-z.

P. McCorduck. Machines who think: A personal inquiry into the history and prospects of artificial intelligence. 1979.

H. McMahan and G. Gordon. Fast exact planning in markov decision processes. In *ICAPS*, 2005a.

H. B. McMahan and G. J. Gordon. Fast exact planning in markov decision processes. In S. Biundo, K. L. Myers, and K. Rajan, editors, *Proceedings of the Fifteenth International Conference on Automated Planning and Scheduling (ICAPS 2005), June 5-10 2005, Monterey, California, USA*, pages 151–160. AAAI, 2005b. URL http://www.aaai.org/Library/ICAPS/2005/icaps05-016.php.

J. Mei, C. Xiao, B. Dai, L. Li, C. Szepesvári, and D. Schuurmans. Escaping the gravitational pull of softmax. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, volume 33, pages 21130–21140. Curran Associates, Inc., 2020a. URL https://proceedings.neurips.cc/paper/2020/hash/f1cf2a082126bf02de0b307778ce73a7-Abstract.html.

J. Mei, C. Xiao, C. Szepesvári, and D. Schuurmans. On the global convergence rates of softmax policy gradient methods. *CoRR*, abs/2005.06392, 2020b. URL https://arxiv.org/abs/2005.06392.

J. Mei, B. Dai, C. Xiao, C. Szepesvári, and D. Schuurmans. Understanding the effect of stochasticity in policy optimization. *CoRR*, abs/2110.15572, 2021. URL https://arxiv.org/abs/2110.15572.

T. Mesnard, T. Weber, F. Viola, S. Thakoor, A. Saade, A. Harutyunyan, W. Dabney, T. Stepleton, N. Heess, A. Guez, M. Hutter, L. Buesing, and R. Munos. Counterfactual credit assignment in model-free reinforcement learning. *ArXiv*, abs/2011.09464, 2020.

E. K. Miller and J. D. Cohen. An integrative theory of prefrontal cortex function. *Annu Rev Neurosci*, 24:167–202, 2001.

M. Minsky. Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30, 1961.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, abs/1312.5602, 2013. URL http://arxiv.org/abs/1312.5602.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. doi: 10.1038/nature14236. URL https://doi.org/10.1038/nature14236.

V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, volume abs/1602.01783, pages 1928–1937, 2016. URL http://arxiv.org/abs/1602.01783.

G. Mogenson, C. Yang, and C. Yim. Influence of dopamine on limbic inputs to the nucleus accumbens. *Annals of the New York Academy of Sciences*, 1988.

M. Mohri and S. Yang. Accelerating optimization via adaptive prediction, 2015.

M. Mohri and S. Yang. Accelerating online convex optimization via adaptive prediction. In A. Gretton and C. C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS 2016, Cadiz, Spain, May 9-11, 2016*, volume 51 of *JMLR Workshop and Conference Proceedings*, pages 848–856. JMLR.org, 2016. URL http://proceedings.mlr.press/v51/mohri16.html.

A. Moore and C. Atkeson. Memory-based reinforcement learning: Efficient computation with prioritized sweeping. In S. Hanson, J. Cowan, and C. Giles, editors, *Advances in Neural Information Processing Systems*, volume 5. Morgan-Kaufmann, 1992.

A. Moore and C. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 13:103–130, 2004.

A. W. Moore and C. G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Mach. Learn.*, 13:103–130, 1993. doi: 10.1007/BF00993104. URL https://doi.org/10.1007/BF00993104.

T. Morimura, E. Uchibe, J. Yoshimoto, J. Peters, and K. Doya. Derivatives of logarithmic stationary distributions for policy gradient reinforcement learning. *Neural Comput.*, 22(2):342–376, Feb. 2010. ISSN 0899-7667. doi: 10.1162/neco.2009.12-08-922. URL https://doi.org/10.1162/neco.2009.12-08-922.

P. S. Muhle-Karbe, H. Sheahan, G. Pezzulo, H. J. Spiers, S. Chien, N. W. Schuck, and C. Summerfield. Goal-seeking compresses neural codes for space in the human hippocampus and orbitofrontal cortex. *Neuron*, 111(23):3885–3899.e6, 2023.

