Normative studies of single-event memories and multitask decision-making

Lucas Silva Simões

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Gatsby Computational Neuroscience Unit

University College London

November 15, 2023

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

Abstract

In recent decades, there have been significant advancements in the study of cognition and behaviour, with various aspects of memory, learning, and decision-making being characterised and analysed. One important approach in this endeavour is computational modelling, which allows researchers to explore the normative mechanisms underlying cognitive phenomena. This thesis is situated within this line of research, and investigates the normative principles governing two aspects of cognition and behaviour: the appropriateness of single-event memories and optimal decision-making in multi-task environments.

Single-event memories are a typical aspect of human life but stand in contrast to common practice in machine learning, where it is better to condense information from several experiences into a set of statistics for robust and generalisable representations. In the first part of this thesis, we build on past research and postulate that single-event memories can support common statistical learning approaches. We provide analytical calculations and simulations to demonstrate that this cannot entirely account for the existence of single-event memories in the human brain.

The second part of the thesis examines the fact that humans and other animals have multiple decisions to make in a day and can decide to move from one to the other at will. While it is understood that they do this to maximise rewards such as food, social validation, and pleasure, current models at times fail to explain behaviour observed in the lab. We use ideas from foraging theory and reinforcement learning to study optimal behaviour in this problem, first in static environments, then in dynamic ones.

Impact Statement

This thesis presents analytical and simulation analyses to, separately, investigate normative aspects of single-event memories and decision-making in multi-task environments.

In the memory part of this thesis, we provide a novel account that sheds light on our understanding of single-event memories from a computational perspective. We offer a comprehensive assessment of the advantages and limitations of this approach, highlighting its potential as a foundation for future research.

A robust grasp of the mechanisms and purpose of single-event memories has the potential to enhance the development of more efficient, less biased artificial agents. These agents can learn to utilize local information effectively while avoiding unwarranted generalizations from out-of-distribution data.

Regarding the decision-making part of this thesis, we present an explanation for the issue of ”overstaying” that has been documented in the literature. Furthermore, our work introduces innovative approaches to understanding optimal behavior in multi-task environments, with a particular focus on dynamic settings. These findings contrast with prior research in the field.

A profound comprehension of optimal behavior in dynamic multi-task environments can empower the creation of artificial agents capable of operating effectively in uncertain and diverse contexts. This has practical applications in robotics and various other domains. Moreover, our research carries potential implications not only in academic and industrial domains but also in clinical applications, as it can be extended to model everyday human decision-making.

Acknowledgements

It has been an incredible journey from the beginning to the completion of this PhD. I could not have gotten this far without the support of numerous remarkable individuals.

Firstly, my parents (Marcio and Adriana) and my brother (Gabriel): for the crucial moments of unwavering support and love, I am immensely grateful. Thank you for always standing by my side.

Then I also want to express my gratitude to friends in Brazil and in England; there are too many to name individually without the risk of overlooking someone important. To my friends well-versed in science, academia, and the unique challenges of this journey – you helped me carry the weight of this distinctive path. To those (lucky) friends less involved in academia – you provided a much-needed escape from it all.

Special thanks to Peter Latham for all the stimulating discussions we've had, and for believing in me even when I didn't; to Alberto Pezzotta, whose support and collaboration during the last year and a half have been invaluable; to Thomas, for helping me not embarrass myself (too much) with mistakes with the English language; and to the Gatsby Unit and Gatsby Charitable Foundation, for the privilege of studying in a thriving research environment with generous funding.

Last but not least, to Carol and Manuela – my true purpose and calling lie with you both.

"In all give thanks to God," and here it could not be different.

Contents

List of Figures

List of Tables

Introduction

Chapter 1

General Introduction

This thesis is divided into two separate parts:

- Part I (Chapter 2) is focused on single-event memories, and trying to understand their advantages in learning;

- Part II (Chapters 3 and 4) presents two studies on optimal decision-making in multi-task problems.

In the first part, we investigate the possible benefits of using single-event memories in learning and prediction. The approach consists in exploring two different algorithms that use information contained in individual datapoints for prediction, and compare those with a standard model of supervised learning. We derive analytical results and compare those with numerical simulations, recovering results previously reported in the literature, now with analytical backing. The conclusion found is that single-event memory can be useful in specific scenario where the standard model of supervised learning is still learning the correct data mapping (very low data regime), or when this model does not contain the model class that generates the data (model mismatch). Nonetheless, single-event memories sufer from the “curse of dimensionality”, and their improvement scales poorly with the dimensionality of the space for the problem.

The second part of the thesis deals with the question of optimal decision-making in multi-task environments. The agent sees one task at a time, where the reward rate obtainable decreases over time, and needs to decide how to behave in the current task, and when to leave to another one. In Chapter 3 we formalise the problem of deciding how to behave optimally in this problem when the environment is static and the agent cannot choose which task to engage with. We prove that an algorithm can learn an optimal policy effectively using only information local to each task and a global reward rate signal. We recover the same results as past work in the foraging literature, where the optimal time to move out of a task can be found by considering the global reward rate in the environment, now in a more general setting where there can be complex policies for behaving in each task. We use this framework to suggest an explanation for apparent suboptimal human behaviour described in the literature.

In Chapter 4 we delve into scenarios where the agent can choose which task to engage with and the environment is now dynamic. We suggest a strategy an agent might choose for optimising their reward rate in this problem, and compare our theoretical predictions with numerical simulations. The resulting optimal policy is different to the one in Chapter 3 and those considered in the foraging literature, with the main distinction being that now the agent can maximise the leaving time considering only the maximum reward rate in the current task, and not the whole environment. We discuss the possible consequences this result might have for behaviour.

Part I

Memory

Chapter 2

A mathematical study of single-event memories in learning

It has long been observed that humans have a remarkable ability to remember specific memories of individual experiences in their lives, and can retell them with an intricate level of detail (Tulving (1972)). Despite the consensus that this capacity exists, a wealth of literature has sought to characterise which features are essential to this type of memory (Mahr and Csibra (18ed); Madan (2019)), which brain regions are responsible for enabling it (McClelland and O'Reilly (1995); Kumaran et al. (2016)), which other species have it too Clayton and Russell (2009); Applegate and Aronov (2022), and what function it serves (Biderman et al. (2020); Mahr and Csibra (18ed)). In this work, we study different learning algorithms that involve using single-event memories, to try to explain the normative appropriateness of those memories and what function they might serve an agent learning about the world.[1]

The question of the function of single-event memories becomes more interesting as one notices that the concept of a memory system that extensively stores individual rich memories of every single lived experience is at odds with the current understanding about learning and memory in the machine learning (ML) and statistics

[1]The name customarily given to memories of individual events is of "episodic memories". Following Mahr and Csibra (18ed), this work keeps a distinction between those and "single-event memories", which is used throughout this thesis. In short, it is understood that when an agent retrieves an episodic memory they are not just thinking about an individual event of the past but also "reliving" that experience and generating further experiences from it. This additional aspect of episodic memories introduces questions about e.g. replay and reliving experiences which are out of the scope for this study, justifying the choice for the alternative name.

communities. It is widely known that, when training a statistical model on data, individual datapoints are not presentative of, and only provide partial information about, the full underlying process generating the data. Even more so, the presence of noise in the observation of the data might limit even further the degree to which one datapoint can contribute to the full understanding of the process that one is trying to learn. Therefore, to better extract meaningful information when learning from data it is useful – and indeed common practice in ML – to consider the aggregate of several datapoints. In light of this, utilizing individual memories for learning would seem like an inneficient approach at best, or even a damaging one.

Others in the computational neuroscience and ML communities have noticed this problem and engaged with it in different ways: A seminal study is the work by Lengyel and Dayan (2008), where authors explored a sequential decision task in which the agent could either utilise an episodic controller (i.e. single-event memory controller) or a model-based controller to decide which action to take next, and had the goal of maximizing total accumulated reward in an episode. The episodic controller consisted of keeping track of past decisions and the reward associated with those, and simply repeating the decisions that led to higher rewards; the model-based one involved learning a model of the world and using this model to determine the best course of action according to its expected trajectory. The authors then found with simulations that the agent endowed with the episodic controller could outperform a non-episodic one in situations of computational complexity and low-data regime, presenting a possible explanation for the benefits for using individual memories in a sequential decision task.

More recently, work by Sprechmann et al. (2018), inspired by neuroscience research and the complementary learning systems (CLS) framework (McClelland and O'Reilly (1995)), proposed an ML model that used memories of individual datapoints to locally augment the predictions of neural networks trained in supervised learning tasks. They found that this procedure improved the performance of the models in common benchmarks and was helpful mitigating catastrophic forgetting in continual learning tasks.

Even though the works referenced above shed light on the possible advantages and limitations of single-event memory systems, neither conducted a comprehensive mathematical investigation of the benefits and limitations associated with the utilisation of single-event memories in learning and prediction. In this chapter we analyse mathematically and validate with computer simulations the influence of using single-event memories in a supervised learning task (i.e. predicting an input-output mapping).

Section 2.1 explains the problem setting and the different models used to explore the question on the usefulness of single-event emmories. Then we calculate the performance for the differen models in Section 2.2, which is incomplete until we calculate different averages over nearest neighbour memories in 2.3. The theoretical and numerical results are presented and analysed in Section 2.4, both presenting the inneficient scaling of the memory methods with the dimensionality of the problem. Then Sections 2.5 and 2.6 extend the results respectively to out-of-model class learning, and multiple memories models. We show how for more complex problems such as the out-of-model class the strategy of using memories to improve ones model could prove helpful. In short, this work presents a novel theoretical approach to study the normative appropriateness of single-event memories in learning, and connects the derived results with prior work in the literature.

2.1 Problem setup

Consider a supervised learning setting, like the one from Section A, and consider that the mapping being learned is linear and without noise,

$$y = f(\mathbf{x}) = \mathbf{w}^* \cdot \mathbf{x}\,, \tag{2.1}$$

where $\mathbf{x}$ and $\mathbf{w}^*$ are vectors in $\mathbb{R}^N$, which we assume are sampled from isotropic Gaussians,

$$\mathbf{x} \sim \mathcal{N}(0, I) \tag{2.2a}$$

$$\mathbf{w}^* \sim \mathcal{N}(0, I)\,. \tag{2.2b}$$

Assuming also that there is no noise in the mapping f from $\mathbf{x}$ to y leads to a joint distribution $p(\mathbf{x},y) = p(\mathbf{x})\delta(y - f(\mathbf{x}))$ and the assumption on the distributions above leads to the variance of the output to scale as the dimensionality of the space,

$$\mathbb{V}[y] = \left\langle (\mathbf{w}^* \cdot \mathbf{x})^2 \right\rangle = \left\langle \|\mathbf{w}^*\|^2 \right\rangle = N \,. \tag{2.3}$$

In line with usual experiences in the life of an animal, we consider the data batches to be seen only once and to consist of a single training pair, $(\mathbf{x}_m, y_m)$, where m indexes the trial number.

To explore the advantages single-event memories might bring in the task presented above, we study two different single-event memory models, and contrast their performance with a standard model of learning (henceforth called "parametric model" or "baseline model"). We present these three models below.

Parametric model

The baseline model is a parametric model, the standard in ML practice, which follows the common ideas of statistical learning for a supervised learning problem, as explained in Appendix A. The model lives in the same parametric family as the true mapping, f, that is,

$$\hat{y}_{\mathbf{w}}(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} \,. \tag{2.4}$$

As this model is in the correct parametric family it is the natural baseline for the performance in the task, so we contrast its performance to the other two models, which make use of single-event memories.

Learning in this model consists of following the direction of the steepest gradient, such that we update the parameter vector with the gradient of the quadratic loss,

$$\ell(\mathbf{w}, \mathbf{x}_m, y_m) = \frac{1}{2}(y_m - \mathbf{w} \cdot \mathbf{x}_m)^2 \tag{2.5a}$$

$$\Delta \mathbf{w} = -\eta \nabla_{\mathbf{w}} \ell(\mathbf{w}, \mathbf{x}_m, y_m) = \eta (y_m - \mathbf{w} \cdot \mathbf{x}_m) \mathbf{x}_m \,, \tag{2.5b}$$

where η is the learning rate – a hyperparameter that one can optimise over depending on the task.

Fully memory-based model

The second model considered is an entirely (single-event) memory-based model. This means it does not have any learned parameters and only uses memories of previously seen data to make its predictions. It does so by comparing the test input $\mathbf{x}$ with the stored "memories" of previously seen inputs $\{\mathbf{x}_m\}$, selecting that of the memories that is the closest in input space to the test input, and outputting as its prediction the output, y_m, relative to that memory. Mathematically, at a training trial t we store the data as a new memory,

$$\mathcal{D}_M \leftarrow \mathcal{D}_{M-1} \cup (\mathbf{x}_m, y_m) \ , \tag{2.6}$$

where $\mathcal{D}_M$ is the collection of memories stored, indexed by the number of memories, M.

Then, at test time, when trying to predict the output for input $\mathbf{x}$, this model uses the nearest neighbour to $\mathbf{x}$ amongst the memories,

$$\hat{y} = y_{\text{NN}} \quad \text{where} \ \ {\text{NN}} = \arg\min_{m \in \mathcal{D}_M} d(\mathbf{x}, \mathbf{x}_m) \,. \tag{2.7}$$

There is a decision that must be made at this point with regards to the criteria that defines "closeness" in input space. For simplicity, we consider the Euclidean distance in $\mathbb{R}^N$ (the space where the input vectors live),

$$d(\mathbf{x}, \mathbf{x}_m) = ||\mathbf{x} - \mathbf{x}_m||^2 \,, \tag{2.8}$$

where the $|| \cdot ||$ denotes the Euclidean or ℓ_2-norm, given by

$$||\mathbf{v}|| = \sqrt{\mathbf{v} \cdot \mathbf{v}} = \sqrt{\sum_i v_i} \,. \tag{2.9}$$

This model is a simple one to describe and represents the extreme case for the hypothesis of the usefulness of single-event memories. We now consider a hybrid approach that mixes ideas from both models above.

Memory-augmented model

The third model is a memory-augmented (or memory-adjusted) model. It is inspired by work in Sprechmann et al. (2018), where the authors used a similar construct, but did not analyse its performance analytically.

In short, the idea is to take the parametric model and improve it using the memories of previously seen data. More specifically, this model follows the parametric model in that it learns a parameter vector $\mathbf{w}$ using gradient descent (see Equation 2.5b), and alongside this it also stores all datapoints used at train-time in a memory database, $\mathcal{D}_M$, like the previous model. Unlike both previous models, it uses both memories and a parameter vector $\mathbf{w}$ to make its prediction: the memories are accessed at test-time to locally adjust $\mathbf{w}$ for the region close to that particular test point.

Mathematically, for each test point, $\mathbf{x}$, one tries to predict, the model takes the memory closest to the test point,

$$\mathbf{x}_{\text{NN}} = \underset{m \in \mathcal{D}_M}{\arg\min}\ d(\mathbf{x}, \mathbf{x}_m), \tag{2.10}$$

and uses it to adjust the parameters of the model from $\mathbf{w}$ to $\widetilde{\mathbf{w}}$. This can be done in different ways, but since our measure of performance is the loss function, the natural choice is to take an extra gradient step along the empirical loss evaluated at that memory,

$$\widetilde{\mathbf{w}} = \mathbf{w} + \Delta\mathbf{w}_{\text{NN}} \tag{2.11a}$$

$$\begin{aligned} \Delta\mathbf{w}_{\text{NN}} &= -\alpha \nabla_{\mathbf{w}} \ell(\mathbf{w}, \mathbf{x}_{\text{NN}}, y_{\text{NN}}) \\ &= \alpha \left(y_{\text{NN}} - \mathbf{w} \cdot \mathbf{x}_{\text{NN}} \right) \mathbf{x}_{\text{NN}}, \end{aligned} \tag{2.11b}$$

where α is the *adjustment rate* that can also be optimised – independently of the learning rate η.

The adjusted parameter vector $\widetilde{\mathbf{w}}$ is used temporarily for the response due to input $\mathbf{x}$,

$$\hat{y}_{\text{adj}} = \widetilde{\mathbf{w}} \cdot \mathbf{x}, \tag{2.12}$$

then it gets discarded and the model returns to $\mathbf{w}$ for the usual gradient update. A depiction of the memory-adjustment procedure can be seen in Figure 2.1.

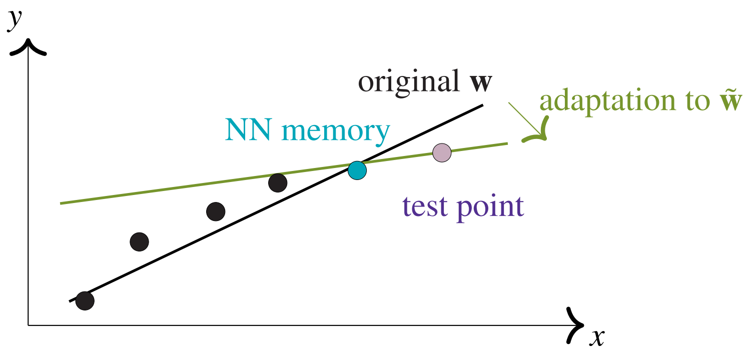

Figure 2.1: A graphical depiction of how the memory-adjustment procedure works. The black and blue dots are previously seen datapoints, that were used to train the parameter vector (the linear fit in black). In order to better predict a given test point (pink point), the closest memory is selected to update the parameters of the model, generating the new linear fit (the green line). This informs a more suitable output for that region of the input space.

2.2 Calculating the performance for the different algorithms

To compare the performance of the different algorithms outlined in the preceding section, we calculate their generalisation error,

$$\mathcal{L}(\mathbf{w}) = \langle \ell(\mathbf{w}, \mathbf{x}, y) \rangle = \frac{1}{2} \left\langle (y - \mathbf{w} \cdot \mathbf{x})^2 \right\rangle . \tag{2.13}$$

By definition this depends on averages over the data distribution, $p(\mathbf{x}, y)$. Furthermore, the nearest neighbour among the stored memories, $\mathbf{x}_{\text{NN}}$, is also stochastic, and for the memory models we need to average over the nearest neighbour contributions also.

We start this section calculating the generalisation error for the baseline. Although it is a common result we include it for completeness and as a warm-up exercise in preparation for the calculation of the improvement given by memory updates in the memory-augmented model. Then we share an intuition behind the nearest neighbour contribution to the performance of the memory algorithms, and calculate their performance. Some details of the calculation of the averages over nearest neighbours are left to Section 2.3.

Parametric model

Here we want to understand how training updates in the parameter vector $\mathbf{w}$ – henceforth referred to as $\Delta\mathbf{w}_m$ to stand for the update due to observing data $(\mathbf{x}_m, y_m)$ –, affect changes in the generalisation error. Following stochastic gradient descent, this update is given by

$$\Delta\mathbf{w}_m = -\eta\nabla_{\mathbf{w}}\ell(\mathbf{w},\mathbf{x}_m,y_m)\,, \tag{2.14}$$

where $(\mathbf{x}_m, y_m)$ is the new training pair, and η is a learning rate.

Recall that we choose the pointwise loss ℓ to be the quadratic loss, so we have

$$\Delta\mathbf{w}_m = -\frac{\eta}{2}\nabla_{\mathbf{w}}\left(y_m - \mathbf{w}\cdot\mathbf{x}_m\right)^2 = +\eta\left(y_m - \mathbf{w}\cdot\mathbf{x}_m\right)\mathbf{x}_m\,. \tag{2.15}$$

After updating from $\mathbf{w}$ to $\mathbf{w}+\Delta\mathbf{w}_m$, the pointwise loss for a given new test pair $(\mathbf{x},y)$ becomes $\ell(\mathbf{w}+\Delta\mathbf{w}_m,\mathbf{x},y)$. However we are not interested in calculating the loss for just one test datapoint, but across all possible datapoints, weighed by their probability, so we want the generalisation error as in Equation 2.13,

$$\mathcal{L}(\mathbf{w}+\Delta\mathbf{w}_m) = \langle\ell(\mathbf{w}+\Delta\mathbf{w}_m,\mathbf{x},y)\rangle_{\mathbf{x},y}\,, \tag{2.16}$$

where the average is taken over the joint distribution

$$p(\mathbf{x},y|\mathbf{w}^*) = p(y|\mathbf{x},\mathbf{w}^*)p(\mathbf{x}) = \delta(\mathbf{w}^*\cdot\mathbf{x}-y)\mathcal{N}(\mathbf{x}|0,I)\,. \tag{2.17}$$

In order to take this average, we need to express the pointwise loss $\ell(\mathbf{w}+\Delta\mathbf{w}_m,\mathbf{x},y)$ in terms of quantities we now. Expanding the squared terms, we have

$$\begin{aligned}\ell(\mathbf{w}+\Delta\mathbf{w}_m,\mathbf{x},y) &= \frac{1}{2}\left[y-\mathbf{w}\cdot\mathbf{x}-\eta\left(y_m-\mathbf{w}\cdot\mathbf{x}_m\right)\left(\mathbf{x}_m\cdot\mathbf{x}\right)\right]^2\\ &= \frac{1}{2}(y-\mathbf{w}\cdot\mathbf{x})^2 - \eta(y-\mathbf{w}\cdot\mathbf{x})(y_m-\mathbf{w}\cdot\mathbf{x}_m)(\mathbf{x}_m\cdot\mathbf{x}) \qquad (2.18)\\ &\quad + \frac{\eta^2}{2}(y_m-\mathbf{w}\cdot\mathbf{x}_m)^2(\mathbf{x}\cdot\mathbf{x}_m)^2\,.\end{aligned}$$

Defining the discrepancy between the true vector generating the data and the parameter vector,

$$\Delta\mathbf{w}^* = \mathbf{w}^* - \mathbf{w}, \tag{2.19}$$

we can write

$$\begin{aligned}\ell(\mathbf{w}+\Delta\mathbf{w}_m,\mathbf{x},y) = \frac{1}{2}(\Delta\mathbf{w}^*\cdot\mathbf{x})^2 - \eta(\Delta\mathbf{w}^*\cdot\mathbf{x})(\Delta\mathbf{w}^*\cdot\mathbf{x}_m)(\mathbf{x}_m\cdot\mathbf{x}) \\ + \frac{\eta^2}{2}(\Delta\mathbf{w}^*\cdot\mathbf{x}_m)^2(\mathbf{x}\cdot\mathbf{x}_m)^2 .\end{aligned} \tag{2.20}$$

Finally, we can average over all the data variables using the known results for the Gaussian variables,

$$\langle \mathbf{x}\mathbf{x}^\intercal \rangle = \langle \mathbf{x}_m\mathbf{x}_m^\intercal \rangle = I \tag{2.21a}$$

$$\left\langle (\mathbf{x}_m\cdot\mathbf{x}_m)^2 \|\mathbf{x}_m\|^2 \right\rangle = (N+2)I, \tag{2.21b}$$

and obtain the generalisation error after a training update,

$$\begin{aligned}\mathcal{L}(\mathbf{w}+\Delta\mathbf{w}_m) &= \langle \ell(\mathbf{w}+\Delta\mathbf{w}_m,\mathbf{x},y)\rangle \\ &= \left\langle \|\Delta\mathbf{w}^*\|^2 \right\rangle \left[1-\eta+\frac{1}{2}\eta^2(N+2)\right] .\end{aligned} \tag{2.22}$$

We further notice that the first term in the expansion of $\ell(\mathbf{w}+\Delta\mathbf{w}_m,\mathbf{x},y)$, after averaging, is equal to the generalisation error before the training update, and write

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2}\left\langle \|\Delta\mathbf{w}^*\|^2 \right\rangle \tag{2.23a}$$

$$\mathcal{L}(\mathbf{w}+\Delta\mathbf{w}_m) = \mathcal{L}(\mathbf{w})\left[1-\eta+\frac{1}{2}\eta^2(N+2)\right] . \tag{2.23b}$$

Since η is a free hyperparameter, we can tune it to maximise the drop in generalisation error with each datapoint. Doing so we find the equation for the optimal learning rate and the corresponding optimal average change in the generalisation error,

$$\eta^* = \frac{1}{N+2} \tag{2.24a}$$

$$\mathcal{L}(\mathbf{w}+\Delta\mathbf{w}_m) = \mathcal{L}(\mathbf{w})\left[1-\frac{1}{N+2}\right] . \tag{2.24b}$$

One can also write the result above as a function of the initial parameter vector, $\mathbf{w}_0$, and number of learning steps M taken up to that point by recurrently applying the same update,

$$\mathcal{L}(\mathbf{w}_0, \mathrm{t}) = \mathcal{L}(\mathbf{w}_0)\left[1 - \frac{1}{N+2}\right]^{\mathrm{t}} \approx \mathcal{L}(\mathbf{w}_0)\exp\left(-\frac{\mathrm{t}}{N+2}\right). \tag{2.25}$$

Intuition behind using nearest neighbours

In both the memory-augmented and fully memory-based models, the idea of using the memory closest to test point $\mathbf{x}$ is informed by the notion that one can use the shared information from the closeness of test point $\mathbf{x}$ and $\mathbf{x}_{\mathrm{NN}}$ to improve the estimation of target y by using the memorised pair $(x_{\mathrm{NN}}, y_{\mathrm{NN}})$. This is an assumption on the continuity of the mapping f.

Ideally there would be a relevant memory near in input space to the test point, $\mathbf{x}_{\mathrm{NN}} \approx \mathbf{x}$, and the use of memory would be maximally informative – this is evidently true in the limit of the number of memories going to infinity, $M \to \infty$. However, for finite M it will generally not be the case that there is a memory close enough to the test point, and memories will only be partially informative about the correct output to the test point. This is especially true for high dimension input spaces, $\mathbb{R}^N$ with large N.

In fact, we can decompose the memory nearest to the test point, $\mathbf{x}_{\mathrm{NN}}$, into a component aligned with the test point - which is the informative part of the memory for the task - and a perpendicular component,

$$\mathbf{x}_{\mathrm{NN}} = z_{\mathrm{NN}}\widehat{\mathbf{x}} + \mathbf{x}_{\mathrm{NN}}^{\perp}, \tag{2.26}$$

where $\widehat{\mathbf{x}} = \mathbf{x}/||\mathbf{x}||$ is the unit vector in the direction of the test point, and $\mathbf{x}_{\mathrm{NN}}^{\perp} \cdot \mathbf{x} = 0$ is the part of the nearest neighbour memory that is orthogonal to the test point. This can be seen graphically in Figure 2.2.

Since each memory $(\mathbf{x}_m, y_m)$ (and in particular the nearest neighbour one) is a stochastic draw from the data distribution, both components will be noisy, and not fully informative about the mapping f. However, the information about the correct prediction for test point can only be contained in the memories in the component that is aligned with it, z_{NN}. In that sense the perpendicular component can be seen as the main source of noise when using the information in the nearest neighbour memories for prediction.

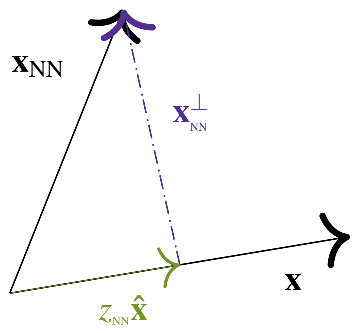
$\mathbf{x}_{\text{NN}}$
$\mathbf{x}_{\text{NN}}^{\perp}$
$\mathbf{x}$
$z_{\text{NN}}\hat{\mathbf{x}}$

Figure 2.2: The decomposition of the selected memory $\mathbf{x}_{\text{NN}}$ into $z_{\text{NN}}\hat{\mathbf{x}}$, a component aligned with the test point $\mathbf{x}$, and $\mathbf{x}_{\text{NN}}^{\perp}$, a perpendicular component. In high-dimensional spaces vectors become further apart, and the largest contribution will be from the perpendicular component, suppressing signal with noise.

Therefore, the scalar z_{NN} is the contribution to $\mathbf{x}_{\text{NN}}$ along the relevant direction, parallel to the test point, and all the rest is accounted by $\mathbf{x}_{\text{NN}}^{\perp}$. If one has the perfect memory for a given test point, $\mathbf{x}_{\text{NN}} = \mathbf{x}$, then the perpendicular term is identically zero, $\mathbf{x}_{\text{NN}}^{\perp} = \mathbf{0}$, and the signal is maximal, $z_{\text{NN}} = ||\mathbf{x}||$, otherwise there is a trade-off between z_{NN} and $||\mathbf{x}_{\text{NN}}^{\perp}||$. This is in line with the intuition articulated before.

Fully memory-based model

Recall that this model simply uses the output of the corresponding nearest neighbour as its prediction,

$$\hat{y} = y_{\text{NN}} \quad \text{where} \quad {}_{\text{NN}} = \arg\min_{m \in \mathcal{M}} d(\mathbf{x}, \mathbf{x}_m)\,. \tag{2.27}$$

As described in Section 2.1, we use the generalisation error, $\mathcal{L}$, to estimate the average error for each model and we want to understand how changes to the model (in this case, the set of memories $\mathcal{M}$) propagate into improvements to the error.

Applying Equation 2.27 into the quadratic loss, we have

$$\begin{aligned}\mathcal{L}(y_{\mathrm{NN}};M) &= \left\langle \frac{1}{2}(y - y_{\mathrm{NN}})^2 \right\rangle \\ &= \frac{1}{2}\left\langle \left[(\mathbf{w}^* \cdot \mathbf{x})^2 - 2(\mathbf{w}^* \cdot \mathbf{x})(\mathbf{w}^* \cdot \mathbf{x}_{\mathrm{NN}}) + (\mathbf{w}^* \cdot \mathbf{x}_{\mathrm{NN}})^2\right] \right\rangle .\end{aligned} \tag{2.28}$$

Assuming that $\mathbf{w}^* \sim \mathcal{N}(0,I)$ and that the standard Gaussian vector $\mathbf{x}$ self-averages as $\|\mathbf{x}\| \approx \sqrt{N}$, we can write the different expectations as

$$\left\langle (\mathbf{w}^* \cdot \mathbf{x})^2 \right\rangle = \langle \mathbf{x} \cdot \mathbf{x} \rangle = N \tag{2.29a}$$

$$\langle (\mathbf{w}^* \cdot \mathbf{x})(\mathbf{w}^* \cdot \mathbf{x}_{\mathrm{NN}}) \rangle = \langle \mathbf{x} \cdot \mathbf{x}_{\mathrm{NN}} \rangle = \langle z_{\mathrm{NN}} \|\mathbf{x}\| \rangle = \sqrt{N}\,\langle z_{\mathrm{NN}} \rangle \tag{2.29b}$$

$$\begin{aligned}\left\langle \mathbf{w}^* \cdot \mathbf{x}_{\mathrm{NN}}^2 \right\rangle &= \left\langle \|\mathbf{x}_{\mathrm{NN}}\|^2 \right\rangle = \left\langle z_{\mathrm{NN}}^2 (\hat{\mathbf{x}} \cdot \hat{\mathbf{x}}) \right\rangle + \left\langle \|\mathbf{x}_{\mathrm{NN}}^{\perp}\|^2 \right\rangle + 2\left\langle z_{\mathrm{NN}} \hat{\mathbf{x}} \cdot \mathbf{x}_{\mathrm{NN}}^{\perp} \right\rangle \\ &= \left\langle z_{\mathrm{NN}}^2 \right\rangle + \left\langle \|\mathbf{x}_{\mathrm{NN}}^{\perp}\|^2 \right\rangle ,\end{aligned} \tag{2.29c}$$

and obtain the result

$$\mathcal{L}(y_{\mathrm{NN}};M) = \frac{1}{2}\left[N - 2\sqrt{N}\,\langle z_{\mathrm{NN}} \rangle + \left\langle z_{\mathrm{NN}}^2 \right\rangle + \left\langle \|\mathbf{x}_{\mathrm{NN}}^{\perp}\|^2 \right\rangle \right] . \tag{2.30}$$

The loss initially is order N, as expected since we defined the output variable y to be order $\sqrt{N}$, in Equation 2.3.

