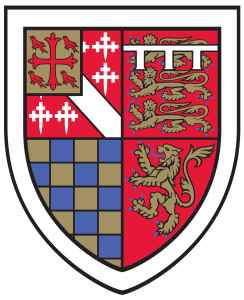
# Sequential decision making with feature-linear models



**David Janz**

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
*Doctor of Philosophy*

St Edmund's College

October 2021

# Errata

This online version of the thesis contains a correction to the proof of lemma 18, page 51, on the monotonicity of predictive variance in Gaussian process regression models.

# Declaration

This thesis is the result of my own work and includes nothing which is the outcome of work done in collaboration except as declared in the preface and specified in the text. It is not substantially the same as any work that has already been submitted before for any degree or other qualification except as declared in the preface and specified in the text. It contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

# Abstract

**Sequential decision making with feature-linear models, by David Janz**

This thesis is concerned with the problem of sequential decision making, where an agent interacts sequentially with an unknown environment and aims to maximise the sum of the rewards it receives. Our focus is on methods that model the reward as linear in some feature space. We consider a bandit problem, where the rewards are linear in a reproducing kernel Hilbert space, and a reinforcement learning setting with features given by a neural network. The thesis is split into two parts accordingly.

In part I, we introduce a new algorithm for the optimisation of continuous functions with a known number of derivatives under noisy bandit feedback. The algorithm, Partitioned GP-UCB, combines ideas from classic kernel-based bandit algorithms and hierarchical optimisation methods to provide near-tight confidence intervals and strong guarantees on computational complexity. As part of our analysis, we develop a novel bound on the effective dimension of kernel ridge regression with the Matérn kernel with respect to the volume of the domain. Part I is mainly concerned with the derivation of this bound.

In part II we tackle practical exploration in deep reinforcement learning, with a focus on methods that combine linear uncertainty estimates with feature embeddings given by deep Q-function networks. We observe a flaw within previous such work: while these methods enforce a temporal difference relationship on the mean state-action values given by the linear model, they do not constrain its inputs—the neural network embeddings—and these determine the uncertainty estimates of linear models. We show that such embeddings need to follow a certain successor feature structure and develop a model based on this insight: Successor Uncertainties. We demonstrate that our model is capable of solving hard tabular exploration challenges and that it scales to the Atari Learning Environment, where it attains strong scores relative to competing methods.

# Contents

# Introduction

In this thesis we develop and analyse algorithms for sequential decision making. We focus on two settings: global optimisation, where we optimise an unknown function with stochastic feedback, and reinforcement learning, where an agent interacts with a stateful environment (effectively a single-player game) and aims to maximise some cumulative reward signal. In both contexts, our interest is fuelled by recent developments in the application of machine learning, but we will approach these topics from a more theoretical perspective, one based on the framework of bandit algorithms.

The bandit framework, introduced formally by T. L. Lai et al. (1985) but considered as early as Thompson (1933), considers an agent (learner) interacting with an environment (bandit) sequentially over a number of time-steps. Each interaction consists of the agent selecting an action (arm) from a given set, and the environment returning a stochastic reward sampled from an unknown action-conditioned reward distribution. The agent's goal is to select actions that maximise the sum of the rewards it receives over some time horizon. At the core of the bandit framework is the exploration-exploitation problem: the agent can either exploit actions it believes yield high rewards, or sample rewards from arms it is less certain about in hope of finding arms with higher mean rewards.

Bandit algorithms and the exploration-exploitation trade-off are perhaps easiest to understand through the lens of their potential real-life applications. These include:

- Medicine, where one of a number of possible treatments is sequentially assigned to each patients in the treatment group based on the outcomes of previous patients (Thompson, 1933; Durand et al., 2018) or where the dose of a drug is adjusted on a per-patient basis depending on their earlier response (Bastani et al., 2020).
- Online advertisement and recommender systems, where adverts or recommendations are displayed in a manner that balances a) future clicks that may result from a well-developed (explored) profile of the user with b) clicks that may be attained

now by exploiting current estimates of the user's interests (Bouneffouf et al., 2013; Bouneffouf, 2014; Q. Zhou et al., 2017).

- Packet routing, where an algorithm sequentially decides on the route that each packet will take from point $A$ to point $B$, with the aim of minimising the overall transfer time (Talebi et al., 2017). Here, exploration may lead to finding a faster route, but an algorithm that explores too much may be slower than one that exploits a known good route.

While our applications of the framework will depart from the classic bandit problems listed here, they are nonetheless problems underpinned by a need to balance exploration with exploitation. We now look at these in turn.

The first problem we study, that of global optimisation, is motivated by the practice of Bayesian optimisation. Here, an unknown function is optimised under first order noisy feedback, by first placing a non-parametric prior over that function, then sequentially updating it under a suitable likelihood and using the resulting posterior for decision making. Bayesian optimisation is known to be very sample-efficient in practice, and has been used for:

- Optimising hyperparameters for other machine learning algorithms (Hutter et al., 2011; Turner et al., 2021), including deep neural networks (Snoek et al., 2012).
- De novo drug and material design (Gómez-Bombarelli et al., 2018), where an algorithm selects new molecules to synthesise and test for a specific purpose.
- Design of large scale scientific experiments (Kirschner et al., 2019).

In contrast with the previous examples, these are problems of pure exploration where no cost is assigned to exploration, and the agent's goal is to identify a good arm within some limited budget of evaluations. However, under very mild conditions, algorithms that solve the bandit problem (max cumulative reward) also solve the global optimisation (max reward) problem (Bubeck et al., 2011).

While Bayesian optimisation with various point-acquisition (action-selection) rules has been previously studied under the bandit framework (Srinivas, Krause, S. Kakade et al., 2010), existing strong guarantees tend to hold only for average performance—under the assumption that the function being optimised is a sample from some assumed prior, and that we wish to do well on average across such samples. Under the criterion of worst-case performance, most existing algorithms are only provably near-optimal when optimising

smooth (infinitely differentiable) functions. Our focus is on developing a method with worst-case guarantees for functions with very weak differentiability properties. Our contribution, the Partitioned GP-UCB algorithm, is effective in practice and has strong theoretical guarantees for both performance and runtime; it is the first method to satisfy these three criteria. The method is based on a combination of insights from a feature-linear modelling perspective (with features based on kernel functions) with ideas from methods for the optimisation of Lipschitz continuous functions (Jones et al., 1993; Bubeck et al., 2011), and certain equivalences between these two approaches.

Reinforcement learning, the second area of our study, is a sequential decision making problem where the chosen actions affect not only the immediate rewards but also those attained in the future. Reinforcement learning has received much attention in applied literature since neural networks, and in particular convolutional networks, have allowed classic reinforcement learning agents to tackle increasingly complex problems. Recent successful applications of reinforcement learning include:

- Games, famously the Atari 2600 games Mnih et al. (2015) and M. G. Bellemare et al. (2013), but since then also modern games such as StarCraft II (Vinyals et al., 2019) as well as classic board games including Go, Chess and Shogi (Silver, Huang et al., 2016; Silver, Hubert et al., 2017; Schrittwieser et al., 2020).
- The protein folding problem, tackled by Jumper et al. (2021), as well as de novo drug and material design (Olivecrona et al., 2017; Guimaraes et al., 2017; Z. Zhou et al., 2019; Simm et al., 2020).
- Industrial control problems, like optimising data centre cooling systems (Evans et al., 2016) and reducing the power usage of water treatment plants (Lilwall, 2021).
- Improving human-computer interaction, including brain-computer interfaces controlling, for example, bionic limbs (Pohlmeyer et al., 2014; Ubelacker, 2019).

While the use of deep neural networks greatly extended the applicability of reinforcement learning algorithms, the advances in function approximation and model flexibility came at the expense of the methods used to explore the environment: the successes of reinforcement learning have mainly come in areas with access to large amount of cheap data, and use simple, naïve exploration methods. Our approach will be to use a bandit-based perspective to improve exploration within such *deep* reinforcement learning algorithms.

We can use the bandit framework to analyse reinforcement learning algorithms by

augmenting the environment to return a state alongside the reward, with that state affecting future states and rewards. Within our work, we use this perspective together with a linear model of the rewards based on neural network embeddings on state-action pairs—this combination allows us to leverage techniques from the well-developed literature on sequential decision making under linear function approximation, while retaining applicability to modern, deep reinforcement learning. Our main contribution is the Successor Uncertainties family of methods. These provide strong performance on hard exploration problems that scales to complex, high dimensional problems. Alongside out algorithmic contribution, we also provide an in-depth analysis and critique of similar methods.

Moving forward, after we briefly outline some notation, the thesis splits into two parts: first covering global/Bayesian optimisation, second reinforcement learning. Each part begins with an overview of the technical contributions therein and their structure. Part I, being more theoretical in nature, sets out many basic results around linear models and sequential decision making. Readers interested in part II, reinforcement learning but unfamiliar with the underlying theory ought to consult chapters 1, 2 and 3 of part I and the references therein.

# Overview of notation

**General** We use $\doteq$ to emphasise an equality that follows directly from definition. $\{a_j\}$ is used as shorthand for $\{a_j \colon j \in J\}$ when the index set $J$ is unambiguous. For $n \in \mathbb{N}$, $[n]$ denotes the set $\{0, \dots, n-1\}$.

**Vectors, matrices and functions** For a finite set $A$, we identify each map $f : A \mapsto \mathbb{R}$ with a vector $f \in \mathbb{R}^{|A|}$ and write either $f(a)$ or $[f]_a$ for the value $f$ takes on $a \in A$. $e_a$ denotes the unit vector in $\mathbb{R}^{|A|}$ with $[e_a]_a = 1$. $\|\cdot\|$ and $\langle \cdot, \cdot \rangle$ denote the $\ell^2/L^2$ norms and inner products. $\langle \cdot, \cdot \rangle_F$ denotes Frobenius inner product for matrices.

**Probability and integration** For random variables $X, Y$, $\mathbf{E}[X|Y = y]$ and $\mathbf{P}(X|Y = y)$ are numbers while $\mathbf{E}[X|Y]$ and $\mathbf{P}(X|Y)$ random variables. $\mathbb{1}\{\mathcal{E}\}$ denotes the indicator of an event $\mathcal{E}$. For a finite set $A$, $\pi$ a measure on $A$ and each $a \in A$, we use $\pi(a)$ to denote $\pi(\{a\})$. For $f \in L^1(\mathbb{R}^d)$, $\int f(x)dx$ denotes its integral with respect to the Lebesgue measure on $\mathbb{R}^d$. We do not discuss measurability and $\sigma$-algebras; the latter can be taken to be finite/Borel as appropriate. Equality of random variables is to be understood as almost sure if not otherwise specified.

**Asymptotic notation** We adopt Hardy's notation. We will use $R_T \leq Cf(T)$ to denote that there exists a constant $C > 0$ independent of $T$ such that $R_T \leq Cf(T)$ for all $T$. We will write $C_d$ to emphasise that this constant depends on some quantity $d$ and $C'_d, C''_d$ when we need more constants. Whenever such otherwise undefined constants appear multiple times, they are not to be interpreted as being related. We will use $\iota$ to denote any terms that have an at most polylogarithmic dependence on the relevant asymptotic quantities. We will use $\varepsilon > 0$ to represent an arbitrary small number and take $\varepsilon + \varepsilon = \varepsilon$.

# Part I

# Kernel-based bandit optimisation

# Overview of part I

Within part I of this thesis, we consider the problem of optimising a continuous function $f\colon [0,1]^d \mapsto \mathbb{R}$ under bandit feedback. Specifically, for $T \in \mathbb{N}$, at each step $t = 1, \ldots, T$ we select an $X_t \in [0,1]^d$ and observe

$$Y_t = f(X_t) + \varepsilon_t,$$

where $\varepsilon_t$ is a $\mathfrak{B}$-subGaussian independent random variable for $\mathfrak{B} > 0$ and $f$ belongs to a ball of a known radius within a Sobolev Hilbert space of order $\nu + d/2$, for a smoothness parameter $\nu > 0$. The latter assumption implies that $f$ has $\lfloor \nu \rfloor$th order mixed partial derivatives.

This optimisation problem emerges from a frequentist analysis of a classic Bayesian bandit optimisation method, the Gaussian Process upper confidence bound algorithm (GP-UCB, Srinivas, Krause, S. M. Kakade et al., 2009), when the Matérn kernel is used to define the covariance function (M. L. Stein, 1999; Matérn, 1960). Most previous results for this and similar Gaussian process optimisation algorithms are restricted to kernels with an exponential or fast polynomial decay spectral density, like the Squared Exponential kernel, and are thus only applicable to the optimisation of smooth or near-smooth functions. While we examine the problem from a bandit perspective, our results translate immediately to the corresponding global optimisation problem that motivated our work (Bubeck et al., 2011).

The problem is of high practical relevance. Gaussian process optimisation is applied in a number of areas including A/B testing, recommender systems, robotics, sensor networks and preference learning (Shahriari et al., 2015). The Matérn family of kernels in particular is the default choice for the purpose of the black-box optimisation of machine learning algorithms (Snoek et al., 2012). New theoretical insights may lead to faster, more robust algorithms in this area that perform well across larger sets of problems.

## Contributions

We develop the Partitioned GP-UCB algorithm, an extension and generalisation of the GP-UCB algorithm (Srinivas, Krause, S. M. Kakade et al., 2009; Srinivas, Krause, S. Kakade et al., 2010) that attains low regret for a large set of smoothness parameters $\nu$ within the outlined setting. In particular, we prove that for all $\nu > 1$ and all $d \geq 1$, the regret $R_T$ incurred by Partitioned GP-UCB over $T$ interactions is bounded as

$$R_T \leq C_\varepsilon T^{\frac{d(2d+3)+2\nu}{d(2d+4)+4\nu}+\varepsilon} \quad (\forall \varepsilon > 0).$$

For values of $\nu$ and $d$ for which the eigenfunctions of a certain associated integral equation are uniformly bounded, we also have the stronger result

$$R_T \leq C_\varepsilon T^{\frac{d+\nu}{d+2\nu}+\varepsilon} \quad (\forall \varepsilon > 0).$$

This matches the corresponding algorithm agnostic lower bound up to the $\varepsilon$ in the exponent (Scarlett, Bogunovic et al., 2017a). We show that the uniform boundedness condition holds for the case $d = 1, \nu = 1/2$. Verifying whether it holds for general $d, \nu$ remains an open problem.

The results we present improve significantly on those available for the original GP-UCB algorithm, which are of the form

$$R_T \leq Z_{T,d,v} \quad \text{with} \quad Z_{T,d,v} \geq C T^{\frac{3d+2\nu}{2d+4\nu}}.$$

Notably, the results for GP-UCB do not guarantee sublinear regret in the regime $d \geq 2\nu$; those for Partitioned GP-UCB do. When the additional uniform boundedness condition holds, Partitioned GP-UCB is also near-optimal as a global optimisation algorithm. Moreover, where the set of arms is constrained to a finite subset of $[0, 1]^d$, the computational complexity of Partitioned GP-UCB is bounded as $C_\varepsilon T^{1+\varepsilon}$ for all $\varepsilon > 0$. This result is near-optimal, and again improves significantly on the corresponding results for GP-UCB and for its sketching-based variants (Calandriello, Carratino et al., 2019; Calandriello, Carratino et al., 2020).