F. Nadim and D. Bucher. Neuromodulation of neurons and synapses. *Current opinion in neurobiology*, 29:48–56, 2014.

F. Najafi, G. Elsayed, R. Cao, E. Pnevmatikakis, P. Latham, J. Cunningham, and A. Churchland. Excitatory and inhibitory subnetworks are equally selective during decision-making and emerge simultaneously during learning. *Neuron*, 105(1):165–179, Jan 2020.

A. Nemirovski. Prox-method with rate of convergence o(1/t) for variational inequalities with lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM J. Optim.*, 15(1):229–251, 2004. doi: 10.1137/S1052623403425629. URL https://doi.org/10.1137/S1052623403425629.

A. S. Nemirovsky, D. B. Yudin, and E. R. Dawson. *Problem Complexity and Method Efficiency in Optimization*. John Wiley, Chichester, 1983.

Y. Nesterov. A method for solving the convex programming problem with convergence rate $\mathcal{O}(1/k^2)$. *Proceedings of the USSR Academy of Sciences*, 269:543–547, 1983.

D. Nettle and M. Bateson. The evolutionary origins of mood and its disorders. *Current Biology*, 22(17):R712–R721, 2012.

X. Nie, E. Brunskill, and S. Wager. Learning when-to-treat policies, 2020.

E. H. Nieh, M. Schottdorf, N. W. Freeman, R. J. Low, S. Lewallen, S. A. Koay, L. Pinto, J. L. Gauthier, C. D. Brody, and D. W. Tank. Geometry of abstract learned knowledge in the hippocampus. *Nature*, 595(7865):80–84, 2021.

Y. Niv, D. Joel, and P. Dayan. A normative perspective on motivation. *Trends in cognitive sciences*, 10(8):375–381

J. O'Byrne and K. Jerbi. How critical is brain criticality? *Trends in Neurosciences*, 45(11): 820–837, 2022.

J. Oh, S. Singh, and H. Lee. Value prediction network. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information*

Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA, pages 6118–6128, 2017. URL http://papers.nips.cc/paper/7192-value-prediction-network.

J. O'Keefe and L. Nadel. *The hippocampus as a cognitive map*. Clarendon Press ; Oxford University Press, Oxford, New York, 1978.

M. Okun and I. Lampl. Instantaneous correlation of excitation and inhibition during ongoing and sensory-evoked activities. *Nature Neuroscience*, 11(5):535–537, 2008.

Y. Pan, M. Zaheer, A. White, A. Patterson, and M. White. Organizing experience: A deeper look at replay mechanisms for sample-based planning in continuous state domains. *CoRR*, abs/1806.04624, 2018. URL http://arxiv.org/abs/1806.04624.

N. Parikh, S. Boyd, et al. Proximal algorithms. *Foundations and trends® in Optimization*, 1 (3):127–239, 2014.

R. Parr, L. Li, G. Taylor, C. Painter-Wakefield, and M. L. Littman. An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 752–759, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4. doi: 10.1145/1390156.1390251. URL http://doi.acm.org/10.1145/1390156.1390251.

J. Peng and R. Williams. Efficient learning and planning within the dyna framework. *IEEE International Conference on Neural Networks*, pages 168–174 vol.1, 1993a.

J. Peng and R. J. Williams. Efficient learning and planning within the dyna framework. *Adaptive Behavior*, 1(4):437–454, 1993b. doi: 10.1177/105971239300100403. URL https://doi.org/10.1177/105971239300100403.

J. Peng and R. J. Williams. Incremental multi-step q-learning. *Machine Learning*, 22:283–290, 1996.

J. Peters and S. Schaal. Policy gradient methods for robotics. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2219–2225, 2006.

S. Pitis. Source traces for temporal difference learning. *ArXiv*, abs/1902.02907, 2018. URL http://arxiv.org/abs/1902.02907.

B. T. Polyak. Gradient methods for the minimisation of functionals. *Ussr Computational Mathematics and Mathematical Physics*, 3:864–878, 1963.