We derive explicit expressions for the averages $\langle z_{\mathrm{NN}} \rangle, \left\langle z_{\mathrm{NN}}^2 \right\rangle$ and $\left\langle \|\mathbf{x}_{\mathrm{NN}}^{\perp}\|^2 \right\rangle$ in Section 2.3.

Memory-augmented model

The generalisation error for the memory-augmented model involves two different gradient steps: the training step, identical to what was done for the the parametric model, leading up to Equation 2.24b; and the memory-adjustment step, which is the distinctive aspect of this model.

The change in the parameter vector due to memory adjustment is given by a step in the direction of gradient descent for the quadratic loss, evaluated on the nearest neighbour memory

$$\Delta\mathbf{w}_{\text{NN}} = -\alpha\nabla_{\mathbf{w}}\ell(\mathbf{w}, \mathbf{x}_{\text{NN}}, y_{\text{NN}}) = \alpha(\Delta\mathbf{w}^* \cdot \mathbf{x}_{\text{NN}})\mathbf{x}_{\text{NN}}\,, \tag{2.31}$$

where we use again the discrepancy between the true vector generating the data and the parameter vector,

$$\Delta\mathbf{w}^* = \mathbf{w}^* - \mathbf{w}\,. \tag{2.32}$$

The memory-adjustment update in Equation 2.31 is similar to the learning update in Equation 2.14, in the sense that both are steps in the direction of gradient descent for the quadratic loss, but with the important distinction that now the loss is evaluated on the nearest neighbour tuple $(\mathbf{x}_{\text{NN}}, y_{\text{NN}})$ instead of a new draw from the training data distribution. This means we need to take averages over the identity of the nearest neighbour among the memories.

As in Equation 2.20, we write the pointwise loss for a test point, $\ell(\mathbf{w} + \Delta\mathbf{w}_{\text{NN}}, \mathbf{x}, y)$, and expand the squared terms that depend on the update,

$$\begin{aligned}
\ell(\mathbf{w} + \Delta\mathbf{w}_{\text{NN}}, \mathbf{x}, y) &= \frac{1}{2}\left[y - \mathbf{w}\cdot\mathbf{x} - \alpha\left(y_{\text{NN}} - \mathbf{w}\cdot\mathbf{x}_{\text{NN}}\right)\left(\mathbf{x}_{\text{NN}}\cdot\mathbf{x}\right)\right]^2 \\
&= \frac{1}{2}(y - \mathbf{w}\cdot\mathbf{x})^2 - \alpha(y - \mathbf{w}\cdot\mathbf{x})(y_{\text{NN}} - \mathbf{w}\cdot\mathbf{x}_{\text{NN}})(\mathbf{x}_{\text{NN}}\cdot\mathbf{x}) \\
&\quad + \frac{\alpha^2}{2}(y_{\text{NN}} - \mathbf{w}\cdot\mathbf{x}_{\text{NN}})^2(\mathbf{x}\cdot\mathbf{x}_{\text{NN}})^2 \\
&= \frac{1}{2}(\Delta\mathbf{w}^*\cdot\mathbf{x})^2 - \alpha(\Delta\mathbf{w}^*\cdot\mathbf{x})(\Delta\mathbf{w}^*\cdot\mathbf{x}_{\text{NN}})(\mathbf{x}_{\text{NN}}\cdot\mathbf{x}) \\
&\quad + \frac{\alpha^2}{2}(\Delta\mathbf{w}^*\cdot\mathbf{x}_{\text{NN}})^2(\mathbf{x}\cdot\mathbf{x}_{\text{NN}})^2\,.
\end{aligned} \tag{2.33}$$

Then, averaging over the relevant quantities we can obtain the generalisation error,

$$\mathcal{L}(\mathbf{w} + \Delta\mathbf{w}_{\text{NN}}) = \langle \ell(\mathbf{w} + \Delta\mathbf{w}_{\text{NN}}, \mathbf{x}, y)\rangle_{\mathbf{x},\mathbf{x}_{\text{NN}}}\,, \tag{2.34}$$

for which we need to use the decomposition from Equation 2.26, $\mathbf{x}_{\text{NN}} = z_{\text{NN}}\widehat{\mathbf{x}} + \mathbf{x}_{\text{NN}}^{\perp}$, noticing that $\mathbf{x}\cdot\widehat{\mathbf{x}} = ||\mathbf{x}||$ and $\mathbf{x}\cdot\mathbf{x}_{\text{NN}}^{\perp} = 0$, so that

$$\begin{aligned}\mathcal{L}(\mathbf{w}+\Delta\mathbf{w}_{\text{NN}}) &= \left\langle \frac{1}{2}(\Delta\mathbf{w}^*\cdot\mathbf{x})^2 - \alpha(\Delta\mathbf{w}^*\cdot\mathbf{x})\left[\Delta\mathbf{w}^*\cdot(z_{\text{NN}}\widehat{\mathbf{x}}+\mathbf{x}^{\perp}_{\text{NN}})\right](z_{\text{NN}}\widehat{\mathbf{x}}\cdot\mathbf{x}))\right\rangle \\ &\quad + \left\langle \frac{\alpha^2}{2}\left[\Delta\mathbf{w}^*\cdot(z_{\text{NN}}\widehat{\mathbf{x}}+\mathbf{x}^{\perp}_{\text{NN}})\right]^2(z_{\text{NN}}\widehat{\mathbf{x}}\cdot\mathbf{x})^2\right\rangle \\ &= \mathcal{L}(\mathbf{w}) - \alpha\left[\left\langle z^2_{\text{NN}}(\Delta\mathbf{w}^*\cdot\mathbf{x})^2\right\rangle + \left\langle z_{\text{NN}}\|\mathbf{x}\|(\Delta\mathbf{w}^*\cdot\mathbf{x})(\Delta\mathbf{w}^*\cdot\mathbf{x}^{\perp}_{\text{NN}})\right\rangle\right] \\ &\quad + \frac{\alpha^2}{2}\left[\left\langle z^4_{\text{NN}}(\Delta\mathbf{w}^*\cdot\mathbf{x})^2\right\rangle + \left\langle z^2_{\text{NN}}\|\mathbf{x}\|^2(\Delta\mathbf{w}^*\cdot\mathbf{x}^{\perp}_{\text{NN}})^2\right\rangle\right. \\ &\quad \left. +2z^3_{\text{NN}}\|\mathbf{x}\|(\Delta\mathbf{w}^*\cdot\mathbf{x})(\Delta\mathbf{w}^*\cdot\mathbf{x}^{\perp}_{\text{NN}})^2\right]. \end{aligned} \tag{2.35}$$

Assuming that averages over the test vector, $\mathbf{x}$, can be done first, we get

$$\begin{aligned}\mathcal{L}(\mathbf{w}+\Delta\mathbf{w}_{\text{NN}}) &= \mathcal{L}(\mathbf{w}) - \alpha\left\langle\|\Delta\mathbf{w}^*\|^2\right\rangle\left\langle z^2_{\text{NN}}\right\rangle \\ &\quad + \frac{1}{2}\alpha^2\left[\left\langle\|\Delta\mathbf{w}^*\|^2\right\rangle\left\langle z^4_{\text{NN}}\right\rangle + \left\langle\|\Delta\mathbf{w}^*\|^2\right\rangle\left\langle z^2_{\text{NN}}\|\mathbf{x}^{\perp}_{\text{NN}}\|^2\right\rangle\right].\end{aligned} \tag{2.36}$$

This last assumption is an approximation, since by definition the nearest neighbour quantities depend on the value of $\mathbf{x}$. When estimating these averages over nearest neighbours in Section 2.3 we do assume that the test point self-averages, which justifies the step above, but can be a source of error. Another approximation made is that the discrepancy vector $\Delta\mathbf{w}^*$ decouples from the nearest neighbour quantities. This also cannot be the case, as the same data points that become the memories were originally used for training, hence they determined the evolution of the parameter vector $\mathbf{w}$. Nonetheless, Equation 2.36 should be approximately correct in high-dimensional spaces and for situations with a slow learning rate, $\eta \approx 0$.

Optimising the adjustment rate to decrease the generalisation error as much as possible per step, we obtain

$$\alpha^* = \arg\min_{\alpha}\mathcal{L}(\mathbf{w}+\Delta\mathbf{w}_{\text{NN}}) = \frac{\left\langle z^2_{\text{NN}}\right\rangle}{\left\langle z^4_{\text{NN}}\right\rangle + \left\langle z^2_{\text{NN}}\|\mathbf{x}^{\perp}_{\text{NN}}\|^2\right\rangle}, \tag{2.37}$$

and the associated improvement ε in the generalisation error

$$\varepsilon = \frac{\left\langle z^2_{\text{NN}}\right\rangle^2}{\left\langle z^4_{\text{NN}}\right\rangle + \left\langle z^2_{\text{NN}}\|\mathbf{x}^{\perp}_{\text{NN}}\|^2\right\rangle}, \tag{2.38a}$$

$$\mathcal{L}(\mathbf{w}+\Delta\mathbf{w}_{\text{NN}}) = \mathcal{L}(\mathbf{w})\left[1-\varepsilon\right]. \tag{2.38b}$$

2.3 Expectations over Nearest Neighbour quantities

Here we present the calculations of the averages over nearest neighbour contributions in more detail. As shown in Equations 2.30, 2.37 and 2.38a, we are interested in averages of the kind,

$$\left\langle z_{\text{NN}}^{p} \|\mathbf{x}_{\text{NN}}^{\perp}\|^{2q} \right\rangle , \tag{2.39}$$

where p, q are integers. As described in the previous section, every memory $\mathbf{x}_m$ can be decomposed into a component aligned with the test point, $z_m\hat{\mathbf{x}}$, and a perpendicular one, $\mathbf{x}_m^{\perp}$, but we do not simply want the averages over the distribution $p(z_m, \|\mathbf{x}_m^{\perp}\|^2 \,|\, \mathbf{x})$ defined over a memory m. Instead, we want those conditioned on the fact that a *particular* memory is the nearest neighbour to $\mathbf{x}$. Therefore, taking those averages is a non-trivial problem, and this work presents a method to derive those quantities.

We start considering the distance between the test point $\mathbf{x}$ and a memory $\mathbf{x}_m$,

$$\gamma_m \equiv \|\mathbf{x}_m - \mathbf{x}\|^2 = (z_m - \|\mathbf{x}\|)^2 + \|\mathbf{x}_m^{\perp}\|^2 . \tag{2.40}$$

The conditioning procedure takes the collection of distances $\{\gamma_m\}$ for all memories, orders them $\{\gamma_{(m)}\}$ – where lower subscript indices represent smaller distances and closer points –, and selects the nearest neighbour, $\gamma_{(1)} \equiv \gamma_{\text{NN}}$. Then the relevant probability distribution over the pair from the decomposition in Equation 2.26, $(z, \|\mathbf{x}^{\perp}\|^2)$ is the one that is conditioned on the identity of the nearest neighbour, $\mathbf{x}_m \equiv \mathbf{x}_{\text{NN}}$, that is,

$$p(z, \|\mathbf{x}^{\perp}\|^2 \mid \text{NN}, \mathbf{x}) , \tag{2.41}$$

where the conditional NN indicates that the memory we picked – say, m – is the nearest neighbour to $\mathbf{x}$. Now we need to decompose this distribution in terms of quantities we can calculate. First we can use Bayes' theorem to write

$$p(z, \|\mathbf{x}^{\perp}\|^2 \mid \text{NN}, \mathbf{x}) = \frac{p(z, \|\mathbf{x}^{\perp}\|^2|\mathbf{x})p(\text{NN}|z, \|\mathbf{x}^{\perp}\|^2, \mathbf{x})}{p(\text{NN}|\mathbf{x})} . \tag{2.42}$$

The probability distributions are: a prior $p(z, \|\mathbf{x}^{\perp}\|^2|\mathbf{x})$, which tells us the prior probability that a memory m has a decomposition of values $(z, \|\mathbf{x}^{\perp}\|^2)$ under the direction of $\mathbf{x}$; the likelihood $p(\text{NN}|z, \|\mathbf{x}^{\perp}\|^2, \mathbf{x})$, which reads: the probability that a given memory m we picked is the nearest neighbour memory to $\mathbf{x}$ given we know its decomposition has the values $(z, \|\mathbf{x}^{\perp}\|^2)$; and the evidence $p(\text{NN}|\mathbf{x})$, which is the probability that the memory we picked, m, is the nearest neighbour memory to $\mathbf{x}$ without knowing anything else about it – this is usually seen as a normalisation.

Now we can use marginalisation to introduce another variable, γ, the distance between the test point $\mathbf{x}$ and the memory $\mathbf{x}_m$ (that becomes the nearest neighbour memory $\mathbf{x}_{\text{NN}}$ after the conditioning NN). This is needed because we only know how to rank memories by closest or farthest in a real line – the scalar space of distances – not in the space of $(z, \|\mathbf{x}^{\perp}\|^2) \in \mathbb{R}^2$ – despite those being the quantities we are interested in. Then,

$$\begin{aligned} p(z, \|\mathbf{x}^{\perp}\|^2|\text{NN}, \mathbf{x}) &= \frac{p(z, \|\mathbf{x}^{\perp}\|^2|\mathbf{x})}{p(\text{NN}|\mathbf{x})} \int \mathrm{d}\gamma\, p(\text{NN}, \gamma|z, \|\mathbf{x}^{\perp}\|^2, \mathbf{x}) \\ &= p(z, \|\mathbf{x}^{\perp}\|^2|\mathbf{x}) \int \mathrm{d}\gamma\, p(\gamma|z, \|\mathbf{x}^{\perp}\|^2, \mathbf{x})\, \frac{p(\text{NN}|\gamma, \mathbf{x})}{p(\text{NN}|\mathbf{x})} \end{aligned} \tag{2.43}$$

where we used the fact that the evidence does not depend on γ, so it can go inside the integral; and also the fact that conditioned on γ the distribution for the identity of the nearest neighbour is independent of the components of the decomposition of the memory.

Here we need results from Order Statistics, a subfield of Statistics which concerns the study of probability distributions for ordered sets. A brief introduction on Order Statistics is given in Appendix B. Using Equations B.5 and B.11 to substitute the densities over the identity of the nearest neighbour, we can write an expression for the uknown terms in the right-hand side of Equation 2.43 as

$$\frac{p(\text{NN}|\gamma, \mathbf{x})}{p(\text{NN}|\mathbf{x})} = M\,[1 - C(\gamma)]^{M-1}\ , \tag{2.44}$$

and using the fact that given z, $\|\mathbf{x}^{\perp}\|$ and $\mathbf{x}$ the distance γ is completely determined through $\gamma = \|\mathbf{x}_m - \mathbf{x}\|^2 = (z - \|\mathbf{x}\|)^2 + \|\mathbf{x}^{\perp}\|^2$ (Equation 2.40), we also have that

$$p(\gamma|z, \|\mathbf{x}^{\perp}\|^2, \mathbf{x}) = \delta\left(\gamma - (z - \|\mathbf{x}\|)^2 - \|\mathbf{x}^{\perp}\|^2\right)\,. \tag{2.45}$$

Collecting those results we can use the Dirac delta to do the integral, and write

$$p(z, ||\mathbf{x}^{\perp}||^2 \mid \textsc{nn}, \mathbf{x}) = p(z, ||\mathbf{x}^{\perp}||^2|\mathbf{x})\, M \left[1 - C_\gamma((z - ||\mathbf{x}||)^2 + ||\mathbf{x}^{\perp}||^2)\right]^{M-1}, \quad (2.46)$$

where M is the number of memories and $C(\gamma)$ is the cumulative distribution function (cdf) for the distance, γ, calculated in Appendix E to be approximately a Gaussian distribution,

$$\gamma \sim \mathcal{N}\left(||\mathbf{x}||^2 + N, 4||\mathbf{x}||^2 + 2N\right). \quad (2.47)$$

A description for Equation 2.46 in words is as follows: given the known prior distribution for the quantities $(z, ||\mathbf{x}^{\perp}||)$ for any memory, what is the shifted distribution for those quantities if we know that the memory picked is the nearest neighbour to the test point, $\mathbf{x}$. This can be done because the pair $(z, ||\mathbf{x}^{\perp}||)$ fully determines the distance γ, which is the scalar quantity used to rank the memories. Through the distance (and indirectly through the components $(z, ||\mathbf{x}^{\perp}||)$) one can estimate how likely it is that the memory picked is the nearest neighbour to the test point. From Order Statistics we obtain that the values more likely are those that make the cdf $C_\gamma((z - ||\mathbf{x}||)^2 + ||\mathbf{x}^{\perp}||^2)$ closer to zero, and more so when there are more memories (because with more memories it is more likely to get a memory close to the test point).

We can then take the expectation of powers of those quantities, as discussed,

$$\begin{aligned} \left\langle z_{\mathrm{NN}}^p ||\mathbf{x}_{\mathrm{NN}}^{\perp}||^{2q} \right\rangle &\equiv \int \mathrm{d}z \mathrm{d}||\mathbf{x}^{\perp}||^2 \, z^p ||\mathbf{x}^{\perp}||^{2q} \, p(z, ||\mathbf{x}^{\perp}||^2|\textsc{nn}, \mathbf{x}) \\ &= \int \mathrm{d}z \mathrm{d}||\mathbf{x}^{\perp}||^2 \, z^p ||\mathbf{x}^{\perp}||^{2q} \, p(z, ||\mathbf{x}^{\perp}||^2) \\ &\quad \times M \left[1 - C_\gamma((z - ||\mathbf{x}||)^2 + ||\mathbf{x}^{\perp}||^2)\right]^{M-1}, \end{aligned} \quad (2.48)$$

where $p, q \in \mathbb{N}$.

The missing piece now is the *prior* distribution for the pair $(z, ||\mathbf{x}^{\perp}||^2)$ before conditioning on the nearest neighbour. The steps involved in finding this distribution and solving the integral in Equation 2.48 do not provide further intuition to the problem that was not given already, so they are omitted from the main text and can be found in Appendix E. In short, the distribution for $(z, \|\mathbf{x}^{\perp}\|^2)$ decouples if one is not conditioning on other quantities, and both can be seen as approximately Gaussian. The results are

$$\left\langle z_{\text{NN}}^{p} \|\mathbf{x}_{\text{NN}}^{\perp}\|^{2q} \right\rangle = \left\langle \left[\frac{u}{\sqrt{3}} - s\sqrt{\frac{2}{3}} \right]^{p} \left[N + \sqrt{\frac{2N}{3}} \left(u\sqrt{2} + s \right) \right]^{q} \zeta_{\text{NN}}(s, M) \right\rangle_{u,s} , \tag{2.49}$$

where u, s are standard Gaussian variables, and we defined the nearest neighbour factor and cdf for a standard Gaussian as

$$\zeta_{\text{NN}}(s, M) = M[1 - \Phi(s)]^{M-1} \tag{2.50a}$$

$$\Phi(s) = \int_{-\infty}^{s} \frac{\mathrm{d}q}{\sqrt{2\pi}} e^{-\frac{1}{2}q^2} . \tag{2.50b}$$

Average functions, $\beta_k(M)$

There are two averages over different standard Gaussian variables, s and u. The average over the u variable depends only on the powers u^k; therefore it is straightforward with the values being the moments of a standard normal distribution.

The average over s is not as immediate, since the nearest neighbour factor, ζ_{NN}, modulates the expectation. In the left panel of Figure 2.3 we see how ζ_{NN} functions similarly to a step function, selecting the negative semi-half of the real line, especially for larger values of M – it becomes sharper and shifts towards selecting more negative values as M increases.

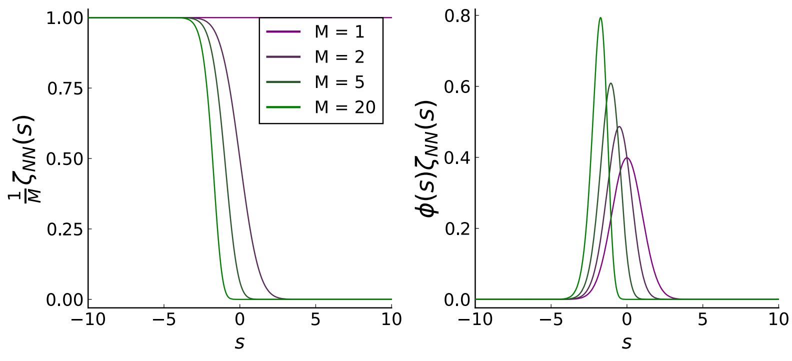

Figure 2.3: The integrands of the expectations in Equation 2.49 as functions of s and M. The nearest neighbour factor, $\zeta_{\text{NN}}(s, M)$, is defined in Equation 2.50a and modulates the expectation; $\phi(s)$ is the pdf for a standard Gaussian, over which we take the expected values.

Alternatively, one can see the effect that ζ_{NN} has on the averages as shifting and reshaping the Gaussian distribution, $\phi(s)$, when calculating its moments, s^k. This leads to a sharper and more negative Gaussian distribution, which can be seen in the right panel of Figure 2.3. The shift and scaling of the expectation values for the powers s^k follows from this: as the number of memories increases the mean becomes more negative and the variance smaller.

Since the averages of different powers of s modulated by the nearest neighbour factor, ζ_{NN}, come up frequently in the next sections, it is convenient to define them as

$$\begin{aligned}\beta_k(M) &\equiv \Big\langle (-1)^k s^k \zeta_{\text{NN}}(s,M) \Big\rangle_s \\ &= (-1)^k M \int_{-\infty}^{\infty} \frac{\mathrm{d}s}{\sqrt{2\pi}} e^{-\frac{1}{2}s^2} s^k \left[1-\Phi(s)\right]^{M-1} .\end{aligned} \tag{2.51}$$

As the β_ks are integrals of just one scalar variable, they can be easily integrated numerically. The results for the lower powers is shown in Figure 2.4. In order to better understand the behaviour of these functions, we derive approximate analytical results for their scaling when the number of memories, M, grows.

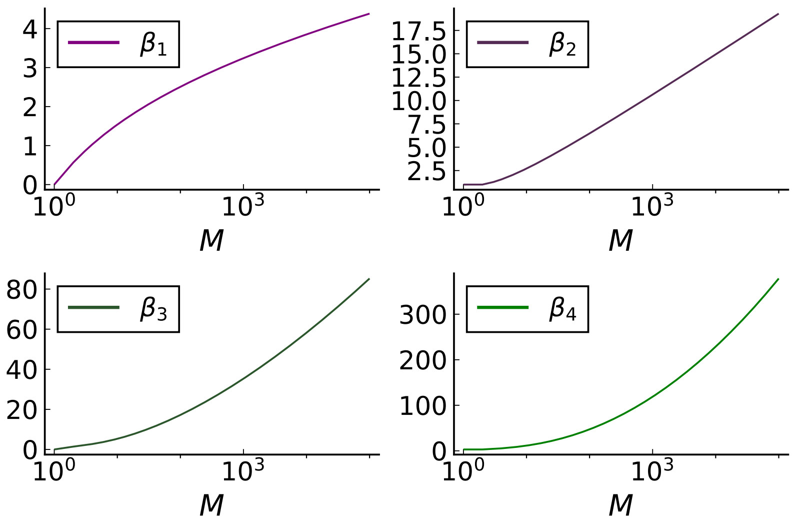

Figure 2.4: Functions $\beta_k(M)$ for selected values of k. We plot the functions for odd values of k with a flipped sign for easiness of comparison. Higher powers of k yield faster scaling with the number of memories, M.

Coming back to the intuition at the beginning of this subsection, we notice from Figure 2.3 how the nearest neighbour modulation, $\zeta_{\text{NN}}(s)$, shifts and scales the standard Gaussian distribution, $\phi(s)$, but its shape still looks approximately Gaussian. We then use Laplace's saddle point method to expand each of those as an exponential of a quadratic function and solve the integral approximately. The full calculation can be found in Appendix D, with the final result being

$$\beta_k(M) \overset{M\to\infty}{\sim} \frac{e^{\frac{1}{2}\log\log M + o(1)} \left[W\left(\frac{M^2}{2\pi}\right)\right]^{\frac{k}{2}}}{2\left[1+W\left(\frac{M^2}{2\pi}\right)\right]^{\frac{1}{2}}}, \tag{2.52}$$

where $W(x)$ is the Lambert W function. The Lambert W function $W(x)$ is such that it satisfies $we^w = x$. For large arguments, $x \gg 1$, it can be proven to scale as $W(x) = \log x - \log\log x + o(1)$. More information about the Lambert W function can be found in Appendix C.

To investigate how precise are the approximations $\hat{\beta}_k$ in Equation 2.52, we plot in Figure 2.5 the ratio between the numerical integration of the true β_k (from Equation 2.51) and the approximations shown above. The error generally decays with M and given large enough M the ratios for different values of k converge. They do not converge exactly to 1 as the calculations done in Appendix D discards some minor terms in favour of interpretability.

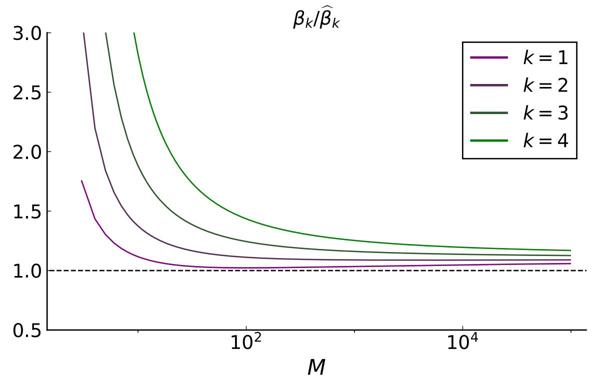

Figure 2.5: Ratio between numerical integration of the functions β_k and the asymptotics $\hat{\beta}_k$ for selected values of k. The error is larger for higher values of k, but all the curves saturate to a constant for high values of M, indicating the validity of the approximation

In summary, we have shown that the contribution from the number of memories, M, for the averages of Equation 2.14 come exclusively through the β_k functions defined in Equation 2.51. Those were shown graphically and analytically to scale close to logarithmically with M, indicating that their influence in the value of the averages can be limited depending on the powers of p and q and the dimensionality of the space of the problem, N.

2.4 Comparing the algorithms

In the previous sections, we provided the theoretical predictions for performances of the three algorithms, restated below for reference. Now we present the experimental results, comparing the algorithm performances both among themselves and with the predictions.

We first compare the generalisation error of the parametric model, as given in Equation 2.25,

$$\mathcal{L}(\mathbf{w}_0,\mathrm{t}) = \mathcal{L}(\mathbf{w}_0)\left[1-\frac{1}{N+2}\right]^{\mathrm{t}} \approx \mathcal{L}(\mathbf{w}_0)\exp\left(-\frac{\mathrm{t}}{N+2}\right), \tag{2.53}$$

where, assuming $\mathbf{w}_0 \sim \mathcal{N}(0,I)$, we have that

$$\mathcal{L}(\mathbf{w}_0) = \frac{1}{2}\left\langle(\mathbf{w}_0\cdot\mathbf{x})^2\right\rangle = \frac{1}{2}\left\langle\mathbf{x}\cdot\mathbf{x}\right\rangle = \frac{N}{2}, \tag{2.54}$$

and that of the fully memory-based model, as first shown in Equation 2.30,

$$\mathcal{L}(y_{\text{NN}};M) = \frac{1}{2}\left[N - 2\sqrt{N}\left\langle z_{\text{NN}}\right\rangle + \left\langle z_{\text{NN}}^2\right\rangle + \left\langle\|\mathbf{x}_{\text{NN}}^{\perp}\|^2\right\rangle\right]. \tag{2.55}$$

The averages over nearest neighbour quantities can be obtained from Equation 2.49,

$$\left\langle z_{\text{NN}}\right\rangle = \sqrt{\frac{2}{3}}\beta_1(M), \tag{2.56a}$$

$$\left\langle z_{\text{NN}}^2\right\rangle = \frac{1}{3}\left(1+2\beta_2(M)\right), \tag{2.56b}$$

$$\left\langle\|\mathbf{x}_{\text{NN}}^{\perp}\|^2\right\rangle = N - \sqrt{\frac{2N}{3}}\beta_1(M), \tag{2.56c}$$

which leads to the expression for the loss,

$$\mathcal{L}(y_{\text{NN}};M) = N - \frac{1}{2}\sqrt{6N}\beta_1(M) + \frac{1}{6}\left(1 + 2\beta_2(M)\right). \tag{2.57}$$

Simulation results and corresponding theoretical curves are presented in Figure 2.6 (for dimensionalities of the space $N = 20$ and $N = 100$). Early in training the fully-memory based approach can outperform the baseline one as the parameter vector **w** is still being learned. After a number of training points are seen, the parametric model starts outperforming the fully-memory based one. This result corroborates Lengyel and Dayan (2008), where authors report the "episodic controller" performed better than the "semantic controller" early in training in a reinforcement learning task. After some trials and efficient learning the performance for the "semantic controller" greatly improves and surpasses the "episodic" one.

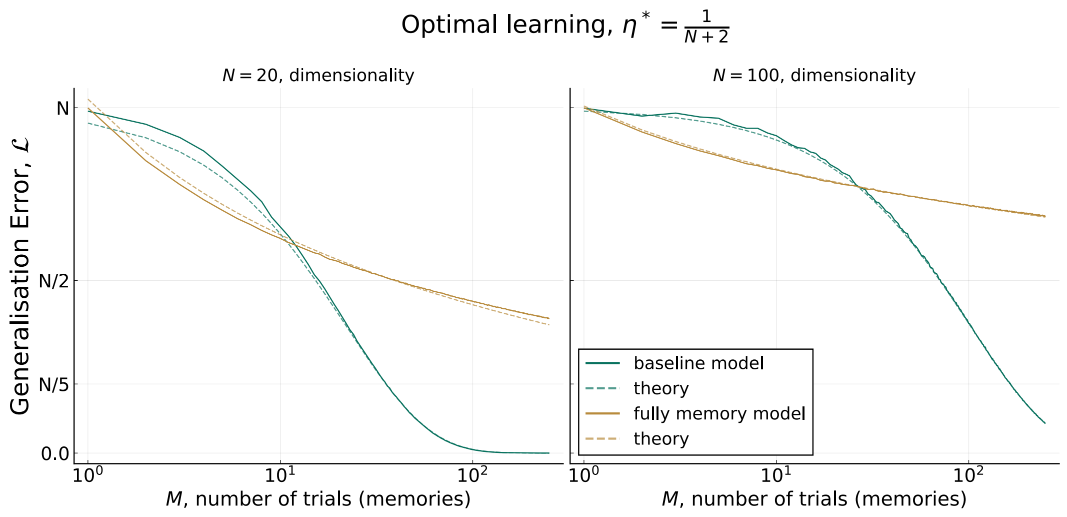

Figure 2.6: The generalisation error for the fully memory-based model, compared with theory and the standard baseline model (which does gradient descent on the loss). Both theoretical curves approach simulations better as the dimesionality of the space, N, is larger.