Partitioned GP-UCB works by partitioning the domain into a cover consisting of axis-aligned hypercubes and constructing GP-UCB–style upper confidence bounds independently on each cover element. Importantly, each upper bound is based only on the observations contained within the corresponding hypercube. The resulting piecewise

upper confidence on the domain is tighter than that given by GP-UCB. At the core of the method is a novel result that gives a quantitative version of the intuition that *there is less to learn on a smaller domain*. Specifically, we bound how information gain for Gaussian process regression with a Matérn kernel scales with the support of that regression problem. This result may find applications outside of our algorithm. The cover construction we use is similar to that of the Hierarchical Optimistic Optimisation algorithm (HOO, Bubeck et al., 2011), but our algorithm applies to a larger range of smoothness parameters $\nu$ than HOO. The subsetting of data we use resembles the trick within the related SupKernelUCB algorithm (Valko, Carpentier et al., 2013). Unlike SupKernelUCB, however, our algorithm performs well in practice.

## Structure and attribution

Part I is structured as follows:

1. Chapter 1 introduces the bandit formalism and standard upper-confidence-bound-based algorithms and outlines the OFUL algorithm (Abbasi-Yadkori et al., 2011) and its kernel-based extension, equivalent to the GP-UCB algorithm.

2. Chapter 2 defines Sobolev spaces, kernels and Gaussian process regression, and outlines related results. The key insights in this chapter are the development of the relation between the Matérn kernel and particular Sobolev spaces, of the equivalence between kernel-based linear regression and Gaussian process regression and of the relation between effective dimension and information gain.

3. Chapter 3 introduces in detail the prior works most relevant to our algorithm: the GP-UCB, SupKernelUCB and HOO algorithms. We discuss the associated theoretical results and compare these with the relevant algorithm agnostic lower bounds.

4. Chapter 4 derives and discusses new results on the information gain associated with conjugate Gaussian process regression using a Matérn family kernel. We use these to provide an improvement on the regret bound for the GP-UCB algorithm under a specific choice of hyperparameters.

5. Chapter 5 describes the Partitioned GP-UCB algorithm. provides an analysis of its regret and computational complexity and presents an experimental validation of the method.

The work presented in this part is based on David Janz, David R. Burt & Javier González (2020) and further collaboration with David R. Burt. We thank Michal Valko and Daniele Calandriello for helpful discussions, and José Miguel Hernández-Lobato for feedback on the writing.

# Chapter 1

# An introduction to bandits

In this chapter, we develop the basic notions and formalisms around bandits and build up to an intuitive understanding of kernelised linear bandit algorithms.

## 1.1 Stochastic bandit formalism

Bandit problems correspond to the repeated interaction between an agent and a environment, taking place over time-steps $t = 1, \dots, T$ with $T \in \mathbb{N}$ known as the horizon. We define a bandit instance $\xi$ by the set of available arms $\mathcal{X}$ and a set of distributions $\{P_x \colon x \in \mathcal{X}\}$. On selecting action $X_t \in \mathcal{X}$, the agent receives a random reward $Y_t | X_t \sim P_{X_t}$ with mean $f(X_t) = \mathbf{E}[Y_t | X_t]$, which we call the payoff (for $X_t$). The agent is defined by a sequence $\pi = \{\pi_t\}$ with each $\pi_t$ mapping from a history of outcomes $\mathcal{H}_t = (X_1, Y_1, \dots, X_t, Y_t)$ to a measure on $X_{t+1}$. We call an algorithmic construction for $\pi$ a bandit algorithm.

The interaction between an agent $\pi$ and a bandit problem instance $\xi$ induces a probability measure over the history $\mathcal{H}_T$. The aim of the agent is then to maximise a statistic, typically the expectation, of the sum of the rewards $Y_1, \dots, Y_T$. For given bandit instance $\xi$, we write this equivalently as minimising a statistic of the regret, given by

$$R(T, \xi, \pi) = \sum_{t=1}^{T} f^\star - f(X_t),$$

where $f^\star \doteq \sup_{x \in \mathcal{X}} \mathbf{E}[Z_x]$ for $Z_x \sim P_x$ is the optimal payoff. We call any $x^\star \in \mathcal{X}$ with $f(x^\star) = f^\star$ an optimal arm.

The challenge facing the agent is that it does not know a priori the instance $\xi$ it faces, only that it belongs to some set $\Xi$, and may also not know $T$. We look for agents that perform well across all instances in $\Xi$ and all horizons $T$. A standard way of formalising this is upper bounding the worst-case expected regret,

$$\hat{R}(T, \pi) = \sup_{\xi \in \Xi} \mathbf{E} R(T, \pi, \xi).$$

An algorithm $\pi$ that satisfies $\hat{R}(T, \pi) = \inf_{\pi'} \hat{R}(T, \pi')$ for all $T \in \mathbb{N}$ is said to be minimax-optimal on $\Xi$. This is a very strong requirement. More pragmatically, we look to find functions $g$ and $f$ such that

$$g(T) \leq \inf_{\pi'} \hat{R}(T, \pi') \leq \hat{R}(T, \pi) \leq f(T).$$

We refer to $g(T)$ as an algorithm-agnostic lower bound on the worst case expected regret and $f(T)$ an upper bound. If for a given $\Xi$ and algorithm $\pi$, $g$ and $f$ differ by less than $C_\varepsilon T^\varepsilon$ for all $\varepsilon > 0$, we say that $\pi$ is *near-minimax-optimal* (on $\Xi$). If $\hat{R}(T, \pi)/T \to 0$ as $T \to \infty$, we will say the algorithm has *sublinear regret*.

### 1.1.1 Constant probability results, adaptivity

The results within part I of this thesis will be stated in terms of $R(T, \pi, \xi)$, which we will denote by $R_T$, and will be of the form:

> For any $\delta \in (0, 1)$, with probability $1 - \delta$, $R_T \leq f(T, \delta)$.

This is standard phrasing within the kernel-based optimisation literature most relevant to our work, but it is a shorthand. A formal version of the statement reads:

> For all $\delta \in (0, 1)$, there exists an event $E$ such that the probability of $E$ is no less than $1 - \delta$ and that on the event $E$, for all $T \in \mathbb{N}$ and all $\xi \in \Xi$, $R(T, \pi, \xi) \leq f(T, \delta)$.

The egregious imprecision of the original statement becomes more excusable once we note that the formal statement is itself the strictest that could be reasonably inferred.

For settings where the maximum expected per-step regret can be bounded uniformly across $\Xi$ by some constant (such as analysed in this thesis) constant-probability results translate trivially to results in expectation. Say the bounding constant is $A$, then we

bound the regret on the complement of $E$ by $AT$, giving an overall bound of the form

$$\mathbf{E}R_T \leq \min_{\delta \in (0,1)} \left( (1-\delta) f(T,\delta) + \delta AT \right),$$

where we may allow $\delta$ to depend on $T$. The other direction, from bounds in expectation to those in probability, can be obtained in a generic manner using Markov's inequality.

Additionally, results may show a dependence of $f$ on the instance $\xi$. Such results are preferable, as they show not only how the algorithm performs on the worst-case problems but also how it adapts when presented with less challenging instances. We say algorithms and analysis that depend explicitly on the instance $\xi$, and not just on $\Xi$, are *adaptive*. Adaptivity will be an important point within our work.

### 1.1.2 Simple and Bayesian regret, Bayesian optimisation

Simple and Bayesian regret are two measures of performance related to regret that come up in settings that are variations of that considered in this work.

First, we might assume that along with $\Xi$ we are given a measure $P_\xi$ over $\Xi$ and that $\xi$ is sampled from $P_\xi$ at the start of all interaction. Then we might consider the problem of minimising instance regret on average across $\Xi$. We call this quantity the Bayesian regret and denote it

$$BR_T \doteq \mathbf{E}\mathbf{E}_\xi R(T, \pi, \xi).$$

This Bayesian formulation of the bandit problem is in a sense easier than the classical formulation, in that worst-case expected regret immediately bounds average regret. The challenge in minimising Bayesian regret is then in optimally using the additional information contained in the prior $P_\xi$.

Second, we may look at global optimisation/pure exploration, where we do not care about intermediate rewards $\{Y_t\}$ but instead wish to quickly identify an optimal arm. This is formalised as looking for an estimate $\hat{X}_T^\star$ such that $SR_T = f^\star - f(\hat{X}_T^\star)$, called the simple regret, is low. Bandit algorithms can be used to tackle the pure exploration problem. In particular, if we let $\hat{X}_T^\star$ be chosen uniformly at random from arms $X_1, \ldots, X_T$ selected by a bandit algorithm with sublinear regret, then

$$\mathbf{E}R_T/T = f^\star - \mathbf{E}\frac{1}{T}\sum_{t=1}^{T} f(X_t) = f^\star - \mathbf{E}f(\hat{X}_T^\star) = SR_T \to 0 \quad \text{as} \quad T \to \infty.$$

The converse does not hold in general.

Combining a prior over the bandit class with a simple regret objective gives Bayesian optimisation, a problem highly related to the ideas discussed in this thesis and one where algorithms such as GP-UCB see most of their practical use (Snoek et al., 2012).

## 1.2 Algorithm design

With the formalities behind us, we now look at the two main algorithm design strategies that underpin our contributions:

- Optimism in the Face of Uncertainty (OFU). Optimism is a key tool for bandit optimisation that reduces a bandit problem to that of constructing high probability estimates for the corresponding payoff function.
- Linear approximation. We will use linear combinations of elements of appropriately chosen feature spaces to approximate the continuous payoff functions we optimise.

We explain these through a series of examples. First, we show how optimism works by applying it to a two-armed bandit. Then, we present OFUL, the now-standard optimistic algorithm for linear bandits, and sketch its extension to feature-linear models. We end by showing how this feature-linear model may be evaluated using kernel functions.

### 1.2.1 Upper confidence bound algorithms

We present upper confidence bound (UCB) algorithms through the lens of the methodology developed in the seminal work by Auer (2002) and Auer, Cesa-Bianchi et al. (2002). The core of these algorithms is the construction of an upper confidence bound for the payoff function based on the data observed thus far. The algorithm then selects an arm that maximises this bound, and the resulting reward acts to bring this bound closer to the empirical average for that arm. In turn, the empirical average converges suitably to the true payoff of that arm, and for suboptimal arms, eventually drops below that associated with an optimal arm. This leads to an optimal arm being selected increasingly often. How quickly such an algorithm converges onto optimal arms depends on the tightness of the upper confidence bounds it uses. Much of research on bandit algorithms, and indeed our development in part I, is focused on developing tighter upper bounds for specific problem classes.

### 1.2.2 A simple UCB algorithm

We illustrate the UCB method on a two-armed bandit with independent Gaussian rewards, based on Auer, Cesa-Bianchi et al. (2002). We focus on breaking the analysis down into modular parts, which we will encounter repeatedly throughout the thesis.

The bandit problem considered consists of two arms, $\mathcal{X} = \{x_1, x_2\}$, with corresponding payoffs $f_1, f_2$. Without loss of generality take $x_1$ to be optimal with $f_1 = f_2 + \Delta$ for some $\Delta > 0$ referred to as the suboptimality gap. For a given horizon $T \in \mathbb{N}$ and each time step $t = 1, \ldots, T$, the agent selects an arm $X_t = x_i$ and observes $Y_t = f_i + \varepsilon_t$ for $\varepsilon_t$ an independent standard normal random variable.

We begin the design of the algorithm by deriving an upper confidence bound for the payoff of each arm for a single time step $t$. Let $\delta > 0$ be a confidence parameter. Then for arm $x_i$ and step $t$, we look for a UCB of the form

$$u_{i,t} = \mu_{i,t} + \beta_{i,t}(\delta),$$

where $\mu_{i,t}$ is the empirical mean of rewards sampled from arm $x_i$ by step $t$ (an unbiased estimator of $f_i$) and $\beta_{i,t}(\delta)$ is the corresponding confidence width, which we will choose such that $\mathbf{P}\{f_i - \mu_{i,t} < \beta_{i,t}(\delta)\} \geq 1 - \delta$. The Cramér-Chernoff method gives us that the average of $n$ standard normal random variables satisfies

$$\mathbf{P}\left(\frac{1}{n}\sum_{i=1}^{n} \varepsilon_i \geq z\right) \leq e^{-nz^2/2} \quad (\forall z > 0),$$

which leads to the choice $\beta_{i,t}(\delta) = \sqrt{2\log(1/\delta)/n_{i,t}}$, where $n_{i,t}$ is the number of times $x_i$ was selected prior to the start of round $t$. Next, we use a union bound to extend this inequality to hold for all time-steps and arms simultaneously. For any sequence $\{b_t > 0\}$,

$$\mathbf{P}\left(\bigcup_{t=1}^{T} \{f_i < \mu_{i,t} + \beta_{i,t}(b_t)\}\right) \geq 1 - \sum_{t=1}^{T} b_{i,t}.$$

Taking $b_t = \delta/(4T)$ for all $t$, we have that $\sum_{i\in\{0,1\}} \sum_{t=0}^{T} b_{i,t} = \delta/2$. The final $1/2$ factor allows us to ask for a symmetric lower bound of the form $\mu_{i,t} - \beta_{i,t}(\delta)$ to hold.

One important direction of research in bandit literature is the development and application of alternative constructions for time-uniform bounds. See Boucheron et al. (2013), chapter 1, for an overview of recent developments in this area.

**Algorithm:** UCB
**Parameters:** $T \in \mathbb{N}$, $\delta \in (0, 1)$
**Initialisation:** $\mu_{i,t} = \infty$, $n_{i,t} = 0$ for $i = 1, 2$

For each $t = 1, \ldots, T$:

1. Compute

$$n_{i,t} = \sum_{\tau<t} \mathbb{1}\{X_\tau = i\} \quad \text{and} \quad \mu_{i,t} = \frac{1}{n_{i,t}} \sum_{\tau<t} Y_\tau \mathbb{1}\{X_\tau = i\}$$

   for $i = 1, 2$.
2. Select $X_t = x_i$ for

$$i \in \arg\max_{b\in\{1,2\}} \mu_{b,t} + \sqrt{2\log(1/\delta)/n_{b,t}},$$

   breaking ties randomly.

Figure 1.1: UCB Algorithm, two-armed bandit.

With the upper confidence in place, the algorithm is ready (fig. 1.1). It remains to analyse its performance. Recall that $f_1 - f_2 = \Delta$. We can write the regret as

$$R_T = \Delta n_{2,T}.$$

To bound the incurred regret, we bound $n_{2,T}$.

Suppose that we are on the *good* event where with probability $1-\delta$ the derived confidence intervals hold; that is

$$f_1 < \mu_{1,t} + \beta_{1,t}(\delta) \quad \text{and} \quad -f_2 < -\mu_{2,t} + \beta_{2,t}(\delta).$$

If $x_2$ was selected at a time step $t$, we know that

$$\mu_{2,t} + \beta_{2,t}(\delta) = u_{2,t} \geq u_{1,t} = \mu_{1,t} + \beta_{1,t}(\delta)$$

and that the per-step regret incurred was $\Delta$. Combining the two inequalities,

$$\Delta = f_1 - f_2 < \mu_{1,t} + \beta_{1,t}(\delta) - \mu_{2,t} + \beta_{2,t}(\delta) \leq 2\beta_{2,t}(\delta).$$

This bound of per-step regret by twice the confidence widths will feature repeatedly in

the analysis of the algorithms we consider.