F. Pouille, A. Marin-Burgin, H. Adesnik, B. V. Atallah, and M. Scanziani. Input normalization by global feedforward inhibition expands cortical dynamic range. *Nature Neuroscience*, 12(12):1577–1585, 2009.

D. Precup, R. S. Sutton, and S. P. Singh. Theoretical results on reinforcement learning with temporally abstract options. In *Machine Learning: ECML-98, 10th European Conference on Machine Learning, Chemnitz, Germany, April 21-23, 1998, Proceedings*, pages 382–393, 1998.

D. Precup, R. Sutton, and S. Dasgupta. Off-policy temporal difference learning with function approximation. In *ICML*, pages 417–424, 2001.

K. Protopapas and A. Barakat. Policy mirror descent with lookahead, 2024. URL https://arxiv.org/abs/2403.14156.

M. L. Puterman. *Markov Decision Processes*. Wiley, 1994. ISBN 978-0471727828. URL http://books.google.com/books/about/Markov_decision_processes.html?id=Y-gmAQAAIAAJ.

M. L. Puterman and S. L. Brumelle. On the convergence of policy iteration in stationary dynamic programming. *Mathematics of Operations Research*, 4(1):60–69, 1979.

G. J. Quirk and D. Mueller. Neural mechanisms of extinction learning and retrieval. *Neuropsychopharmacology*, 33(1):56–72, 2008.

A. Rakhlin and K. Sridharan. Online learning with predictable sequences. In S. Shalev-Shwartz and I. Steinwart, editors, *COLT 2013 - The 26th Annual Conference on Learning Theory, June 12-14, 2013, Princeton University, NJ, USA*, volume 30 of *JMLR Workshop and*

Conference Proceedings, pages 993–1019. JMLR.org, 2013a. URL http://proceedings.mlr.press/v30/Rakhlin13.html.

A. Rakhlin and K. Sridharan. Optimization, learning, and games with predictable sequences. *CoRR*, abs/1311.1869, 2013b. URL http://arxiv.org/abs/1311.1869.

A. Rakhlin and K. Sridharan. Online learning with predictable sequences, 2014.

G. Richter-Levin and I. Akirav. Emotional tagging of memory formation—in the search for neural mechanisms. *Brain Research Reviews*, 43(3):247–256, 2003.

J. P. Roach, A. K. Churchland, and T. A. Engel. Choice selective inhibition drives stability and competition in decision circuits. *Nature Communications*, 14(1):147, 2023.

H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 9 1951.

O. Robinson and H. Chase. Learning and choice in mood disorders: Searching for the computational parameters of anhedonia. *Computational Psychiatry*, 1:1–26, 10 2017.

K. Rømer Thomsen, P. C. Whybrow, and M. L. Kringelbach. Reconceptualizing anhedonia: novel perspectives on balancing the pleasure networks in the human brain. *Front Behav Neurosci*, 9:49, 2015.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

D. Russo. Approximation benefits of policy gradient methods with aggregated states, 2022.

R. B. Rutledge, N. Skandali, P. Dayan, and R. J. Dolan. A computational and neural model of momentary subjective well-being. *Proceedings of the National Academy of Sciences*, 111 (33):12252–12257, 2024/10/23 2014.

H. Satija, P. Amortila, and J. Pineau. Constrained markov decision processes via backward value functions, 2020. URL https://openreview.net/forum?id=S1lyyANYwr.

A. J. Schaefer, M. D. Bailey, S. M. Shechter, and M. S. Roberts. *Modeling Medical Treatment Using Markov Decision Processes*, pages 593–612. Springer US, Boston, MA, 2004. ISBN 978-1-4020-8066-1. doi: 10.1007/1-4020-8066-2_23. URL https://doi.org/10.1007/1-4020-8066-2_23.

T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015. URL http://arxiv.org/abs/1511.05952.

B. Scherrer and M. Geist. Local policy search in a convex space and conservative policy iteration as boosted policy search. In *Machine Learning and Knowledge Discovery in Databases*, page 35–50, Berlin, Heidelberg, 2014. Springer-Verlag. ISBN 978-3-662-44844-1. doi: 10.1007/978-3-662-44845-8_3. URL https://doi.org/10.1007/978-3-662-44845-8_3.