We proceed to compare the parametric model with the memory-augmented one. From Section 2.2, the improvement due to the memory-adjustment is given by Equations 2.37 – 2.38a,

$$\alpha^* = \frac{\langle z_{\text{NN}}^2 \rangle}{\langle z_{\text{NN}}^4 \rangle + \langle z_{\text{NN}}^2 \|\mathbf{x}_{\text{NN}}^{\perp}\|^2 \rangle}, \tag{2.58a}$$

$$\mathcal{L}(\mathbf{w} + \Delta\mathbf{w}_{\text{NN}}) = \mathcal{L}(\mathbf{w})\,[1-\varepsilon]\,, \tag{2.58b}$$

$$\varepsilon = \frac{\langle z_{\text{NN}}^2 \rangle^2}{\langle z_{\text{NN}}^4 \rangle + \langle z_{\text{NN}}^2 \|\mathbf{x}_{\text{NN}}^{\perp}\|^2 \rangle}. \tag{2.58c}$$

Returning to Equation 2.49, the relevant averages are

$$\langle z_{\text{NN}}^2 \rangle = \frac{1}{3}\left(1 + 2\beta_2(M)\right) \tag{2.59a}$$

$$\langle z_{\text{NN}}^4 \rangle = \frac{1}{3}\left(1 + 4\beta_2(M) + \frac{4}{3}\beta_4(M)\right) \tag{2.59b}$$

$$\left\langle z_{\text{NN}}^2 \|\mathbf{x}_{\text{NN}}^{\perp}\|^2 \right\rangle = \frac{N}{3}\left[1 + 2\beta_2(M)\right] + \sqrt{\frac{2N}{27}}\left[5\beta_1(M) + 2\beta_3(M)\right]. \tag{2.59c}$$

Keeping only the dominant terms for large $N \gg 1$, the average $\langle z_{\text{NN}}^2 \|\mathbf{x}_{\text{NN}}^{\perp}\|^2 \rangle$ dominates the denominators with its $N\beta_2(M)$ term, implying that the scalings for the optimal learning rate and improvement are

$$\alpha^* \sim \frac{1}{N}, \tag{2.60a}$$

$$\varepsilon \sim \frac{\beta_2(M)}{N}. \tag{2.60b}$$

This means that the optimal adjustment rate, α^*, should have, to first order, the same scaling as the optimal learning rate, η^*, calculated in Equation 2.24a, and the improvement brought by the memory-adjustment step scales slowly with the number of memories (through the $\beta_2(M)$ term in the numerator), the effect being suppressed by the dimensionality of the space in the denominator.

Figure 2.7 shows these effects in the improvement ε for different values of N as a function of the number of memories M. This plot was done considering $\eta = 0$, i.e. without any learning by the underlying parametric model. This is to verify that the approximations done to derive the results presented earlier were valid.

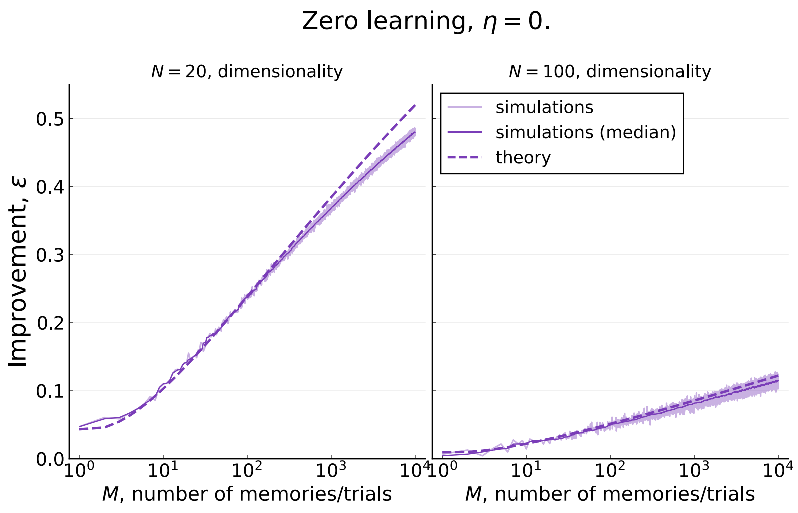

Figure 2.7: Improvement ε as a function of the number of memories M, for different values of dimensionality of the space N. Here there is no learning of the parameter vector $\mathbf{w}$. Light purple lines are the average of different runs of the simulation ($\sim 10^4$ runs); dark purple lines are medians over windows of size 100 of the light curves. The value for improvement predicted by the theory (purple, dashed) agrees well with simulations especially as N increases. Also, as the dimensionality increases, the effectiveness of using memories *decreases*, making this method unsuitable for high-dimensional tasks in this setting.

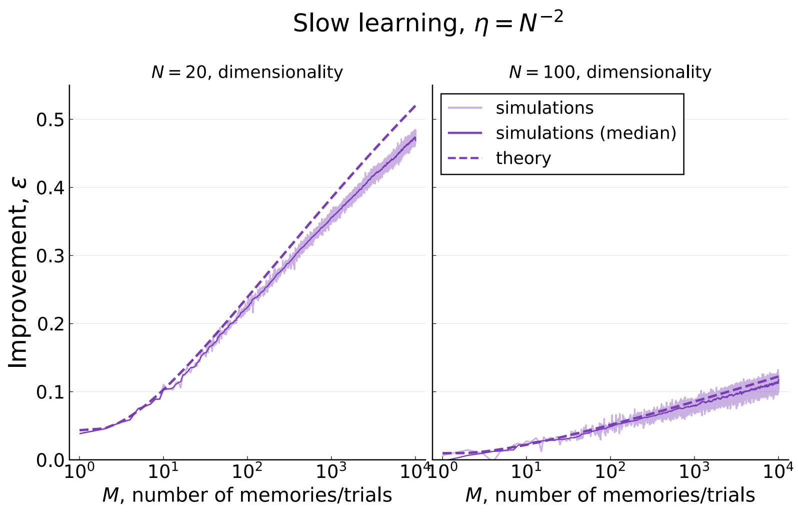

Figure 2.8: Improvement ε as a function of the previously seen data, used both for training the parametric model $\mathbf{w}$ and for the memory-adujstment procedure. Light purple lines are the average of different runs of the simulation ($\sim 10^4$ runs); dark purple lines are medians over windows of size 100 of the light curves. The different panels correspond to different dimensionalities of the space, N. Slow learning, $\eta = 1/N^2$.

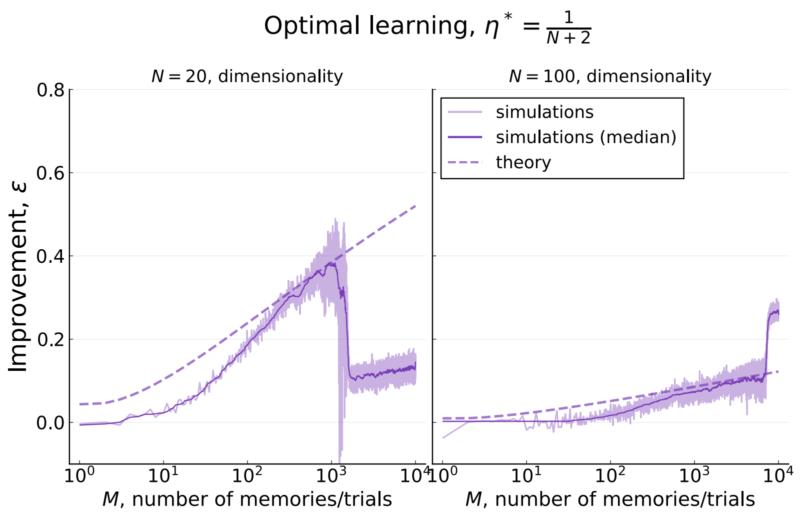

Figure 2.9: Improvement ε as a function of the previously seen data, used both for training the parametric model $\mathbf{w}$ and for the memory-adujstment procedure. Light purple lines are the average of different runs of the simulation ($\sim 10^4$ runs); dark purple lines are medians over windows of size 100 of the light curves. The different panels correspond to different dimensionalities of the space, N. Optimal learning $\eta^* = 1/(N+2)$.

Figures 2.8 and 2.9 show, respectively, how the simulation results change when the learning rate is slow, $\eta \sim 1/N^2$, and optimal, $\eta^* = \frac{1}{N+2}$. We see that the approximations assuming that there are no strong correlations between the discrepancy vector $\Delta\mathbf{w}^*$ and the nearest neighbour quantities still stand qualitatively correct for the values of M and N shown for the slow learning case, but break down when learning becomes stronger. We notice that there are clear nonlinear aspects in the optimal learning plots (Figure 2.9) which cannot be accounted by the theoretical results.

Then, Figures 2.10 – 2.12 present the generalisation error of the baseline and the memory-adjustment models, which displays the same results as the other figures: for no or slow learning there is a noticeable improvement from using memories to adjust the parameter vector, but in the case of slow learning this improvement scales poorly with the dimesionality of the space, N, and does not affect the performance considerably when learning is optimal. While the right panel of Figure 2.9 jumps upward at the end of the regime might seem to indicate a relevant improvement from memories in the optimal learning scenario, one needs to keep in mind the associated plot in the right panel of Figure 2.12, which shows that the generalisation error is essentially zero, and the decrease in the loss due to the memory-adjustment procedure does not yield a relevant change in absolute terms.

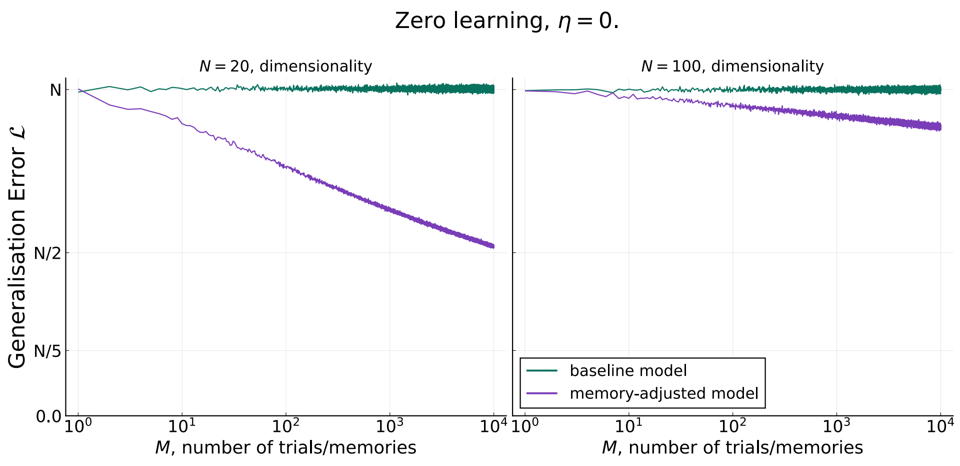

Figure 2.10: Generalisation error for baseline and the memory-adjustment models, considering no learning $\eta = 0$.

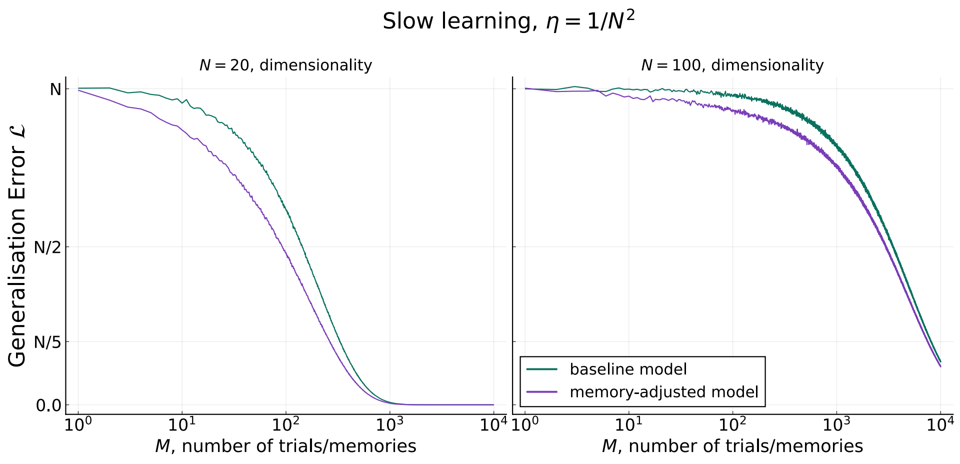

Figure 2.11: Generalisation error for baseline and the memory-adjustment models, considering slow learning $\eta^* = 1/N^2$

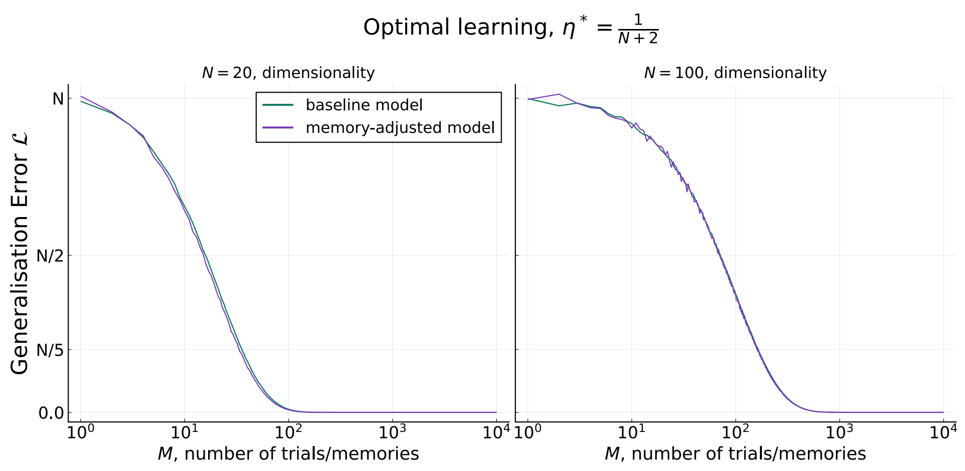

Figure 2.12: Generalisation error for baseline and the memory-adjustment models, considering optimal learning $\eta = 1/(N+2)$

2.5 Model mismatch

Now we modify the task the algorithms need to learn by introducing a non-zero bias in the true mapping between input and output, i.e. $y_0 \neq 0$ in the generative process that describes the learning problem,

$$y = f(\mathbf{x}) = \mathbf{w}^* \cdot \mathbf{x} + y_0 \,. \tag{2.61}$$

This is to make the problem more complicated in general, and to introduce a model mismatch for the baseline. Depending on the size of the bias term this change can drastically impair the performance of that model, as presented later in this section. The hypothesis here is that the small improvements presented in the previous section were due to the parametric model being in the same family of functions as the true model, and instead memories would be useful in cases with more complicated mappings between input and output, such as when there is a model mismatch.

This seems to make intuitive sense: if an agent has a good model of the mapping it is trying to predict then it does not need to remember individual events anymore, it must only follow its model. Conversely, if the agent does not have a good model of the mapping, then having memories of individual events can fix those gaps in knowledge and help make better predictions.

Following the same procedure as in preceding sections, we can derive the optimal adjustment rate α^*,

$$\alpha^* = \frac{\left\langle z_{\text{NN}}^2 \right\rangle + \frac{\langle y_0^2 \rangle}{2\sqrt{N}} \langle z_{\text{NN}} \rangle}{\frac{\langle y_0^2 \rangle}{2} \langle z_{\text{NN}}^2 \rangle + \langle z_{\text{NN}}^4 \rangle + \left\langle z_{\text{NN}}^2 \|\mathbf{x}_{\text{NN}}^\perp\|^2 \right\rangle}, \tag{2.62}$$

and the average improvement associated with it,

$$\varepsilon(y_0) = \frac{\left[\left\langle z_{\text{NN}}^2 \right\rangle + \frac{\langle y_0^2 \rangle}{2\sqrt{N}} \langle z_{\text{NN}} \rangle \right]^2}{\left[1 + \frac{\langle y_0^2 \rangle}{2N} \right] \left[\frac{\langle y_0^2 \rangle}{2} \langle z_{\text{NN}}^2 \rangle + \langle z_{\text{NN}}^4 \rangle + \left\langle z_{\text{NN}}^2 \|\mathbf{x}_{\text{NN}}^\perp\|^2 \right\rangle \right]}. \tag{2.63}$$

The results are mostly similar to those in Equation 2.59, with important additions due to the bias term, y_0. The terms that depend only on powers of z_{NN} scale as

$$\langle z_{\text{NN}}^p \rangle \sim \beta_p(M), \tag{2.64}$$

and therefore do not contribute to the overall scaling unless the number of memories is exponentially large. The largest term in the denominator is typically $\left\langle z_{\text{NN}} \|\mathbf{x}_{\text{NN}}^\perp\|^2 \right\rangle \sim N\beta_2(M)$. But now, if the bias term is of an order comparable with the dimensionality of the space, $y_0^2 \sim N$, then there would be a contribution to the numerator of Equation 2.63 such that it would be of the same order as the denominator, making the effect from the memories more noticeable. This scaling is in line with Equation 2.3, where we saw that the scaling of the output is in general order $\sqrt{N}$ for randomly sampled isotropic Gaussian variables $\mathbf{w}^*$ and $\mathbf{x}$.

The comparison between theoretical calculation for the improvement and numerical simulations can be seen in Figures 2.13 – 2.15. The results mostly recapitulate what was shown in Section 2.4, with the main difference that now the improvement in the higher-dimensional case, $N = 100$, isn't negigible.

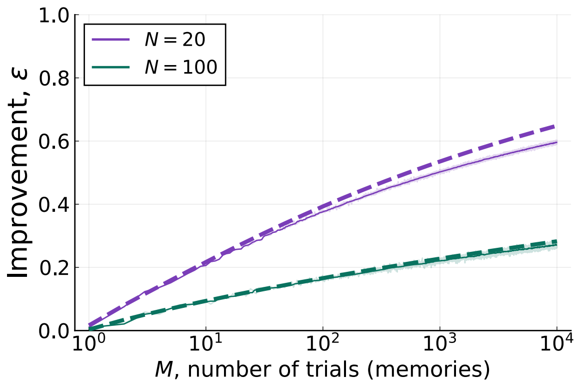

Figure 2.13: Improvement in the model-mismatch condition, $y_0 = \sqrt{N}$, for zero learning. Simulation plots (solid lines) and theory (dashed lines) are plotted for dimensionalities $N = 20$ (purple) and $N = 100$ (green).

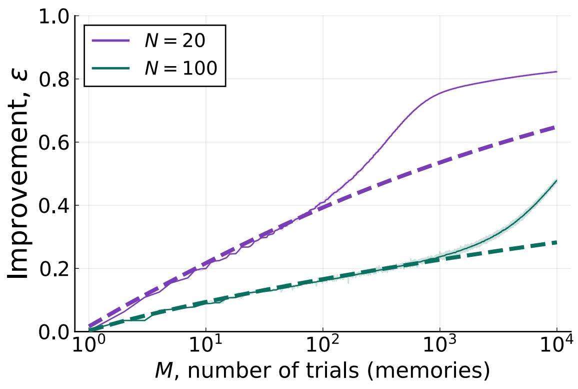

Figure 2.14: Improvement in the model-mismatch condition, $y_0 = \sqrt{N}$, for slow learning. Simulation plots (solid lines) and theory (dashed lines) are plotted for dimensionalities $N = 20$ (purple) and $N = 100$ (green).

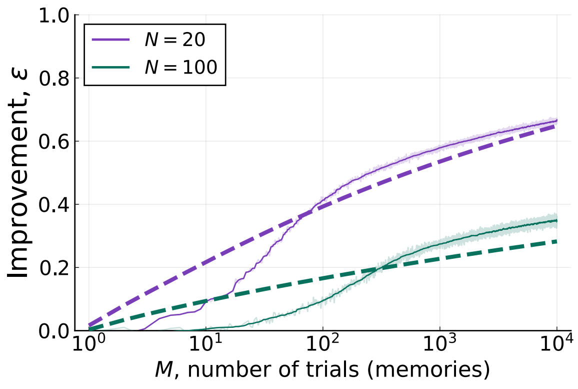

Figure 2.15: Improvement in the model-mismatch condition, $y_0 = \sqrt{N}$, for optimal learning. Simulation plots (solid lines) and theory (dashed lines) are plotted for dimensionalities $N = 20$ (purple) and $N = 100$ (green).

Furthermore, there are clear regions of the improvement curves that the theory misses. In order to understand those we look into the generalisation error curves, shown in Figure 2.16. As expected from the mismatch between the class of models that the parametric model can represent and the true model generating the data, every learning profile plateaus at a certain level. For slow learning rate $\eta = \frac{1}{N^2}$ this is at the optimal value the parametric model can go, where $\mathbf{w} = \mathbf{w}^*$ but there is still an error of order $\frac{\langle y_0^2 \rangle}{2} = \frac{N}{2}$. It seems the slower learning rate does a better job in the case of mismatch, as in the simulations using η^* lead to a worse plateau. This could be caused by the faster learning rate method falling into a local optima.

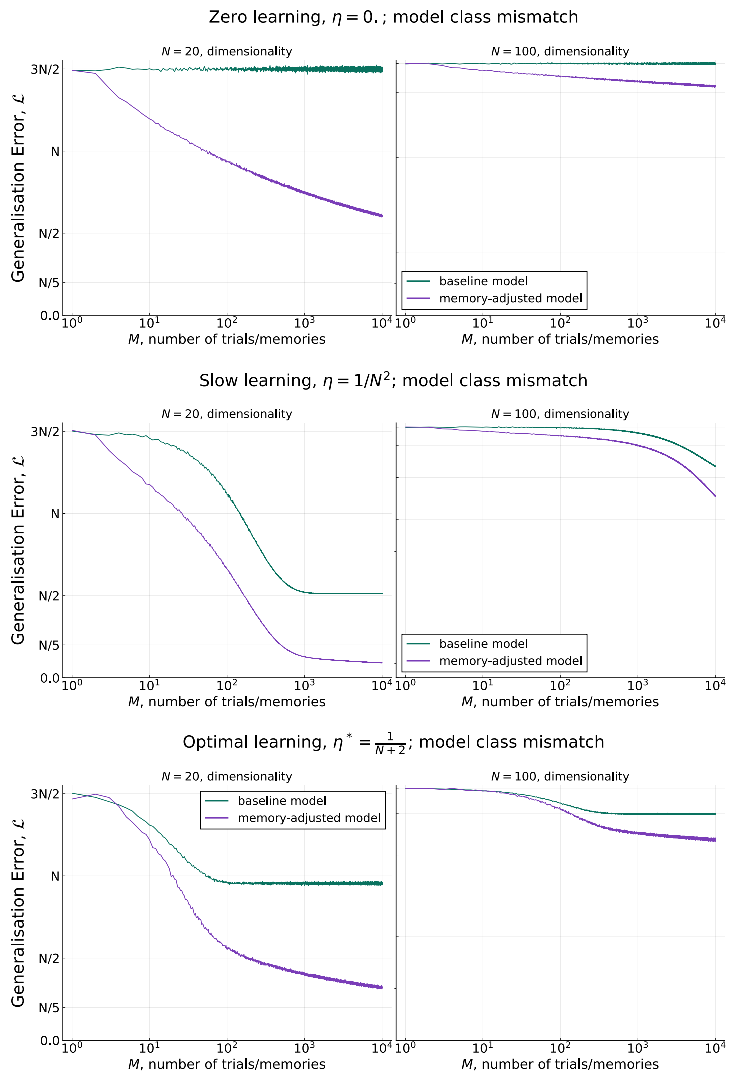

Figure 2.16: Generalisation error curves for numerical simulations of baseline model and memory-adjusted model

In order to quantify how much improvement, ε, is gained in the mismatch condition *on top* of what was originally gained in the within model class situation, we quantify it in the simulations by looking at the difference between those,

$$\Delta\varepsilon(N,M) = \varepsilon(N,M,y_0 = \sqrt{N}) - \varepsilon(N,M,y_0 = 0)\,. \tag{2.65}$$

As one can see in Figures 2.17 – 2.19, the benefit from using memories to adjust the model remains of considerable effect for a range of values of N, even when learning is not slow. However, as in the case within model class, the learning causes correlations between the discrepancy vector $\Delta\mathbf{w}^*$ and the nearest neighbour quantities that are not fully accounted for in the theoretical results.

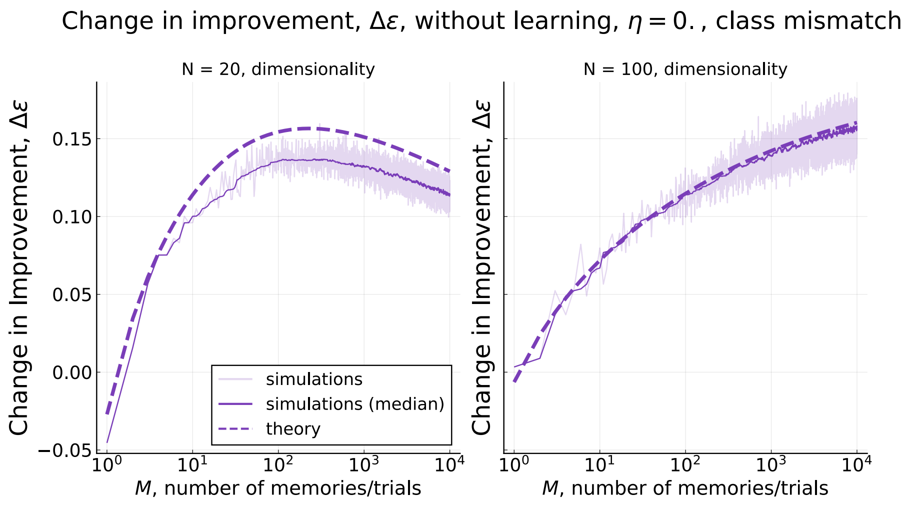

Figure 2.17: Change in improvement $\Delta\varepsilon$ when comparing model mismatch $y_0 = \sqrt{N}$ and within model-class $y_0 = 0$ scenarios. Again, shown for different values of dimensionality of the space N as a function of the number of memories M, with the darker purple line being the median of the lighter purple curves. As before, the theory agrees well with the no-learning simulations

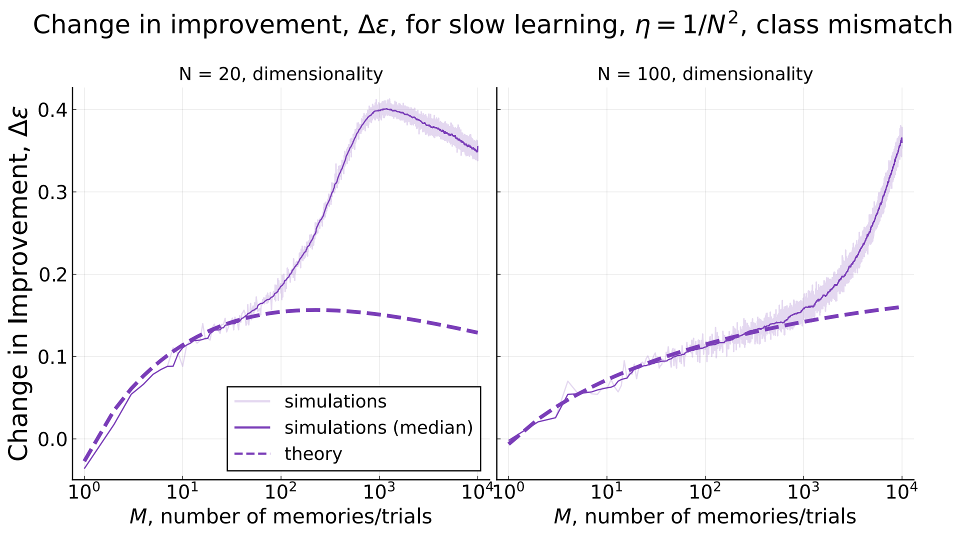

Figure 2.18: As we move to the other learning profiles, here shown $\eta = \frac{1}{N^2}$, the theory captures well the simulation results up to a certain point, where there seems to be a transition. In Figure 2.16 we see this is due to a saturation of the learning.

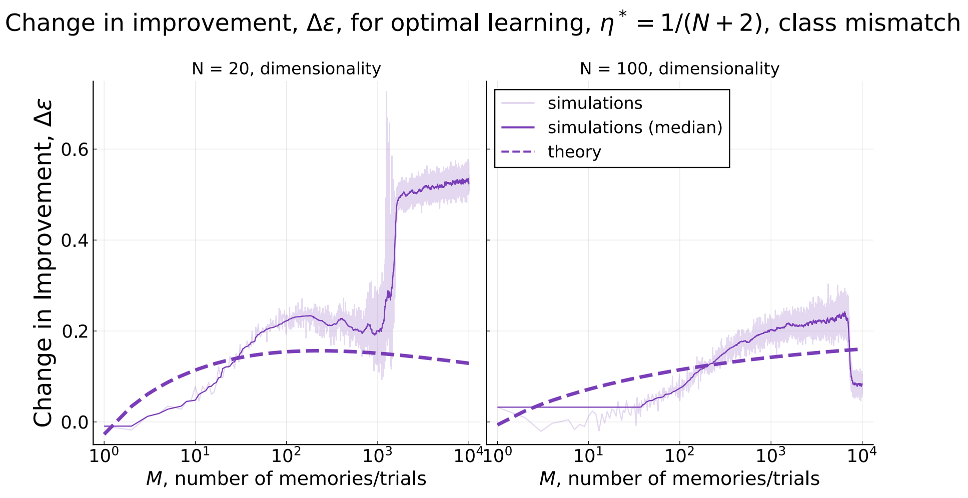

Figure 2.19: Similarly for the situation where $\eta^* = \frac{1}{N+2}$, the theory captures the overall trend of the (change in) improvement, but fails to do so as the learning reaches a plateau, as seen in Figure 2.16

2.6 Extension to L closest neighbours

In the preceding sections, we considered how and when using one single-event memory could be an effective strategy for prediction or for updating a parametric model used in prediction. A natural extension is to extrapolate the reasoning to using L single-event memories – this seems to still consider the different memories individually, while also perhaps extracting from each their differential contribution to prediction. In this section we consider this development of having the L closest memories adjust the parametric model. We update the equations presented before, noting their differences, and present the new results, again comparing with simulations.