The final step is specific to finite-armed bandits. If $\Delta > 2\beta_{2,\tau}(\delta)$ for some $\tau \in \mathbb{N}$ then $X_\tau \neq x_2$, and if this holds for some $\tau$ then, by the monotonicity of $\beta_{2,t}$ with respect to $t$, it holds for all $t$ satisfying $\tau \leq t \leq T$. Thus $n_{2,T} = n_{2,\tau} \leq \tau$. Let $\tau(\Delta, \delta)$ be the lowest such $\tau$ for a given choice of $\Delta$ and $\delta$ (we can solve for $\tau(\Delta, \delta)$ from the definition of $\beta_{2,\tau}(\delta)$). Then

$$R_T \leq \Delta \left(\tau(\Delta, \delta) \vee T\right).$$

This is an instance-dependent constant-probability regret bound. We can obtain a worst-case bound on expected regret using the previously described method of bounding the regret on the complement of the good event by $\Delta T$. A worst-case is then attained by maximising this bound over $\Delta$ and minimising over $\delta$.

Working through the argument fully (and with slightly more careful analysis), Lattimore et al. (2020) show that taking $\delta = 1/T$,

$$\hat{R}(T, \pi) \leq 8\sqrt{2T \log T} + 3\Delta$$

for the thus defined algorithm $\pi$. The additive $\Delta$ term is unavoidable. Any algorithm must pull the suboptimal arm at least once. Furthermore, lower bounds show that for $k$ arms, a $\sqrt{kT}$ term is also unavoidable (Lattimore et al., 2020, chapter 15). It is clear therefore that for $k \geq T$, we will need additional structure to obtain sublinear regret.

### 1.2.3 Linear UCB algorithms

We now turn to bandits with infinitely many arms, but where the payoff depends linearly on the arm location. Specifically, we take $\mathcal{X} = [0, 1]^d$ and assume that there exists a $\theta^\star \in \mathbb{R}^d$ such that

$$f(x) = \langle x, \theta^\star \rangle$$

for all $x \in \mathcal{X}$ and that $\|\theta^\star\|_2 \leq \mathfrak{R}$ for some known constant $\mathfrak{R} > 0$. We observe $Y_t = f(X_t) + \varepsilon_t$ with $\varepsilon_t$ independent for each $t$ and subGaussian.

The bound on the norm of $\theta^\star$ can be interpreted as an assumption on the smoothness of the payoff function, since

$$|f(x_\star) - f(x)| = |\langle \theta^\star, x_\star - x \rangle| \leq \|\theta^\star\|_2 \|x_\star - x\|_2 \leq \mathfrak{R} \|x_\star - x\|_2 \quad (x \in \mathcal{X}).$$

Crucially, this smoothness allows us to use only a finite number of observations to construct upper confidence bounds for $f$ that hold jointly across all of $[0,1]^d$.

We proceed to bound $f$ by constructing sets $\{B_t\}$ such that with constant probability, $\theta^\star$ is contained within all $B_t$ for $t \geq 1$ and that the sets $B_t$ are small. We do so using linear ridge regression estimates for $\theta^\star$. For each $t \geq 1$, we find a $\hat{\theta}_t \in \mathbb{R}^d$ that minimises the ridge regression objective

$$\sum_{\tau=1}^{t} \left(Y_\tau - \langle \hat{\theta}_t, X_\tau \rangle\right)^2 + \lambda \|\hat{\theta}_t\|_2^2,$$

for some regularisation parameter $\lambda > 0$.[1] Stacking $X_1, \ldots, X_t$ into a design matrix $X_{1:t} \in \mathbb{R}^{t \times d}$ and $Y_1, \ldots, Y_t$ into a vector $Y_{1:t}$, the solution to linear ridge regression is given by

$$\hat{\theta}_t = \Sigma_t X_{1:t}^T Y_{1:t} \quad \text{where} \quad \Sigma_t^{-1} = X_{1:t}^T X_{1:t} + \lambda I.$$

We can bound the error in our estimation of $f$ at location $x$ and time step $t$ using Cauchy-Schwarz:

$$|f_t(x_\star) - f(x)| = \langle \hat{\theta}_t - \theta^\star, x \rangle \leq \|x\|_{\Sigma_t^{-1}} \|\hat{\theta}_t - \theta^\star\|_{\Sigma_t}.$$

The term $\|x\|_{\Sigma_t^{-1}}$ is known the predictive standard deviation of the linear model at $x$, and can be bounded using standard methods. For the other term, Abbasi-Yadkori et al. (2011) show through a method-of-mixtures argument (Pena et al., 2004) that with probability $1 - \delta$, for all $t \geq 1$,

$$\|\hat{\theta}_t - \theta^\star\|_{\Sigma_t} \leq \sqrt{\log \frac{\det \Sigma_t}{\det \lambda I} + 2 \log(1/\delta)} + \lambda^{1/2} \|\theta^\star\|_2. \tag{1.1}$$

The resulting confidence ellipsoid serves as the set $B_t$. The quantity $\log(\det \Sigma_t / \det \lambda I)$ featuring here is the log reduction in the volume of a confidence ellipsoid for $\theta^\star$ after observing $\mathcal{H}_t$, and is known as the information gain for $\theta^\star$. It generalises $n_{i,t}$ for this linear setting: when all selected arms are axis-aligned and at unit distance, $\log \det \Sigma_t = \sum_{i=1}^d n_{i,t}$, for $n_{i,t}$ a per-dimension arm-pull count. Generalisations of information gain will be a core focus of our work.

Using thus constructed confidence intervals within the standard UCB algorithmic frame-

[1] The regularisation term $\lambda \|\hat{\theta}_t\|_2^2$ corresponds to constraining $\hat{\theta}_t$ to lie on a sphere of finite radius; this corresponds to our prior knowledge that $\|\theta^\star\|_2 \leq \mathfrak{R}$ for some $\mathfrak{R} > 0$.

work results in regret bounded as

$$R_T \leq d\sqrt{T}\log T + \sqrt{dT\log(T/\delta)},$$

with the resulting algorithm known as 'optimism in the face of uncertainty, linear' Abbasi-Yadkori et al. (OFUL, 2011). The corresponding lower bound for this subGaussian linear bandit problem on a convex hypercube is of the form $Cd\sqrt{T}$. OFUL is thus near-minimax optimal for this setting (Lattimore et al., 2020, page 250).

### 1.2.4 OFUL with feature embeddings

To generalise OFUL to continuous functions, we now extend the contextual linear bandit framework to the case where the mean reward function is linear in some given feature embedding of the arm. That is, $\mathcal{X} = [0,1]^d$ and

$$Y_t = \langle \theta^\star, \varphi(X_t)\rangle + \varepsilon_t$$

for a feature embedding $\varphi\colon \mathbb{R}^d \mapsto \mathbb{R}^{d'}$ and weights $\theta^\star \in \mathbb{R}^{d'}$ with $\|\theta^\star\|_2 \leq \mathfrak{R}$. This is a very expressive representation. By Stone-Weierstrass, for an appropriate choice of $\varphi$ (for example, a sufficiently high order polynomial) this feature-linear formulation can be used to uniformly approximate any continuous function on a compact domain.

For a given choice of $\varphi$ and writing $\Phi_{1:t} \in \mathbb{R}^{t\times d'}$ for the design matrix formed by the embeddings of the selected arms, $\varphi(X_1), \ldots, \varphi(X_t)$, the standard ridge regression linear model yields an estimate for $\hat{\theta}_t$ of the form

$$\hat{\theta}_t = \Sigma_t \Phi_{1:t}^T Y_{1:t} \quad \text{where} \quad \Sigma_t = \left(\Phi_{1:t}^T \Phi_{1:t} + \lambda I\right)^{-1},$$

which can in turn be used within the OFUL algorithm. Such extension is however of limited utility. To be able to accurately express a broad set of continuous functions in this feature-linear form, we require $d'$ to be large. On the other hand, running OFUL requires the computation of $\Sigma_t \in \mathbb{R}^{d'\times d'}$, an inversion with a cost that scales as ${d'}^3$. This gives an unpleasant trade-off between statistical accuracy and computational efficiency.

To get around this difficulty, we employ the *kernel trick*: we rewrite the solution to the regularised linear regression such that it depends on the embeddings $\varphi$ only through an inner product $\langle \varphi(x), \varphi(x')\rangle \doteq$ and identify a closed form expression for this inner product that does not require evaluating $\varphi$. We refer to $k\colon \mathcal{X}\times\mathcal{X} \mapsto \mathbb{R}$ as the kernel function

associated with $\varphi$.

We now give an example of a kernel function. Consider the second order polynomial feature embedding $\varphi\colon \mathbb{R}^d \mapsto \mathbb{R}^{d'}$, with $d' = d(d+1)/2$, given by

$$\varphi\colon x \mapsto \left(x_1^2, \binom{d}{2} x_1 x_2, \ldots, \binom{d}{2} x_d x_{d-1}, x_d^2\right)^T.$$

Computing $\langle \varphi(x), \varphi(x') \rangle$ by explicitly evaluating $\varphi(x)$ and $\varphi(x')$ and taking the $\mathbb{R}^{d'}$ inner product requires order $d'$ operations. However,

$$\langle \varphi(x), \varphi(x') \rangle = \sum_{i,j=1}^{d} A_{ij}^2 x_i x_j x_i' x_j' = (x^T x')^2 \quad \text{where} \quad A_{ij} = \begin{cases} 1 & i = j, \\ \binom{d}{2} & \text{otherwise,} \end{cases}$$

and so taking $k(x, x') = (x^T x')^2$, we can evaluate this inner product in order $d$ operations instead. This generalises to order $p$ polynomial features, with $k(x, x') = (x^T x')^p$ giving a complexity linear in $d$ and independent of $d'$ and $p$.

To use kernels for regression, we rewrite the solution of feature-based linear ridge regression so that it depends on $X_{1:t}$ only through the inner products of the corresponding embeddings. Recall that for matrices $U, V$ of appropriate sizes, $(I + UV)^{-1}U = U(I + VU)^{-1}$, a result known as the push-through identity. Thus,[2]

$$\langle \varphi(x), \hat{\theta}_t \rangle = \varphi(x)(\Phi_{1:t}^T \Phi_{1:t} + \lambda I)^{-1} \Phi_{1:t}^T Y_{1:t} = \varphi(x) \Phi_{1:t}^T (\Phi_{1:t} \Phi_{1:t}^T + \lambda I)^{-1} Y_{1:t}.$$

With a suitable kernel $k$, we then have

$$[\varphi(x)\Phi_{1:t}^T]_i = k(x, X_i) \quad \text{and} \quad [\Phi_{1:t}\Phi_{1:t}^T]_{ij} = k(X_i, X_j),$$

for an overall computational complexity of order $C_d t^3$, independent of $d'$. The resulting method is referred to as *kernel ridge regression*.

With an adjustment in the construction of the parameter confidence sets, kernel ridge regression can be integrated within the OFUL framework, yielding the GP-UCB algorithm for the optimisation of continuous functions (Chowdhury et al., 2017). We return to this, after the next chapter, where we develop a more formal understanding of kernels and kernel-based regression.

[2]This formulation of ridge regression is referred to as the function-space view, since we are estimating the function $f(x) = \langle \varphi(x), \theta^\star \rangle$ directly. In contrast, the previous approach where we focus on the weights $\theta^\star$ is the weight-space view.

# Chapter 2

# Kernels and regression

In this chapter, we provide the theoretical background necessary for the analysis of kernel-based optimisation algorithms for functions with a known number of derivatives. In particular:

- We define Sobolev spaces and show how these correspond to specifying desired smoothness conditions. We identify Sobolev Hilbert spaces as suitable for kernel-based regression and show how these can be defined on convex sets using the Fourier transform and restrictions.
- We define kernels and their corresponding reproducing kernel Hilbert spaces (RKHS). We show multiple characterisation of RKHSs, and in particular one using the Fourier transform and restrictions. We use this to establish a connection between Sobolev Hilbert spaces and the RKHSs of kernels belonging to the Matérn family.
- We provide a more rigorous introduction to kernel ridge regression, with a focus on quantities that appear within the kernel-based version of OFUL (Abbasi-Yadkori, 2009), and show its relation to conjugate Gaussian process regression.
- We introduce the effective dimension of a kernel ridge regressor and the information gain associated with conjugate Gaussian process regression and discuss the manner in which these can be considered measures of the complexity of the regression problem. We demonstrate their near-equivalence.

We recap key results from this chapter at the start of the next.

## 2.1 Function spaces

The optimisation problem we tackle assumes a payoff function $f\colon [0,1]^d \mapsto \mathbb{R}$ with a known number of derivatives. Here, we build up some definitions and results around Sobolev Hilbert spaces, which turn out to be the *right* spaces in which to reason about $f$.

### 2.1.1 Continuity classes

For a multi-index $\alpha \in \mathbb{N}^d$, we refer to $|\alpha| = \sum_{i=1}^d \alpha_i$ as its degree and denote the standard partial differentiation operator by

$$D^\alpha = \left( \frac{\partial^{\alpha_1}}{\partial x^{\alpha_1}}, \ldots, \frac{\partial^{\alpha_d}}{\partial x^{\alpha_d}} \right)^T.$$

We denote the set of all continuous functions on $U \subset \mathbb{R}^d$ open by $C(U)$ and we write $C^s(U)$ for the $s$-continuity class on $U$, the set of functions $u\colon U \mapsto \mathbb{R}$ such that $D^\alpha u \in C(U)$ for all $|\alpha| \leq s$. That is, functions with $s$-many continuous derivatives on $U$.

We then define $C(\bar{U})$ as the set of functions $u \in C(U)$ that can be extended continuously to $\bar{U}$, the closure of $U$. Equipped with the norm

$$\|u\|_{C(\bar{U})} = \sup_{x \in U} |u(x)|,$$

$C(\bar{U})$ forms a Banach space. Similarly, we define $C^s(\bar{U})$ as the set of functions $u \in C^s(U)$ such that $D^\alpha u$ continuously extends to $\bar{U}$ for all $|\alpha| \leq s$. Equipped with the norm

$$\|u\|_{C^s(\bar{U})} = \max_{|\alpha| \leq s} \sup_{x \in U} |D^\alpha u(x)|,$$

these too form Banach spaces.

We also consider the following more fine-grained partition of $C(\bar{U})$ for continuous functions without (strong) derivatives. We say that a function $u \in C(\bar{U})$ is $\gamma$-Hölder continuous if

$$|u(x) - u(y)| \leq L|x - y|^\gamma$$

for some $L > 0$ and $0 < \gamma \leq 1$. When $\gamma = 1$, we say $u$ is Lipschtiz continuous. We formalise the class of all $\gamma$-Hölder continuous functions by defining the $\gamma$-Hölder seminorm

of $u\colon U \mapsto \mathbb{R}$,

$$[u]_{C^{0,\gamma}(\bar{U})} = \sup_{x,y\in U, x\neq y} \left( \frac{|u(x) - u(y)|}{|x-y|^\gamma} \right),$$

and constructing the $\gamma$-Hölder norm as

$$\|u\|_{C^{0,\gamma}(\bar{U})} = \|u\|_{C(\bar{U})} + [u]_{C^{0,\gamma}(\bar{U})}.$$

$\gamma$-Hölder continuous functions on $\bar{U}$ are then functions with finite $C^{0,\gamma}(\bar{U})$ norm.