J. Schimmelpfennig, J. Topczewski, W. Zajkowski, and K. Jankowiak-Siuda. The role of the salience network in cognitive and affective deficits. *Front Hum Neurosci*, 17:1133367, 2023.

D. C. Schreiner, R. Renteria, and C. M. Gremel. Fractionating the all-or-nothing definition of goal-directed and habitual decision-making. *Journal of Neuroscience Research*, 98(6): 998–1006, 2024/12/15 2020.

J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. P. Lillicrap, and D. Silver. Mastering atari, go, chess and shogi by planning with a learned model. *CoRR*, abs/1911.08265, 2019. URL http://arxiv.org/abs/1911.08265.

Y. Schroecker, M. Vecerík, and J. Scholz. Generative predecessor models for sample-efficient imitation learning. *CoRR*, abs/1904.01139, 2019. URL http://arxiv.org/abs/1904.01139.

J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015. URL http://arxiv.org/abs/1502.05477.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

W. Schultz, P. Dayan, and P. R. Montague. A neural substrate of prediction and reward. *Science*, 275(5306):1593–1599

W. W. Seeley. The salience network: A neural system for perceiving and responding to homeostatic demands. *The Journal of Neuroscience*, 39(50):9878, 12 2019.

S. A. Shankman, A. C. Katz, A. A. DeLizza, C. Sarapas, S. M. Gorka, and M. L. Campbell. *The Different Facets of Anhedonia and Their Associations with Different Psychopathologies*, pages 3–22. Springer Netherlands, Dordrecht, 2014.

C. E. Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

L.-X. Shao, C. Liao, I. Gregg, P. A. Davoudian, N. K. Savalia, K. Delagarza, and A. C. Kwan. Psilocybin induces rapid and persistent growth of dendritic spines in frontal cortex in vivo. *Neuron*, 109(16):2535–2544.e4, 2024/09/17 2021.

R. M. Shapley and D. Xing. Local circuit inhibition in the cerebral cortex as the source of gain control and untuned suppression. *Neural Netw*, 37:172–181, Jan 2013.

A. Shwartz. Death and discounting. *IEEE Transactions on Automatic Control*, 46(4):644–647, 2001.

D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, page I–387–I–395. JMLR.org, 2014.

D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017a. doi: 10.1038/nature24270. URL https://doi.org/10.1038/nature24270.

D. Silver, H. van Hasselt, M. Hessel, T. Schaul, A. Guez, T. Harley, G. Dulac-Arnold, D. P. Reichert, N. C. Rabinowitz, A. Barreto, and T. Degris. The predictron: End-to-end learning and planning. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 3191–3199. PMLR, 2017b. URL http://proceedings.mlr.press/v70/silver17a.html.

H. A. Simon. A behavioral model of rational choice. *Quarterly Journal of Economics*, 69: 99–118, 1955.

S. P. Singh, T. S. Jaakkola, and M. I. Jordan. Learning without state-estimation in partially observable markovian decision processes. In *Proceedings of the Eleventh International Conference on International Conference on Machine Learning*, ICML'94, page 284–292, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc. ISBN 1558603352.

H. F. Song, G. R. Yang, and X.-J. Wang. Training excitatory-inhibitory recurrent neural networks for cognitive tasks: A simple and flexible framework. *PLOS Computational Biology*, 12(2):e1004792–, 02 2016.

R. E. Suri and W. Schultz. A neural network model with dopamine-like reinforcement signal that learns a spatial delayed response task. *Neuroscience*, 91(3):871–890

D. Sussillo. Neural circuits as computational dynamical systems. *Curr Opin Neurobiol*, 25: 156–163, Apr 2014.

D. Sussillo and O. Barak. Opening the black box: Low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural Comput.*, 25(3):626–649, mar 2013.

R. Sutton. Temporal credit assignment in reinforcement learning. 1984.

R. Sutton and A. G. Barto. An adaptive network that constructs and uses an internal model of its world. In *An adaptive network that constructs and uses an internal model of its world*, 1981.