The main change we make is to consider the adjustment update now as an average update over directions corresponding to different memories,

$$\Delta \mathbf{w}_L = -\frac{\alpha}{L}\sum_{l=1}^{L} \nabla_{\mathbf{w}} \ell(\mathbf{w}, \mathbf{x}_{(l)}, y_{(l)}) = \frac{\alpha}{L}\sum_{l} (\Delta \mathbf{w}^* \cdot \mathbf{x}_{(l)})\, \mathbf{x}_{(l)} \,. \tag{2.66}$$

From this change we can obtain new equations for the optimal adjustment step α^* and the associated improvement ε by following the same procedure as in Section 2.2 (i.e. calculating the change in the pointwise loss and taking the average to obtain the generalisation error). The formal results are

$$\alpha^* = \frac{L \sum_l \langle z_l^2 \rangle}{2\sum_l \langle z_l^4 \rangle + \sum_l \langle z_l^2 \|\mathbf{x}_l^{\perp}\|^2 \rangle + 2\sum_{l' \neq l} \langle z_l^2 z_{l'}^2 \rangle + \sum_{l' \neq l} \langle z_l z_{l'} (\mathbf{x}_l^{\perp} \cdot \mathbf{x}_{l'}^{\perp}) \rangle} \tag{2.67a}$$

$$\varepsilon = \frac{\left[\sum_l \langle z_l^2 \rangle\right]^2}{2\sum_l \langle z_l^4 \rangle + \sum_l \langle z_l^2 \|\mathbf{x}_l^{\perp}\|^2 \rangle + 2\sum_{l' \neq l} \langle z_l^2 z_{l'}^2 \rangle + \sum_{l' \neq l} \langle z_l z_{l'} (\mathbf{x}_l^{\perp} \cdot \mathbf{x}_{l'}^{\perp}) \rangle} \,. \tag{2.67b}$$

Also similar to before, we need to take expectations over the nearest L neighbours. The procedure – once more applying results from Order Statistics, as described in Section B – is similar to the one that led to Equation 2.46, except that now we are interested in each of the lth closest neighbours, where $l \in \{1, \dots, L\}$, instead of just the nearest one. The marginal distributions for each l are given by

$$p\left(z, \|\mathbf{x}^{\perp}\|^2 \middle| \, l^{th}, \, \mathbf{x}\right) = p(z, \|\mathbf{x}^{\perp}\|^2) \int \mathrm{d}\gamma \, p(\gamma | z, \|\mathbf{x}^{\perp}\|^2, \mathbf{x}) \times \frac{M!}{(l-1)!(M-l)!} C(\gamma)^{l-1} \left[1 - C(\gamma)\right]^{M-l} , \tag{2.68}$$

and the joint distributions for neighbours $l < k$ are

$$p\left(z_1, \|\mathbf{x}^{\perp}\|_1^2, z_2, \|\mathbf{x}^{\perp}\|_2^2 \middle| \, l^{th}, \, k^{th}, \, \mathbf{x}\right) = p(z_1, \|\mathbf{x}^{\perp}\|_1^2, z_2, \|\mathbf{x}^{\perp}\|_2^2) \times \int \mathrm{d}\gamma_l \mathrm{d}\gamma_k \, \mathbb{1}\{\gamma_l \leq \gamma_k\} \, p\left(\gamma_l, \gamma_k \middle| z_l, \|\mathbf{x}^{\perp}\|_l^2, z_k, \|\mathbf{x}^{\perp}\|_k^2\right) \times \frac{M! \, C(\gamma_l)^{l-1} \left[C(\gamma_k) - C(\gamma_l)\right]^{k-l-1} \left[1 - C(\gamma_k)\right]^{M-k}}{(l-1)!(k-l-1)!(M-k)!} . \tag{2.69}$$

To follow from the distributions above to the averages needed to compute Equations 2.67a and 2.67b, one needs to follow the same approach as the one described in Appendix E, but adapted for $L > 1$ memories: do the γ integrals using delta functions, make change of variables and approximations, to obtain

$$\left\langle z_l^p |x_l^{\perp}|^{2q} \right\rangle = \left\langle \left[\frac{u}{\sqrt{3}} - s\sqrt{\frac{2}{3}}\right]^p \left[N + \sqrt{\frac{2N}{3}}\left(u\sqrt{2} + s\right)\right]^q \zeta_l(s, M) \right\rangle_{u,s} \tag{2.70}$$

and, for $l < k$,

$$\left\langle z_l^p z_k^r |x_l^{\perp}|^{2q} |x_k^{\perp}|^{2u} \right\rangle = \left\langle \left[\frac{u_l}{\sqrt{3}} - s_l\sqrt{\frac{2}{3}}\right]^p \left[N + \sqrt{\frac{2N}{3}}(u_l\sqrt{2} + s_l)\right]^q \left[\frac{u_k}{\sqrt{3}} - s_k\sqrt{\frac{2}{3}}\right]^r \left[N + \sqrt{\frac{2N}{3}}(u_k\sqrt{2} + s_k)\right]^u \zeta_{lk}(s_l, s_k, m) \right\rangle_{u_l, u_k, s_l, s_k} \tag{2.71}$$

where the marginal factors, ζ_l, and joint factors, ζ_{lk}, are given by

$$\zeta_l(s) = \frac{M!}{(l-1)!(M-l)!} \Phi(s)^{l-1} \left[1 - \Phi(s)\right]^{M-l} , \tag{2.72a}$$

$$\zeta_{lk}(s_l, s_k) = \frac{M! \;\; \mathbb{1}\{s_l \leq s_k\}}{(l-1)!(k-l-1)!(M-k)!} \Phi(s_l)^{l-1} \left[\Phi(s_k) - \Phi(s_l)\right]^{k-l-1} \left[1 - \Phi(s_k)\right]^{M-k} . \tag{2.72b}$$

The averages using just one of the neighbours, l, are relatively simple and similar to the averages we had before, but the averages for joint terms are more complicated and require further consideration. In particular, it is not possible to calculate exactly the quantity $\langle z_l z_{l'}(\mathbf{x}_l^\perp \cdot \mathbf{x}_{l'}^\perp)\rangle$ with the averages we know from Equation 2.71, but we can make the following approximations that should be reasonable for large N and small M. First, the dot product between the perpendicular components of both memories can be rewritten as $\mathbf{x}_l^\perp \cdot \mathbf{x}_{l'}^\perp = \|\mathbf{x}_l^\perp\|\|\mathbf{x}_{l'}^\perp\| \cos\theta_{ll'}$. So far this is exact, but we don't know the cosine, this is where the approximation comes: initially the vectors $\mathbf{x}_l^\perp$ and $\mathbf{x}_{l'}^\perp$ are approximately orthogonal as they are random vectors in a high dimensional space, so $\cos\theta_{ll'} \sim \frac{1}{\sqrt{N}}$. This breaks down later in training, since memories will be more correlated due to the fact that they are conditioned on both being close neighbours to the test point.

Figures 2.20 – 2.22 present, as before, the improvement and generalisation error for different dimensionalities of the space and different learning condition (no learning, slow learning, and optimal learning), in a within model class situation.

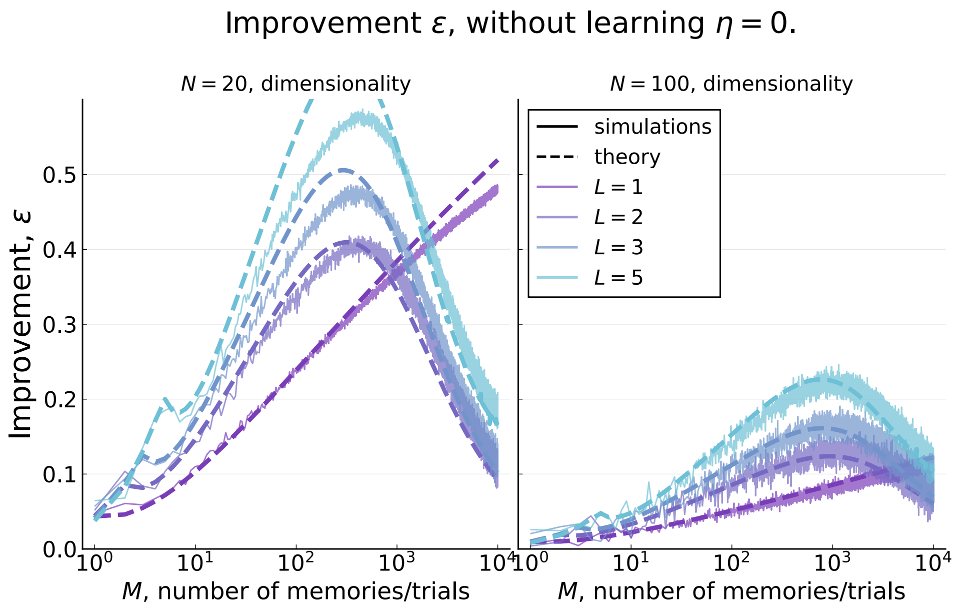

Figure 2.20: As the number of memories, L, being used in the adjustment procedure increases, the improvement, ε, is larger. As M increases the approximations done to calculate the learning rate α become less precise, and a suboptimal adjustment rate might explain the fall in improvement.

A couple of points are worth noticing about the improvement curves when using $L > 1$ nearest neighbours for the adjustment update: first, as before, the theoretical predictions match the numerical results reasonably well for large dimensionality spaces and slow learning regimes; then, the increased improvement from using more memories in the adjustment procedure is present for low and intermediate data regimes, but there is a sharp decrease as the number of available memories grows, presumably due to a suboptimal adjustment rate α^*, leading to an over-adjustment of the model.

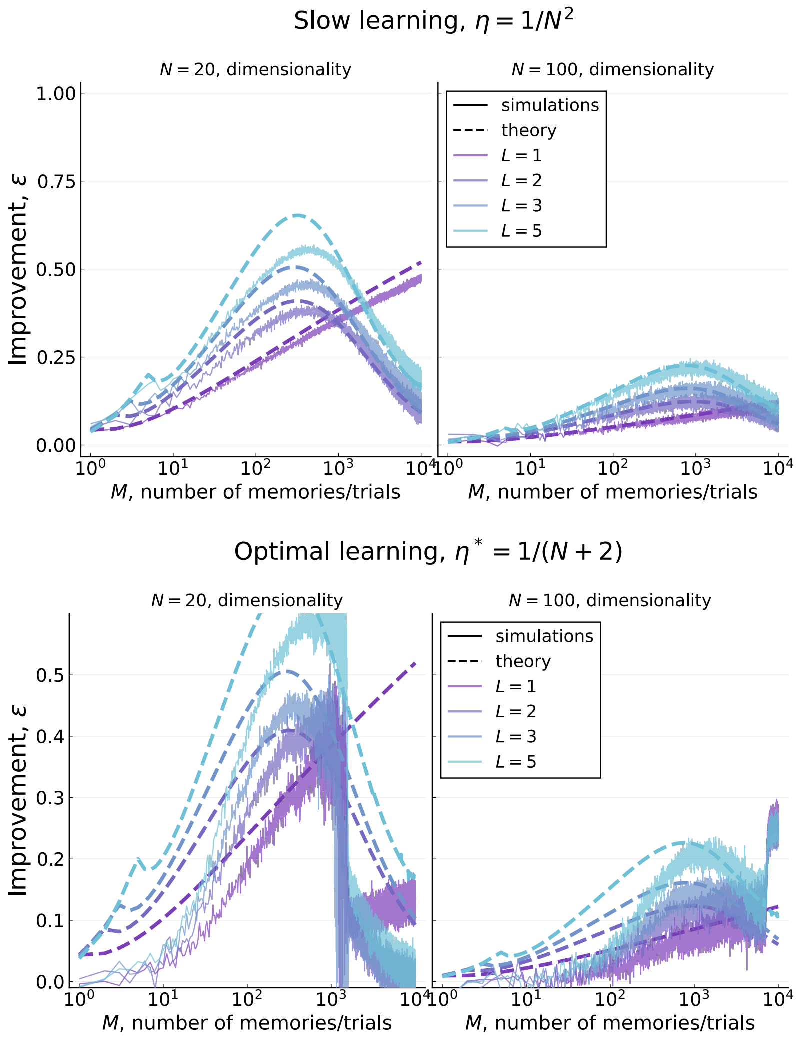

Figure 2.21: As the learning rate becomes non-zero, the theory still captures most of the simulation results. Again, the improvement decays after a number of memories/trials being stored/seen, perhaps indicating the suboptimal adjustment rate due to the aproximations made. Also, similar effects to before can be observed due to learning not being a negligible effect anymore.

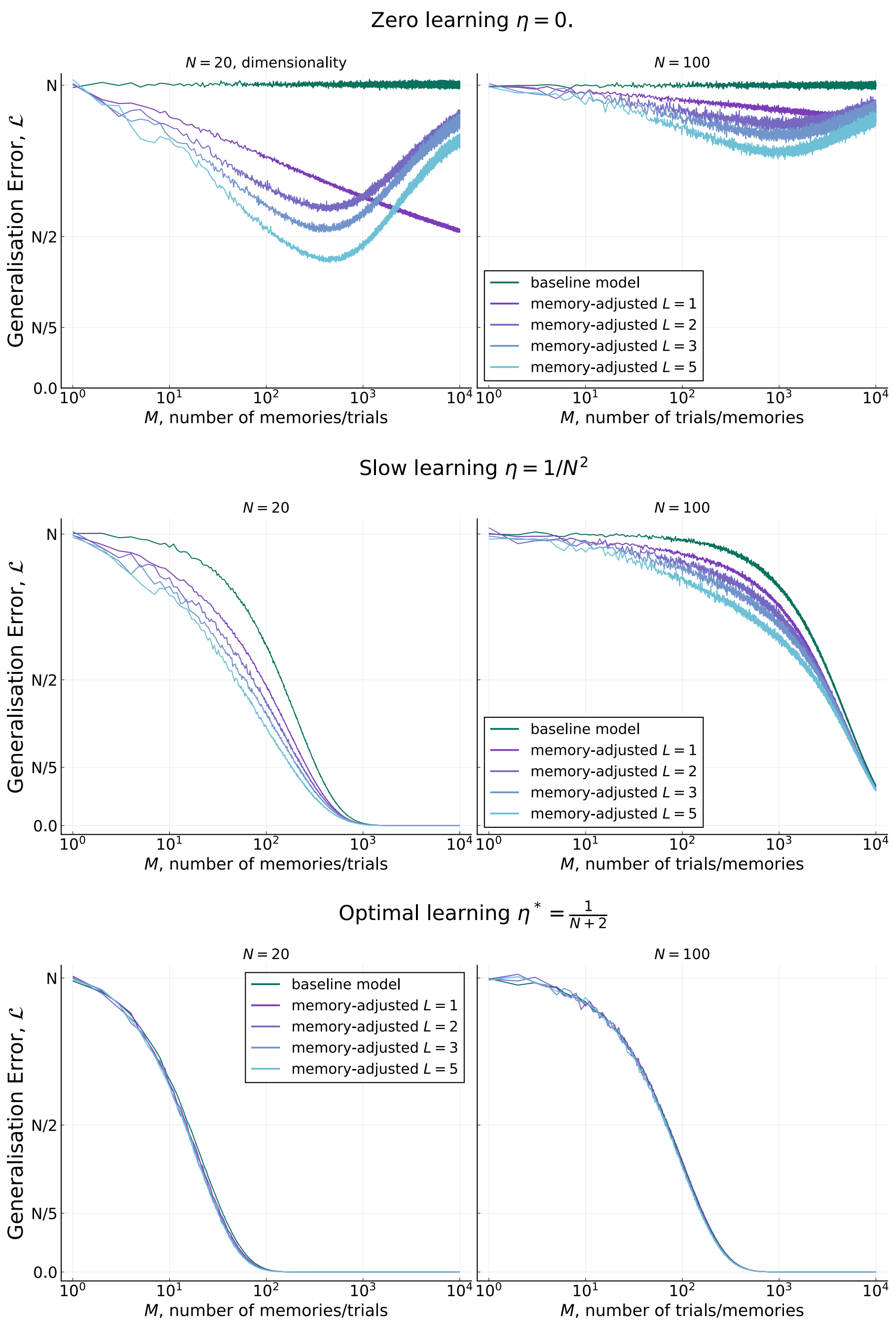

Figure 2.22: Generalisation error for different dimensionalities of the space, N, and different learning rates, η. As before, we see how the effect of learning renders the contribution from the memories as almost irrelevant.

2.7 Discussion

In this chapter, our objective was to study computational models in order to understand the possible uses for the capacity observed in humans and other animals to recall individual past experiences. For that we compared two different supervised learning algorithms that used memories of single-events with an algorithm that did not use single-event memories. Our main contribution was to calculate analytically the performance of those models, and the improvement that was gained from using single-event memories.

Employing techniques from order statistics and machine learning, we found an analytical expression for the scaling of this improvement with the dimensionality of the problem and the number of memories in a supervised learning problem. The result was corroborated with simulations proving to be robust ouside of the main assumptions, and precise when the dimensionality of the problem was large. However, the scaling was inneficient with the number of memories stored, rendering the proposed procedure impractical to use in simple and/or high-dimensional environments. In fact, the limitedness of the brain (be it in number of neurons or in physical space) has been proven to be tightly linked to its storage capacity (see e.g. Amit et al. (1985)), constraining the number of rich memories such a system can reliably store. Additionally, it is not clear that an exponentially large number of memories is available for most tasks, and outside of this regime the benefits of the single-event memory augmentation are small.

However, we cannot completely rule out the usefulness of single-event memories in more complex tasks, or situations of model mismatch, as evidenced in Section 2.5. In fact, work by Sprechmann et al. (2018) proposes a similar mechanism as a solution to prevent catastrophic forgetting and enable fast acquisition of new information in a continual learning task. Their setting is more comparable with our model mismatch setting, as in both cases the parametric model for learning (here, the vector **w**; there, a deep neural network) is not necessarily in the correct model family for describing the data. In their work, the memories present a non-negligible improvement over the other method, which might indicate that a model mismatch scenario is the more likely to showcase the benefits of using memories for improvement. Furthermore, Nagy and Orbán (2017) illustrates how having memories of single events can be helpful when the agent has a collection of imperfect models and has to decide online which one best represent the data at hand. The theoretical framework provided in this chapter corroborates those findings, with the benefit of allowing approximate calculations to be made, rendering the intuition more precise through the scalings with the relevant quantities of the problem, N and M. With that in mind, further analyses are needed to consolidate whether strategies employing memories are in fact more effective in situations where it is hard to learn the mapping generating the data, or that mapping changes over time.

The approach presented in this chapter might be improved by exploring other extensions: taking more gradient steps with the memories (which indeed was done in Sprechmann et al. (2018)), or performing preprocessing in the data before using it as an input to the model. While we expect the "curse of dimensionality" and its trade-off with the number of memories to remain true, further studies are needed to validate or eliminate the possibility that single-event memories are more useful in more complex scenarios, and the mathematical tools derived in our work should be useful for those analyses.

Part II

Decision-making

Chapter 3

Optimal reward-rate in foraging-like multi-task environments

In this chapter we consider reward-maximiser agents, and a type of task where the agent gathers information over time to make a decision. For instance, this could involve collecting information to choose a brand of cereal, pick a vacation destination, or determine whether there are more dots moving left or right in a visual task. In all these these tasks taking more time to gather information often leads to better decisions and the potential for greater rewards, like making the right choice or enjoying a better holiday. However, time is also valuable, and the expected reward rate from a single decision drops as the agent spends more time in the task.

Furthermore, biological agents have different tasks available to them, and cannot spend endless amounts of time on a single decision. By devoting too much time to make one decision they miss out on other opportunities, and, as a result, do not accrue as much reward as they could. In this research, we investigate how the optimal decision to stay or leave a single task is influenced by the existence of different tasks in the environment.

The notion that an agent cannot optimise jointly for both speed and accuracy is at times called the Speed-Accuray Tradeoff, and has been studied extensively both theoretically and experimentally. For a review of this literature, see e.g. Heitz (2014); Bogacz (2022). Past research has shown both theoretically (Bogacz et al. (2006)) and experimentally (Gold and Shadlen (2007)) that it is possible to formalise mathematically the notion of optimally accumulating evidence for a decision, and that it can be quantified in animal brains. However, researchers have found that humans don't always behave optimally as prescribed by theory. Both Bogacz et al. (2010); Simen et al. (2009) found that some participants would spend more time in a decision (therefore being more accurate) than expected of an optimal agent. Nonetheless, further work by Balci et al. (2011) presents contrasting evidence that subjects could learn to behave optimally after enough practice.

Typically, these evidence accumulation tasks are studied in controlled experimental settings where subjects (usually humans or macaques) repeat the same task during a session. However, in real life, humans and animals have a variety of tasks they can engage with simultaneously. We refer to this as a multi-task environment.

The availability of multiple potential tasks can significantly impact behaviour. An agent's performance in a particular task may be influenced not only by their considerations on how to optimise the speed-accuracy tradeoff for that specific task but also by the presence of other available tasks. For instance, a (hypothetical) person considering the upcoming decision of buying a yacht may not spend too much time pondering which brand of bleach to purchase at the supermarket right now.

Another situation where animals are presented with multiple opportunities at their disposal and must decide how much time to allocate in each of them is foraging. When foraging, animals can either exploit the food patch closest to them, or move on to seek food elsewhere. In order to decide between these options, the animal needs to estimate how much food they expect to obtain by staying in the current patch, and compare with the amount of food expected from the rest of the environment. Considering an animal with complete information about the environment, Charnov (1976) calculates that the optimal behaviour consists in leaving the current patch when the instantaneous energy rate falls below the environment's average rate. Extensions to Charnov's result have been proposed, such as allowing revisitation of patches (Possingham and Houston (1990)), and limited information on patch identity (Kilpatrick et al. (2021)), but so far there has been no study for when the agent can select a general policy for deciding when to leave the current patch type.

In this chapter we merge the two lines of thought presented in this introduction: the speed-accuracy tradeoff in evidence accumulation decision-making tasks, and multi-task decision-making from Foraging Theory. This allows studying the optimal behaviour for an agent seeking to maximise reward rate in a multi-task foraging-like environment. We extend the usual foraging setting to include more complex tasks and policies for behaving in a food patch, or task. This makes the previous solution to the problem unfeasible in practice, so in Section 3.2 we discuss a new algorithm for solving the problem efficiently. Then, in Section 3.3 we investigate a particular evidence accumulation decision-making task as an example of tasks that can be studied with this new formulation, and in Section 3.4 use the framework proposed previously to provide an alternative explanation for apparent suboptimal decision-making by humans, inspired by analogous results reported in the literature (Bogacz et al. (2010); Simen et al. (2009); Balci et al. (2011)).

3.1 A framework for multi-task decision-making

Let us consider an agent that is in an environment where there are $i = 1, \ldots, N$ tasks available, and only one of those can be visited at any given time. For example, as in foraging, where the agent has different "food patches" they can collect food from, or in real life, where they have different tasks where they need to accumulate evidence to make informed decisions. For now we consider the case where the tasks cannot be chosen by the agent, but are randomly assigned with probabilities p_i. Even though this is an artificial assumption – which will not be used in Chapter 4 – it is helpful to develop intuition and is sufficient to explain seemingly suboptimal behaviour observed in humans (Section 3.4). This setting up to now is also similar to the one studied by Charnov (1976) and reviewed in Appendix F.

What the agent can choose in this multi-task environment are the actions *inside* the task they're currently in, in order to accumulate evidence and eventually leave the task. We represent these actions the agent can take with policies, π_i, also indexed by i as in for each task.

We subsume the reward structure and stochasticity of a task into an average reward function, $\overline{R}$, that only yields reward to the agent upon leaving to a new task. The reward functions of the tasks depend explicitly on the agent's policies, so that when an agent follows policy π_i in task i they can expect to obtain an average reward of $\overline{R}_i(\pi_i)$.

Time is also a crucial factor. We consider tasks in which the average reward does not increase indefinitely with time spent in task, but, on the contrary, the reward rate decreases with time after a while in that task. Let us represent by $\overline{T}_i(\pi_i)$ the time the agent spends in task i when following policy π_i. Then, in line with the above, we assume that the quantity the agent seeks to maximize is the amount of reward collected for the minimum amount of time spent, i.e. the reward rate

$$r(\pi) = \frac{\sum_i p_i \overline{R}_i(\pi_i)}{\sum_i p_i \overline{T}_i(\pi_i) + T_0}, \tag{3.1}$$

as a function of the policies in each task $\pi = (\pi_1, \ldots, \pi_N)$ through their impact on the average rewards $\overline{R}_i$ and average times $\overline{T}_i$. A transit time, $T_0 > 0$, is introduced so that the reward rate cannot be infinite.

This quantity can be shown to correspond to the overall reward rate over a large number of trials, M, which is used in the Average-Reward setting in Reinforcement Learning (see e.g. Blackwell (1962); Puterman (1994); Dewanto et al. (2021); Sutton and Barto (2018)). The simplifying factor that the problem in this chapter assumes, and a generic Average-Reward RL problem does not, is that here the only state variable is the task the agent is currently in, which is visited stochastically with fixed probability p_i. This means we can exchange the average over time (sums over trials) for averages over states (tasks), as shown in the following calculation. Let us denote superscript t, $\bullet^t$, as an index over trials, and subscript i, $\bullet_i$, as an index over tasks,

$$r_{\text{ARL}}(\pi) = \lim_{M\to\infty} \frac{\sum\limits_{t=1}^{M} R^t}{\sum\limits_{t=1}^{M} T^t} = \lim_{M\to\infty} \frac{\sum\limits_{i=1}^{N}\sum\limits_{t_i=1}^{M_i} R^{t_i}}{\sum\limits_{i=1}^{N}\sum\limits_{t_i=1}^{M_i} T^{t_i}}, \tag{3.2}$$

where M_i represents the number of trials that visitation to task i if the total number of trials was M.

Then, for a fixed policy, we can use the law of large numbers to equate the sum of different R's coming from a same task into their average conditioned on the policy for the limit of many trials in that task. This is valid since we assume the distribution that generates the rewards and times is the same for all those times (i.e. the state of the environment is stationary). The final result becomes

$$\lim_{M\to\infty} \frac{\sum_i \sum_{t_i} R^{t_i}}{\sum_i \sum_{t_i} T^{t_i}} = \lim_{M\to\infty} \frac{\sum_i M_i \overline{R}_i}{\sum_i M_i \overline{T}_i} = \frac{\sum_i p_i \overline{R}_i}{\sum_i p_i \overline{T}_i} = r(\pi)\,. \tag{3.3}$$

Note that the objetive function $r(\pi)$ is also similar to the one assumed by Charnov (1976) (see Appendix F). There the animal explicitly selects the amount of time to stay in a task, whereas here the focus is on the *policy*, with the time spent in task being a consequence of the policy. In both formulations, the agent is making similar decisions – to decide when to leave a task and move (stochastically) to the next one – but the apparently superficial change of introducing policies supports modelling essentially arbitrary behaviour, and also allows a stochastic mapping from behaviour to time spent in the task.

In order to optimise $r(\pi)$, we consider, in the first instance, that the $\overline{R}_i$s and $\overline{T}_i$s are differentiable functions of the policies, and look for the extrema of $r(\pi)$ with respect to the policies through differentiation,

$$\nabla_{\pi_i} r(\pi) = \frac{p_i}{\sum_i p_i \overline{T}_i(\pi_i) + T_0} \left(\nabla_{\pi_i} \overline{R}_i - r(\pi) \nabla_{\pi_i} \overline{T}_i \right) \tag{3.4a}$$

$$\nabla_{\pi_i} r(\pi) \Big|_{\pi^*} = 0 \;\Rightarrow\; \nabla_{\pi_i} \overline{R}_i \Big|_{\pi_i^*} = r(\pi^*) \nabla_{\pi_i} \overline{T}_i \Big|_{\pi_i^*}\,. \tag{3.4b}$$

The extremum above recovers the solution found by Charnov (1976) for their analogous problem – the optimal strategy being to leave a task when the instantaneous reward rate in a task matched the global reward rate (see Equation F.2b). In Section F we proved how Charnov's solution is a maximum when the rewards are concave functions. In theory one can follow the same procedure and differentiate Equation 3.4a once more to look at the Hessian matrix and see if this is a maximum.

In practice the procedure above can be difficult to carry out in general, as policies can be very high-dimensional objects, and because the explicit dependences of the $\overline{R}_i$s and $\overline{T}_i$s with the policies are needed. Hence, although mathematically correct, using Equation 3.4b to improve a policy is not a feasible option for an agent. In the next section we propose a local algorithm to solve this optimisation problem in a practical way. We prove that the algorithm cannot decrease the global reward rate, making it a suitable option for a biological agent.

3.2 A local algorithm to optimise the reward rate

Let us consider an agent currently in task i, and consider also the following local problem where the agent tries to maximize the reward under that particular task discounted by a cost of time ρ, to account for the opportunity cost of spending time in task i,

$$\pi_i^*(\rho) = \arg\max_{\pi_i} \left[\overline{R}_i(\pi_i) - \rho\, \overline{T}_i(\pi_i)\right] . \tag{3.5}$$

We can prove that if an agent updates ρ such as to map the estimated global reward rate $r(\pi)$ at each step,

$$\rho \equiv r(\pi) = \frac{\sum_i p_i \overline{R}_i(\pi_i)}{\sum_i p_i \overline{T}_i(\pi_i)} , \tag{3.6}$$

and iteratively follows through the process of (i) improving the policy through Equation 3.5 and (ii) updating the cost of time ρ, then the global reward rate $r(\pi^t)$, as a sequence in t, cannot decrease.

This process defines an effective algorithm that an agent can use to improve their performance in the multi-task problem described in the preceding section, where the optimal performance in a given task depends on the other tasks only through the global scalar quantity ρ, and not on the other various aspects of those tasks.

To show this, let us consider an improvement on the policy for task j, and consider the shorthand notation $\overline{R}_j^t = \overline{R}_j(\pi_j^t)$ and $\overline{T}_j^t = \overline{T}_j(\pi_j^t)$, and define the auxiliary quantities $a_j^t \equiv \sum_{i \neq j} p_i \overline{R}_i^t$ and $b_j^t \equiv \sum_{i \neq j} p_i \overline{T}_i^t$. Then,

$$\begin{aligned}\rho^{t+1} &= \frac{a_j^{t+1} + p_j \overline{R}_j^{t+1}}{b_j^{t+1} + p_j \overline{T}_j^{t+1}} \\ &= \frac{\left(b_j^t + p_j \overline{T}_j^t\right) r(\pi^t) + p_j\left(\overline{R}_j^{t+1} - \overline{R}_j^t\right)}{b_j^t + p_j \overline{T}_j^{t+1}} .\end{aligned} \tag{3.7}$$

Now we can use the fact that the policy update $\pi_j^t \overset{\text{Eq. 3.5}}{\longrightarrow} \pi_j^{t+1}$ improves on the local problem,

$$\overline{R}_j^{t+1} - \rho^t\, \overline{T}_j^{t+1} \geq \overline{R}_j^t - \rho^t\, \overline{T}_j^t , \tag{3.8}$$

and insert Equation 3.8 into Equation 3.7 to obtain that the reward rate cannot decrease,

$$\begin{aligned}\rho^{t+1} &\geq \frac{\left(b_j^t + p_j \overline{T}_j^t\right)\rho^t + p_j\left(\rho^t \overline{T}_j^{t+1} + \overline{R}_j^t - \rho^t \overline{T}_j^t - \overline{R}_j^t\right)}{b_j^t + p_j \overline{T}_j^{t+1}} \\ &= \frac{\left(b_j^t + p_j \overline{T}_j^t\right)\rho^t + p_j\left(\overline{T}_j^{t+1} - \overline{T}_j^t\right)\rho^t}{b_j^t + p_j \overline{T}_j^{t+1}} \\ &= \rho^t .\end{aligned} \tag{3.9}$$

This result indicates that for the agent to optimise their *global* reward rate performance in the environment they can simply solve *local* optimisation problems while tracking the global variable ρ. By employing ρ as a cost per time the agent can tradeoff the amount of reward they expect to get from staying longer in the current task with the amount of reward they could obtain outside of it. See Figure 3.1 for a graphical representation of this concept.

This is crucial because ρ reflects the overall richness of the environment. When ρ is high, it indicates that the cost per unit of time is also high. In such cases, the agent tends to leave a task earlier, anticipating greater overall gains from other opportunities in the environment.

The algorithm presented above is independent of the particular way in which an agent might optimise the local problem. As long as the local objective

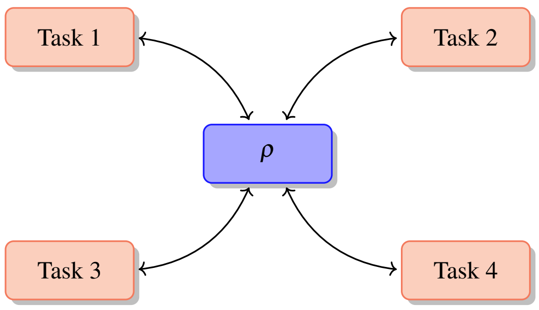

Figure 3.1: Optimal behaviour in a task is effectively decoupled from other tasks. The coupling is only done through the global variable ρ, which tracks the estimate of the overall reward rate across the whole environment. By optimising ρ locally on each task, one at a time, the agent is guaranteed to never decrease their overall reward rate.