We use the notion of Hölder continuity to construct function classes $C^{s,\gamma}(\bar{U})$ between $C^s(\bar{U})$ and $C^{s+1}(\bar{U})$ for any $s \in \mathbb{N}$ and $0 < \gamma \leq 1$. These consist of all $u \in C^s(\bar{U})$ such that

$$\|u\|_{C^{s,\gamma}} = \sum_{|\alpha|\leq s} \|\partial^\alpha u\|_{C(\bar{U})} + \sum_{|\alpha|=s} [\partial^\alpha u]_{C^{0,\gamma}(\bar{U})}$$

is finite.

### 2.1.2 Sobolev spaces

The inner product form of the kernel regressor will require that we work in a Hilbert space. We now define Sobolev spaces, which suitably generalise continuity classes, and then look at a subset of these that form Hilbert spaces.

We begin with Lebesgue spaces. For $U \subset \mathbb{R}^d$ and $p$, $1 \leq p < \infty$, the Lebesgue space $L^p(U)$ is the set of functions $p$-integrable with respect to the Lebesgue measure. That is, measurable functions $f\colon U \mapsto \mathbb{R}$ for which the seminorm

$$\|f\|_{L^p(U)} = \left( \int_U |f(x)|^p dx \right)^{1/p}$$

is finite. $L^p(U)$ can be extended to $p = \infty$ using the notion of essential boundedness. A function $f$ is essentially bounded if for some $M < \infty$, $|f| \leq M$ almost everywhere. The space $L^\infty(U)$ is then the space of all essentially bounded functions, with the lowest essential bound for a given element serving as the seminorm. For a measure $\mu$, we will write $L^p_\mu(U)$ to denote the space of functions $p$-integrable with respect to $\mu$.

Lebesgue spaces can be used to construct corresponding Banach spaces, with norm given by the Lebesgue seminorm and considering the elements as equivalence classes of functions equal Lebesgue-almost-everywhere. We will do this throughout and ignore the distinction between functions and equivalence classes. Where it exists, we will always

work with the unique continuous representative of each class. For $p = 2$, the resulting Banach space equipped with the inner product

$$\langle f, g \rangle_{L^2(U)} = \int g(x) f(x) dx$$

forms a Hilbert space.

With that, we are ready to define Sobolev spaces.

**1 Definition.** *For $s \in \mathbb{N}$, $1 \leq p \leq \infty$ and $U \subset \mathbb{R}^d$ open, the Sobolev space $L^{s,p}(U)$ is given by*

$$L^{s,p}(U) = \{f \in L^p(U) \colon D^\alpha f \in L^p(U) \text{ for all } \alpha \text{ with } |\alpha| \leq s\},$$

*equipped with the norm $\|f\|_{L^{s,p}(U)} = \sum_{|\alpha| \leq s} \|D^\alpha\|_{L^p(U)}$.*

It is easy to verify then that the Sobolev spaces $L^{s,p}(U)$ are Banach spaces. Moreover, $L^{s,2}(U)$ spaces can be made into Hilbert spaces by taking the inner product

$$\langle f, g \rangle_{L^{s,2}(U)} = \sum_{|\alpha| \leq s} \langle D^\alpha f, D^\alpha g \rangle_{L^2(U)}.$$

We refer to such spaces as Sobolev Hilbert spaces. Taking $p = \infty$, we recover (up to measure zero) the standard continuity classes.

Sobolev spaces allow us to trade-off integrability and smoothness. To make this more precise, we will need the following definition.

**2 Definition.** *For $U \subset \mathbb{R}^d$ open, its boundary is the set $\partial U = \bar{U} \setminus U$. We say that $U$ is Lipschitz if $\partial U$ can be expressed as the graph of a Lipschitz function.*

Crucially, any convex set is Lipschitz. With that in place, we have:

**3 Rellich-Kondrachov theorem** (Adams et al. (2003))**.** *Let $U \subset \mathbb{R}^d$ be open with a Lipschitz boundary. Then for $mp > d > (m-1)p$ and all $\lambda$ satisfying $0 < \lambda \leq m - d/p$, $L^{s+m,p}(U)$ can be continuously embedded in $C^{s,\lambda}(\bar{U})$.*

Informally, the Rellich-Kondrachov theorem states that we can trade $L^p(U)$ derivatives for $L^\infty(U)$ derivatives at a cost of $d/p$ orders of smoothness.

### 2.1.3 Fourier characterisation of Sobolev Hilbert spaces

We now focus on the Sobolev Hilbert spaces $L^{s,2}(U)$. We provide an alternative characterisation of these that uses the Fourier transform and a suitable extension operator.

We define the Fourier transform as the unitary operator on $L^1(\mathbb{R}^d)$ given by

$$(\mathcal{F}f)(\omega) = (2\pi)^{-d/2} \int_{\mathbb{R}^d} f(x) e^{i\langle x,\omega\rangle} \mathrm{d}x$$

with a corresponding inverse

$$(\mathcal{F}^{-1}g)(x) = (2\pi)^{-d/2} \int_{\mathbb{R}^d} g(\omega) e^{-i\langle x,\omega\rangle} \mathrm{d}\omega.$$

The pair $\mathcal{F}$, $\mathcal{F}^{-1}$ can be continuously extended to operators on $L^2(\mathbb{R}^d)$, a result referred to as Plancherel's theorem. We do not distinguish between $\mathcal{F}$, $\mathcal{F}^{-1}$ and their extensions. Throughout, $\omega$ will denote a generic argument for the Fourier transform of a function.

For $u, v \in L^2(\mathbb{R}^d)$, the Fourier transform operator $\mathcal{F}$ has the following useful properties:

1. Unitary, $\langle u, v\rangle = \langle \mathcal{F}u, \mathcal{F}v\rangle$
2. Differentiation, $(\mathcal{F}D^\alpha u)(\omega) = (i\omega)^{|\alpha|}\mathcal{F}u$
3. Convolution, $u * v = (2\pi)^{d/2}\langle \mathcal{F}u, \mathcal{F}v\rangle$

With that, we have the following characterisation of Sobolev Hilbert spaces on $\mathbb{R}^d$.

**4 Definition.** *We define the Sobolev Hilbert space $H^s(\mathbb{R}^d)$ by*

$$H^s(\mathbb{R}^d) = \{f \in L^2(\mathbb{R}^d) \colon (\mathcal{F}f)(\omega)(1 + \|\omega\|_2^2)^{s/2} \in L^2(\mathbb{R}^d)\}$$

*equipped with the inner product*

$$\langle f, g\rangle_{H^s} = (2\pi)^{-d/2} \int_{\mathbb{R}^d} (\mathcal{F}f)(\omega)(\mathcal{F}g)(\omega)(1 + \|\omega\|_2^2)^s d\omega$$

*and the norm induced by the inner product.*

Using the unitary and differentiation property of Fourier transforms, we can confirm that the thus defined spaces $H^s(\mathbb{R}^d)$ are norm equivalent to $L^{s,2}(\mathbb{R}^d)$. However, since the Fourier transform is a global operation, a similar statement for $U \subset \mathbb{R}^d$ is not immediate. For that, we will need the following extension theorem.

**5 Theorem** (E. M. Stein (1970), Theorem 5′)**.** *Let $U \subset \mathbb{R}^d$ be open and Lipschitz. Then there exists a linear operator $E$ mapping functions on $U$ to functions on $\mathbb{R}^d$ with the properties*

*i. $E(f)|_U = f$, that is, $E$ is an extension operator;*

*ii. $E$ maps $L^{s,p}(U)$ continuously into $L^{s,p}(\mathbb{R}^d)$ for all $p$, $1 \leq p \leq \infty$, and all non-negative integral $s$.*

*Moreover the norms of these mappings have bounds which depend only on $s, d$ and the boundary of $U$.*

Theorem 5 allows us to define $H^s(U)$ as the set of measurable functions $u\colon U \mapsto \mathbb{R}$ with finite quotient norm $\|u\|_{H^s(U)}$, given by

$$\|u\|_{H^s(U)} = \inf \left\{ \|f\|_{H^s(\mathbb{R}^d)} \colon f|_U = u \right\}.$$

Since the norm of the relevant extension operators is bounded independently of $u$, the thus defined space $H^s(U)$ is norm-equivalent to $L^{s,2}(U)$.

## 2.2 Kernels and reproducing kernel Hilbert spaces

In our work kernels will be used to define both the problem class and our solutions. We now formalise the concept of a kernel. The following definitions and theorems are from chapter 4 of Christmann et al. (2008) and chapter 10 of Wendland (2004)—we strongly recommend both for a more in-depth introduction to the topic.

**6 Definition.** *For $\mathcal{X}$ a non-empty set, a function $k\colon \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is called a* kernel *on $\mathcal{X}$ if there exists a Hilbert space $H$ and a map $\varphi\colon \mathcal{X} \mapsto H$ such that for all $x, y \in \mathcal{X}$ we have*

$$k(x, y) = \langle \varphi(x), \varphi(y) \rangle_H .$$

*We call $\varphi$ a feature map and $H$ a feature space of $k$.*

The following alternative characterisation is easier to verify in practice.

**7 Theorem.** *A function $k\colon \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is a kernel if and only if it is symmetric and positive definite.*

While multiple feature maps and feature spaces can correspond to an individual kernel, with each kernel we can associate a canonical feature space $H_k$ called the Reproducing

Kernel Hilbert Space (RKHS) of $k$. To define this canonical feature space, we need:

**8 Definition.** *Let $H$ be a real Hilbert space of functions $f\colon \mathcal{X} \mapsto \mathbb{R}$. A function $k\colon \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is called a reproducing kernel for $H$ if:*

*i. $k(\cdot, y) \in H$ for all $y \in \mathcal{X}$, and*

*ii. $f(y) = \langle f, k(\cdot, y)\rangle_H$ for all $f \in H$ and all $y \in \mathcal{X}$.*

Reproducing kernels and kernels coincide. If $k$ is the reproducing kernel for $H_k$, then $k$ is symmetric and positive definite (Wendland, 2004, Theorem 10.4). On the other hand, each symmetric and positive definite $k$ is the reproducing kernel for some RKHS $H_k$ with the feature map $\varphi(y) = k(\cdot, y)$. For a given kernel $k$, we can construct $H_k$ directly.

**9 Theorem.** *Let $H'_k(\mathcal{X}) = \operatorname{span}\{k(\cdot, x)\colon x \in \mathcal{X}\}$ and equip this space with the bilinear form*

$$\left\langle \sum_{j=1}^{N} \alpha_j k(\cdot, x_j), \sum_{k=1}^{M} \beta_k k(\cdot, y_k) \right\rangle_{H'_k} = \sum_{j=1}^{N} \sum_{k=1}^{M} \alpha_j \beta_k k(x_j, y_k).$$

*Then $\langle \cdot, \cdot \rangle_{H'_k}$ is an inner product on $H'_k$. Furthermore, $H'_k$ is a pre-Hilbert space with reproducing kernel $k$ and the completion of $H'_k$ is $H_k$.*

From this construction we see that $H_k$ corresponds to the space of functions that can be represented by kernel ridge regression with kernel $k$. We now provide alternative constructions for $H_k$ using Bochner's and Mercer's theorems.

### 2.2.1 Bochner's theorem

Bochner's theorem characterises stationary kernels in terms of their Fourier transform. We call a kernel $k$ stationary if $k(x, y) = \kappa(x - y)$ for some $\kappa\colon \mathbb{R}^d \mapsto \mathbb{R}$.

**10 Theorem** (Bochner)**.** *A continuous function $\kappa\colon \mathbb{R}^d \mapsto \mathbb{R}$ is positive semi-definite if and only if it is the Fourier transform of a finite non-negative Borel measure on $\mathbb{R}^d$.*

The following is an immediate consequence of Bochner's theorem.

**11 Theorem.** *Suppose $k$ is a stationary kernel given by function $\kappa \in C(\mathbb{R}^d) \cap L^1(\mathbb{R}^d)$. Define*

$$H = \left\{ f \in L^2(\mathbb{R}^d) \cap C(\mathbb{R}^d)\colon \frac{\mathcal{F}f}{\sqrt{\mathcal{F}\kappa}} \in L^2(\mathbb{R}^d) \right\}$$

*and equip $H$ with the inner product*

$$\langle f, g\rangle_H = \left\langle \frac{\mathcal{F}f}{\sqrt{\mathcal{F}\kappa}}, \frac{\mathcal{F}g}{\sqrt{\mathcal{F}\kappa}} \right\rangle_{L^2(\mathbb{R}^d)} = \int_{\mathbb{R}^d} \frac{(\mathcal{F}f)(\omega)(\mathcal{F}g)(\omega)}{(\mathcal{F}\kappa)(\omega)} d\lambda(\omega).$$

*Then $H$ and $H_k$ are isomorphic.*

The quantity $\mathcal{F}\kappa$ is referred to as the *spectral density* of $k$. Henceforth, we will drop the distinction between $k$ and $\kappa$ and write $k(x, y) = k(x - y)$ when $k$ is a stationary kernel.

As in the case of Sobolev Hilbert spaces, we can generalise this result for subsets of $\mathbb{R}^d$ through the use of restrictions:

**12 Theorem** (Aronszajn (1950), section I.5)**.** *Let $S$ be a set and $H_k(S)$ an RKHS on $S$ with reproducing kernel $k$ and norm $\|\cdot\|_{H_k(S)}$. Then for a set $U \subset S$, $k|_U$ is the reproducing kernel of $H_k(U)$ with the quotient norm $\|u\|_{H_k(U)} = \min\{\|f\|_{H_k(S)} \colon f|_U = u\}$.*

Indeed, we shall shortly use Bochner's theorem and this restriction result to demonstrate norm equivalence between certain Sobolev Hilbert spaces and the RKHSs of Matérn family kernels.