R. Sutton and S. Singh. On step-size and bias in temporal-difference learning. In *Proceedings of the Eighth Yale Workshop on Adaptive and Learning Systems*, pages 91–96. Yale University, New Haven, CT., 1994.

R. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artif. Intell.*, 112:181–211, 1999a.

R. Sutton, C. Szepesvari, A. Geramifard, and M. Bowling. Dyna-style planning with linear function approximation and prioritized sweeping. *ArXiv*, abs/1206.3285, 2008.

R. Sutton, A. R. Mahmood, D. Precup, and H. V. Hasselt. A new q(lambda) with interim forward view and monte carlo equivalence. In *ICML*, pages 568–576, 2014.

R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, 1988.

R. S. Sutton. Integrated modeling and control based on reinforcement learning. In R. Lippmann, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems 3, [NIPS Conference, Denver, Colorado, USA, November 26-29, 1990]*, pages 471–478. Morgan Kaufmann, 1990a. URL http://papers.nips.cc/paper/388-integrated-modeling-and-control-based-on-reinforcement-learning.

R. S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *In Proceedings of the Seventh International Conference on Machine Learning*, pages 216–224. Morgan Kaufmann, 1990b.

R. S. Sutton. TD models: Modeling the world at a mixture of time scales. In *Machine Learning, Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California, USA, July 9-12, 1995*, pages 531–539, 1995.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, Cambridge, MA, USA, 2018. ISBN 0262039249.

R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In S. Solla, T. Leen, and K. Müller, editors, *Proceedings of the 12th International Conference on Neural Information Processing Systems*, volume 12 of *NIPS'99*, page 1057–1063, Cambridge, MA, USA, 1999b. MIT Press. URL https://proceedings.neurips.cc/paper_files/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf.

R. S. Sutton, J. Modayil, M. Delp, T. Degris, P. M. Pilarski, A. White, and D. Precup. Horde: a scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011), Taipei, Taiwan, May 2-6, 2011, Volume 1-3*, pages 761–768, 2011.

R. S. Sutton, C. Szepesvári, A. Geramifard, and M. Bowling. Dyna-style planning with linear function approximation and prioritized sweeping. *CoRR*, abs/1206.3285, 2012. URL http://arxiv.org/abs/1206.3285.

R. S. Sutton, A. R. Mahmood, and M. White. An emphatic approach to the problem of off-policy temporal-difference learning. *ArXiv*, abs/1503.04269, 2016a.

R. S. Sutton, A. R. Mahmood, and M. White. An emphatic approach to the problem of off-policy temporal-difference learning. *The Journal of Machine Learning Research*, 17(1): 2603–2631, 2016b.

G. Tesauro. TD-Gammon, a Self-Teaching Backgammon Program, Achieves Master-Level Play. *Neural Computation*, 6(2):215–219, 03 1994.

E. L. Thorndike. *Animal intelligence: Experimental studies.* Macmillan Press, Lewiston, NY, US, 1911.

E. C. Tolman. Cognitive maps in rats and men. *Psychological review*, 55(4):189

M. Tomar, L. Shani, Y. Efroni, and M. Ghavamzadeh. Mirror descent policy optimization. *CoRR*, abs/2005.09814, 2020. URL https://arxiv.org/abs/2005.09814.

R. Tremblay, S. Lee, and B. Rudy. Gabaergic interneurons in the neocortex: From cellular properties to circuits. *Neuron*, 91(2):260–292, Jul 2016.

P. Tseng. Applications of a splitting algorithm to decomposition in convex programming and variational inequalities. *SIAM Journal on Control and Optimization*, 29(1):119–138, 1991.

J. Tsitsiklis and B. Van Roy. Analysis of temporal-diffference learning with function approximation. In M. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, pages 1075–1081. MIT Press, 1996.

D. Tsuboi, T. Nagai, J. Yoshimoto, and K. Kaibuchi. Neuromodulator regulation and emotions: insights from the crosstalk of cell signaling. *Front Mol Neurosci*, 17:1376762, 2024.