$\overline{R}_i(\pi_i) - \rho\, \overline{T}_i(\pi_i)$ is increased at every step, any stable optimisation procedure should work. In particular, a wide range of Reinforcement Learning approaches can be used to improve the policy. For the remainder of the current chapter we focus on environments in which the policy is simple enough that one can optimise with simpler methods, such as Newton-Raphson method.

3.3 Evidence accumulation tasks

In order to make the problem more concrete we apply the concepts from the preceding section into a class of evidence accumulation tasks. In particular, we are inspired by *random-dot kinematogram* tasks, which are common in experimental settings.

On a random-dot kinematogram task the subject is shown a screen with randomly moving dots, where the velocities of the dots are sampled independently from one of a limited set of different distributions. In each trial, the subject must integrate visual information to determine which distribution is generating the data, specifically, the predominant direction of dot movement. A sketch exemplifying the task can be seen in Figure 3.2.

We work with a simplified version of the random-dot kinematogram, presented in Figure 3.3. Instead of moving dots there are real scalar values that the agent needs to integrate as evidence. The generative model that generates the data is hierarchical: for each trial the parameters of a distribution are sampled, then the datapoints are sampled from that distribution. In our case the distributions are Gaussians, $\mathcal{N}(\mu, \sigma^2)$,

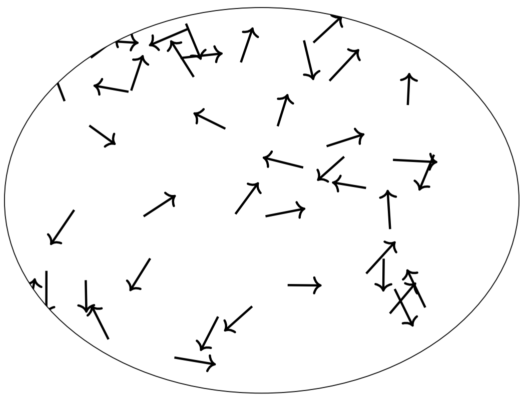

Figure 3.2: A static sketch of the dynamic input a subject faces in a random-dot kinematogram task. The arrows represent the velocities of the dots, and the goal is to integrate information from those velocities in order to determine which distribution generated the input.

and the only parameter that can change is the mean of the distribution, which can be either of $\pm\mu_0$, as in

$$\mu \sim p(\mu) = \frac{1}{2}\delta(\mu - \mu_0) + \frac{1}{2}\delta(\mu + \mu_0)\,. \tag{3.10}$$

For simplicity we assume that the possible mean values are symmetrically opposite, $\pm\mu_0$, such that the possible means for the Gaussian differ only by a sign flip. This is not as restrictive as it might first seem, since when given a generic pair $\mu_1 > \mu_2$ one can always stretch and shift the real numbers to arrive at the pair $\pm\mu_0$. We also assume that the variance of the distributions, σ^2, (the only other hyperparameter) is known to the agent.

Then, within a trial the agent observes samples $x_1, \ldots, x_n, \ldots$ that they use in the evidence accumulation. Conditioned on knowing that the data was generated by a particular mean μ, the generative model for the samples is given by

$$x_n|\mu \sim p(x|\mu) = \mathcal{N}\left(\mu, \sigma^2\right)\,. \tag{3.11}$$

In fact, the agent does not know which mean is generating the data, and the point of the task is to use the sampled data to invert the generative model described by

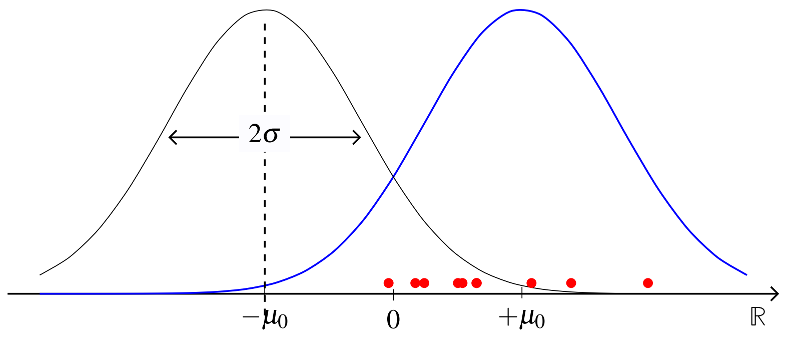

Figure 3.3: A sketch of the task actually implemented, that captures the important features from the random-dot kinematogram task. The Gaussian curves are the different options of probability distributions that could be generating the inputs. In this plot the blue curve is the one generating the data (red dots).

Equations 3.10 and 3.11 and compare the possible options of distributions generating the data to make the correct decision – in the same way that a subject in a random-dot kinematogram task needs to identify the most common direction the dots are moving. This is done via the log ratio of the posterior probabilities, a quantity that is positive if the probability of $\mu = +\mu_0$ is larger, and negative otherwise,

$$\log\left[\frac{p(+\mu_0|\mathcal{D})}{p(-\mu_0|\mathcal{D})}\right]. \tag{3.12}$$

Using Bayes' rule to express the posterior distributions in terms of the likelihood of independently sampled data and the prior distributions, we have

$$\log\left[\frac{p(+\mu_0|\mathcal{D})}{p(-\mu_0|\mathcal{D})}\right] = \log\left[\frac{p(+\mu_0)\prod_n p(x_n|+\mu_0)}{p(-\mu_0)\prod_n p(x_n|-\mu_0)}\right] = \sum_n \log\left[\frac{p(x_n|+\mu_0)}{p(x_n|-\mu_0)}\right], \tag{3.13}$$

where we use the assumption that, in the absence of further information, the alternatives must be equally likely, i.e. $p(+\mu_0) = p(-\mu_0) = 1/2$.

The ratios inside the sum in Equation 3.13, $\log\left[\frac{p(x_n|+\mu_0)}{p(x_n|-\mu_0)}\right]$, is the evidence from datapoint x_i in favour of option $+\mu_0$ ($-\mu_0$) if it is positive (negative). For the particular case of Gaussian distributions, this is given by

$$\log\left[\frac{p(x_n|+\mu_0)}{p(x_n|-\mu_0)}\right] = -\frac{1}{2\sigma^2}\left[[x_n-(+\mu_0)]^2-[x_n-(-\mu_0)]^2\right] = \frac{2\mu_0}{\sigma^2}x_n. \tag{3.14}$$

This means that by doing the inversion of the generative model and comparing the evidence for all datapoints for the different alternatives, one finds that evidence accumulation depends only on the estimated average over the samples, $\sum_n x_n$, so that we can define, for a given instant t, the evidence accumulator variable

$$\mathrm{e}_m = \sum_{n=1}^{m} x_n \,. \tag{3.15}$$

Combining Equation 3.12-3.15, the posterior log ratio depends only on the sampled data through the evidence accumulator variable, e_m, as in

$$\log\left[\frac{p(+\mu_0|\mathcal{D}_m)}{p(-\mu_0|\mathcal{D}_m)}\right] = \frac{2\mu_0}{\sigma^2}\mathrm{e}_m \,, \tag{3.16}$$

which means that the optimal decision policy must depend on the data only through the e_m as well. With that in mind, we choose a family policy that has fixed boundaries $\pm B$ and decides to choose the alternative whenever $\mathrm{e}_m = \sum_{n=1}^{m} x_n$ reaches either side of the decision boundary. See Figure 3.4 for an illustration of this process. The symmetric boundaries result from the symmetric nature of the problem. Given that the value B is the sole parameter for this class of policies, optimising the policy merely involves tuning it.

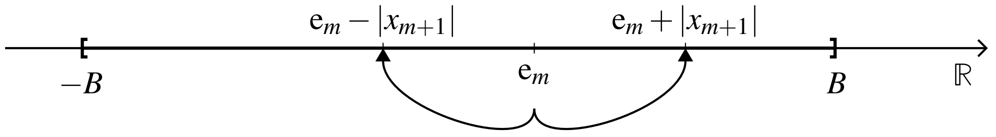

Figure 3.4: Depiction of the evidence accumulation process for the problem described in Equations 3.10 and 3.11, and Figure 3.3. As the evidence accumulator e_m incorporates the information from datapoint x_{m+1} it can go towards either the positive or negative boundaries, $\pm B$, depending on its sign.

Now, in order to connect the evidence accumulation process described above with the multi-task decision-making framework from the previous sections, we need to compute average rewards and times as functions of the policy, $\overline{R}(\pi)$ and $\overline{T}(\pi)$, both necessary for the optimization of objective 3.5. For this problem it can be done analytically, which we present in Appendix H. The final results become

$$\overline{R}(\pi) = \frac{R_+}{\exp\left(-\frac{2\mu_0 B}{\sigma^2}\right)+1} + \frac{R_-}{\exp\left(+\frac{2\mu_0 B}{\sigma^2}\right)+1} \tag{3.17a}$$

$$\overline{T}(\pi) = \frac{B}{\mu_0}\tanh\left(\frac{\mu_0 B}{\sigma^2}\right) \tag{3.17b}$$

where R_+ is the reward for a correct choice, and R_- (usually negative) is for an incorrect one. As expected, having a larger threshold B implies taking more time to make a decision and on average acquiring more reward.

We can then apply Equations 3.17 into the objective in Equation 3.5 to optimise the policy. The optimisation of B depends on the reward rate ρ, which is the variable through which all the tasks are coupled. This means that the same task, when embedded in different multi-task environments, can yield different optimal policies. We explore this in more detail in the next section.

3.4 Explaining apparent suboptimal human behaviour

In everyday life, individuals often encounter situations where they must choose between two closely comparable options, such as deciding to go on holiday either to Paris or to Lisbon. At first it might not be clear to an agent which of those is the better alternative, so they need to accumulate evidence (e.g. browse the Internet for ideas of things to do in each city), which takes time. If the alternatives are equally attractive, an ideal agent that maximises reward rate would not spend a long time accumulating evidence, as the difference in reward would be negligible compared with the extra time gained.

In this section we situate the problem of deciding between two similar alternatives within the context of multi-task environments, applying the results from preceding sections. We claim that humans do not necessarily maximise the reward rate in each task they engage with, but they maximise the global reward rate. This leads to behaviour that is different to what was previously expected in the literature, and could explain the apparent suboptimality just described.

Let us consider an environment where there are two decision tasks: a choice between two high-value options, and a choice between two low-value options. Following the assumption laid down in Section 3.1, we consider an agent in this environment cannot choose which of the tasks to engage with at a given time, but they are presented with the high-value decision task with probability p_1, and with probability $1-p_1$ for the low-value one. For example, the high-value decision could be the choice for the destination of a future holiday, and the low-value decision could be what dessert to have.

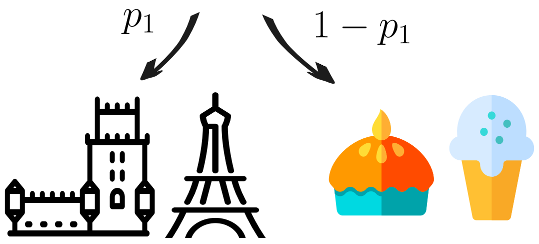

Figure 3.5: The environment considered in our example: there is a high-value decision task that is accessed with probability p_1 and a low-value task that the agent encounters with probability $1-p_1$. Open-licensed icons were taken from SVG Repo.

We consider the decision tasks to be similar to those in Section 3.3: the agent continually accumulates evidence until deciding to go with one of the alternatives. For example, they want to spend time researching about the possible venues for the holiday, or they ask the waiter for more information about the ingredients in the dessert. The objective is to maximise reward rate over long periods of time. For that, for either task, the agent receives signals that come from a Gaussian distribution, $x_n \sim \mathcal{N}(\mu, \sigma^2)$, where μ can be either of $\{+R_0, -R_0\}$ and must be inferred by the agent. The noise level σ^2 is fixed and assumed known.

Upon corrrectly choosing which gaussian is generating the data, the agent receives reward R^a; alternatively they receive R^b if they make the wrong choice. Furthermore, the difficulty of the task (i.e. how large R_0 is, if it is close to zero the task is a hard one) depends on the distance between those rewards,

$$R_0 = \frac{R^a - R^b}{2}. \tag{3.18}$$

The parameters chosen for the environment in the numerical simulations are shown in Table 3.1, where Task 1 is a high-value task, and Task 2 is a low-value one.

	R^a	R^b	R_0	σ^2
Task 1	100	90	5	5
Task 2	11	-7	9	15

Table 3.1: Parameters used for the environment with a high-value and a low-value tasks.

Since we have the explicit dependencies of the average rewards and times on the policies (which are parametrised only by the thresholds for each task, B_1 and B_2) we can optimise the global reward rate directly. Rewriting the objective from Equation 3.1, we seek to optimise

$$r(B_1,B_2) = \frac{p_1\overline{R}_1(B_1) + (1-p_1)\overline{R}_2(B_2)}{p_1\overline{T}_1(B_1) + (1-p_1)\overline{T}_2(B_2) + T_0}, \tag{3.19}$$

where T_0 is the average time that takes for the agent to switch tasks, which we set to $T_0 = 0.1$, and the average rewards and times are given by Equations 3.17, repeated below with the variables adjusted for this problem,

$$\overline{R}(B) = \frac{R^b}{\exp\left(-\frac{2R_0B}{\sigma^2}\right)+1} + \frac{R^a}{\exp\left(+\frac{2R_0B}{\sigma^2}\right)+1} \tag{3.20a}$$

$$\overline{T}(B) = \frac{B}{R_0}\tanh\left(\frac{R_0B}{\sigma^2}\right) \tag{3.20b}$$

We denote the optimal thresholds after maximising over Equation 3.19 as

$$B_1^*, B_2^* = \underset{B_1,B_2}{\arg\max}\ r(B_1,B_2), \tag{3.21}$$

and the corresponding optimal rewards and times are given by

$$\overline{R}_1^* = \overline{R}_1(B_1^*), \quad \overline{T}_1^* = \overline{T}_1(B_1^*) \tag{3.22a}$$

$$\overline{R}_2^* = \overline{R}_2(B_2^*), \quad \overline{T}_2^* = \overline{T}_2(B_2^*). \tag{3.22b}$$

In order to compare the solution found by a maximiser of the global reward rate (Equation 3.19), and the performance of an agent that maximises the reward rate per task, we consider denote the latter as

$$B_1^0 = \arg\max_{B_1} \overline{R}(B_1)/\overline{T}(B_1) \tag{3.23a}$$

$$B_2^0 = \arg\max_{B_2} \overline{R}(B_2)/\overline{T}(B_2)\,, \tag{3.23b}$$

and the corresponding optimal rewards and times are given by

$$\overline{R}_1^0 = \overline{R}_1(B_1^0), \quad \overline{T}_1^0 = \overline{T}_1(B_1^0) \tag{3.24a}$$

$$\overline{R}_2^0 = \overline{R}_2(B_2^0), \quad \overline{T}_2^0 = \overline{T}_2(B_2^0)\,. \tag{3.24b}$$

As explained earlier in this section, we expect the overall performance of the global maximiser to be better than that of the agent that maximises reward rate per task. The results are shown Figure 3.6.

The results in the top panel of Figure 3.6 show how the leaving times that maximise the global reward rate deviate from those that maximise the reward rate on each task separately. For the high-value task the global optimal leaving time takes longer than the local optimal, $\overline{T}_1^* \geq \overline{T}_1^0$. The opposite is true for the leaving time of the low-value task, $\overline{T}_2^* \leq \overline{T}_2^0$. The equalities happen in the limits of visiting one of the tasks only, respectively $p_1 = 1$ or $p_1 \to 0$. This is because the agent visiting just one of the tasks only has the one task to optimise. In the bottom panel we see the rewards also do not follow the result expected from a maximiser of the reward rate for a single task. This is shown with more detail in Figure 3.7, with the relative difference in the rewards,

$$\frac{(R_i^* - R_i^0)}{R_i^0}\,. \tag{3.25}$$

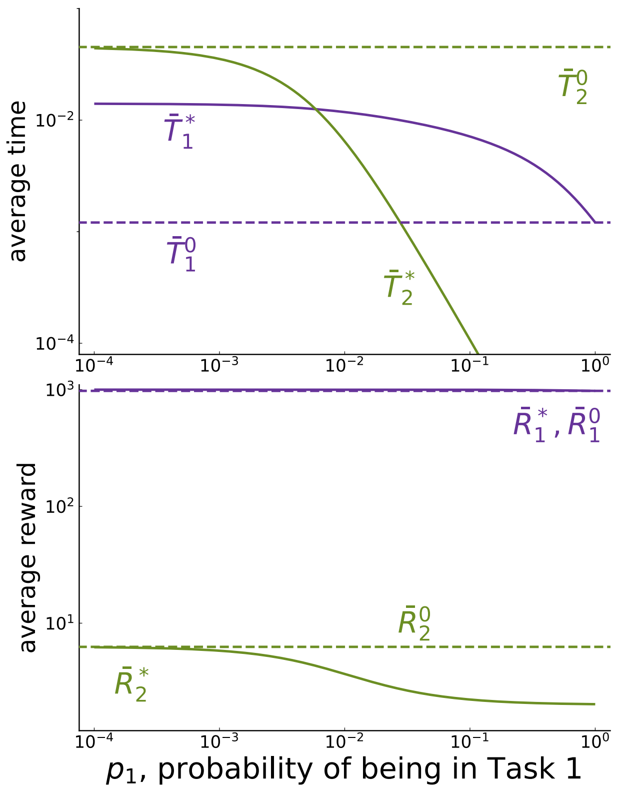

Figure 3.6: Optimal reward gained, $\overline{R}^*$, and time spent in task $\overline{T}^*$, for the high-value decision (Task 1) and the low-value one (Task 2), as a function of p_1, the probability of visiting Task 1. The dashed lines are the baseline set by an agent that maximises the reward rate per task, instead of the global one, as defined in Equations 3.23

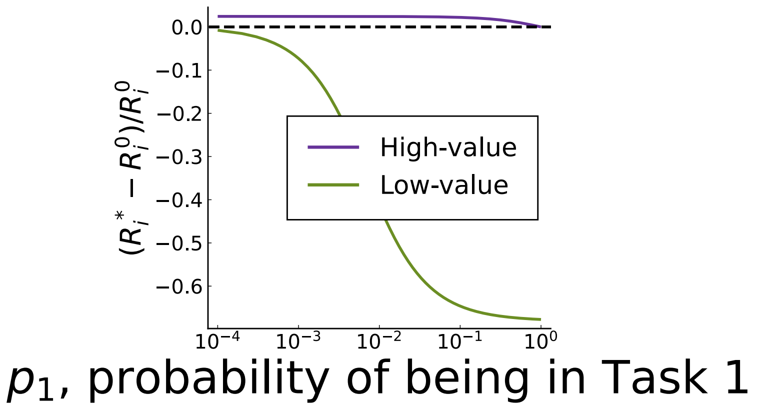

Figure 3.7: Relative difference between the optimal reward, R_i^*, as calculated in Equation 3.21 and the reward from the strategy that maximises reward rate per task, R_i^0.

The result in Figure 3.7 corroborates that of Figure 3.6, presenting how the global reward rate maximiser agent does not maximise the local reward rate per task in this problem (for $p_1 \neq 0, 1$). For visitation probabilities $p_1 < 1$, the global maximiser agent spends more time in the high-value task and collects more reward than the local optimiser agent (indicated by the black dashed line). Again, the opposite is true in the low-value task.

Intuitively, from the perspective of an agent currently in the high-value task, the low-value one can be seen as part of the transit time – it adds extra time that one cannot be in the high-value task. The less frequent the high-value task ($p_1 \to 0$), the longer the agent stays far from it, and the more time they should spend on it. This is because in the extreme of very long waiting times, spending some extra time on the high-value task is beneficial, as it can be very valuable and the agent does not incur in a large cost. If, on the other hand, the high value task happens almost every time ($p_1 \approx 1$), the effective transit time is negligible, and one should just optimize the high value task. In this case, spending longer on the high-value task decerases the average reward rate. This can be seen in Figure 3.8, which compares the reward rates of the different strategies.

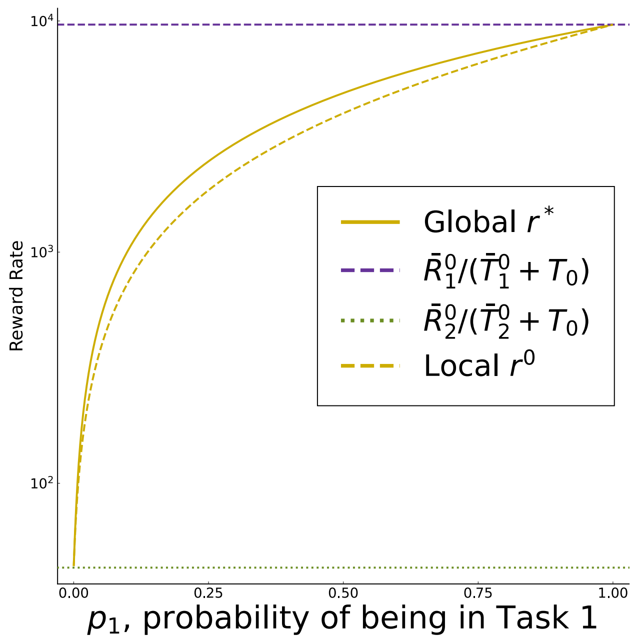

Figure 3.8: Reward rates for the strategy that maximises the global reward rate, r^* (solid gold line), and for the strategy that maximises the local reward rates, r^0 (dashed gold line), for different values of the probability of visiting the high-value task, p_1. Dashed purple line on the top is the maximum reward rate for the high-value task, and the dotted green line on the bottom is the maximum reward rate for the low-value task, which are the limiting cases when $p_1 = 1$ and $p_1 \to 0$, respectively.

These results provide another possible explanation as to why humans spend more time in high-value tasks: when the availability of the high-value tasks is low and the amount of reward one can expect to obtain in the rest of the environment is also low, it is better to perform as best as one can (therefore spending more time to make a more informed decision) in the high-value task to accumulate more reward. As the high-value task becomes more frequent, the agent can safely spend less time in each visitation, since they visit that task more often.

3.5 Discussion

This study contributes to the expanding field of research focused on understanding the decision-making behaviour of optimal agents. Specifically, we explored a category of problems wherein an agent operates within an environment containing multiple tasks, or food patches, and the goal of the agent is to determine the actions that maximize their reward or food acquisition. We developed an algorithm designed to enhance the global performance of the agent by optimising each local problem (i.e. each task) individually, and connecting those via one single scalar parameter, ρ. This parameter tracks the total reward rate that can be achieved in the environment, and functions as an opportunity cost, or cost per unit of time, in the optimisation procedure.

Previous work by Charnov (1976) has indeed offered a solution for determining optimal departure times in environments featuring multiple tasks. This was achieved through the direct optimisation of the global objective. However, these findings do not extend to scenarios where the agent has the liberty to adopt a more complex policy. Consequently, the algorithm outlined in Section 3.2 provides an alternative approach to optimising the global reward rate, so that it might be more suitable in certain applications. Nevertheless, in instances where the policy can be simplified to merely selecting the departure time, both approaches should yield consistent results, as demonstrated in Equation 3.4b.

Inspired by prior studies (Bogacz et al. (2010); Simen et al. (2009); Balci et al. (2011)), which observed subjects in similarly-valued two-alternative forced choice tasks (2AFC) making decisions at a slower pace than expected from reward rate maximisers, we proposed a hypothesis to explain similar seemingly suboptimal human behaviour. We argued that individuals taking too long to make high-value decisions could be optimising their behaviour to maximise reward rate in a multi-task environment and, in this context, we proved that it is in fact optimal to allocate more time to a high-value task when the overall expected reward rate in the environment is low, for example when the high-value task is rarely visited. This is because in an environment with low overall average reward it is crucial to maximise reward when one can, whereas in a rich environment this is not as imperative. While this does not explain the specific examples in Bogacz et al. (2010); Simen et al. (2009); Balci et al. (2011) (as those involve an overstaying behaviour presumably in a low-value task), the importance of examining multi-task considerations as possible confound factors should not be neglected.

Other explanations have been proposed in the literature: results by Balci et al. (2011), Masís et al. (2020) suggest that subjects might take some time to learn the contingencies of the task, and if left longer in the experiment they might learn to behave as expected of an agent that maximizes reward rate; alternatively, Constantino and Daw (2015) hypothesise that subjects possibly make use of a nonlinear utility function when evaluating the different options. At this stage there is no conclusive evidence to either explanation, and an account that combines those factors should also not be discarded.

While past work has proposed investigating foraging and RL conjunctively (Constantino and Daw (2015); Kolling and Akam (2017)), and others have studied extensions of Foraging Theory in cases where the agent must accumulate evidence in order to make more informed decisions (Davidson and Hady (2019)), the work presented in this chapter is unique to have studied the setting of multi-task environments where the policy can be more complex than merely deciding when to leave a task or a food patch. The results presented are consistent with and expand the literature on foraging and decision-making.

Chapter 4

Multi-task decision-making with time-dependent environments

In their daily lives, people engage in a wide range of activities, such as writing a PhD thesis, preparing meals, or watching a movie. Often, these tasks cannot be performed simultaneously, requiring individuals to transition sequentially and make decisions about how to allocate their time and effort. These decisions are influenced by the expected enjoyment or reward associated with each activity, leading individuals to adjust their time investments accordingly.

In real-life environments, dynamic factors can also influence the reward levels associated with different tasks. For instance, factors like satiation and motivation levels fluctuate, affecting how much an agent values activities such as eating or working long hours. Additionally, resource depletion can occur in tasks like foraging, where the availability of resources diminishes as the agent exploits them. Moreover, new opportunities may emerge or disappear, necessitating rapid adaptation.

Given these complexities, how should an agent decide to transition between tasks, considering the various rewarding activities available for them to choose, and taking into account the changes in the environment?

As discussed in Chapter 3, previous work by Charnov (1976) addressed a similar problem in the context of foraging. However, that work considered a static environment where the agent's only decision was the time to leave a food patch, and very little is currently known about decision-making in scenarios where (i) the agent can also choose the next task they engage with, and (ii) the state of the environment changes in response to the agent's decisions.

Possingham and Houston (1990) develops a theoretical account for optimal behaviour when the agent revisits a number of identical food patches. This introduces a dynamics in the environment, as the availability of resources decreases with time spent visiting a given patch, but this formulation does not consider an environment with a variety of food patches, where the agent can choose the next place to visit. The only other study found addressing similar conditions is Hall-McMaster et al. (2021), where authors contrast the behaviour of human participants between two experimental setups: when the subjects can decide which task to visit next, and when they cannot. The authors found that when the subjects could choose where to go next they chose to visit fast-replenishing sites, and attained higher reward rate. When comparing amongst different models, this behaviour was best captured by one that uses information about both average reward rate for the environment and reward information.

While Hall-McMaster et al. (2021) makes a significant contribution to characterising the behaviour for this problem, there remains an opportunity to further explore the underlying theoretical questions about optimal behaviour in multi-task dynamic environments. The research presented in this chapter builds upon this work, with a particular focus on delving deeper into the theoretical aspects. The findings in this chapter offer compelling evidence that the deviations from the original foraging formulation create a setting in which optimal behaviour is strikingly different from that presented by Charnov (1976). Whereas in Charnov's work the optimal leaving time depends on the overall reward rate of the environment, the results obtained for this new setting indicate that policies where the leaving time optimises the reward rate in each task separately are the optimal ones.

The structure of the chapter is as follows: Section 4.1 presents the problem setup and key definitions. Next, in Section 4.2 we derive the theoretical predictions about the optimal policy. For that we make an assumption about the speed of the environment dynamics, and an informed ansatz about the optimal leaving times an agent should pick. In Section 4.3 we explain the Policy-Gradient Reinforcement Learning (RL) methods used to validate the theory, presenting, and comparing, the numerical results for the RL methods and the theory in Section 4.4.

4.1 Problem setup

Similar to Chapter 3, we examine an environment where an agent faces multiple tasks, denoted $i = 1, \ldots, N$, and at each trial t the agent must decide how long to stay in the current task, $s^t = i$, before moving to the next one. In this chapter, we add two new elements to the problem: now the agent can choose the next task they visit; and the environment is dynamic.

This second property means, in a foraging setting, that an animal spending time in one food patch depletes the resources in that patch, and allows resources in other patches to restore; or, a person with different levels of motivation for different tasks gets tired after engaging with the same one repeatedly.

We assume that the dynamics of the environment happens over a vector $\theta \in [0,1]^N$, where each entry, θ_i, is associated with one task, i, and the dynamics is governed by the differential equations

$$\tau_i^{in}\dot{\theta}_i = -\theta_i \qquad \text{if } s^t = i \tag{4.1a}$$

$$\tau_i^{out}\dot{\theta}_i = 1 - \theta_i \qquad \text{if } s^t = j \neq i\,. \tag{4.1b}$$

The time constants τ_i^{in} and τ_i^{out}, respectively for when the agent is in task i or in task $j \neq i$, can be set to be different. This flexibility enables different rates of depletion and replenishment, which is especially valuable when modeling concepts like hunger, where appetite decreases rapidly but recovers slowly. For results in the following section the time constants are set to be equal unless specified otherwise. We conveniently define the dynamics to ensure that θ_i is constrained between $[0,1]$.

The variables θ are state variables, and necessarily part of the environment. However their interpretation can vary depending on the specific problem being modeled: the level of motivation for the reward that can be obtained in a given task; or the amount of resources in a food patch, in the case of foraging.

Still in the context of foraging, we define the reward functions for each task. The resource level θ_i modulates the amount of food (and, therefore, of reward) that the agent can obtain in food patch i. For a lower level, $\theta_i \approx 0$, the agent can find less food than for a higher level, $\theta_i \approx 1$. We consider reward functions of the kind

$$R_i(\theta_i, T_i) = \theta_i \left(R_i^M \tanh(k_i T_i) - R_i^0 \right) , \tag{4.2}$$

where the parameters $R_i^M - R_i^0$ and R_i^0 define the maximum and minimum levels of reward, and k_i defines the steepness of the hyperbolic tangent. This reward function for a choice of parameters can be seen Figure 4.1.

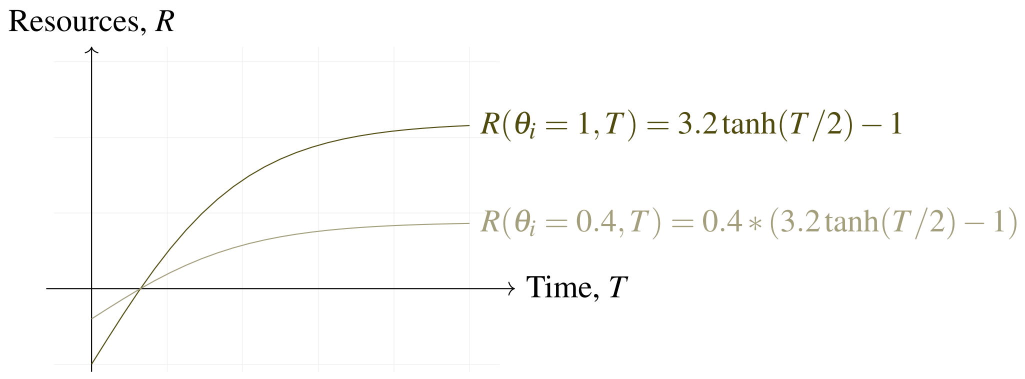

Figure 4.1: Example of a reward function from Equation 4.2 for a set of parameters and different values of the resource level θ_i, which acts multiplying the whole function. The effect of the modulation, since θ_i is limited between $[0, 1]$ is to bring the reward closer to zero, keeping the point where $R = 0$ the same. As explained in Equation 4.3, it also means the time for maximum reward is unaffected.