### 2.2.2 Mercer's theorem

For $\mathcal{X}$ a compact set and $H_k(\mathcal{X})$ a separable RKHS on $\mathcal{X}$ with reproducing kernel $k$, Mercer's theorem constructs a basis for $H_k(\mathcal{X})$ consisting of the eigenfunctions of an associated kernel integral operator. For $\mu$ a measure with full support on $\mathcal{X}$, we define $\widetilde{T}_k \colon L^2_\mu \mapsto C(\mathcal{X})$ to be the integral operator

$$\widetilde{T}_k \colon f \mapsto \int k(\cdot, x) f(x) \mathrm{d}\mu(x), \tag{2.1}$$

and $T_k \colon L^2_\mu(\mathcal{X}) \mapsto L^2_\mu(\mathcal{X})$ to be the composition of $\widetilde{T}_k$ and the inclusion $C(\mathcal{X}) \hookrightarrow L^2_\mu(\mathcal{X})$. $T_k$ is referred to as the integral operator of $k$ on $L^2_\mu(\mathcal{X})$. It is straightforward to show $T_k$ is compact, positive and self adjoint. Therefore, by the spectral theorem, there exists a countable orthonormal set $\{\widetilde{\vartheta}_j\}_{j\in J}$ of $L^2_\mu(\mathcal{X})$ and a sequence $\{\lambda_j\}_{j\in J}$ with $\lambda_1 \geq \lambda_2, \cdots > 0$ such that

$$T_k f = \sum_{j\in J} \lambda_j \left\langle f, \widetilde{\vartheta}_j \right\rangle \widetilde{\vartheta}_j \quad (f \in L^2_\mu(\mathcal{X})). \tag{2.2}$$

Moreover, each $\widetilde{\vartheta}_j$ has a continuous representative $\vartheta_j = \lambda_j^{-1} \widetilde{T}_k \widetilde{\vartheta}_j$. It is immediate that $T_k \vartheta_i = \lambda_i \vartheta_i$ for all $i \in J$, and thus $\{(\lambda_i, \vartheta_i) \colon i \in J\}$ form an eigensystem for $T_k$. This

diagonalisation result allows us to express the kernel $k$ in terms of the series

$$k(x, y) = \sum_{j \in J} \lambda_j \vartheta_j(x) \vartheta_j(y),$$

which converges uniformly on $\mathcal{X} \times \mathcal{X}$ and absolutely for each $(x, y) \in \mathcal{X} \times \mathcal{X}$. This latter result is known as Mercer's theorem. The following is a direct consequence:

**13 Theorem.** *With the assumptions and notations of Mercer's theorem, define*

$$H = \left\{ \sum_{j \in J} a_j \vartheta_j : \left( \frac{a_j}{\sqrt{\lambda_j}} \right) \in \ell^2 \right\}$$

*and equip $H$ with the inner product*

$$\left\langle \sum_{i \in J} a_i \vartheta_i, \sum_{j \in J} b_j \vartheta_j \right\rangle_H = \sum_{j \in J} \frac{a_j b_j}{\lambda_j}$$

*and the norm induced by $\langle \cdot, \cdot \rangle_H$. Then $H$ and $H_k$ are isomorphic.*

Mercer's theorem and the associated integral operator will form the basis of our analysis of kernel-based optimisation in chapter 4. For that later use, it is important to note that the operator $T_k$ depends both on the kernel $k$ and the measure with respect to which it is defined ($\mu$ in the above notation), and therefore so does its eigendecomposition.

We will make frequent use of the following observations around Mercer's theorem:

1. For $\mu$ the uniform measure on a finite set $\mathcal{X} = \{x_1, \ldots, x_m\}$, the corresponding integral operator is the Gram matrix $[K]_{ij} = k(x_i, x_j)$ and the diagonalisation result corresponds to the finite-dimensional eigendecomposition of $K$.

2. For $k$ a stationary kernel, the integral operator $T_k$ is a convolution against $k$ and can thus be evaluated using the convolution property of the Fourier transform.

3. When $k$ is stationary and periodic on the interval $I_d = [0, 2\pi]^d$, we can represent it by the Fourier series as

$$k(x) = \sum_{n=-\infty}^{\infty} c_n e^{-i\langle n, x \rangle} \quad \text{where} \quad c_n = \int k(x) e^{-i\langle n, x \rangle} dx.$$

   The sequence $\{c_n\}$ are the Fourier coefficients for $k$. Note that $\{(c_n, e^{-i\langle n, x \rangle}) : n \in \mathbb{Z}\}$ is an eigendecomposition of $k$.

4. For $U$ an enlargement of $\mathcal{X}$ (Minkowski sum with a ball), any stationary kernel on $\mathcal{X}$ can be written as the restriction of a periodic kernel on $U$.

### 2.2.3 The Matérn family of kernels

We now look at Matérn kernels, introduced by M. L. Stein (1999) and named after the work of Matérn (1960) on spatial interpolation. For $\nu > 0$, the Matérn-$\nu$ kernel on $\mathbb{R}^d$ is the stationary kernel given by

$$k_\nu \colon x \longmapsto \frac{2^{1-\nu}}{\Gamma(\nu)} \left( \frac{\|x\|_2 \sqrt{2\nu}}{\ell} \right)^\nu B_\nu \left( \frac{\|x\|_2 \sqrt{2\nu}}{\ell} \right),$$

where $\ell > 0$ is a lengthscale parameter, $\Gamma$ denotes the Gamma function and $B_\nu$ is the modified Bessel function of the second kind. Its spectral density is

$$S_\nu(\omega) \doteq (\mathcal{F} k_\nu)(\omega) = \frac{2^d \pi^{d/2} \Gamma(\nu + d/2)(2\nu)^\nu}{\Gamma(\nu) \ell^{2\nu}} \left( \frac{2\nu}{\ell^2} + 4\pi \|\omega\|_2^2 \right)^{-(\nu + d/2)}.$$

That is, the Lebesgue density of a student-t distribution. From theorem 11 and definition 4, the RKHS $H_{k_\nu}(\mathbb{R}^d)$ is norm equivalent to $H^{\nu + d/2}(\mathbb{R}^d)$. Therefore by the 3, the functions in $H_{k_\nu}(\mathbb{R}^d)$ have $\lfloor \nu \rfloor$ strong derivatives. For $\nu \leq 1$, they are $\nu$-Hölder continuous. By theorem 12 and theorem 5, the Matérn kernel for $U \subset \mathbb{R}^d$ convex (and the corresponding RKHS) can be obtained by restriction.

Evaluating the Matérn kernel for general $\nu$ has to be done numerically. However, where $\nu = p + 1/2$ for $p \in \mathbb{N}$, the Matérn kernel is given by the expression

$$k_{p+1/2} \colon x \mapsto \exp\left( -\frac{\|x\|_2 \sqrt{2p+1}}{\ell} \right) \frac{p!}{(2p)!} \sum_{i=0}^{p} \frac{(p+i)!}{i!(p-i)!} \left( \frac{2\|x\|_2 \sqrt{2p+1}}{\ell} \right)^{p-i},$$

and is therefore easy to compute. The corresponding RKHS contains $p$-times differentiable functions. Practical use of the Matérn kernel is generally limited to the first three Matérn integer-and-a-half kernels, given by

$$k_\nu = M_{\nu,\ell}(x) \exp\left( -\frac{\|x\|_2}{\ell} \right) \quad \text{for} \quad M_{\nu,\ell}(x) = \begin{cases} 1 & \nu = 1/2, \\ 1 + \frac{\|x\|_2}{\ell} & \nu = 3/2, \\ 1 + \frac{\|x\|_2}{\ell} + \frac{\|x\|_2^2}{3\ell^2} & \nu = 5/2. \end{cases}$$

Functions with three or more derivatives are often modelled using the Squared Exponential

kernel

$$k_{\mathrm{SE}}(x) = \lim_{\nu \to \infty} \kappa_\nu(x) = \exp\left(\frac{\|x\|_2^2}{2\ell^2}\right),$$

which has a Gaussian spectral density and RKHS $H_{k_{\mathrm{SE}}}(\mathbb{R}^d) = \bigcap_{\nu>0} H_{k_\nu}(U)$ containing smooth functions only.

## 2.3 Function estimation in kernel spaces

With the benefit of a more rigorous introduction to kernels, we now formalise the kernelised linear ridge regressor sketched in the previous chapter.

We begin with some definitions. For observations $\mathcal{H}_t$ and a regularisation parameter $\lambda > 0$, we write $(\mu_t, \sigma_t) = \mathrm{GPR}(\lambda, \mathcal{H}_t)$ for the output of kernel ridge regression, where

$$\mu_t(x) = k_t(x)(K_t + \lambda I)^{-1} Y_{1:t} \quad \text{and} \quad \sigma_t^2(x) = k(x,x) - k_t(x)(K_t + \lambda I)^{-1} k_t^T(x)$$

with $k_t(x) = (k(x, X_1), \ldots, k(x, X_t))^T$ and $[K_t]_{ij} = k(X_i, X_j)$. The kernel used will be specified in context where relevant. We call $\mu_t$ the mean function, $\sigma_t^2$ the variance function and $K_t$ the Gram matrix. We will associate with the regressor the quantity

$$\gamma_t = \frac{1}{2} \log\det\left(I + \lambda^{-1} K_t\right),$$

which we call the *information gain*.

With the notation in place, consider the following extension of the OFUL concentration inequality to kernel ridge regression, derived originally in Abbasi-Yadkori (2009) and later rederived (in a weaker form) and popularised by Chowdhury et al. (2017).

**14 Theorem.** *Let $H_k(\mathcal{X})$ be an RKHS on $\mathcal{X} \subset \mathbb{R}^d$ with reproducing kernel $k$. For $T \in \mathbb{N}$, let $\mathcal{H}_1 \subset \ldots \subset \mathcal{H}_T$ be a sequence of histories with $\mathcal{H}_t = (X_1, Y_1, \ldots, X_t, Y_t)$, where $Y_t = f(X_t) + \varepsilon_t$ for $f \in H_k(\mathcal{X})$ with $\|f\|_{H_k(\mathcal{X})} \leq \mathfrak{R}$ and $\varepsilon_t$ conditionally subGaussian with constant $\mathfrak{B}$. Then for $(\mu_t, \sigma_t) = GPR(1 + 1/T, H_t)$, with probability $1 - \delta$,*

$$|\mu_t(x) - f(x)| \leq \sigma_t(x)\left(\mathfrak{R} + \mathfrak{B}\sqrt{2(\gamma_t + \log(1/\delta))}\right)$$

*for all $x \in \mathcal{X}$ and all $t \leq T$.*

This concentration inequality is used to prove regret bounds for a kernelised version of

OFUL. We will look at its use in detail over the course of the next two chapters, but first we build some intuition for the quantities featuring within it. We do so from two perspectives:

1. A classical take based on kernel ridge regression. This is common in literature on frequentist statistics, optimal experimental design, control and bandit algorithms.
2. A Bayesian perspective, through an equivalence with conjugate Gaussian process regression, used in the context of Bayesian statistics and Bayesian optimisation.

For a broader overview of the relationship between the two perspectives, see the excellent monograph by Kanagawa et al. (2018). We will end the chapter on an implementation note, in particular an outline of a useful online method of updating Cholesky decompositions of Gram matrices. For more general information on the implementation of these regression methods consult chapter 2 of Rasmussen et al. (2006).

### 2.3.1 Kernel ridge regression

We begin with the classical take. We assume $\mathcal{X}$ is a compact subset of $\mathbb{R}^d$ and denote by $X_{1:t} = (X_i)_{i=1}^t \subset \mathcal{X}$ some input locations with corresponding values $Y_{1:t} = (Y_i)_{i=1}^t \subset \mathbb{R}$. Then, for a kernel $k$ on $\mathcal{X}$ and a regularisation parameter $\lambda > 0$, the kernel ridge regression problem is that of finding a function $m_t \in H_k(\mathcal{X})$ minimising

$$\sum_{i=1}^{t} (Y_i - m_t(X_i))^2 + \lambda \|m_t\|^2_{H_k(\mathcal{X})}$$

This can be interpreted as finding a function that fits the observed values well while remaining smooth, in a sense defined by the RKHS norm. Perhaps somewhat surprisingly (or not), the minimiser of the kernel ridge regression objective lies in an $t$-dimensional dimensional subspace of $H_k(\mathcal{X})$.

**15 Theorem** (Representer Theorem)**.** *The minimiser of the kernel ridge regression objective is of the form* $\sum_{i=1}^t \alpha_i k(X_i, \cdot)$ *with* $\alpha_1, \dots, \alpha_t \in \mathbb{R}^t$.

*Proof.* Let $S = \operatorname{span}\{k(x, \cdot) \colon x \in X\}$. $S$ is finite dimensional and therefore a closed subset of $H_{k(\mathcal{X})}$. By the projection theorem, $H_{k(\mathcal{X})} = S \oplus S^\perp$. That is, all $f \in H_{k(\mathcal{X})}$ can be written as the sum of some $f_S \in S$ and $f_{S^\perp} \in S^\perp$ with $\langle f_S, f_{S^\perp} \rangle = 0$. Let $f$ be a minimiser of the objective. Then $f(X_i) = \langle k(x_i, \cdot), f_S + f_{S^\perp} \rangle = f_S(X_i) + \langle k(X_i, \cdot), f_{S^\perp} \rangle = f_S(X_i)$. Moreover, $\|f\|_{H_{k(\mathcal{X})}} = \|f_S\|_{H_{k(\mathcal{X})}} + \|f_{S^\perp}\|_{H_{k(\mathcal{X})}} \geq \|f_S\|_{H_{k(\mathcal{X})}}$. On the other hand, since

$f$ minimises the objective and agrees with $f_S$ on $X$, $\|f\|_{H_{k(\mathcal{X})}} \leq \|f_S\|_{H_{k(\mathcal{X})}}$. Therefore $f = f_S$. □

This result reduces solving the kernel ridge regression problem to identifying a $t$-dimensional vector. We use this to write

$$m_t(X_j) = \left\langle k(x_j, \cdot), \sum_{i=1}^{n} \alpha_i k(x_i, \cdot) \right\rangle_H = [K_t \alpha]_j,$$

where $\alpha = [\alpha_1, \ldots, \alpha_t]^T$ and we used that $k$ is the reproducing kernel of $H_k(\mathcal{X})$. This allows us to rewrite the ridge regression objective in the form

$$\|y - K_t \alpha\|_2^2 + \lambda \alpha^T K_t \alpha.$$

This is a sum of two convex terms and therefore convex. Differentiating shows that the optimal parameters are given by $\alpha^\star = (K_t + \lambda I)^{-1} Y_{1:t}$, a linear combination of the observations $Y_{1:t}$. The resulting optimal $m_t$ is then given by

$$m_t(x) = k_t(x)\alpha^\star = k_t(x)(K_t + \lambda I)^{-1} Y_{1:t},$$

and thus $m_t = \mu_t$.

We can can obtain a measure of confidence within kernel ridge regression by using ridge leverage scores. The ridge leverage score for a data point $(Y_i, X_i)$ is the derivative of $m(X_i)$ with respect to $Y_i$ and is given by

$$\tau_i \doteq \frac{\partial m(X_i)}{\partial Y_i} = [K_t (K_t + \lambda I)^{-1}]_{ii}.$$

Clearly, $0 < \tau_i < 1$. Where $\tau_i$ is near to zero, changing $Y_i$ does not change $m(X_i)$ by much and we can think of the regression at that point as being strongly determined by previous observations. The opposite holds for $\tau_i \approx 1$.