C. E. Vaaga, M. Borisovska, and G. L. Westbrook. Dual-transmitter neurons: functional implications of co-release and co-transmission. *Curr Opin Neurobiol*, 29:25–32, Dec 2014.

H. van Hasselt and R. S. Sutton. Learning to predict independent of span. *CoRR*, abs/1508.04582, 2015.

H. van Hasselt, Y. Doron, F. Strub, M. Hessel, N. Sonnerat, and J. Modayil. Deep reinforcement learning and the deadly triad. *CoRR*, abs/1812.02648, 2018.

H. van Hasselt, M. Hessel, and J. Aslanides. When to use parametric models in reinforcement learning? *CoRR*, abs/1906.05243, 2019. URL http://arxiv.org/abs/1906.05243.

H. van Hasselt, S. Madjiheurem, M. Hessel, D. Silver, A. N. Barreto, and D. Borsa. Expected eligibility traces. *ArXiv*, abs/2007.01839, 2020.

H. van Seijen and R. S. Sutton. Planning by prioritized sweeping with small backups. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 361–369. JMLR.org, 2013. URL http://proceedings.mlr.press/v28/vanseijen13.html.

H. van Seijen and R. S. Sutton. True online TD(λ). In *International Conference on Machine Learning*, pages 692–700, 2014.

H. Van Seijen and R. S. Sutton. A deeper look at planning as learning from replay. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 2314–2322. JMLR.org, 2015. URL http://dl.acm.org/citation.cfm?id=3045118.3045364.

C. van Vreeswijk and H. Sompolinsky. Chaos in neuronal networks with balanced excitatory and inhibitory activity. *Science*, 274(5293):1724–1726, Dec 1996.

S. Vaswani, O. Bachem, S. Totaro, R. Mueller, M. Geist, M. C. Machado, P. S. Castro, and N. L. Roux. A functional mirror ascent view of policy gradient methods with function approximation. *CoRR*, abs/2108.05828, 2021. URL https://arxiv.org/abs/2108.05828.

S. Vaswani, A. Kazemi, R. Babanezhad, and N. L. Roux. Decision-aware actor-critic with function approximation and theoretical guarantees, 2023.

N. Vieillard, B. Scherrer, O. Pietquin, and M. Geist. Momentum in reinforcement learning. *CoRR*, abs/1910.09322, 2019. URL http://arxiv.org/abs/1910.09322.

S. Vyas, M. D. Golub, D. Sussillo, and K. V. Shenoy. Computation through neural population dynamics. *Annual Review of Neuroscience*, 43(Volume 43, 2020):249–275, 2020.

A. J. Wakhloo, W. Slatton, and S. Chung. Neural population geometry and optimal coding of tasks with shared latent structure, 2024.

M. E. Walton, T. E. J. Behrens, M. J. Buckley, P. H. Rudebeck, and M. F. S. Rushworth. Separable learning systems in the macaque brain and the role of orbitofrontal cortex in contingent learning. *Neuron*, 65(6):927–939, Mar 2010.

Y. Wan, M. Zaheer, A. White, M. White, and R. S. Sutton. Planning with expectation models. *CoRR*, abs/1904.01191, 2019. URL http://arxiv.org/abs/1904.01191.

J. Wang and J. D. Abernethy. Acceleration through optimistic no-regret dynamics. *CoRR*, abs/1807.10455, 2018. URL http://arxiv.org/abs/1807.10455.

J. Wang, J. D. Abernethy, and K. Y. Levy. No-regret dynamics in the fenchel game: A unified framework for algorithmic convex optimization. *CoRR*, abs/2111.11309, 2021. URL https://arxiv.org/abs/2111.11309.

J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. M. Botvinick. Learning to reinforcement learn. *CoRR*, abs/1611.05763, 2016. URL http://arxiv.org/abs/1611.05763.

J. X. Wang, Z. Kurth-Nelson, D. Kumaran, D. Tirumala, H. Soyer, J. Z. Leibo, D. Hassabis, and M. Botvinick. Prefrontal cortex as a meta-reinforcement learning system. *Nature Neuroscience*, 21(6):860–868, 2018.