Ideally an agent seeking to maximise rewards would like to keep the levels fixed at the maximum $\theta_i = 1$ (for all i), but this is not possible due to the dynamics described by Equations 4.1.

The functional form chosen in Equation 4.2 decouples the influence of resources level θ_i on the reward obtained from the effect due to the leaving time T_i,

$$R_i(\theta_i, T_i) \equiv \theta_i R_i(T_i) , \tag{4.3}$$

where from now on we refer to R_i as the function depending only on the leaving time, T_i.

This modelling choice has the advantage that an agent can optimise θ_i separately from optimising T_i, as we explore later on.

Then, we assume that the goal of an agent in this environment is to find the best policy that maximises the expected reward rate over long stretches of time,

$$r(\pi) = \lim_{M\to\infty} \mathbb{E}_\pi \left[\frac{\sum_{t=1}^{M} R^t}{\sum_{t=1}^{M} T^t} \middle| s_0, \theta_0 \right] \tag{4.4}$$

where M is the number of trials, which we take on the limit to infinity, and the expectation is taken over the possible trajectories when following policy π. We assume that the dependence on the initial state (s_0, θ_0) is only temporary and does not carry on in the limit to infinity.

By taking the limit to infinity we expect the sums over time to self-average and we can consider the expectation of the ratio to be well approximated by the ratio of the expectations,

$$r(\pi) = \lim_{M\to\infty} \frac{\sum_{t=1}^{M} \mathbb{E}_\pi \left[R^t | s_0, \theta_0 \right]}{\sum_{t=1}^{M} \mathbb{E}_\pi \left[T^t | s_0, \theta_0 \right]}. \tag{4.5}$$

In fact, in the next section we prove sufficient conditions for this to be the case (see the calculation leading from Equation 4.9 to Equation 4.13).

The objective in Equation 4.4 is similar to the objective in Chapter 3 (Equations 3.1 – 3.3), as both involve optimising the reward rate. Therefore, it is also comparable to the Average-Reward setting in Reinforcement Learning (for which a few relevant references are Blackwell (1962); Puterman (1994); Dewanto et al. (2021); Sutton and Barto (2018)). The difference from the previous chapter is that now we must take into account the environment's dynamics and the agent's freedom to select their next destination when calculating the expectations. Those are dealt with in the next section.

4.2 Theoretical results

Here we propose a policy for the problem defined by Equation 4.4, where the choice of leaving times is given by an informed ansatz, and the choice of where to go next is a calculated theoretical optimal. This informs theoretical predictions for the optimal policy, which are then validated in Section 4.4.

Let us restrict ourselves to situations in which the dynamics in the environment (as set by the τs in Equations 4.1) is slow when compared with the leaving times. This means that a single decision imparts a small effect on the long-term state of the environment. This simplifies the problem letting us further assume that by following the optimal policy the agent ends up spending stationary fractions of times in each task, f_i, and the environment reaches stationary values for the resource levels, θ_i^*. The stationary fractions f_i define a probability distribution of task visitation, for which the optimal values are calculated in this section. Given that we restrict ourselves to the slow dynamics regime, this stationarity assumption is a reasonable one, but there is no guarantee that this will indeed be the case, and this is validated with numerical simulations in Section 4.4.

Due to the slow environment assumption, we assume the agent can choose the best leaving times, T^*, without changing the resource levels substantially,

$$\theta_i^{post} = \theta_i^{pre} e^{-\frac{T_i^*}{\tau_i}} \approx \theta_i^{pre} \left(1 - \frac{T_i^*}{\tau_i}\right) \approx \theta_i^{pre}. \tag{4.6}$$

In this case, a reasonable ansatz is that the agent can optimise the leaving time for the current task without considering the repercussions over the trajectory in the environment, and, to a first approximation, the best thing an agent can do is to optimise the reward being received now. We then propose the ansatz that the optimal agent chooses the leaving time for current task i as the time that maximises the reward rate for that task,

$$T_i^* = \arg\max_{T_i} \frac{R_i(T_i)}{T_i}, \tag{4.7}$$

independently of the current resource levels in the environment, and independently of the parameters for the other tasks. Again, this is a heuristic that is validated in Section 4.4. Furthermore, this ansatz only makes sense if the agent can then choose the transitions between tasks in order to maximise the reward rate globally, which is done by tuning the amount of time spent in each task.

Let us denote by f_i the stationary quantity that describes the fraction of time the agent spends in task i,

$$f_i = \frac{\text{amount of time in task } i}{\text{total time}} = \frac{\sum_t T^t \mathbb{1}\{s^t = i\}}{\sum_t T^t}, \tag{4.8}$$

where T^t is the time spent in trial t.

This hypothesis for the optimal policy is starkingly different from what Charnov (1976) and the results of Chapter 3 suggest for the agent's optimal behaviour. In those, the optimal time to leave a food patch or a task is when the instantaneous reward rate surpasses the level set by the average reward rate in the environment. This time is typically different to the optimal time for a single task.

To validate this proposal, we derive theoretical predictions for the optimal stationary frequency of time spent in each task, $\mathbf{f} = [f_1, \ldots, f_N]$, and the associated reward rate associated, and then test those with numerical simulations in Section 4.4.

Let us consider the stochastic reward rate for a single trajectory,

$$\hat{r} = \lim_{M \to \infty} \frac{\sum_{t=1}^{M} R^t}{\sum_{t=1}^{M} T^t}. \tag{4.9}$$

We now prove that this quantity can be simplified to depend only on the reward rates for each task, ρ_i, the fractions of time the agent spends in each task, f_i, and the stationary values for the resource levels, θ_i^*. This happens because the sums over time self-average into sums over states, and the $\hat{r}$ becomes equivalent (in the limit to infinity) to the expected reward rate $r(\pi)$. Grouping the rewards and times by tasks,

$$\hat{r} = \lim_{M\to\infty} \frac{\sum_{i=1}^{N} \sum_{t_i=1}^{M_i(M)} R^{t_i}}{\sum_{i=1}^{N} \sum_{t_i=1}^{M_i(M)} T^{t_i}}, \tag{4.10}$$

where the $M_i(M)$ are the number of visits to task i, indexed by the trials t_i.

Assuming that the leaving times are those given by Equation 4.7, and that the resource levels are the stationary ones, the expression becomes

$$\hat{r} = \lim_{M\to\infty} \frac{\sum_{i=1}^{N} M_i(M)\theta_i^* R_i(T_i^*)}{\sum_{i=1}^{N} M_i(M) T_i^*}. \tag{4.11}$$

Then we can multiply the numerator inside the sum by T_i^*/T_i^* and obtain

$$\hat{r} = \lim_{M\to\infty} \frac{\sum_{i=1}^{N} M_i T_i^* \theta_i^* \frac{R_i(T_i^*)}{T_i^*}}{\sum_{i=1}^{N} M_i T_i^*} = \lim_{M\to\infty} \sum_{i=1}^{N} f_i(M)\theta_i^* \rho_i. \tag{4.12}$$

Assuming the stationarity of the frequencies of time spent in each task, we obtain the expression for the ensemble average defining the reward rate:

$$\begin{aligned} \hat{r} &= \sum_{i=1}^{N} f_i \theta_i^* \rho_i \\ &= r(\mathbf{f}; \theta, \rho), \end{aligned} \tag{4.13}$$

which is equal to the expected reward rate in Equation 4.4 since the quantities in the right-hand side of Equation 4.13 were already self-averaged by the sums over trials.

We can further simplify the expression for the expected reward rate by considering another aspect of the stationarity assumption. In the steady-state, the resource levels do not deviate substantially from the optimal value, and we can consider that after moving from task i to other tasks and back the level θ_i^* should oscillate and return to the same value. After spending $f_i\mathcal{T}$ time in task i and $(1-f_i)\mathcal{T}$ on other tasks, we have

$$\theta_i^* = 1 - \left(1 - \theta_i^* \, e^{-\frac{f_i\mathcal{T}}{\tau_i}}\right) e^{-\frac{(1-f_i)\mathcal{T}}{\tau_i}} . \tag{4.14}$$

Solving for θ_i^*, we obtain

$$\theta_i^* = \frac{1 - e^{-\dfrac{(1-f_i)\mathcal{T}}{\tau_i}}}{1 - e^{-\left[\dfrac{(1-f_i)}{\tau_i} + \dfrac{f_i}{\tau_i}\right]\mathcal{T}}} . \tag{4.15}$$

As long as the duration $\mathcal{T}$ is long enough to accomodate the oscillation of levels θ_i back to stationary values, but not too long so that it is still smaller than the time constants, $\mathcal{T} \ll \tau_i$, we can Taylor expand the exponentials and obtain

$$\theta_i^* = 1 - f_i , \tag{4.16}$$

which leads to the expression for the reward rate as a function of the frequencies only,

$$r(\mathbf{f}) = \sum_i f_i(1-f_i)\rho_i . \tag{4.17}$$

Now the expression for the reward rate depends only on the stationary distribution f_i, which we can optimise over, and on the optimal reward rates ρ_i, which are assumed to be known. Since the f_is need to describe a probability distribution we need to consider a constrained optimisation problem,

$$\begin{aligned} \max_{\mathbf{f}} \quad & \sum_i f_i(1-f_i)\rho_i \\ \text{s.t.} \quad & f_i \geq 0, \text{ for all } i \\ & \sum_i f_i = 1 . \end{aligned} \tag{4.18}$$

This can be done using the Karush–Kuhn–Tucker (KKT) conditions for the

Lagrangian

$$\mathfrak{L}(\mathbf{f}, \lambda, \nu) = r(\mathbf{f}) + \lambda \left(\sum_i f_i - 1 \right) + \sum_i \nu_i f_i \,. \tag{4.19}$$

Then the corresponding KKT conditions are: the stationarity condition,

$$\frac{\partial \mathfrak{L}}{\partial f_i} = (1 - 2f_i)\rho_i - \lambda + \nu_i = 0\,, \tag{4.20}$$

the primal and dual feasibility conditions,

$$\sum_i f_i = 1 \tag{4.21a}$$

$$f_i \geq 0 \tag{4.21b}$$

$$\nu_i \geq 0\,, \tag{4.21c}$$

and the complemetnary slackness condition,

$$\nu_i f_i = 0\,. \tag{4.22}$$

From the KKT stationarity condition (Equation 4.20) we can isolate the multiplier ν_i, and then use the dual feasibility (Equation 4.21c) to obtain the inequality

$$\lambda \geq (1 - 2f_i)\rho_i\,. \tag{4.23}$$

Applying the equation from the KKT stationarity condition to that of the complementary slackness condition (Equation 4.22), we obtain that

$$f_i[\lambda - (1 - 2f_i)] = 0\,. \tag{4.24}$$

From those we know that the frequency f_i is either zero, or such that it zeros the quantity in square brackets, leading us to the result

$$f_i = \max\left\{0,\ \frac{1}{2} - \frac{1}{2}\lambda\rho_i^{-1}\right\}\,. \tag{4.25}$$

Then, using the condition that the frequencies mut be normalised, we get that

$$1 = \sum_i f_i = \sum_i \max\left\{0,\, \frac{1}{2} - \frac{1}{2}\lambda\rho_i^{-1}\right\}. \tag{4.26}$$

This normalisation condition implies that the value of the multiplier λ sets the scale of overall reward rate for the environment, which in turn defines which tasks are visited and with which frequency (Equation 4.25). If one of the tasks has very low reward rate in comparison with the rest of the environment, $\rho_i \ll \lambda$, the optimal solution is to not visit that task, $f_i = \max\{0, \text{something negative}\} = 0$.

The result in Equation 4.26 is similar to the "water-filling" problem, commonly discussed in the constrained optimisation literature (see e.g. (Boyd and Vandenberghe, 2004, Section 5.5, Example 5.2)), and one cannot find a closed-form solution for it. For a fixed set of $\{\rho_i\}$ it can be solved approximately using numerical methods, iterating over different values for λ.

Nonetheless, it is the case that for some choices of the parameters $\{\rho_i\}$, the solution is in the interior of the feasible domain (that is, $f_i \neq 0$ for all tasks i) which means the multipliers associated with inequalities must be zero (due to complementary slackness, Equation 4.22). Assuming we are in this case, Equation 4.20 simplifes to

$$(1-2f_j)\rho_j - \lambda = 0. \tag{4.27}$$

Isolating the f_j in order to use the normalisation constraint, we find an expression for the Lagrange multiplier,

$$\lambda = (N-2)\left(\sum_i \rho_i^{-1}\right)^{-1}, \tag{4.28}$$

which can then be used in Equation 4.27, to obtain an expression for the optimal frequencies,

$$f_i = \frac{1}{2}\left[1 + (2-N)\left(\rho_i \sum_j \rho_j^{-1}\right)^{-1}\right], \tag{4.29}$$

and from those the optimal reward rate (using Equation 4.17).

This solution is only valid if all the tasks are visited by the agent, $f_i > 0$. This will be the case for the numerical results we discuss in Section 4.4. In particular, some special cases are straightforward to calculate and present intuitive results.

For example, in the case where the are only two tasks in the environment, Equation 4.29 prescribes that the optimal strategy is to visit each task equally frequently, $f_1 = f_2 = 1/2$, regardless of the reward rates in each task. This leads to an average reward of $r(\mathbf{f}) = 2\frac{1}{2}\frac{1}{2}(\rho_1 + \rho_2) = \overline{\rho}$, an average between the reward rates.

Another easy case is one in which there are N tasks with similar reward rates, $\rho_i = \rho$ for all i. Then the sum over the reward rates per task (in Equation 4.29) simplifies, and we obtain the uniform frequencies $f_i = \frac{1}{N}$, reasonable for when all the tasks are effectively the same. The reward rate is then $r(\mathbf{f}) = (1 - \frac{1}{N})\rho$, which makes sense in light of the interim results we used to get to the final equation, namely Equation 4.16, $\theta_i = 1 - f_i$, and Equation 4.13, $r(\mathbf{f}, \theta) = \sum_i f_i$.

4.3 Policy Gradient methods

In order to test the theoretical results derived in Section 4.2, we employ Policy-Gradient Reinforcement Learning methods to solve the problem defined by Equation 4.4. The reasons for choosing a policy gradient approach are twofold: first, it naturally handles continuous actions, such as the leaving time, T^t; second, policy gradient methods are more suitable for the goal of studying the behavior of an agent as they do not need to learn the policy through a value function.

In this section we explain the basic concepts behind the algorithms used in the simulations, and also clarify specific nuances needed to use them in the multi-task problem described in this chapter. In the next section we show that the RL methods explore similar solutions to the ones predicted by the theory for a number of different environments.

The objetive function we seek to optimise is the expected reward rate for a parameterised policy, $r(\pi_{\mathbf{w}})$, which the Policy Gradient Theorem (see (Sutton and Barto, 2018, Section 13.6)) asserts can be optimised by following the direction of the gradient of the policy,

$$\nabla_{\mathbf{w}} r(\pi_{\mathbf{w}}) \propto \mathbb{E}_{\pi}\left[G^t \nabla_{\mathbf{w}} \log \pi_{\mathbf{w}}(a^t, T^t | s^t, \theta^t t)\right], \tag{4.30}$$

where G^t is the differential return, defined as

$$G^t = R^t - r(\pi)T^t + G^{t+1}. \tag{4.31}$$

Intuitively, Equation 4.30 affirms that to change parameters $\mathbf{w}$ in order to follow the direction that increases the reward rate, $\nabla_{\mathbf{w}} r(\pi_{\mathbf{w}})$, one can follow the direction that increases the probability of visiting state s_t and choosing action (a_t, T_t), weighed by the differential return, G_t, observed after following that state-action pair. This is then averaged over different trajectories when following policy π.

In the following sections, we present results using two policy gradient algorithms: REINFORCE, which is a Monte Carlo method; and Actor-Critic, which is a Temporal Difference method. For an introduction to Monte Carlo and TD methods in RL see Appendix G.

REINFORCE is the simplest policy gradient method, and directly optimises the policy without a critic – that is, without the need to estimate the value function –, which means it requires batches of data to estimate the return in Equation 4.30.

As a Policy-Gradient method, REINFORCE requires an explicit parameterisation for the policy. We use a lognormal distribution for the leaving times, T_t, and a softmax on a scores table for the choice of next task, a_t, such that

$$\pi(T|s,\theta,\mathbf{w}) = \frac{1}{T\sigma_{\mathbf{w}}(s,\theta)\sqrt{2\pi}} \exp\left(-\frac{(\ln T - \mu_{\mathbf{w}}(s,\theta))^2}{2\sigma_{\mathbf{w}}(s,\theta)^2}\right) \tag{4.32a}$$

$$\pi(a|s,\theta,\mathbf{w}) = \frac{e^{S(a|s,\theta)}}{\sum_{a'} e^{S(a'|s,\theta)}}, \tag{4.32b}$$

where the parameters of the distributions – μ, σ and S – are constructed through a forward pass on feedforward neural networks parameterised by $\mathbf{w}$. See Appendix I for more details.

The learning rules for REINFORCE are given by

$$\delta^t = \sum_{k=1}^{t_{max}-t} \left(R^{t+k} - \rho T^{t+k}\right) \tag{4.33a}$$

$$\Delta \mathbf{w} = \eta_{\mathbf{w}} \left\langle \delta^t \nabla_{\mathbf{w}} \log \pi(a^t | s^t, \theta^t, \mathbf{w}) \right\rangle_t \tag{4.33b}$$

$$\Delta \rho = \eta_\rho \left\langle \delta^t \right\rangle_t , \tag{4.33c}$$

where the averages $\langle \cdot \rangle_t$ are over the trials in a batch. The calculation of the gradients is done via automatic differentiation (see Appendix I for more details).

The Actor-Critic algorithm, as an online algorithm, updates its parameters at every trial instead of only in batches. Additionally, it is a Temporal-Difference method, meaning it learns a value function as well as the policy, in order to estimate the differential return. Defining the differential value as the expected differential return from Equation 4.31, we have

$$\begin{aligned} v_\pi(s,\theta) &= \mathbb{E}\left[G_t | s^t = s, \theta^t = \theta\right] \\ &= \overline{R}_\pi(s,\theta) - r(\pi)\overline{T}_\pi(s,\theta) \\ &\quad + \sum_{s',\theta'} \pi(a,T|s,\theta) p(s',\theta'|s,\theta,a,T)\; v_\pi(s',\theta') . \end{aligned} \tag{4.34}$$

For the Actor-Critic we assume the leaving times are fixed at the optimal times from Equation 4.7,

$$T_i^* = \arg\max_{T_i} \frac{R_i(T_i)}{T_i} , \tag{4.35}$$

and the decision of the next task is sampled from a softmax on the table $h(a|s)$, with the addition of a linear layer on the resource levels θ,

$$\pi(a|s,\theta) = \frac{e^{h(a|s)+\mathbf{w}_a \cdot \theta^t}}{\sum_j e^{h(j|s)+\mathbf{w}_j \cdot \theta}} . \tag{4.36}$$

The update rules for the Actor-Critic with eligibility traces, in a continuing problem, are then given by

$$\delta^t = R^t - \rho T^t + V(s^{t+1}) - V(s^t) \tag{4.37a}$$

$$\Delta z^h = (1-\lambda^h)\nabla_h \log \pi(a^t|s^t,\theta^t) \tag{4.37b}$$

$$\Delta z^{\mathbf{w}} = (1-\lambda^{\mathbf{w}})\nabla_{\mathbf{w}} \log \pi(a^t|s^t,\theta^t) \tag{4.37c}$$

$$\Delta z^V = (1-\lambda^V)\delta_{s,s^t} \tag{4.37d}$$

$$\Delta h = \eta^h \delta^t z^h \tag{4.37e}$$

$$\Delta \mathbf{w} = \eta^{\mathbf{w}} \delta^t z^{\mathbf{w}} \tag{4.37f}$$

$$\Delta V = \eta^V \delta^t z^V \tag{4.37g}$$

$$\Delta \rho = \eta^\rho \delta^t\,, \tag{4.37h}$$

where all trace-decay parameter, the λs, and learning rates, ηs, are hyperparameters that take value in the interval $[0,1]$. Notice that we decide to not give information about θ^t to the value table V, despite those being state variables – this is an approximation to avoid needing to use function approximation due to the continuous variables θ.

The gradients in Equations 4.37b and 4.37c can be calculated analytically, and are given by

$$\partial_{h(a|s)} \log \pi(a^t|s^t) = \left[\delta_{a,a^t} - \pi(a|s^t,\theta^t)\right]\delta_{s,s^t} \tag{4.38a}$$

$$\partial_{w_{a,s}} \log \pi(a^t|s^t) = \left[\delta_{a,a^t} - \pi(a|s^t,\theta^t)\right]\theta^t_s\,. \tag{4.38b}$$

4.4 Numerical results

In this section we compare the performance expected from the theoretical predicitons and that of the Policy Gradient method described in Section 4.3 across 6 different environments, described in Table 4.1. These environments differ by the choice of parameters – R^M and R^0, that define the maximum and minimum reward, k, the steepness of the hyperbolic tangent in Equation 4.2, and τ, the time constant for the dynamics – for which a full account of task parameter values is given in Appendix J.

Name	# Tasks	Short description
Env 1	2	Optimal times in different timescales ($T_2^* > T_1^*$); Task 1 faster and more rewarding
Env 2	2	$T_2^* \sim 3\ T_1^*$; Task 2 is *more* rewarding than Task 1
Env 3	2	$T_2^* \sim 3\ T_1^*$; Task 2 is *less* rewarding than Task 1
Env 4	3	All 3 tasks are equal, but with different time constants τ_i
Env 5	5	All 5 tasks are identical
Env 6	11	Task 1 is very rewarding; other 10 tasks are less rewarding; optimal time are similar

Table 4.1: Description of the different environments studied in Section 4.4. A more complete description is given in Appendix J.

By running for a number of trials the policies learnt by REINFORCE and Actor-Critic after a number of training iterations, we can assess how these policies compare with the prediction set in Section 4.2. Firstly, Figure 4.2 shows a comparison between the leaving times learnt by REINFORCE and those given by the ansatz in Equation 4.7. The leaving times for the Actor-Critic model are not presented since those were hardcoded to be the same as the ansatz. The leaving times are mostly congruent, with the notable exception of that associated with Task 2 in Environment 1. As can be seen in the right panel of Figure 4.3, this is due to the flatness of the reward rate for that task around the optimal time, making it a harder optimisation problem.

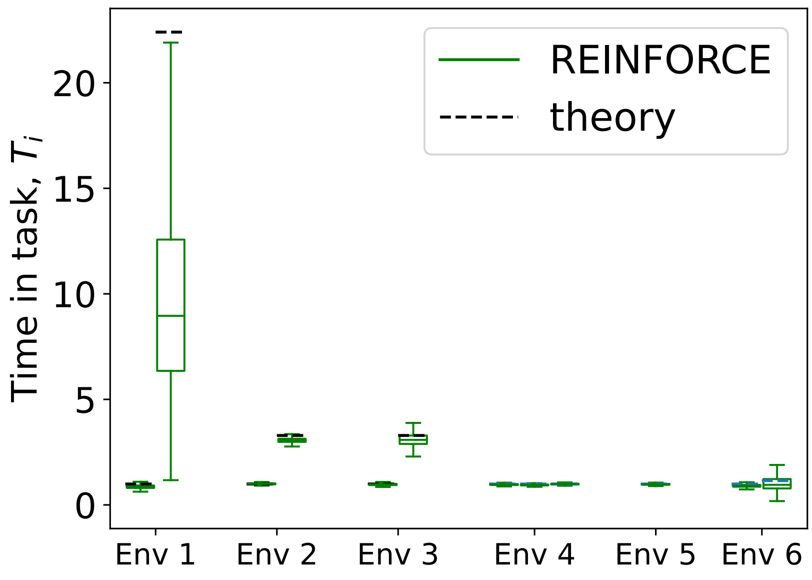

Figure 4.2: Comparisons of leaving times between the REINFORCE-learnt policies (green) and the local optimal leaving time (purple).

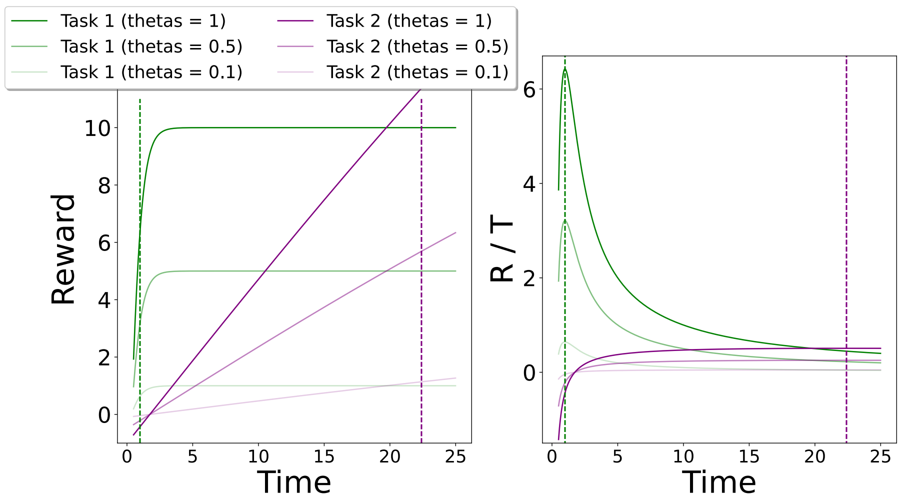

Figure 4.3: Environment 1. **Left.** Reward as function of leaving time for varying levels of θ. **Right.** Reward *rate*, as function of leaving time and for a few levels of θ. In both panels the dashed lines are the leaving times that maximise reward rate for each task

Then, Figure 4.4 presents the performance of the theoretical optimal and that of the policies found by the algorithms. The results are consistent with the theory in most cases, and the discrepancies between theory and simulations might be attributable to stopping the learning too early.

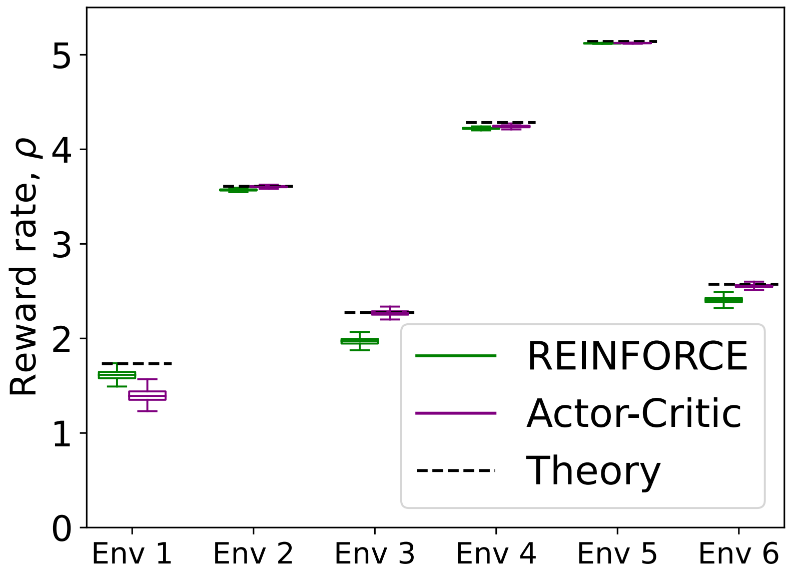

Figure 4.4: Comparison between the reward rates found by REINFORCE and Actor-Critic, and those predicted by theory.

The relative frequencies of time spent in each task, f_i, are shown in Figure 4.5. The figure shows that simulation results are moderately consistent with the theoretical predictions: overall the policies found by Actor-Critic had a better agreement with the theory, but neither REINFORCE or Actor-Critic achieved the levels set by the theory for Environment 1.

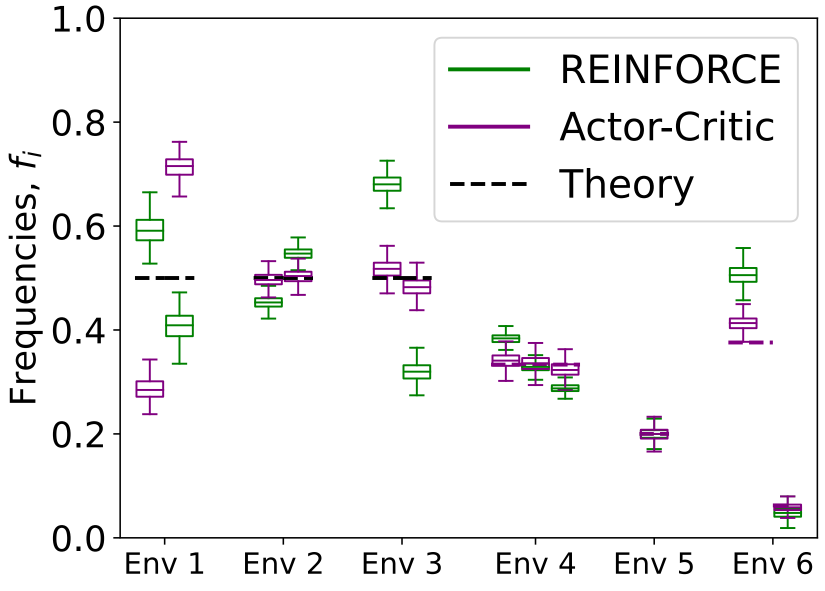

Figure 4.5: Comparison between predicted frequencies of time spent in each task, f_i, and measured ones.

Overall the results presented above indicate that the Policy Gradient methods seem to obtain policies that are similar amongst them, and similar to the theoretical predictions. The discrepancies between them might be explained mostly by stopping the algorithms before they had fully learned the optimal policy. In order to better assess this, Figures 4.6 – 4.8 present the performance of the policies learnt by REINFORCE through learning. Panels C and D from these figures show that REINFORCE quickly (in the order of tens of thousands of episodes) learns a policy for which the reward rate is close to the theoretical optimal. The baseline ρ, which is learned by the algorithm to track the reward rate correctly does so. Panels A and B from the figures show that the solutions encountered by REINFORCE in fact maintain the resource levels θ around stationary values, although not always exactly the ones predicted by the theory. The results are relatively robust throughout different environments with varying: number of tasks, optimal leaving times, maximum reward rate, and time constants.

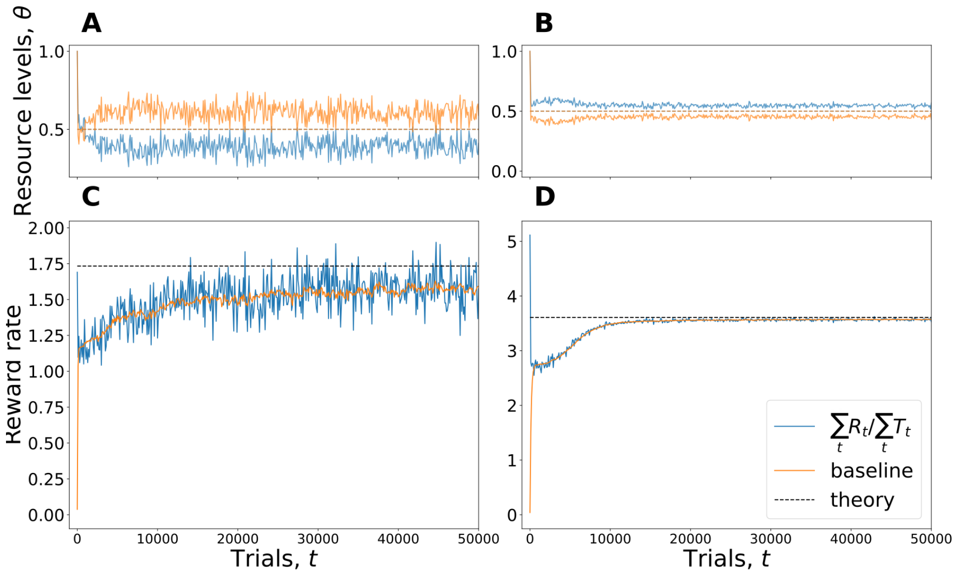

Figure 4.6: REINFORCE through learning. Environments 1 and 2 are the same from Table 4.1. **A, C.** Environment 1 **B, D.** Environment 2. Dashed lines are theory predictions for the stationary θ_i^*s and stationary reward rates, $r(\pi^*)$.