Ridge leverage scores are proportional to the variance function evaluated on the data points:

**16 Theorem.** *With the established notation, $\sigma_t^2(X_i) = \lambda \tau_i$.*

*Proof.* Let $P$ be the symmetric square root of $K_t$ and let $e_i \in \mathbb{R}^t$ be the $i$th unit vector

in the standard basis of $\mathbb{R}^t$. Then

$$\begin{aligned}\sigma_i^2(X_i) &= e_i^T K_t e_i - e_i K_t^T (K_t + \lambda I)^{-1} K_t e_i \\ &= e_i^T P(I - P(K_t + \lambda I)^{-1} P) P e_i \\ &= e_i^T P(I - (K_t + \lambda I)^{-1} K_t) P e_i, \end{aligned} \tag{2.3}$$

where the last line uses the push-through identity. Next, premultiplying $(K_t + \lambda I) - K_t = \lambda I$ by $(K_t + \lambda I)^{-1}$ we obtain $I - (K_t + \lambda I)^{-1} K_t = \lambda (K_t + \lambda I)^{-1}$. Substituting this into eq. (2.3) and applying the push-through identity gives

$$\sigma_i^2(X_i) = \lambda e_i^T P(K_t + \lambda I)^{-1} P e_i = \lambda [K_t (K_t + \lambda I)^{-1}]_{ii} = \lambda \tau_i. \qquad \square$$

Another quantity of interest is the *effective dimension*, which is the sum of the ridge leverage scores,

$$d_t' \doteq \sum_{i=1}^{t} \tau_i = \operatorname{Tr}\left[E\Lambda(\Lambda + \lambda I)^{-1} E^T\right] = \sum_{i=1}^{t} \frac{\lambda_i}{\lambda_i + \lambda},$$

where $E\Lambda E^T$ is the eigendecomposition of $K_t$ and we used that $K_t$ and $K_t + \lambda I$ are simultaneously diagonalisable, and $\lambda_1 \geq \ldots \geq \lambda_t \geq 0$ are the eigenvalues of $K_t$. Consider the case where $\tau_i \approx 1$ for each $i$. Then each prediction $y_i^\star$ is near-orthogonal to any other $y_j^\star$ with $i \neq j$, and hence the regression problem can be thought to have approximately $\dim y^\star$ degrees of freedom. On the other hand, when this sum of ridge leverage scores is low relative to the number of observations, the regression can be determined by a small subset of the dimensions. The effective dimension is an object of frequent interest in sketching algorithms, where it is approximately the number of points that need to be retained to accurately approximate the full regression (Yang et al., 2017). We will shortly show that effective dimension is equal to information gain, up to a logarithmic factor.

### 2.3.2 Gaussian process regression

We now look at the same regressor from the perspective of Bayesian inference with a conjugate Gaussian model. Here, we assume that the target function is a random process $F$ on the domain $\mathcal{X}$, for which any finite marginal $F_A$ for $A \subset \mathcal{X}$ has a Gaussian

distribution with moments

$$\mathbf{E}[F_{A_i}] = 0 \quad \text{and} \quad \text{Cov}[F_{A_i}, F_{A_j}] = k(A_i, A_j)$$

for a given symmetric positive definite covariance function $k\colon \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$. That is, we assume $F$ is a centred Gaussian process with covariance kernel $k$. We assume that we have observed noisy realisations of the Gaussian process at locations $X_1, \ldots, X_t$, of the form

$$Y_{X_i} = F_{X_i} + \varepsilon_i,$$

where $\varepsilon_1, \ldots, \varepsilon_t$ are independent zero mean Gaussian random variables with variance $\lambda > 0$, and that we are interested in finding the least-squares estimator of the realisation of $F$. By definition, this is the conditional expectation of $F$ given $Y_X$.

Consider the random vector $[Y_X; F_Z]^T$. From the problem definition,

$$\begin{bmatrix} Y_X \\ F_Z \end{bmatrix} \sim \mathcal{N}\left( \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} K_{XX} + \lambda I & K_{XZ} \\ K_{ZX} & K_{ZZ} \end{bmatrix} \right), \tag{2.4}$$

where $K_{ZX}$ is a matrix with entries $[K_{ZX}]_{ij} = k(Z_i, X_j)$. The conditional $F_Z|Y_X$ can be computed from this joint using standard results on Gaussian conditioning (see, for example, K. B. Petersen et al., 2008). It is again Gaussian, with moments

$$\mathbf{E}[F_Z|Y_X] = K_{XZ}(K_{ZZ} + \lambda I)^{-1} Y_X, \quad \text{and} \tag{2.5}$$

$$\text{Cov}\,[F_Z, F_Z|Y_X] = K_{ZZ} - K_{ZX}(K_{XX} + \lambda I)^{-1} K_{XZ}. \tag{2.6}$$

Since $Z$ was a finite but otherwise arbitrary subset of $\mathcal{X}$, Kolmogorov's extension theorem gives that $F|Y_X$ is a Gaussian process on $\mathcal{X}$ with posterior mean $\mu_t$ and posterior variance $\sigma_t^2$, that is those corresponding exactly to our previously defined regressor.

We make two observations:

1. Computing the Gaussian process posterior distribution requires only the manipulation of finite dimensional Gaussian distributions.

2. The expression for the posterior variance is the Schur complement of the covariance matrix in eq. (2.4); it depends only on the input locations $X$ and not on $Y_X$.

This perspective on kernel-based regression provides insight on information gain. Denote by $I(A; B)$ the mutual information between random variables $A$ and $B$. Then:

**17 Lemma.** *For conjugate Gaussian process regression on a domain $\mathcal{X}$ with a gram matrix $K_{XX}$ and likelihood variance $\lambda$,*

$$I(F; Y_X) = I(F_X; Y_X) = \frac{1}{2} \log \det(1 + \lambda^{-1} K_{XX}) \doteq \gamma_{|X|}.$$

That is, $\gamma_{|X|}$ is the decrease in entropy of $F$ after observing $Y_X$. The proof relies on standard properties of mutual information (see, for example, chapter 2 of Cover (1999)).

*Proof.* $I(F; Y_X) = I(F_X; Y_X)$ follows from Fubini's theorem. To find a closed form expression for $I(F_X; Y_X)$, we use symmetry of mutual information and write it as

$$I(F_X; Y_X) = I(Y_X; F_X) = H(Y_X) - H(Y_X | F_X),$$

where $H(A)$ denotes entropy of a random variable $A$. Consider $Y_X$ and $Y_X | F_X$ in turn:

1. Since $Y_X \sim \mathcal{N}(0, K_{XX} + \lambda I)$, a standard result gives

$$\begin{aligned} H(Y_X) &= \frac{1}{2} \log \det(2\pi e \left(\lambda I + K_{XX}\right) \\ &= \frac{|X|}{2} \log\left(2\pi e \lambda\right) + \frac{1}{2} \log \det\left(I + \lambda^{-1} K_{XX}\right). \end{aligned}$$

2. For $Y_X | F_X$, note first $Y_X = F_X + \varepsilon_X$ for some $\varepsilon_X \sim \mathcal{N}(0, \lambda I)$. Therefore

$$H(Y_X | F_X) = H(F_X + \varepsilon_X | F_X) = H(\varepsilon_X | F_X) = H(\varepsilon_X) = \frac{|X|}{2} \log\left(2\pi e \lambda\right),$$

   where we used that for $a$ constant, $H(a + A) = H(A)$ and $\varepsilon_X$ is independent of $F_X$.

Combining the two results, we obtain the second equality. □

Like effective dimension, information gain can be thought to quantify the complexity of a regression problem. When the maximum amount of information that can be gained with each new observation decreases quickly, a small number of well-chosen observations can be used to accurately approximate the result of the full conjugate Gaussian process regression. Crucially for our later contributions, maximum information gain depends on the domain $\mathcal{X}$. Quantifying this relationship for the Matérn kernel will be one of our main contributions in this part of the thesis.

The log ratio of confidence ellipsoid volumes featuring in the OFUL concentration

inequality is a special case of the information gain defined here. Specifically, for $X \in \mathbb{R}^{t\times d}$ and with a linear kernel, such that $K_{XX} = XX^T$, we have

$$\log\det(I + \lambda^{-1}K_{XX}) = \log\frac{\det(X^TX + \lambda I)}{\det(\lambda I)},$$

which is exactly the $\log\frac{\det\Sigma_t}{\det\lambda I}$ term featuring in the OFUL confidence sets.

### 2.3.3 Connecting information gain and effective dimension

We now relate information gain and effective dimension. First, we need that predictive variance is monotonous in the number of observations:

**18 Lemma.** *For $\mathcal{H}_{t-1} \subset \mathcal{H}_t$, $\sigma^2_{t-1}(x) \geq \sigma^2_t(x)$ for all $x \in \mathcal{X}$.*

*Proof.* Fix an arbitrary $x \in \mathcal{X}$ and $\Delta_t = \sigma^2_{t-1}(x) - \sigma^2_t(x)$. We wish to show that $\Delta_t \geq 0$. By definition,

$$\Delta_t = k_t^T(x)\left(K_t + \lambda I\right)^{-1}k_t(x) - k_{t-1}^T(x)\left(K_{t-1} + \lambda I\right)^{-1}k_{t-1}(x),$$

which we write as $b_t^T\left(A_t^{-1} - \widetilde{A}_{t-1}^{-1}\right)b_t$ for $A_t = K_t + \lambda I$, $b_t = k_t(x)$ and $\widetilde{A}_{t-1}^{-1} = \begin{bmatrix} A_{t-1}^{-1} & 0 \\ 0 & 0 \end{bmatrix}$. The result follows by observing that $A_t^{-1} - \widetilde{A}_{t-1}^{-1} \succeq 0$. Specifically, writing $m_{t-1} = A_{t-1}b_{t-1}$, by the Woodbury identity,

$$A_t^{-1} - \widetilde{A}_t^{-1} = \begin{bmatrix} A_{t-1} & b_{t-1} \\ b_{t-1}^T & \|b_{t-1}\|^2 + \lambda \end{bmatrix}^{-1} - \widetilde{A}_t^{-1} = (A_t \setminus A_{t-1})^{-1}\begin{bmatrix} m_{t-1}m_{t-1}^T & -m_{t-1} \\ -m_{t-1}^T & 1 \end{bmatrix},$$

which is clearly positive semi-definite. □

Next, we bound sums of online predictive variances by information gain. The result is of independent interest for the analysis of kernel-based algorithms. It first appeared in Srinivas, Krause, S. M. Kakade et al. (2009) where it was proved using probabilistic and information theoretic arguments. We provide an algebraic proof.

**19 Lemma** (Sum of variances and information gain)**.** *For a kernel $k$ with $k(x,x) \leq \lambda$ for all $x \in \mathcal{X}$, we have*

$$\lambda^{-1}\sum_{i=1}^{t}\sigma_i^2(X_i) \leq 2\log\det(I + \lambda^{-1}K_t).$$

*Proof.* With the usual notation, we write each $K_i$ for $i \leq t$ recursively as

$$K_i = \begin{bmatrix} K_{i-1} & k_i(X_i) \\ k_i^T(X_i) & k(X_i, X_i) \end{bmatrix}.$$

Then, by Schur's determinant lemma,

$$\det(I + \lambda^{-1} K_i) = \det(I + \lambda^{-1} K_{i-1})(1 + \lambda^{-1} K_i / K_{i-1}), \tag{2.7}$$

where $K_i/K_{i-1}$ denotes the Schur complement of $K_i$ and we take $K_1/K_0 = K_1$. Applying the lemma recursively and noting that $K_i/K_{i-1} = \sigma_i^2(X_i)$, we have

$$\log\det(I + \lambda K_i) = \log \prod_{i=1}^{t} (1 + \lambda^{-1} K_i / K_{i-1}) = \sum_{i=1}^{t} \log(1 + \lambda^{-1} \sigma_i^2(X_i)).$$

The result follows by noting that

$$0 < \lambda^{-1}\sigma_i^2(X_i) \leq \lambda^{-1} k(x, x) \leq 1 \quad \text{and so} \quad 2\log(1 + \lambda^{-1}\sigma_i^2(X_i)) \geq \lambda^{-1}\sigma_i^2(X_i)$$

for all $i \leq t$. □

The relation between information gain and effective dimension is then:

**20 Theorem** (Calandriello, Carratino et al. (2019))**.** *For a kernel $k$ with $k(x, x) \leq \lambda$ for all $x \in \mathcal{X}$, the effective dimension $d'_t$ and information gain $\gamma_t$ are related by*

$$d'_t \leq 4\gamma_t \leq 4d'_t(1 + \log(\lambda^{-1} t + 1)).$$

Calandriello, Carratino et al. (2019) give the above theorem with stronger constants and cite Hazan et al. (2007) for a key inequality in their proof. We could not find the cited inequality in Hazan et al. (2007) or reproduce their stated constants. Our proof of the second inequality follows that of Calandriello, Lazaric et al. (2017b).

*Proof.* For the first inequality, since $\sigma_{i+1}^2(x) \leq \sigma_i^2(x)$ for all $x$ on the domain, we have

$$d'_t = \lambda^{-1} \sum_{i=1}^{t} \sigma_t^2(X_i) \leq \lambda^{-1} \sum_{i=1}^{t} \sigma_i^2(X_i) \leq 2\log\det(I + \lambda^{-1} K_t) = 4\gamma_t.$$

To prove the second inequality, let $\{\lambda_i\}$ be the eigenvalues of $K_t \in \mathbb{R}^{t\times t}$. Then,

$$\log\det(I+\lambda^{-1}K_t) = \log\prod_{i=1}^{t}(1+\lambda^{-1}\lambda_i) = \sum_{i=1}^{t}\log(1+\lambda^{-1}\lambda_i),$$

where we used that the determinant of a matrix is the product of its eigenvalues. Now,

$$\begin{aligned}
\sum_{i=1}^{t}\log(\lambda_i/\lambda+1) &= \sum_{i=1}^{t}\log(\lambda_i/\lambda+1)\left(\frac{\lambda_i/\lambda+1}{\lambda_i/\lambda+1}\right)\\
&= \sum_{i=1}^{t}\log(\lambda_i/\lambda+1)\frac{\lambda_i/\lambda}{\lambda_i/\lambda+1} + \sum_{i=1}^{t}\frac{\log(\lambda_i/\lambda+1)}{\lambda_i/\lambda+1}\\
&\le \log(\lambda_1/\lambda+1)\sum_{i=1}^{t}\frac{\lambda_i}{\lambda_i+\lambda} + \sum_{i=1}^{t}\frac{\lambda_i}{\lambda_i+\lambda}\\
&= d'_t(1+\log(\lambda_1/\lambda+1)),
\end{aligned}$$

where we used the linear bound $\log(x+1)\le x$ for $x\ge 0$. The result follows by noting that $\lambda_1 \le \sum_{i=1}^{t}\lambda_i = t$. □

### 2.3.4 A note on implementation: online Cholesky decomposition

The main cost in computing the GP regressor given some data $X, Y$ consisting of $t$ observations comes from the term $(K_{XX}+\lambda I)^{-1}$, which requires $Ct^3$ operations to compute directly from scratch. Usually, a Cholesky decomposition and subsequent back-solving is used in place of an explicit inversion (see Rasmussen et al. (2006), chapter 2 for details). In the case of sequential algorithms, where we have nested datasets $\mathcal{H}_1 \subset \ldots \subset \mathcal{H}_t$ and need to compute a regressor for each, we can significantly improve the complexity of computing the $t$ regressors with online updates. Specifically, writing

$$K_i = \begin{bmatrix} K_{i-1} & k_i(X_i)\\ k_i^T(X_i) & k(X_i,X_i)\end{bmatrix} = \begin{bmatrix} L_{i-1} & 0\\ b^T & a\end{bmatrix}\begin{bmatrix} L_{i-1}^T & b\\ 0 & a\end{bmatrix} = L_iL_i^T$$

we see that we can take $b = L_{i-1}k_i(X_i)$ and $a = \sqrt{k(X_i,X_i)-b^Tb}$ in the above to obtain $L_i$ recursively from $L_{i-1}$ in $Ci^2$ operations for each $i \le t$.