T. Wang, D. J. Lizotte, M. Bowling, and D. Schuurmans. Dual representations for dynamic programming. 2008. URL https://api.semanticscholar.org/CorpusID:15837107.

X.-J. Wang. Probabilistic decision making by slow reverberation in cortical circuits. *Neuron*, 36(5):955–968, Dec 2002.

C. Watkins. Learning from delayed rewards. 01 1989.

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

C. J. C. H. Watkins and P. Dayan. Technical note: q -learning. *Mach. Learn.*, 8(3–4):279–292, May 1992b. ISSN 0885-6125. doi: 10.1007/BF00992698. URL https://doi.org/10.1007/BF00992698.

M. White. Unifying task specification in reinforcement learning. In *International Conference on Machine Learning*, volume abs/1609.01995, pages 3742–3750. PMLR, 2017.

M. White and A. White. A greedy approach to adapting the trace parameter for temporal difference learning. *ArXiv*, abs/1607.00446, 2016.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8:229–256, 1992. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

H. R. Wilson and J. D. Cowan. Excitatory and inhibitory interactions in localized populations of model neurons. *Biophys J*, 12(1):1–24, Jan 1972.

L. Xiao. On the convergence rates of policy gradient methods, 2022.

Z. Xiong, M. Fazel, and L. Xiao. Dual approximation policy optimization, 2024. URL https://arxiv.org/abs/2410.01249.

Z. Xu, H. V. Hasselt, and D. Silver. Meta-gradient reinforcement learning. *ArXiv*, abs/1805.09801:2402–2413, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/2715518c875999308842e3455eda2fe3-Abstract.html.

H. H. Yin and B. J. Knowlton. The role of the basal ganglia in habit formation. *Nature Reviews Neuroscience*, 7(6):464–476, 2006.

H. H. Yin, B. J. Knowlton, and B. W. Balleine. Lesions of dorsolateral striatum preserve outcome expectancy but disrupt habit formation in instrumental learning. *Eur J Neurosci*, 19(1):181–189, Jan 2004.

H. H. Yin, S. B. Ostlund, B. J. Knowlton, and B. W. Balleine. The role of the dorsomedial striatum in instrumental conditioning. *Eur J Neurosci*, 22(2):513–523, Jul 2005.

R. Yuan, S. S. Du, R. M. Gower, A. Lazaric, and L. Xiao. Linear convergence of natural policy gradient methods with log-linear policies. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=-z9hdsyUwVQ.

T. Zahavy, Z. Xu, V. Veeriah, M. Hessel, J. Oh, H. P. van Hasselt, D. Silver, and S. Singh. A self-tuning actor-critic algorithm. *Advances in Neural Information Processing Systems*, 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/f02208a057804ee16ac72ff4d3cec53b-Abstract.html.

T. Zahavy, V. Veeriah, S. Hou, K. Waugh, M. Lai, E. Leurent, N. Tomasev, L. Schut, D. Hassabis, and S. Singh. Diversifying ai: Towards creative chess with alphazero, 2023.

S. Zhang, B. Liu, H. Yao, and S. Whiteson. Provably convergent two-timescale off-policy actor-critic with function approximation. In *International Conference on Machine Learning*, pages 11204–11213. PMLR, 2020a.

S. Zhang, V. Veeriah, and S. Whiteson. Learning retrospective knowledge with reverse reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, 2020b.

Z. Zheng, R. Vuorio, R. L. Lewis, and S. Singh. Pairwise weights for temporal credit assignment. *CoRR*, abs/2102.04999, 2021. URL https://arxiv.org/abs/2102.04999.

Y. Zhu, Y. Zeng, J. Ren, L. Zhang, C. Chen, G. Fernandez, S. Qin, T. Kahnt, L. L. Colgin, V. Murty, and A. Tambini. Emotional learning retroactively promotes memory integration through rapid neural reactivation and reorganization. *eLife*, 11:e60190, 2022.

K. Zühlsdorff, J. W. Dalley, T. W. Robbins, and S. Morein-Zamir. Cognitive flexibility: neurobehavioral correlates of changing one's mind. *Cerebral Cortex*, 33(9):5436–5446, 11 2022.