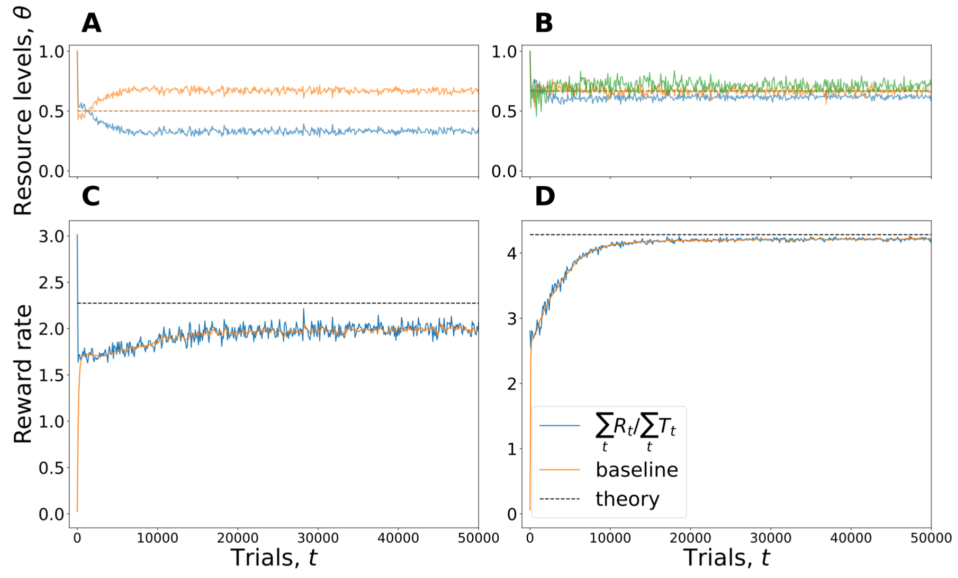

Figure 4.7: REINFORCE through learning. Environments 3 and 4 are the same from Table 4.1. **A, C.** Environment 3 **B, D.** Environment 4. Dashed lines are theory predictions for the stationary θ_i^*s and stationary reward rates, $r(\pi^*)$.

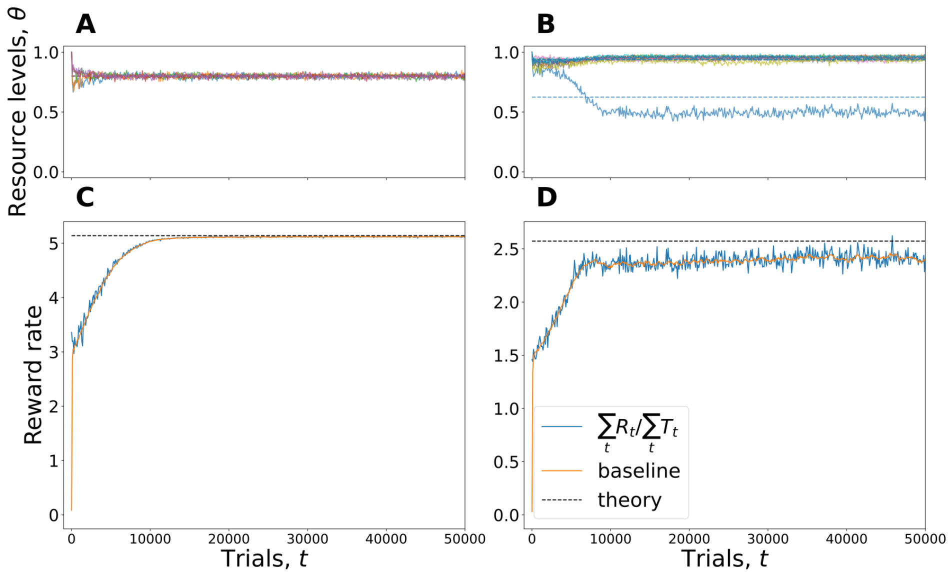

Figure 4.8: REINFORCE through learning. Environments 5 and 6 are the same from Table 4.1. **A, C.** Environment 5 **B, D.** Environment 6. Dashed lines are theory predictions for the stationary θ_i^*s and stationary reward rates, $r(\pi^*)$.

Finally, as shown in Figure 4.9, the policies found by the algorithms are internally consistent with the theoretical predictions for the relation beween stationary values θ_i^* and frequencies of time spent in each task, f_i, despite not achieving the theoretical optimal policy.

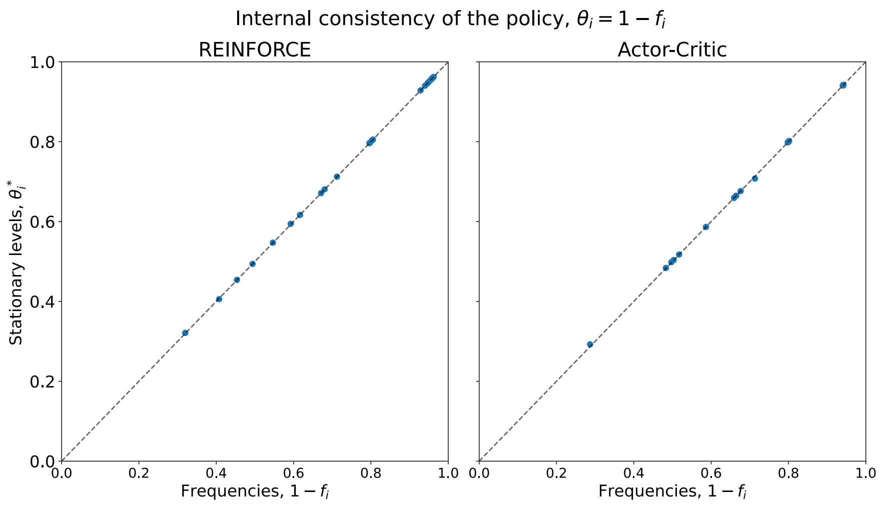

Figure 4.9: Internal consistency between stationary resource levels θ_i^* and measured visitation frequency of different tasks, f_i. Dashed lines are the identity. Both are averages over $M = 500,000$ trials.

These results indicate that in the regime of slow dynamics, $\tau \gg T^*$, an agent that chooses the leaving times as to optimise the local reward rate for each task can indeed achieve a stationary state for the resource levels, θ^*, and task visitation, $\mathbf{f}$, as postulated in Section 4.2. Furthermore, generic Policy-Gradient algorithms from RL for learning optimal policies learn similar policies than the ones predicted by theory, but perhaps due to early stopping of their training procedures, fail to reach the level of performance expected from the theoretical predictions.

4.5 Discussion

In this chapter, we extended the foraging problem to encompass more naturalistic environments, where the state of the world evolves over time, varying based on task visits. Additionally, we let the agent select their next destination, aligning with scenarios commonly encountered by biological agents. This extension of the problem led to notably different outcomes compared to prior formulations of the multi-task leaving time decision problem.

While in Charnov (1976) and in Chapter 3 the optimal time for an animal to leave a food patch was determined by the global reward rate, in the formulation explored in the current chapter the best policy prescribes a departure time based solely on local factors. In this way, in order to maximise the global reward rate, the agent needs to adjust the transition probabilities only.

This chapter provided theoretical results to support this claim, and verified those with simulation experiments. The main assumptions used to derive the results were: a slow dynamics in the environment (when comparing the time constants τ with the leaving times), and the stationarity of the environment for long durations. Those proved sufficient to derive results for the optimal policy, which were then corroborated by simulations using Policy-Gradient RL methods under the same assumptions.

This behavior is convenient in the context of a dynamic environment where the agent faces a variety of tasks. Rather than attempting to simultaneously optimise all tasks, it is more efficient to initially master the optimal leaving time for each task separately, and subsequently learn how to better integrate them by selecting future tasks among the available options. Consequently, if there are any changes to the availailty of tasks (say the agent encounters a new task, or one of the previous tasks becomes unavailable), the optimal leaving tasks can still be the same; rather, the agent only needs to adjust the transition probabilities between tasks. In Charnov (1976) and in Chapter 3 this is not the case, and the agent needs to review the leaving times for all tasks whenever there is a change to the environment.

It is left for further research to assess whether one can relax the assumptions made in this chapter and still obtain similar results. Biological agents often encounter situations in which the dynamics of the environment is fast, or situations in which the rates of depletion of resources are different to those of replenishment.

In sum, this work presents a new multi-task decision-making setting, extending the usual foraging problem where an animal moves between food patches. In our setting, the agent can decide not only when to leave a food patch (or a task), but also which one to engage with next. We further introduce a slow dynamics to the environment, and provide theoretical and numerical evidence to substantiate the novel claim the the optimal behaviour in this environment differs from the behaviour described in the literature, in particular that of Charnov (1976). This has implications for the broad understanding of optimal behaviour in multi-task problems decision-making problems, and raises questions for future research.

Bibliography

Amit, D. J., Gutfreund, H., and Sompolinsky, H. (1985). Storing Infinite Numbers of Patterns in a Spin-Glass Model of Neural Networks. *Phys. Rev. Lett.*, 55(14):1530–1533.

Applegate, M. C. and Aronov, D. (2022). Flexible use of memory by food-caching birds. *eLife*, 11:e70600.

Balci, F., Simen, P., Niyogi, R., Saxe, A., Hughes, J. A., Holmes, P., and Cohen, J. D. (2011). Acquisition of decision making criteria: Reward rate ultimately beats accuracy. *Atten Percept Psychophys*, 73(2):640–657.

Biderman, N., Bakkour, A., and Shohamy, D. (2020). What Are Memories For? The Hippocampus Bridges Past Experience with Future Decisions. *Trends in Cognitive Sciences*, 24(7):542–556.

Blackwell, D. (1962). Discrete Dynamic Programming. *Ann. Math. Statist.*, 33(2):719–726.

Bogacz, R. (2022). Speed Accuracy tradeoff. In Jaeger, D. and Jung, R., editors, *Encyclopedia of Computational Neuroscience*. Springer New York, New York, NY.

Bogacz, R., Brown, E., Moehlis, J., Holmes, P., and Cohen, J. D. (2006). The physics of optimal decision making: A formal analysis of models of performance in two-alternative forced-choice tasks. *Psychological Review*, 113(4):700–765.

Bogacz, R., Hu, P. T., Holmes, P. J., and Cohen, J. D. (2010). Do humans produce the speed–accuracy trade-off that maximizes reward rate? *The Quarterly Journal of Experimental Psychology*, 63(5):863–891.

Boyd, S. P. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press, Cambridge, UK ; New York.

Charnov, E. L. (1976). Optimal foraging, the marginal value theorem. *Theoretical Population Biology*, 9(2):129–136.

Clayton, N. S. and Russell, J. (2009). Looking for episodic memory in animals and young children: Prospects for a new minimalism. *Neuropsychologia*, 47(11):2330–2340.

Constantino, S. M. and Daw, N. D. (2015). Learning the opportunity cost of time in a patch-foraging task. *Cogn Affect Behav Neurosci*, 15(4):837–853.

Corless, R. M., Gonnet, G. H., Hare, D. E. G., Jeffrey, D. J., and Knuth, D. E. (1996). On the LambertW function. *Adv Comput Math*, 5(1):329–359.

David, H. A. and Nagaraja, H. N. (2004). *Order Statistics*. John Wiley & Sons.

Davidson, J. D. and Hady, A. E. (2019). Foraging as an evidence accumulation process. *PLOS Computational Biology*, 15(7):e1007060.

Dewanto, V., Dunn, G., Eshragh, A., Gallagher, M., and Roosta, F. (2021). Average-reward model-free reinforcement learning: A systematic review and literature mapping.

Gold, J. I. and Shadlen, M. N. (2007). The Neural Basis of Decision Making. *Annual Review of Neuroscience*, 30(1):535–574.

Hall-McMaster, S., Dayan, P., and Schuck, N. W. (2021). Control over patch encounters changes foraging behavior. *iScience*, 24(9).

Heitz, R. P. (2014). The speed-accuracy tradeoff: History, physiology, methodology, and behavior. *Frontiers in Neuroscience*, 8:150.

Kilpatrick, Z. P., Davidson, J. D., and El Hady, A. (2021). Uncertainty drives deviations in normative foraging decision strategies. *Journal of The Royal Society Interface*, 18(180):20210337.

Kolling, N. and Akam, T. (2017). (Reinforcement?) Learning to forage optimally. *Current Opinion in Neurobiology*, 46:162–169.

Kumaran, D., Hassabis, D., and McClelland, J. L. (2016). What Learning Systems do Intelligent Agents Need? Complementary Learning Systems Theory Updated. *Trends in Cognitive Sciences*, 20(7):512–534.

Lengyel, M. and Dayan, P. (2008). Hippocampal Contributions to Control: The Third Way. *Neural Information Processing Systems*, page 8.

Madan, C. R. (2019). Rethinking the definition of episodic memory. *PsyArXiv*.

Mahr, J. B. and Csibra, G. (2018/ed). Why do we remember? The communicative function of episodic memory. *Behavioral and Brain Sciences*, 41.

Masís, J., Chapman, T., Rhee, J. Y., Cox, D. D., and Saxe, A. M. (2020). Rats strategically manage learning during perceptual decision making.

McClelland, J. L. and O'Reilly, R. C. (1995). Why There Are Complementary Learning Systems in the Hippocampus and Neocortex:InsightsFrom the Successesand Failuresof Connectionist Models of Learning and Memory. *Psychological Review*, 3(102):419–457.

Nagy, D. G. and Orbán, G. (2017). Episodic memory for continual model learning. *arXiv:1712.01169 [cs, stat]*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Possingham, H. P. and Houston, A. I. (1990). Optimal patch use by a territorial forager. *Journal of Theoretical Biology*, 145(3):343–353.

Puterman, M. (1994). Average Reward and Related Criteria. In *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, chapter 8, pages 331–440. John Wiley & Sons, Ltd.

Simen, P., Contreras, D., Buck, C., Hu, P., Holmes, P., and Cohen, J. D. (2009). Reward rate optimization in two-alternative decision making: Empirical tests of theoretical predictions. *Journal of Experimental Psychology: Human Perception and Performance*, 35(6):1865–1897.

Sprechmann, P., Jayakumar, S. M., Rae, J. W., Pritzel, A., Badia, A. P., Uria, B., Vinyals, O., Hassabis, D., Pascanu, R., and Blundell, C. (2018). Memory-based Parameter Adaptation. *arXiv:1802.10542 [cs, stat].*

Sugiyama, M. (2015). *Statistical Reinforcement Learning: Modern Machine Learning Approaches*. CRC Press.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning, Second Edition: An Introduction*. MIT Press.

Tulving, E. (1972). Episodic and Semantic Memory. In *Organization of Memory*. Academic Press.

Appendices

Appendix A

Introduction to Supervised Learning

In a supervised learning scenario the goal is to learn a mapping f from an input variable $\mathbf{x}$ to an output one $y = f(\mathbf{x})$. This is a general framework and applies to the case where one uses information about the state of the world $\mathbf{x}$ to estimate an approximation of the value of being in that state $f(\mathbf{x})$.

Data for training comes in input-output pairs $(\mathbf{x}, y)$, and we denote collections of such pairs with a full dataset $\mathcal{D}_M$, where M is the number of pairs in the dataset. Different models are used to learn the mapping from input to output data, and each of them makes (possibly different) prediction $\hat{y}(\mathbf{x})$ of the output given the input.

In order to quantify how much error a given model is making, one needs to define a **pointwise error** function, such as the square loss:

$$\ell(\hat{y}, y) = \frac{1}{2}(y - \hat{y})^2 . \tag{A.1}$$

Furthermore, we do not want a given model simply to perform well on a particular pair $(\mathbf{x}, y)$, but on the whole set of possible inputs and outputs $\mathcal{X} \times \mathcal{Y}$, weighted by the probabilities that a given pair can be observed, $p(\mathbf{x}, y)$. Therefore, we want the prediction of the model to minimise the **generalisation error**, which is the loss function of the model under the probability distribution that generates the data,

$$\mathcal{L}(\hat{y}(.)) = \langle \ell(\hat{y}(\mathbf{x}), y) \rangle_{\mathbf{x},y} = \left\langle \frac{1}{2}(y - \hat{y}(\mathbf{x}))^2 \right\rangle_{\mathbf{x},y} , \tag{A.2}$$

where $\langle . \rangle_{\mathbf{x},y}$ represents an average taken over $p(\mathbf{x}, y)$.

However, by definition, one does not have access to the joint distribution $p(\mathbf{x}, y)$ in actual train time, so the generalisation loss cannot be calculated. This implies that a proxy must be sought. Simply substituting the analytical average $\langle . \rangle_{\mathbf{x},y}$ for the sampling average one obtains the **empirical loss**,

$$\widehat{\mathcal{L}}(\hat{y}(.), \mathcal{D}_P) = \sum_{(\mathbf{x},y)\in\mathcal{D}_P} \ell(\hat{y}(\mathbf{x}), y) = \frac{1}{2}\sum_{p=1}^{P} (y_p - \hat{y}(x_p))^2 , \tag{A.3}$$

where $\mathcal{D}_P$ is the train dataset with P training pairs as examples.

Typically one works with parametric models, where we express the collection of parameters generically as a vector $\mathbf{w}$,

$$\hat{y} \equiv \hat{y}_{\mathbf{w}}(\mathbf{x}) . \tag{A.4}$$

The goal of learning the mapping f becomes that of optimising the parameter vector $\mathbf{w} \to \mathbf{w}^*$ such that the parameters minimise the loss

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \mathcal{L}(\hat{y}_{\mathbf{w}}(.)) . \tag{A.5}$$

Since the generalization loss cannot be calculated directly the empirical loss is used. A common way to learn with this model is to follow gradients of the loss for training datapoints $(\mathbf{x}, y)$ in the dataset $\mathcal{D}$, either online or in batches of data. This procedure is known as **stochastic gradient descent** (SGD).

$$\Delta\mathbf{w} = -\eta \nabla_{\mathbf{w}} \widehat{\mathcal{L}}(\mathbf{w}, \mathcal{D}_t) , \tag{A.6}$$

where η is the *learning step*, a hyperparameter than can be tuned.

Appendix B

Introduction to Order Statistics

Order Statistics is a subfield of Statistics that is concerned with the study of probability distributions of ordered sets. We refer the interested reader to the book by David and Nagaraja (2004) for a thorough exposition of the basic results presented here.

Let us consider a distribution $p(x)$, from which we sample $X_1, X_2, \cdots X_M \overset{i.i.d.}{\sim} p(x)$, and we order them in ascending order

$$X_{(1)} \leq X_{(2)} \leq \cdots \leq X_{(M)} \,. \tag{B.1}$$

By definition,

$$X_{(1)} = \min\{X_1,\ \cdots,\ X_M\} \tag{B.2a}$$

$$X_{(N)} = \max\{X_1,\ \cdots,\ X_M\} \tag{B.2b}$$

A common result from Order Statistics describes how to obtain the distributions for each of the $X_{(i)}$, or their joints.

We present in particular the cdf for the first statistic, $X_{(1)}$, as we are interest in Chapter 2 on the distribution of the nearest neighbour,

$$C_{(1)}(x) = 1 - [1 - C(x)]^M \,, \tag{B.3}$$

where $C(x)$ is the cumulative distribution function (cdf) for the X random variable,

$$C(x) = \int_{-\infty}^{x} \mathrm{d}x' p(x') \,, \tag{B.4}$$

which is non-decreasing by definition.

By the Fundamental Theorem of Calculus, one can differentiate the cdf to recover the pdf,

$$p_{(1)}(x) = \frac{\mathrm{d}}{\mathrm{d}x} C_{(1)}(x) = M\left[1 - C(x)\right]^{M-1} p(x) \,. \tag{B.5}$$

Another way to derive this pdf is as follows: from all the M samples only one can be the smallest[1], but a priori the smallest can be any of them, then we need to sum over all of the possible samples. Following this reasoning, the smallest sample is associated with the pdf given by the orginal p, and by construction all the other samples need to have larger values, hence the converse of the cdf. Expressing mathematically the procedure above, we have

$$\begin{aligned} p_{(1)}(x) &= \sum_{i=1}^{M} P(x_i = x,\ x_{\neg i} > x) \\ &= M p(x) \left[\int_{x}^{\infty} \mathrm{d}x' p(x') \right]^{M-1} \\ &= M\left[1 - C(x)\right]^{M-1} p(x). \end{aligned} \tag{B.6}$$

One can follow similar procedures in general and obtain the marginal for any order statistic,

$$p_{(l)}(x) = \frac{1}{1} \left[C(x)\right]^{l-1} \left[1 - C(x)\right]^{M-l} p(x) \,, \tag{B.7}$$

and the joint for a pair of order statistics $X_{(l)}$ and $X_{(m)}$ (where $l < m$),

$$\begin{aligned} p_{(l)(m)}(x,y) = & \frac{M!}{(l-1)!(m-l-1)!(M-m)!} \left[C(x)\right]^{l-1} p(x) \\ & \left[C(y) - C(x)\right]^{m-l-1} p(y) \left[1 - C(y)\right]^{M-m} . \end{aligned} \tag{B.8}$$

[1]We are considering continuous variables, so the chance that two or more have the exact same value has measure zero.

A different way to interpret the result for the first order statistic in Equation B.5 is to see it through a bayesian approach. Let us consider that the density $p_{(1)}$ is the posterior distribution of a given sample $X_i \sim \{X_1, \ldots, X_M\}$ conditioned on the fact that this particular sample is the smallest element of the set, i.e. the first order statistic. Then we write

$$p(X_i = x | i = 1^{st}) = \frac{p(i = 1^{st} | X_i = x) p(X_i = x)}{p(i = 1^{st})} \,. \tag{B.9}$$

A priori, without any information, we know that X_i was sampled from the same distribution as any other sample in the set, with the density $p(x)$. This defines the prior, $p(X_i = x) = p(x)$.

The evidence, $p(i = 1^{st})$, refers to the probability that the sample we picked, X_i, for an unkown i, is the smallest in the set of M samples. As we don't have any further information about it, this can only be a uniform distribution, $p(i = 1^{st})$, leaving us with

$$p(X_i = x | i = 1^{st}) = M p(i = 1^{st} | X_i = x) p(x) \,. \tag{B.10}$$

We can notice this expression is remarkably similar to the density in Equation B.5, with the left-hand side describing the same probability density. Equating them, we obtain an expression for the likelihood,

$$p(i = 1^{st} | X_i = x) = [1 - F(x)]^{M-1} \,. \tag{B.11}$$

This is used in Equation 2.48 to obtain an expression for the density of nearest neighbour quantities, $p(z_{\text{NN}}, ||\mathbf{x}^{\perp}_{\text{NN}}||)$.

Appendix C

About Lambert W functions

Lambert W function $W_k(z)$ is a multivalued function that satisfies the equation

$$we^w = z\,. \tag{C.1}$$

Here we explain the basic behaviour of this function and approximations that can be made to its value in specific regimes, so that one can understand the results presented in the main part of the thesis. For a more thorough presentation of the function and other applications for its use please refer to e.g. Corless et al. (1996).

This function has 2 branches in the real domain: W_0, the principal branch, that has codomain from $w = -1$ to positive infinity; and W_{-1}, with codomain from $w = -1$ to minus infinity. We show the graphical representation of those two branches in Figure C.1.

We start by differentiating the expression defining the Lambert W function,

$$\frac{\mathrm{d}}{\mathrm{d}x}\left(We^W\right) = \frac{\mathrm{d}x}{\mathrm{d}x} \tag{C.2a}$$

$$W'e^W + We^W W' = W'(e^W + W) = x \tag{C.2b}$$

$$\Rightarrow\; W' = \frac{e^{-W}}{1+W} = \frac{1}{e^W + x}\,. \tag{C.2c}$$

From the above and from the fact that $W(e) = 1$ we conclude that the derivative is positive for $x > e$. Trivially also we know that $\log W(x > e) > 0$.

Now taking the logarithm on both sides of the definition, and using the result

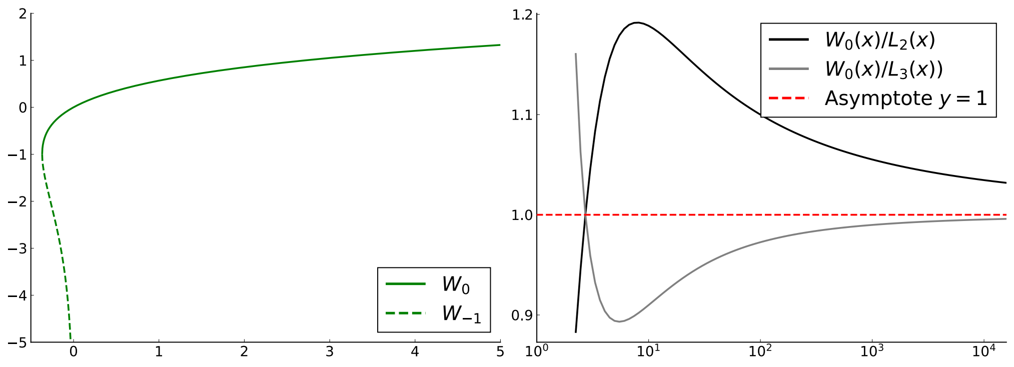

Figure C.1: Left: Real branches of the Lambert W function. The split between W_0 and W_{-1} happens at $(-1/\mathrm{e}, -1)$. We only work with W_0 in this work since we are concerned with real-valued x **Right:** The asymptotics of the Lambert W function as the argument goes to infinity, $x \to \infty$ gets sandwiched between $L_2(x) = \log x - \log\log x$ and $L_3(x) = \log x - \log\log x - \log\left(1 - \frac{\log\log x}{\log x}\right)$.

above, we can bound the logarithm of the Lambert W function for arguments above e,

$$\begin{aligned} &\log w + w = \log x \\ &\Rightarrow \; w = \log x - \log w < \log x, \quad \text{for } x > e\,. \end{aligned} \tag{C.3}$$

We can improve this bound even further by successively taking logarithms and using the results from before, such that one obtains, for large arguments, that

$$\log x - \log\log x < W_0(x) < \log x - \log\log x - \log\left(1 - \frac{\log\log x}{\log x}\right). \tag{C.4}$$

Appendix D

Asymptotics of the functions $\beta_k(M)$

As explained in the main text (Section 2.3), we want to calculate the asymptotic behaviour of the averages $\beta_k(M) = \left\langle s^k \zeta_{\text{NN}}(s, M)\right\rangle_{s\sim\mathcal{N}(0,1)}$ as the number of memories goes to infinity. We want to show that

$$\beta_k(M) = M\int_{-\infty}^{\infty} \frac{\mathrm{d}s}{\sqrt{2\pi}} e^{-\frac{1}{2}s^2} s^k \left[1-\Phi(s)\right]^{M-1} \tag{D.1a}$$

$$\overset{M\to\infty}{\sim} \frac{(-1)^k \mathrm{e}^{\frac{1}{2}\log\log M + o(1)} \left[W\left(\frac{M^2}{2\pi}\right)\right]^{\frac{k}{2}}}{\left[1+W\left(\frac{M^2}{2\pi}\right)\right]^{\frac{1}{2}}}, \tag{D.1b}$$

where $\Phi(s) = \int_{-\infty}^{s} \frac{\mathrm{d}1}{\sqrt{2\pi}} e^{-\frac{1}{2}q^2}$ is the cumulative distribution function for a standard Gaussian.

We start with the definition of the integral we want to estimate,

$$\beta_k(M) = \frac{M}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \mathrm{d}s\, s^k \exp\left(f(s;M)\right) \tag{D.2a}$$

$$f(s;M) = -\frac{1}{2}s^2 + (M-1)\log\left[1-\Phi(s)\right] \approx -\frac{1}{2}s^2 - M\Phi(s) \tag{D.2b}$$

and assume there is a global maximum of the integrand, such that the contributions that matter for the integral are only those close to that maximum. By Taylor expanding the exponent in the integrand up to second order, we obtain a Gaussian distribution times another exponential term. The exponential term goes out of the integral, and the integral can be done with the moments of the gaussian distribution.

This is called Laplace's method [1]. Differentiating f to obtain the extrema,

$$\left.\frac{\partial f}{\partial s}\right|_{s^*} = -s^* - M\frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}(s^*)^2} = 0 \tag{D.3a}$$

$$\Rightarrow \quad s^* = -\frac{M}{\sqrt{2\pi}}e^{-\frac{1}{2}(s^*)^2} \tag{D.3b}$$

$$\Rightarrow \quad (s^*)^2 e^{(s^*)^2} = \frac{M^2}{2\pi} \tag{D.3c}$$

$$\Rightarrow \quad (s^*)^2 = W\left(\frac{M^2}{2\pi}\right) \tag{D.3d}$$

$$\Rightarrow \quad s^* = -\sqrt{W_*}\,, \tag{D.3e}$$

where we selected the negative sign for s^* due to the expression in Equation D.3b, and renamed $W_* = W\left(\frac{M^2}{2\pi}\right)$ to ease the reading. For a quick review on Lambert W funtions, please refer to Appendix C.

Differentiating f once more to check the curvature of the function around the extremum,

$$\begin{aligned} \frac{\partial^2 f}{\partial s^2}(s^*) &= -1 + \frac{M}{\sqrt{2\pi}}\, s^*\, e^{-\frac{1}{2}(s^*)^2} \\ &= -1 - (s^*)^2 = -\left[1 + W_*\right] < 0\,, \end{aligned} \tag{D.4}$$

which means our solution is the maximum of the function for large M.

We can then expand f around its maximum up to second order, and approximate the integral as a Gaussian integral,

$$\begin{aligned} \beta_k(M) &\approx \frac{M}{\sqrt{2\pi}}\int_{-\infty}^{\infty} \mathrm{d}s\, s^k \exp\left[f(s^*) - \frac{1}{2}|f''(s^*)|(s-s^*)^2\right] \\ &= e^{\log M + f(s^*)}\int_{-\infty}^{\infty}\frac{\mathrm{d}s}{\sqrt{2\pi}}\, s^k \exp\left\{-\frac{1}{2}\left[1+W_*\right](s-s^*)^2\right\}\,. \end{aligned} \tag{D.5}$$

[1] See e.g. `https://en.wikipedia.org/wiki/Laplace%27s_method`.

We do a change of variables $\xi = s\sqrt{1+W_*} + s^*$ to perform the integral,

$$\begin{aligned}
\int_{-\infty}^{\infty} \frac{\mathrm{d}s}{\sqrt{2\pi}}\, s^k \exp\left\{-\frac{1}{2}\left[1+W_*\right](s-s^*)^2\right\} \\
&= \int_{-\infty}^{\infty} \frac{\mathrm{d}\xi}{\sqrt{2\pi}\sqrt{1+W_*}} \left(\frac{\xi}{\sqrt{1+W_*}} + s^*\right)^k e^{-\frac{1}{2}\xi^2} \\
&= \frac{1}{(1+W_*)^{\frac{k+1}{2}}} \left\langle \left(\xi + s^*\sqrt{1+W_*}\right)^k \right\rangle_\xi \\
&\approx \frac{(-1)^k W_*^{k/2}}{(1+W_*)^{1/2}},
\end{aligned} \tag{D.6}$$

where, for the approximation in the end, we used that ξ was order 1 and the dominating factor would be the constant aspect of the integral, $s^*\sqrt{1+W_*}$.