# Chapter 3

# Optimisation of functions in a Matérn kernel RKHS

In this last introductory chapter of part I, we look at three bandit algorithms for continuous functions: GP-UCB, SupKernelUCB (Valko, Korda et al., 2013) and Hierarchical Optimistic optimisation (Bubeck et al., 2011). We also examine relevant algorithm-agnostic lower bounds. Before doing so, we briefly restate our optimisation problem in terms of the RKHS of a Matérn kernel and establish some notation.

**The standard assumptions.** *We optimise a function $f\colon [0,1]^d \mapsto \mathbb{R}$ in the RKHS of a Matérn-$\nu$ kernel for $\nu > 0$ known, with corresponding RKHS norm $\|f\|_{k_\nu}$ bounded by some known constant $\mathfrak{R} > 0$. At each step $t = 1, \dots, T$ we select an $X_t \in [0,1]^d$ and observe $Y_t = f(X_t) + \varepsilon_t$, where $\varepsilon_t$ is an independent $\mathfrak{B}$-subGaussian random variable.*

We use the word *optimise* here to refer to both minimising regret and simple regret. We will refer to $f$ as the target function and $H_{k_\nu} \doteq H_{k_\nu}([0,1]^d)$ as the target function space. We denote the corresponding RKHS norm by $\|\cdot\|_{k_\nu}$ and inner product by $\langle \cdot, \cdot \rangle_{k_\nu}$. We recall the following results on $H_{k_\nu}$, established in chapter 2:

1. For $0 < \nu < 1$, functions in $H_{k_\nu}$ are $\nu$-Hölder continuous.
2. For $\nu \in (1, \infty) \setminus \mathbb{N}$, functions in $H_{k_\nu}$ have $\lfloor \nu \rfloor$ continuous derivatives everywhere.

Throughout, we will write $(\mu_t, \sigma_t) = \mathrm{GPR}(\lambda, \mathcal{H}_t)$ for a conjugate Gaussian process regressor with kernel $k_\nu$, the reproducing kernel of $H_{k_\nu}$, and some likelihood variance $\lambda > 0$, conditioned on data $\mathcal{H}_t = \{(X_i, Y_i)\colon i < t\}$. We denote by $\gamma_t$ and $d'_t$ the associated

information gain and effective dimension respectively, which we recall differ by at most a logarithmic factor (theorem 20 on page 52).

## 3.1 Lower bounds for Matérn RKHS regret

Scarlett, Bogunovic et al. (2017a) show that under the standard assumptions, the worst-case expected regret for any algorithm $\pi$ after $T$ steps is lower bounded as

$$\hat{R}(T, \pi) \geq CT^{\frac{d+\nu}{d+2\nu}}. \tag{3.1}$$

Their proof is constructive. They define a basic bump function $g(x) \in [2\varepsilon, -2\varepsilon]$, $\varepsilon > 0$ with a maximum at 0 with value $2\varepsilon$ and with value less than $\varepsilon$ outside of a ball of radius $w_0 > 0$. They then construct a set of potential target functions $\{f_m\}$ by offsetting the bump functions $g$ such that they form a $w_0$-spaced grid on $[0, 1]^d$. Under that construction, any $\varepsilon$-optimal point for some $f_m$ is not $\varepsilon$-optimal for $f_{m'}$ with $m' \neq m$ and there are

$$M = \left\lfloor \left(\frac{1}{w_0}\right)^d \right\rfloor$$

potential target functions. Evaluating any point on the domain yields information as to whether $f = f_m$ for one index $m$ only. For given $M$, standard hypothesis testing inequalities established the required number of samples to identify $f \in \{f_m\}$ as a function of $M$. Higher $M$ leads to a harder hypothesis testing problem, but the required decrease in $w_0$ leads to an increase in the RKHS norm of the target functions $\{f_m\}$. The lower bound is established by minimising $w_0$ subject to the RKHS norm constraint.

In minimising worst-case expected regret under the standard assumptions, we are effectively designing algorithms that do well at this type of needle-in-the-haystack bump problem. A proof of worst case regret for this case could be interpreted as a guarantee that, when faced with a hard instance, the algorithm will revert to what is effectively a grid-search. How fine that grid search needs to be is determined by the assumed bound on the RKHS norm, $\mathfrak{R}$. Clearly, however the constructed example is an extreme edge-case and algorithms that treat every problem as if it was this type of edge-case will perform sub-optimally on easier problems. Much theoretical work now focuses explicitly on specifying a measure of the hardness of a problem on a per-instance manner[1] and deriving bounds that depend explicitly on that quantity. A starting point for such

[1]For example, by the suboptimality gaps $\{\Delta_i\}$ for each arm in a finite bandit.

**Algorithm:** GP-UCB
**Parameters:** $\delta \in (0,1)$, $T \in \mathbb{N}$
**Initialisation:** let $\mathcal{H}_1 = \emptyset$

For each $t = 1, \dots, T$:
1. Compute $(\mu_t, \sigma_t) = \mathrm{GPR}(1 + 1/T, \mathcal{H}_t)$
2. Find $X_t$, a maximiser of

$$u_t(x) = \mu_t(x) + \beta_t(\delta)\sigma_t(x)$$

over $[0,1]^d$, where $\beta_t(\delta) = \mathfrak{R} + \mathfrak{B}\sqrt{2(\gamma_t + \log(1/\delta))}$.
3. Observe $Y_t$ corresponding to $X_t$ and let $\mathcal{H}_{t+1} = \mathcal{H}_t \cup \{(X_t, Y_t)\}$.

Figure 3.1: The base GP-UCB algorithm.

adaptivity when constructing algorithms that target worst-case regret is to ensure that worst-case quantities used in the analysis of the algorithm do not appear in the algorithm itself. We will return to this point frequently.

## 3.2 Gaussian process UCB algorithms

The term *Gaussian process UCB algorithm* could refer to any of a family of optimisation algorithms that use Gaussian processes to construct confidence intervals for the target function and act optimistically. The first analysis of such algorithms was given in Srinivas, Krause, S. M. Kakade et al. (2009) and was an extension of the ConfidenceBall linear algorithm Dani et al. (2008). For the purposes of this work, we will consider the extension of OFUL introduced in Abbasi-Yadkori (2009) and later rederived by Chowdhury et al. (2017) as *the* GP-UCB algorithm; we outline it in fig. 3.1.

### 3.2.1 Regret analysis

The following theorem gives the regret of GP-UCB in terms of information gain.

**21 Theorem** (Chowdhury et al. (2017))**.** *The regret of the GP-UCB algorithm under the standard assumptions is bounded as* $R_T \leq C\sqrt{T\gamma_T}(\mathfrak{R} + \mathfrak{B}\sqrt{\gamma_T})$ *with constant probability.*

The proof of this result is simple (given the concentration inequality in theorem 14), and follows exactly the structure of the proof by Srinivas, Krause, S. M. Kakade et al. (2009), which in turn closely resembles the proof for a finite-armed Gaussian bandit we

presented in section 1.2, itself due to Auer, Cesa-Bianchi et al. (2002).

*Proof of theorem 21.* From the acquisition rule,

$$\mu_t(X_t) + \beta_t \sigma_t(X_t) = u_t(X_t) \geq u_t(x^\star) = \mu_t(x^\star) + \beta_t \sigma_t(x^\star).$$

On the $1 - \delta$ event associated with theorem 14,

$$f(x^\star) \leq \mu_t(x^\star) + \beta_t \sigma_t(x^\star) \quad \text{and} \quad -f(X_t) \leq -\mu_t(X_t) + \beta_t \sigma_t(X_t).$$

Henceforth consider only this event. From our two observations, we have

$$r_t = f(x^\star) - f(X_t) \leq 2\beta_t \sigma_t(X_t).$$

By monotonicity of $\beta_t$, $R_T = \sum_{t=1}^T r_t \leq \beta_T \sum_{t=1}^T \sigma_t(X_t)$. Applying Cauchy-Schwarz,

$$\sum_{t=1}^T \sigma_t(X_t) \leq C \left( T \sum_{t=1}^T \sigma_t^2(X_t) \right)^{1/2}.$$

Now note that the sum on the right hand side is the online effective dimension, and by theorem 20 is bounded up to constant factors by the information gain $\gamma_T$. So,

$$R_T \leq C \beta_T \sqrt{T \gamma_T}.$$

□

We shall shortly show that this bound is at least a factor of $\sqrt{\gamma_t}/\iota$ away from the algorithm-agnostic lower bound given in eq. (3.1). Examining the proof for the cause:

1. The regret of GP-UCB is bounded by the sum of the confidence intervals at the sampled locations $X_1, \ldots, X_T$. This same step features in the finite-armed case and is therefore likely tight.

2. Sums of squared confidence intervals are bounded by information gain. From theorem 20, this is tight up to log factors.

Therefore any improvements will need to come in the form of tighter confidence intervals.

Worst case regret for GP-UCB can be established by bounding the the maximal information gain for a given kernel and domain. But is the GP-UCB algorithm adaptive? Recall

the expression for information gain,

$$\gamma_T \doteq \frac{1}{2} \log \det \left( I + \lambda^{-1} K_T \right).$$

While $\gamma_T$ has no explicit dependence on the rewards $Y_1, \ldots, Y_T$, it depends on the sequence $X_1, \ldots, X_T$, which itself is given by the interaction of the bandit problem and the algorithm. If we consider the algorithm as fixed, it is quite apparent that different bandit problems will induce a different distribution over the selected arms and thus over $\gamma_T$. Indeed, on a hard problem, we would likely see the algorithm sample near-uniformly across the domain. Since points would be distant from each other on average, this would lead to high information gain. On the other hand, in a problem with an easy to identify maximum, points are likely to be clustered around this maximum and information gain low. Hence, GP-UCB does adapt to the problem instance.

### 3.2.2 Computational complexity

The computational complexity of GP-UCB can be decomposed into two parts:

1. Conditioning: constructing $(\mu_t, \sigma_t)$ requires using an inverse of $K_t + \lambda I$. With online Cholesky updates, this takes order $t^2$ computation per step.
2. Acquisition: finding $X_t$ requires optimising $u_t$ over $\mathcal{X}$. This can be done approximately by discretising $\mathcal{X}$ into a finite set of points $D$, making predictions for all $x \in D$ and taking the maximum. Each prediction requires an order $t^2$ vector matrix product; computing the maximum requires order $|D| \log |D|$ operations.

The overall complexity is therefore bounded by

$$CT \left( T^2 + |D| T^2 + |D| \log |D| \right),$$

where the required cardinality of the discretisation $D$ depends on the smoothness of the functions in the RKHS. Smoother functions require a finer discretisation. We now look at each of the two costs in more detail.

The cost of conditioning for Gaussian processes and that of prediction for a fixed number of locations has been studied in-depth, with many methods proposed to reduce these. In the Bayesian literature, these fall under the category of sparse input methods/variational methods (M. Seeger et al., 2003; Quinonero-Candela et al., 2007; Bauer et al., 2016), while in the kernel literature, under matrix sketching (Drineas et al., 2005; Alaoui et al.,

2015; Yang et al., 2017). Approaches from both of these fields tend to involve selecting a small number of input points such that a (potentially modified) posterior based on those points gives a good approximation to that resulting from using all of the data.

Calandriello, Carratino et al. (2019) integrate a sketching method from Calandriello, Lazaric et al. (2017a) with GP-UCB and show that the resulting algorithms retains the same regret bound (up to polylogarithmic factors) while providing provably improved runtime complexity. Calandriello, Carratino et al. (2020) combine this with ideas from Batch GP-UCB (Desautels et al., 2014) to obtain an even faster algorithm. The computational complexity of the latter is bounded as

$$C_T \leq \iota T \left( |D| d_T'^2 + d_T'^3 \right)$$

while again retaining the original regret bound up to polylogarithmic factors. This gives a near-linear-time algorithm in the case of a finite domain with smooth target functions, since then $|D|$ is a constant and $d'_T$ is polylogarithmic in $T$.

However, the acquisition cost usually dominates that of conditioning. In the original GP-UCB paper, Srinivas, Krause, S. M. Kakade et al. (2009) use a discretisation with $|D| = CT^{2d}$ to derive their results. In the case of the Matérn kernel RKHS, we can use a much coarser discretisation:

**22 Theorem.** *The worst case regret of running GP-UCB on $H_{k_\nu}$ with $0 < \nu \leq \infty$ for $T \in \mathbb{N}$ steps is increased by a factor no greater than $C_\varepsilon T^\varepsilon$ for any $\varepsilon > 0$ when using a discretisation consisting of a grid of at least $CT^{\frac{d(\nu \vee 1)}{d+2\nu}}$ evenly spaced points.*

In limit $\nu \to \infty$ this still recovers an exponential dependence on $d$. However, for any fixed $\nu > 0$ the required discretisation size is smaller than $CT^{\nu \vee 1}$.

*Proof of theorem 22.* Let $z_t \in D$ be the point selected by the discretised algorithm at time-step $t$ and $a^\star$ be a maximiser of $f$ on $D$. Then we can bound the regret incurred as

$$R_{T,A} = \sum_{t=1}^{T} f(x^\star) - f(z_t) \leq \sum_{t=1}^{T} |f(x^\star) - f(a^\star)| + \sum_{t=1}^{T} |f(a^\star) - f(z_t)|.$$

The second term is the regret incurred by running exact GP-UCB on $D$, which is bounded by $R_T$, the regret of running exact GP-UCB on $\mathcal{X}$ (maximal information gain is non-decreasing with the size of the domain). We turn to bounding the first term.

Take $b \in \arg\min_{z \in D} \|x^\star - z\|$. Then, by definition of $a^\star$,

$$|f(x^\star) - f(a^\star)| \leq |f(x^\star) - f(b)|.$$

Since $f \in H_{k_\nu}$, it satisfies the Hölder condition

$$|f(x^\star) - f(b)| \leq C_\gamma \|x^\star - b\|^\gamma$$

for $\gamma = \nu \wedge 1 - \varepsilon$ and all $0 < \varepsilon < 1$; and since the grid consists of evenly spaced points,

$$\|x^\star - b\| \leq \min\{\|z - z'\| \colon z, z' \in D\} \doteq \Delta_D.$$

and so $|f(x^\star) - f(a^\star)| \leq C_\gamma \Delta_D^\gamma$.

The overall discretised regret $R_{T,D}$ can therefore be bounded as $R_{T,D} \leq R_T + C_\gamma T \Delta_D^\gamma$. We seek a maximal $\Delta_D$ such that $R_T + CT\Delta_D^\gamma \leq C'_\varepsilon R_T T^\varepsilon$ still holds. Using the lower bound on regret in the Matérn RKHS, eq. (3.1) on page 56, it suffices that $\Delta_D \leq CT^{\frac{\nu \vee 1}{d+2\nu}}$, which requires using at most $C_d \Delta_D^d$ uniformly spaced points. □

It is important to note, however, that since we used the worst-case lower bounds from Scarlett, Bogunovic et al. (2017b) and Scarlett (2018) in our derivation and these now feature directly in the choice of discretisation for the algorithm, the algorithm is no longer adaptive. A better approach, pursued in for example Salgia et al. (2020), is to consider the discretisation scheme as a part of the algorithm, rather than applying a discretisation separately. This may allow for a discretisation size scaling better than $T^d$.