Returning to the other part of the original expression, we can simplify it as follows:

$$\begin{aligned}
\log M + f(s^*) &= \log M - \frac{1}{2}(s^*)^2 - M\Phi(s^*) \\
&\approx \log M - \frac{1}{2}W_* - M\frac{1}{\sqrt{2\pi}}\frac{1}{s^*}e^{-\frac{1}{2}(s^*)^2} \\
&\overset{D.3b}{=} \log M - \frac{1}{2}W_* + 1 \\
&\approx \log M - \frac{1}{2}\left[\log\left(\frac{M^2}{2\pi}\right) - \log\log\left(\frac{M^2}{2\pi}\right)\right] + o(1) \\
&\approx \frac{1}{2}\log\log M + o(1)\,.
\end{aligned} \tag{D.7}$$

Putting everything together, we obtain the result as declared before,

$$\beta_k(M) \overset{M\to\infty}{\sim} \frac{(-1)^k \mathrm{e}^{\frac{1}{2}\log\log M + o(1)} \left[W\left(\frac{M^2}{2\pi}\right)\right]^{\frac{k}{2}}}{\left[1+W\left(\frac{M^2}{2\pi}\right)\right]^{\frac{1}{2}}}\,. \tag{D.8}$$

Appendix E

Approximating the integrals for expectations over the nearest neighbour quantities

We want to derive the results from Equation 2.49. Let us start with the prior distribution $p(z, ||\mathbf{x}^{\perp}||^2) = p(z)p(||\mathbf{x}^{\perp}||^2)$. We know that, before conditioning on the nearest neighbour,

$$z = \mathbf{x}_m \cdot \hat{\mathbf{x}} \sim \mathcal{N}(0,1) \tag{E.1}$$

The prior for $||\mathbf{x}^{\perp}||^2$ is a chi-squared distribution, which for large N we can assume is well approximated by a Gaussian. Using the definition $||\mathbf{x}^{\perp}||^2 = ||\mathbf{x}^m||^2 - z^2$ we can find the mean and variance, such that

$$||\mathbf{x}^{\perp}||^2 \sim \mathcal{N}(N-1,\ 2N-2)\,. \tag{E.2}$$

We further assume that the (prior) cumulative distribution (cdf) for γ, $C(\gamma)$, is the cdf of a gaussian,

$$C(\gamma) = \Phi\left(\frac{\gamma - \mu_\gamma}{\sigma_\gamma}\right) \tag{E.3}$$

where $\mu_\gamma, \sigma_\gamma$ are the mean and standard deviation, given below.

Again, for large N this will be a reasonable approximation as the sum of gaussians is a gaussian and only non-gaussian, non-constant term in the definition for γ is the z^2 term, which is negligible compared to the others:

$$\gamma = \|\mathbf{x}_m - \mathbf{x}\|^2 = (z - \|\mathbf{x}\|)^2 + \|\mathbf{x}^{\perp}\|^2 = \|\mathbf{x}\|^2 + z^2 - 2\|\mathbf{x}\|z + \|\mathbf{x}_{\perp}\|^2 . \tag{E.4}$$

Taking the relevant expectations in the equation above, we find the mean and variance for γ,

$$\mu_\gamma = \|\mathbf{x}\|^2 + N \tag{E.5a}$$

$$\sigma_\gamma^2 = 4\|\mathbf{x}\|^2 + 2N \tag{E.5b}$$

The relevant equation is 2.48, repeated below for convenience,

$$\begin{aligned} \left\langle z_{\text{NN}}^p \|\mathbf{x}_{\text{NN}}^{\perp}\|^{2q} \right\rangle &\equiv \int \mathrm{d}z \mathrm{d}\|\mathbf{x}^{\perp}\|^2 \, z^p \|\mathbf{x}^{\perp}\|^{2q} \, p(z, \|\mathbf{x}^{\perp}\|^2|_{\text{NN}}, \mathbf{x}) \\ &= \int \mathrm{d}z \mathrm{d}\|\mathbf{x}^{\perp}\|^2 \, z^p \|\mathbf{x}^{\perp}\|^{2q} \, p(z, \|\mathbf{x}^{\perp}\|^2) \\ &\quad \times \int \mathrm{d}\gamma \, \delta\left(\gamma - (z - \|\mathbf{x}\|)^2 - \|\mathbf{x}^{\perp}\|^2\right) M \left[1 - C(\gamma)\right]^{M-1} . \end{aligned} \tag{E.6}$$

Doing the γ integral with the Delta function and substituting the results from earlier, we obtain

$$\left\langle z_{\text{NN}}^p |x_{\text{NN}}^{\perp}|^{2q} \right\rangle = \int \mathrm{D}z \mathrm{D}\|\mathbf{x}^{\perp}\|^2 \, z^p \|\mathbf{x}^{\perp}\|^{2q} M \left[1 - \Phi\left(\frac{z^2 - \mathbb{E}z^2 - 2\|\mathbf{x}\|z + \|\mathbf{x}^{\perp}\|^2 - \mathbb{E}\|\mathbf{x}^{\perp}\|^2}{\sqrt{4\|\mathbf{x}\|^2\sigma_z^2 + \sigma_{\perp}^2 + 2}} \right) \right]^{M-1} , \tag{E.7}$$

where the notation

$$\int \mathrm{D}q = \int \mathrm{d}q \; \phi(q) = \int \frac{\mathrm{d}q}{\sqrt{2\pi}} e^{-\frac{1}{2}q^2} \tag{E.8}$$

denotes an average over the standard normal distribution.

We can ignore the terms $z^2 - \mathbb{E}z^2$ in the numerator, and 2 in the denominator, as negligible compared to the others they are summing, and perform a change of variables to obtain

$$\left\langle z_{\text{NN}}^p |x_{\text{NN}}^{\perp}|^{2q} \right\rangle = \int \mathrm{D}\xi_z \mathrm{D}\xi_{\perp} \, (\sigma_z \xi_z)^p \, (N + \sigma_{\perp}\xi_{\perp})^q M \left[1 - \Phi\left(\rho_{\perp}\xi_{\perp} - \rho_z \xi_z\right)\right]^{M-1} , \tag{E.9}$$

where $\rho_\perp \propto \sigma_\perp$ and $\rho_z \propto 2|\mathbf{x}|\sigma_z$ with $\rho_\perp^2 + \rho_z^2 = 1$. We can also make a self-averaging assumption, $\|\mathbf{x}\|^2 = \sum_{i=1}^N x_i^2 \approx N$, to obtain $\rho_\perp^2 \approx 1/3$ and $\rho_z^2 \approx 2/3$.

Another change of variables to rotate the space, together with some simple algebra yields the result

$$\left\langle z_{\text{NN}}^p |x_{\text{NN}}^\perp|^{2q} \right\rangle = \langle\langle [(\rho_\perp t - \rho_z s)\sigma_z]^p \, [(\rho_z t + \rho_\perp s)\sigma_\perp + \mu_\perp]^q \rangle_t \, \zeta_{\text{NN}}(s,M)\rangle_s \,, \tag{E.10}$$

where $t, s \sim \mathcal{N}(0,1)$ and $\zeta_{\text{NN}}(s,M) = M[1-\Phi(s)]^{M-1}$.

Appendix F

Foraging and the Marginal Value Theorem

When should an agent foraging in a particular area of the world determine that the current area is depleted of resources, and that they should have easier access to food elsewhere?

This question was initially formulated by Charnov (1976) as an optimisation problem: an animal in the environment randomly encounters different varieties of food patches ($i = 1, \ldots, N$) at fixed probabilities $\{p_i\}$, and they decide how much time to spend in each food patch type in order to maximise the reward rate intake averaged over the whole environment. Mathematically,

$$E_{rate}(\mathbf{T}) = \frac{\sum_i p_i R_i(T_i) - E_0}{\sum_i p_i T_i + T_0}, \tag{F.1}$$

where T_0 is the travel time incurred by the animal when moving between food patches, and E_0 the cost in energy for travelling and foraging.

By optimising for the actions the animal can take, i.e. the times in each task, T_i, we can find that the optimal strategy is to leave a food patch at time T_i^* such that the instantaneous energy intake falls below the average across all food patches,

$$\frac{\partial E_{rate}}{\partial T_i} = \frac{p_i}{\sum_i p_i T_i + T_0}\left(\frac{\partial R_i}{\partial T_i} - E_{rate}(\mathbf{T})\right) \tag{F.2a}$$

$$\left.\frac{\partial E_{rate}}{\partial T_i}\right|_{\mathbf{T}^*} = 0 \;\Rightarrow\; \left.\frac{\partial R_i}{\partial T_i}\right|_{T_i^*} = E_{rate}(\mathbf{T}^*)\,. \tag{F.2b}$$

Now, to check whether the solution is a maximum we can differentiate once more to obtain the entries of the Hessian matrix; performing a small amount of algebra, we find that

$$\left.\frac{\partial^2 E_{rate}}{\partial T_i^2}\right|_{\mathbf{T}^*} = \frac{p_i}{\sum_i p_i T_i + T_0}\left.\frac{\partial^2 R_i}{\partial T_i^2}\right|_{T_i^*}, \tag{F.3}$$

$$\left.\frac{\partial^2 E_{rate}}{\partial T_i \partial T_j}\right|_{\mathbf{T}^*} = 0\,. \tag{F.4}$$

Therefore, if the R_is are concave functions of the T_is their second derivative will be negative, and the Hessian matrix will be negative-definite, meaning the extremum $\mathbf{T}^*$ we found earlier is a maximum. This assumption is usually true in foraging models, as it is commonly modelled that the amount of reward that can be obtained in a food patch is a monotonically non-decreasing (often saturating) function of time, as seen in Figure F.1.

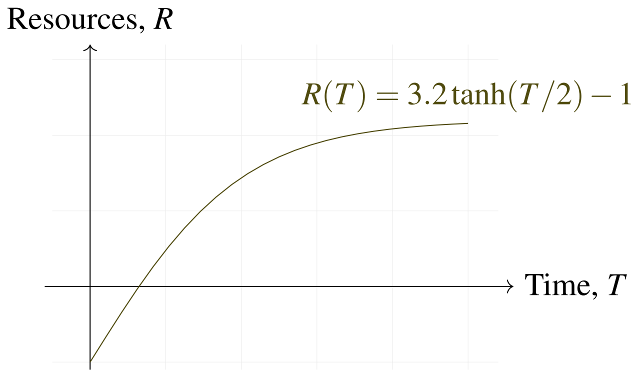

Figure F.1: Example of a monotonically increasing function that can be used in the modelling proposed by Charnov (1976). It has the characteristic of being concave, as necessary by the theory. Furthermore it saturates at $R(T \to \infty) = 2.2$, so the effect of diminishing returns is very marked.

Appendix G

Introduction to Reinforcement Learning

A common, general, learning problem is that of Reinforcement Learning (RL). In RL, an agent moves in the world trying to choose actions that maximise a quantity called the return, which is the sum of all rewards from now into the future[1]. Mathematically,

$$G_t = R_t + R_{t+1} + \cdots + R_T = \sum_{i=t}^{T} R_t, \tag{G.1}$$

where $\{R_i\}$ are the rewards obtained at each timestep, and T is the terminal time. Since there is a clear end to the task the agent is optimising, this delimitation is called an Episode, and this setting is called the *Episodic setting*[2].

When the task does not have a clear terminal time, but continues (possibly) indefinitely, the quantity G_t as described above is not well defined, and the most common solution to this problem is to introduce a discount factor, $\gamma \in [0,1]$, to the expression of the return so that the contributions of rewards obtained in the far future are vanishingly small:

$$G_t = R_t + \gamma G_{t+1} + \gamma^2 G_{t+1} + \cdots = \sum_{i=0}^{\infty} \gamma^i R_{t+i}. \tag{G.2}$$

This is called the *Discounted setting*.

[1]Alternatively, one can think of *minimising* an opposite quantity, usually considered as a sum of costs. This is a more common convention in the adjacent field of Control Theory.

[2]Not to be confused with the Episodic memory of Part I

Perhaps the most important aspect of the return[3] in RL is the fact that it is, in either of the forms discussed above, a recursive quantity:

$$G_t = R_t + \gamma G_{t+1}\,, \tag{G.3}$$

meaning that one can know the return at time t by knowing the return at time $t+1$ and iterating back with the reward at time t. This recursive aspect is a cornerstone of the theory, and is used in several places in order to derive algorithms that learn to behave optimally in RL.

It would be difficult for any agent to make sense of the world and maximise their return if the world had no structure that could be learned and understood, so the next relevant step for defining any RL problem is to describe the state and action spaces. Taking actions is the agent's way of interacting with the environment, and the state represents the information that the agent has available (together with the reward received at each time step) in order to help them learn the effects of their actions in the world. See Figure G.1 below.

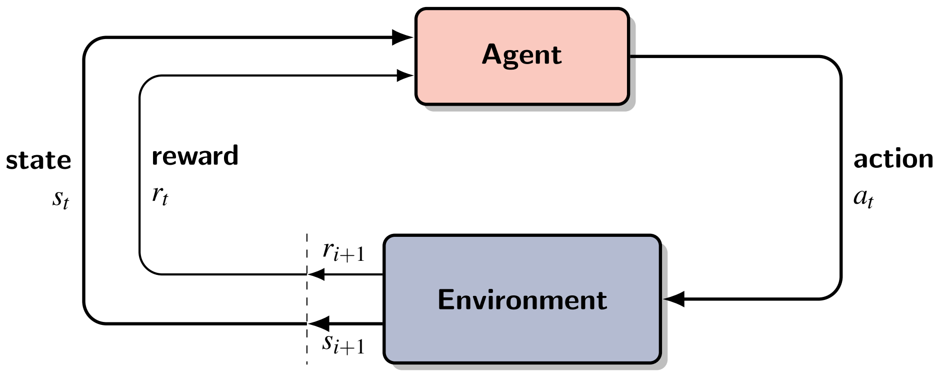

Figure G.1: Diagram of the basic formulation of an MDP. The agent selects actions that influence the state of the environment. The environment then "gives" the agent information about its state and a reward conditioned on the previous state and the action taken by the agent. Figure adapted from TeX StackExchange.

Those ideas can be formalised with the concept of a Markov Decision Process (MDP). An MDP is characterised by 4 components: the state space, $\mathcal{S}$; the action space, $\mathcal{A}$; the reward function, $\mathcal{R}$; and the transition function, $\mathcal{P}$. It is usually

[3]To simplify notation we call both quantities *return* and denote them by G. The meaning of which is being referred to can be inferred from context.

represented as the 4-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P})$, as given all of those a modeller can describe what states are relevant to describe the problem, which actions are available, and how the actions and states determine (perhaps stochastically) the rewards and the next states.

Another important concept is that of a *policy*, usually represented by the greek letter π. A policy is a description of which actions an agent would take given they are in a certain state. It can be deterministic (usually a greedy policy) or stochastic. The objective of RL is to optimise the policy π in order to maximise the return G, while following that policy.

In order to increase the return – since the agent usually does not have access to the future rewards before they obtain those –, the agent must use a proxy that considers only the information available at that point in time. The relevant quantity that represents this current state of knowledge of the agent is the *value function*,

$$V_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s] \,. \tag{G.4}$$

In other words, the value of state s under a policy π is the expected return given that on time t the agent is at the state $S_t = s$ and they will follow policy π to select their actions.

We can use the expression for the return in Equation G.3 to write

$$V_\pi(s) = \sum_a \pi(a|s) \sum_{r,s'} p(r,s'|s,a)[r + \gamma V_\pi(s')] \,. \tag{G.5}$$

In general it is not the value of a generic policy that is relevant, but that which is associated with the *optimal* policy,

$$\begin{aligned} V^*(s) &= \max_\pi \sum_a \pi(a|s) \sum_{r,s'} p(r,s'|s,a)[r + \gamma V^*(s')] \\ &= \max_a \sum_{r,s'} p(r,s'|s,a)[r + \gamma V^*(s')] \,. \end{aligned} \tag{G.6}$$

There are different algorithms that one can use to estimate the optimal value.

Most can be summarised as an iterative update of the value function at each state,

$$\hat{V}(S_t) \leftarrow \hat{V}(S_t) + \eta\delta\,, \tag{G.7}$$

where η is a learning rate and δ is the prediction error, that can take different forms depending how one wishes to estimate the value. The most common ways of estimating the value are: Monte-Carlo (MC) methods, such as REINFORCE; or Bootstrapping methods, such as Temporal Difference (TD).

An MC-like method could be given by $\delta_{MC} = G_t$, where the estimation of the value is given by directly estimating the average of the return but empirically averaging different realizations of the problem. A bootstrapping method on the other hand, would use the current estimate of the value to "bootstrap" the updated estimate. The Temporal Difference (TD) prediction error is then given by

$$\delta_{TD} = R_{t+1} + \gamma\hat{V}(S_{t+1}) - \hat{V}(S_t)\,. \tag{G.8}$$

In either of those methods, the policy is then derived from the value function. It can be a *greedy* policy, which chooses the action the maximises the expected value of future rewards,

$$\pi_{\text{greedy}}(s) = \arg\max_{a} \sum_{s',r} p(s',r|s,a)\left[r + \gamma V_{\pi_{\text{greedy}}}(s')\right]\,, \tag{G.9}$$

or it can be an stochastic policy, which samples the argmax action with higher probability, but allows other actions to be chosen with low probability as well, usually to encourage exploration.

Instead of learning the optimal value function, and from there deriving the optimal policy, an alternative approach is to optimise the policy directly. To do that, one needs to define a parameterisation for the policy $\pi = \pi_{\mathbf{w}}(a|s)$, and an objective to be optimised.

The obvious choice of an objective to be optimised is the value function. Since the purpose of policy methods was to circumvent the need for a value function, this might seem counterintuitive. However, the Policy Gradient theorem proves that, for a non-random initial state s_0, the gradient of the value function is proportional to an expectation of the gradient of the policy,

$$\nabla_{\mathbf{w}} V_{\pi_{\mathbf{w}}}(s_0) \propto \mathbb{E}\left[G_t \nabla_{\mathbf{w}} \log \pi_{\mathbf{w}}(A_t|S_t)\right] . \tag{G.10}$$

Therefore, by following in the direction of the right-hand side of Equation G.10, one is following in the direction that optimises the value function, without needing to keep track of the value. This approach is what leads to the definition of the REINFORCE algorithm, a Monte Carlo method,

$$\Delta \mathbf{w} = \eta G_t \nabla \log \pi(A_t|S_t, \mathbf{w}) . \tag{G.11}$$

There are numerous other methods that optimise the policy directly, usually called Policy Gradient methods due to the result in Equation G.10. Some of them, like the Actor-Critic method, use estimation of the value function on top of the policy update to bootstrap the estimation and improve performance over learning. An interested reader can find a thorough account of the diversity of RL methods in Sutton and Barto (2018); Sugiyama (2015).

Appendix H

Deriving average reward and time for the evidence accumulation task

Here we derive Equations 3.17, the average reward and average time for the constant boundary policy in the evidence accumulation task from Chapter 3. Recalling the task: there are 2 options that could be generating the data x_i, and the goal is to integrate the available evidence to determine which one is generating the data for a given trial. Mathematically, the generative model is given by Equations 3.10 and 3.11, repeated below for convenience,

$$\mu \sim p(\mu) = \frac{1}{2}\delta(\mu-\mu_0) + \frac{1}{2}\delta(\mu+\mu_0) \tag{H.1a}$$

$$\{x_i\}|\mu \sim p(x|\mu) = \mathcal{N}\left(\mu, \sigma^2\right), \tag{H.1b}$$

where i is the index for samples within a trial.

The policy chooses the option that generates data from the gaussian with mean $+\mu_0$ (or, $-\mu_0$) as soon as the evidence accumulator $\mathrm{e}_t = \sum_{i=1}^{t} x_i$ reaches the boundary $+B$ ($-B$). Given this information, we want to calculate the values of 4 different variables: the probabilities $p_+ = p(\mathrm{e}_t = B|\mu_0)$ and $p_- = p(\mathrm{e}_t = -B|-\mu_0)$, and the average stopping times $\overline{T}_+$ and $\overline{T}_-$, as functions of the boundary value B and the parameters of the gaussian.

The problem we want to solve is simple to enunciate: if the evidence accumulator variable e_t accumulates the samples x, and we know the distribution that generates these samples (conditioned on knowing the mean μ), then we can find the probability of hitting each bound, and the average time it takes to hit the bound.

The strategy to obtain those quantities consists in applying a result called the *Optional Stopping Theorem*[1] to our setup, and following a bit of algebra. Let us define the martingale

$$Y_t(s) = \frac{e^{s\,\mathrm{e}_t}}{[M(s)]^t}\,, \tag{H.2}$$

where e_t is the evidence acucmulator variable and $M(s)$ is the moment generating function of a gaussian variable, given by

$$M(s) = \int_{-\infty}^{\infty} \mathrm{d}x\, e^{sx} p(s|\mu,\sigma^2) = e^{s\mu+\frac{1}{2}s^2\sigma^2}\,. \tag{H.3}$$

It is important to condition on knowing the identity of the distribution generating the data, i.e. we know which of $\pm\mu_0$ is being used to generate the data for that trial. This is particularly important to make Y_t a martingale, which is needed for the optional stopping theorem. In fact, we can convince ourselves that Y_t is a martingale by noticing that

$$\mathbb{E}|Y_t| = \frac{\mathbb{E}|e^{s\sum_i x_i}|}{[M(s)]^t} \leq \frac{|\mathbb{E}e^{s\sum_i x_i}|}{[M(s)]^t} = \frac{|\mathbb{E}e^{sx}|^t}{[M(s)]^t} = 1 < \infty\,, \tag{H.4}$$

and proving that its value conditioned on past values $Y_1,\dots,Y_{t-1}$ depends only on the last one, as in

$$\mathbb{E}\left[Y_t|Y_1,\dots Y_{t-1}\right] = \mathbb{E}\left[\frac{e^{sx_t}}{M(s)}\frac{e^{s\,\mathrm{e}_{t-1}}}{[M(s)]^{t-1}}\middle| Y_1,\dots Y_{t-1}\right] = \mathbb{E}\left[\frac{e^{sx_t}}{M(s)}\right] Y_{t-1} = Y_{t-1}\,. \tag{H.5}$$

Then, the optional stopping theorem tells us that, for the stopping time T,

$$\mathbb{E}Y_T = \mathbb{E}Y_1 = \mathbb{E}\left[\frac{e^{sx_1}}{M(s)}\right] = 1\,. \tag{H.6}$$

[1]Explaining this result in depth is out of the scope for this thesis. Suffice to say it describes conditions under which the expected value of a martingale at a stopping time is equal to its initial expected value. We assume those conditions are valid for our problem (essentially, that we always reach either of the boundaries, $\pm B$) and use the result for a martingale defined below.

From the definition of the policy there are only 2 stopping possibilities, when the evidence e_T is either of $\pm B$, therefore we can write the left-hand side as

$$\mathbb{E}Y_T = \mathbb{E}\left[\frac{e^{s\mathrm{e}_T}}{[M(s)]^T}\right] = \frac{1}{[M(s)]^T}\left[p(\mathrm{e}_T = +B|\mu)e^{+sB} + p(\mathrm{e}_T = -B|\mu)e^{-sB}\right] = 1\,. \tag{H.7}$$

Now, the result above should be valid for any value of s, in particular we can choose $s^* = -2\mu/\sigma^2$ so that $M(s^*) = e^0 = 1$, and the equation simplifies to

$$p(\mathrm{e}_T = +B|\mu)e^{-\frac{2\mu B}{\sigma^2}} + p(\mathrm{e}_T = -B|\mu)e^{+\frac{2\mu B}{\sigma^2}} = 1\,. \tag{H.8}$$

Defining the probabilities of different outcomes

$$p_+ = p(\mathrm{e}_T = +B|\mu = +\mu_0) = 1 - p(\mathrm{e}_T = -B|\mu = +\mu_0) \tag{H.9a}$$

$$p_- = p(\mathrm{e}_T = -B|\mu = -\mu_0) = 1 - p(\mathrm{e}_T = +B|\mu = -\mu_0)\,, \tag{H.9b}$$

$$\tag{H.9c}$$

we can select the possible options for $\mu = \pm\mu_0$ and find that

$$p_+ = p_- = \frac{1-e^{\xi}}{e^{-xi} - e^{\xi}} = \frac{1}{e^{-\xi}+1}\,, \tag{H.10}$$

where we defined the auxiliary variable

$$\xi = \frac{2\mu_0 B}{\sigma^2}\,. \tag{H.11}$$

In order to obtain the average times $\overline{T}$ we can differentiate Equation H.7 with respect to s and select the value $s = 0$,

$$0 = \mathbb{E}\left[\frac{(-T)}{[M(s)]^{T+1}}M'(s)\left[p(+|k)e^{+sB} + p(-|k)e^{-sB}\right] + \frac{B}{[M(s)]^T}\left[p(+|k)e^{+sB} - p(-|k)e^{-sB}\right]\right]$$

$$= (-\overline{T})\mu + B\Big[p(+|k) - p(-|k)\Big] \tag{H.12a}$$

$$\Rightarrow\ \overline{T} = \frac{B}{\mu}\Big(p(+|k) - p(-|k)\Big) \tag{H.12b}$$

Once again we then select the possible options for k, now to obtain the average times

$$\overline{T}_+ = \frac{B}{\mu_0}(p_+ - (1-p_+)) = \frac{B}{\mu_0}\frac{1-e^{-\xi}}{1+e^{-\xi}}, \tag{H.13a}$$

$$\overline{T}_- = \frac{B}{\mu_0}((1-p_-) - p_-) = \frac{B}{\mu_0}\frac{e^{-\xi}-1}{e^{-\xi}+1}. \tag{H.13b}$$

Defining the reward for a correct (incorrect) choice to be R_+ ($R_- < 0$), the probability of choosing correctly the $+\mu_0$ option to be $p_+ = p(\mathrm{e}_t = B|\mu_0)$, and the probability of choosing correctly when the true option was $-\mu_0$ to be $p_- = p(\mathrm{e}_t = B|-\mu_0)$, we have the expression for the average reward,

$$\begin{aligned}\overline{R}(\pi) &= \sum_{\mu=\pm\mu_0} p(\mu)\left[R_+ p(\mathrm{correct}|\mu) + R_- p(\mathrm{wrong}|\mu)\right] \\ &= \Big[p(+\mu_0)p(\mathrm{e}_t = B|+\mu_0) + p(-\mu_0)p(\mathrm{e}_t = -B|-\mu_0)\Big]R_+ \\ &\quad + \Big[p(+\mu_0)p(\mathrm{e}_t = -B|+\mu_0) + p(-\mu_0)p(\mathrm{e}_t = B|-\mu_0)\Big]R_- \\ &= \frac{1}{2}R_+(p_+ + p_-) + \frac{1}{2}R_-(2 - p_+ - p_-),\end{aligned} \tag{H.14}$$

and for the average time,

$$\begin{aligned}\overline{T}(\pi) &= \sum_{\mu=\pm\mu_0} p(\mu)\overline{T}_\mu \\ &= p(+\mu_0)\overline{T}_+ + p(-\mu_0)\overline{T}_- = \frac{1}{2}\left[\overline{T}_+ + \overline{T}_-\right].\end{aligned} \tag{H.15}$$

We can finally collect the results obtained so far to state the final equations used in the main text,

$$\overline{R}(B) = \frac{R_+}{1+e^{-\frac{2\mu_0 B}{\sigma^2}}} + \frac{R_-}{1+e^{+\frac{2\mu_0 B}{\sigma^2}}} \tag{H.16a}$$

$$\overline{T}(B) = \frac{B}{\mu_0}\tanh\left(\frac{\mu_0 B}{\sigma^2}\right) \tag{H.16b}$$

Appendix I

Neural Network for REINFORCE

Artificial Feedforward Neural Networks were used to compute the policies for REINFORCE, in Sections 4.3 and 4.4. The networks were defined using the library PyTorch (Paszke et al. (2019)) version 1.12.1 and a linux environment with CUDA version 11.3, Python 3.10.11, GCC 11.3.0, packaged by conda-forge.

For each task in the environment, the agent had two feedforward neural networks, totalising $2N$ networks. Each of the networks had a hidden layer of width $W = 20$, and an output layer with variable dimension: the time policy network had output of dimension equal to 2, the parameters of the lognormal distribution; the task policy network had output of dimension equal to N, the number of possible tasks the agent could move to.

In order to properly learn the joint policy distribution $\pi(T, a|s, \theta, \mathbf{w}) = \pi(T|s, \theta, \mathbf{w})\pi(a|T, s, \theta, \mathbf{w})$, we gave information about the leaving time chosen as an input to the task policy network. Therefore the inputs were different for different networks: θ, with dimensionality N, for the time policy; and $(\theta, \langle T \rangle)$, with dimensionality $N+1$, for the task policy, where $\langle T \rangle$ is the average time according to the lognormal distribution given by the parameters outputted by the time policy network.

Appendix J

Description of the environments for Chapter 4

Chapter 4 presented simulation results using a number of different environments. Here we present with more detail the parameters for those environments described briefly in Table 4.1. For all tasks described below, the relevant equation for the reward function is given by Equation 4.2, repeated below ommitting the index *i* and the dependence on θ,

$$R(T) = R^M \tanh(k\,T) - R^0\,. \tag{J.1}$$

We also report the local optimal leaving time,

$$T^* = \arg\max_T R(T)/T \tag{J.2}$$

and the corresponding local optimal reward rate $R(T^*)/T^*$, and the depletion and replenishment time constants, β^{in} and β^{out}, for the θ dynamics in Equation 4.1.

	R^M	R^0	k	β^{in}	β^{out}	T^*	$R(T^*)/T^*$
Task 1	15.0	-5.0	1.0	300.0	300.0	0.99	6.42
Task 2	36.0	-1.0	0.016	300.0	300.0	22.4	0.51

Table J.1: Environment 1. Short duration task with higher optimal reward rate , and long duration task with lower optimal reward rate.

	R^M	R^0	k	β^{in}	β^{out}	T^*	$R(T^*)/T^*$
Task 1	15.0	-5.0	1.0	2000.0	2000.0	0.99	6.42
Task 2	67.5	-4.5	0.15	2000.0	2000.0	3.29	8.01

Table J.2: Environment 2.

	R^M	R^0	k	β^{in}	β^{out}	T^*	$R(T^*)/T^*$
Task 1	15.0	-5.0	1.0	2000.0	2000.0	0.99	6.42
Task 2	22.5	-1.5	0.15	2000.0	2000.0	3.29	2.67

Table J.3: Environment 3.

	R^M	R^0	k	β^{in}	β^{out}	T^*	$R(T^*)/T^*$
Task 1	15.0	-5.0	1.0	300.0	300.0	0.99	6.42
Task 2	15.0	-5.0	1.0	150.0	150.0	0.99	6.42
Task 3	15.0	-5.0	1.0	50.0	50.0	0.99	6.42

Table J.4: Environment 4.

	R^M	R^0	k	β^{in}	β^{out}	T^*	$R(T^*)/T^*$
Tasks 1-5	15.0	-5.0	1.0	300.0	300.0	0.99	6.42

Table J.5: Environment 5.

	R^M	R^0	k	β^{in}	β^{out}	T^*	$R(T^*)/T^*$
Task 1	15.0	-5.0	1.0	300.0	300.0	0.99	6.42
Tasks 2-11	5.0	-0.5	0.5	300.0	300.0	1.15	1.82

Table J.6: Environment 6.