## 3.3 The SupKernelUCB algorithm

SupKernelUCB is a construction on-top of a GP-UCB-like algorithm that yields a near-optimal bound on regret for the problem we consider (Valko, Korda et al., 2013). The algorithm, described in fig. 3.2, splits the observed data into multiple subsets $\{\Psi_t^s\}$, constructed such that for any $s, t$, the observations in $\Psi_t^s \subset \mathcal{H}_t$ are independent of each other, and uses regressors of the form $\mathrm{GPR}(\,\cdot\,, \Psi_t^s)$ to construct upper confidence bounds. The independence of observations in each subset allows for the use of concentration inequalities for independent random variables (Azuma-Hoeffding), leading to much stronger and easier to derive results. Specifically, the regret of SupKernelUCB after $T$

steps is bounded as

$$R_T \leq \iota\sqrt{Td'_T},$$

giving roughly a factor $\sqrt{d'_T}$ improvement over GP-UCB. While the algorithm requires the set of arms to be finite, it can be extended to convex domains using a discretisation.

**Algorithm:** SupKernelUCB
**Parameters:** $\delta \in (0,1)$, $T \in \mathbb{N}$
**Initialisation:** let $S = \log T$ and set $\Psi_1^1 = \ldots = \Psi_1^S = \emptyset$

For each $t = 1, \ldots, T$,

1. Initialise $A_1 = \{x_1, \ldots, x_m\}$. Set $s = 1$.
2. Repeat the following until $X_t$ is chosen:
   (a) Compute $(\mu_t^{(s)}, \sigma_t^{(s)}) = \mathrm{GPR}(\mathfrak{K}, \Psi_t^s)$ and evaluate

   $$b_t^{(s)}(x) = \beta(\delta)\sigma_t^{(s)}(x) \quad \text{and} \quad u_t^{(s)}(x) = \mu_t^{(s)}(x) + b_t^{(s)}(x)$$

   for each $x \in A_s$ where $\beta(\delta) = \sqrt{2\log 2Tm/\delta}$.
   (b) If $b_t^{(s)}(x) > 2^{-s}$ for any $x \in A_s$, then let it be $X_t$, observe $Y_t$ and store the pair in $\Psi^s$, that is

   $$\Psi_{t+1}^s = \Psi_t^s \cup \{(X_t, Y_t)\},\ \Psi_{t+1}^{s'} = \Psi_t^{s'} \text{ for } s' \neq s.$$

   (c) Else if $b_t^{(s)}(x) \leq 1/\sqrt{T}$ for all $x \in A_s$ then choose $X_t$ as the maximiser of $u_t^{(s)}$ and do not store the result. That is,

   $$\Psi_{t+1}^{s'} = \Psi_t^{s'} \text{ for all } s' = 1, \ldots, S.$$

   (d) Else, set

   $$A_{s+1} = \left\{x \in A_s \colon u_t^{(s)}(x) \geq \max_{x' \in A_s} u_t^{(s)}(x') - 2 \times 2^{-s}\right\}$$

   and $s \leftarrow s + 1$.

Figure 3.2: The SupKernelUCB algorithm.

The trick to constructing the independent subsets is that whether an observation $(X_t, Y_t)$ is included in a set $\Psi_t^s$ depends only on the the confidence width $b_t^{(s)}(X_t)$, and is therefore independent of the observed value $Y_t$. The construction is based on the the work of Auer (2002), who introduces LinRel, a linear bandit algorithm, and extends it to SupLinRel. There, the extension improves the regret bound by a factor of $\sqrt{d}$. While the construction of independent sets of observations allows us to derive strong bounds for SupKernelUCB,

the constant factors involved are large, and in practice SupKernelUCB suffers near-linear empirical regret (demonstrated in table 5.1 on page 102). However, we will use a similar subsetting trick to attain tighter confidence bounds in our work.

## 3.4 Hierarchical Optimistic Optimisation

The third method we consider is Hierarchical Optimistic Optimisation (HOO, Bubeck et al., 2011). HOO is based on constructing an adaptive tree-like discretisation of the domain, an idea with roots in the classical *method of dividing rectangles* (Jones et al., 1993). Its analysis is closely related to that of Auer, Ortner et al. (2007) and Kleinberg et al. (2008). Many works have since extended the HOO algorithm, including SOO (Munos, 2011), StoSOO (Valko, Carpentier et al., 2013) and POO (Shang et al., 2019).

The assumptions made within HOO are very general. It assumes only that the target function $f$ is Lipschitz with respect to some dissimilarity (like a metric, but not requiring the point-separating property) and that this dissimilarity is known a priori. When the chosen dissimilarity is $\|x - y\|^\nu$ and the target function $f$ is $\nu$-Hölder continuous, the results for HOO translate into a worst-case bound of the form

$$R_T \leq C_\varepsilon T^{\frac{d+\nu}{d+2\nu}+\varepsilon} \quad (\forall \varepsilon > 0).$$

HOO is therefore near-minimax-optimal for the Matérn kernel RKHS when $\nu \leq 1$. However, it is unclear whether there exists a dissimilarity for which HOO recovers near-minimax-optimal regret for $\nu > 1$.

We briefly describe the algorithm specialised to our setting. First, we specify a hierarchy of dyadic covers of the domain, a set $\{A_{i,h} \subset [0,1]^d \colon i < \infty, h = 1, \ldots, 2^i\}$ such that

$$\bigcup_h A_{i,h} = [0,1]^d \quad (\forall i \in \mathbb{N}).$$

We take $A_{0,0} = [0,1]^d$, and for each level $i > 0$, we let $A_{i+1,2h}$ and $A_{i+1,2h+1}$ be given by splitting $A_{i,h}$ in half along its longest direction (with some tie breaking rule). We call $A_{i+1,2h}$ and $A_{i+1,2h+1}$ the children nodes of $A_{i,h}$. At each step $t = 1, \ldots, T$, the algorithm begins at the root node $A_{0,0}$ and recursively moves to the child with the higher upper confidence bound $U_{i,h}$ (which we define shortly). It stops when it reaches a node that it has not previously visited, say $A_{I,H}$. It then selects $X_t$ as the centre of $A_{I,H}$ and uses the observation $Y_t$ to update $U_{i,h}$ for all $(i,h)$ on the path from $(0,0)$ to $(I,H)$.

To construct the upper confidence bounds $U_{i,h}$, we first make a Lipschitz-type estimate

$$L_{i,h} = \mu_{i,h} + \sqrt{\frac{2 \log T}{n_{i,h}}} + \rho_{i,h}$$

for each $(i, h)$, where $\mu_{i,h}$ is the average value of all observed values within $A_{i,h}$ (including within subsets of $A_{i,h}$), $n_{i,h}$ is the number of those observations and $\rho_{i,h}$ is the diameter of $A_{i,h}$ with respect to the chosen dissimilarity metric. We then take $U_{i,h}$ to be the maximum of $L_{i,h}$ and the $L$-value of the children nodes of $(i, h)$ and set the $U$-value of any node that has not been previously visited to infinity. This leads the algorithm to, at each iteration, traverse from the root of the tree to the node with the most promising $L$-value, deviating if it comes across a child node that it has not previously visited.

As in SupKernelUCB, the strong theoretical guarantees of HOO arise from a clever use of subsets of independent observations. In this case, we have that for any $A_{i,j}$,

$$\frac{1}{n_{i,j}} \sum_{\tau=1}^{t} \left(f(X_\tau) - Y_\tau\right) \mathbb{1}\{X_\tau \in A_{i,j}\}$$

is a martingale difference sequence. This allows for the $n_{i,h}^{-1/2} \iota$ rate of concentration seen in the definition of the $L$-values.

Bubeck et al. (2011) show that the regret incurred by HOO is adaptive with respect to a measure of problem difficulty called the *near-optimality dimension* of $f$, which measures how quickly the pre-image of $\varepsilon$-suboptimal function values grows near the optimum with respect to $\varepsilon$ and the dissimilarity metric (this can be seen as generalising the suboptimality gaps of finite-armed bandits). However, since the near-optimality dimension depends on the dissimilarity metric, the algorithm does not adapt to the smoothness of the function. In an interesting development, Locatelli et al. (2018) shows that while it is possible to obtain rates for simple regret that are optimal with respect the best possible dissimilarity metric (as done in Shang et al. (2019)), an a priori specification of a dissimilarity metric is necessary to achieve optimal rates for regret. In particular, this means that for $\nu \leq 1$, a priori knowledge of $\nu$ is required to attain optimal regret.

## 3.5 Discussion

We have introduced three important algorithms, GP-UCB, SupKernelUCB and HOO, and examined algorithm agnostic lower bounds for the problem. Looking at the results for GP-UCB and SupKernelUCB, we see that the upper bounds for regret GP-UCB cannot match the algorithm agnostic lower bounds. Specifically, comparing the lower bounds with the upper bound from SupKernelUCB we see that for the Matérn-$\nu$ kernel,

$$\gamma_t \geq T^{\frac{d}{d+2\nu}}/\iota.$$

In section 4.4, we show that $\gamma_T \leq \iota T^{\frac{d}{d+2\nu}}$ holds for a particular choice of $\nu$ and $d$. Therefore, with the current analysis and form of upper confidence bounds, the best possible bound on the regret of GP-UCB is

$$R_T \leq \iota \gamma_T \sqrt{T} \leq \iota T^{\frac{3d+2\nu}{2d+4\nu}}.$$

This is vacuous for $\nu \leq d/2$. The same analysis implies that the regret bound for SupKernelUCB cannot be improved by more than a polylogarithmic factor. Similarly, recall that the regret of HOO is near-minimax-optimal for $\nu \leq 1$.

Moreover, GP-UCB is slow. Even with the sketching and batching techniques from Calandriello, Carratino et al. (2019) and Calandriello, Carratino et al. (2020), and combining the best-case bound for $\gamma_t$, and the discretisation from theorem 22, the complexity bound for the resulting algorithm, $C_T$, is of the form

$$C_T \leq \iota T^{\frac{d(\nu \vee 1+3)+2\nu}{d+2\nu}}.$$

This exceeds $\iota T^2$ for all $\nu, d$. In contrast, the runtime of HOO is on the order of $T \log T$.

The theoretical results for GP-UCB are therefore weaker than those for SupKernelUCB and HOO. Both the latter algorithms use upper bounds based on subsets of observations, which are carefully constructed to allow for the use of the strong Azuma-Hoeffding concentration inequality. Can we use a similar idea to construct a GP-UCB-style algorithm that combines the strengths of all three algorithms? In the next chapter, we develop the basis for a concentration inequality that will allow us to do exactly that.

# Chapter 4

# Bounding information gain for regression with the Matérn kernel

In this chapter, we bound the maximum information gain for a Matérn kernel as a function of the lengthscale kernel parameter and the diameter of the domain. In effect, we prove a quantitative version of the intuitive result that there is *less to learn on a smaller domain*. Specifically:

- In section 4.1, we present a result based on the work of Widom (1963), bounding the tail sum of eigenvalues of the Matérn kernel with respect to the kernel lengthscale (the proof is presented in appendix section 4.A). By a change of variables argument, we show that this implies a bound with respect to the diameter of the domain.
- In section 4.2, we use our result on eigenvalues to derive bounds for information gain with explicit dependence on the kernel lengthscale and diameter of the domain. We use take two distinct approaches: using the discretisation methodology of Srinivas, Krause, S. M. Kakade et al. (2009), which applies broadly; and following the approach of Vakili et al. (2021), which requires an additional assumption of uniformly bounded operator eigenfunctions.
- Then, in section 4.3, we show how our new bounds on information gain yield a tighter regret bound for GP-UCB when using a carefully chosen lengthscale parameter. This regret bound is sublinear for effectively all $\nu, d$.
- Finally, in section 4.4, we discuss the assumption of uniformly bounded eigenfunctions and establish for which combinations of $\nu$ and $d$ it is known to hold.

## 4.1 Tail sums of Matérn kernel eigenvalues

The following result on tail sums of Matérn kernel eigenvalues will be key to our bounds on the associated information gain:

**23 Theorem.** *Let $k$ be a Matérn-$\nu$ kernel on a $d$-dimensional interval of diameter $\rho \leq 1$. Write $\{\lambda_n\}$ for the eigenvalues of the kernel integral operator associated with $k$ with respect to a uniform measure. Then, for any $\varepsilon > 0$ and $n_0 > C_\varepsilon \rho^{-\varepsilon}$,*

$$\sum_{n>n_0} \lambda_n \leq C'_\varepsilon \rho^{2\nu-\varepsilon} n_0^{-\frac{2\nu}{d}}.$$

That a result of this kind ought to hold can be seen by examining the asymptotics of the spectrum of the Matérn kernel operator for a measure with Lebesgue density $V$. Specifically, let $V\colon \mathbb{R}^d \mapsto \mathbb{R}$ be a bounded, non-negative, Riemann-integrable function, $M_V$ an operator corresponding to multiplication by $V$ and $T$ the kernel integral operator associated with a Matérn kernel and the Lebesgue measure. Then Widom (1963) shows that the operator eigenvalues of $TM_V$ satisfy

$$\lambda_n \sim \pi^\nu \left(\frac{2}{d}\right)^{\frac{d+2\nu}{d}} \frac{\Gamma(d/2+\nu)}{\Gamma(-\nu)\,\Gamma(d/2)^{\frac{d+2\nu}{d}}} \left\{\int V(x)^{\frac{d}{d+2\nu}} dx\right\}^{\frac{d+2\nu}{d}} n^{-\frac{d+2\nu}{d}}.$$

Taking $V$ to be the Lebesgue density of a uniform measure on $[0,\rho]^d$ (all results in this chapter are translation invariant) and absorbing all terms that do not depend on $\rho$ or $n$ into a constant, we have

$$\lambda_n \sim C\rho^{2\nu} n^{-\frac{d+2\nu}{d}} \quad \text{and so} \quad \sum_{n>n_0} \lambda_n \leq C'\rho^{2\nu} n_0^{-\frac{2\nu}{d}} + C'' n_0^{-\frac{2\nu}{d}}.$$

The leading term on the right hand side is the result we want. However, the additional additive error term here does not have an explicit dependence on $\rho$. That it should depend on $\rho$ is clear: letting $\rho \to 0$, we expect the mass of the eigenvalues to concentrate on $\lambda_0$ and, therefore, for a fixed $n_0 > 0$, the tail sum to tend to 0. However, the proof technique of Widom (1963) does not appear to yield a way of establishing this dependence.

Our approach to proving theorem 23 takes a different route. We use the methodology of Widom (1963) to derive explicit upper bounds for the the tail sums of eigenvalues of the Matérn kernel with respect to both $n_0$ and the kernel lengthscale parameter $\ell$, while keeping $V$ (and hence $\rho$) fixed. We prove this result, stated in the following lemma,

in appendix 4.A.

**24 Lemma.** *Let $k$ be a Matérn-$\nu$ kernel on a $d$-dimensional unit interval with lengthscale $\ell \geq 1$. Write $\{\lambda_n\}$ for the eigenvalues of the kernel integral operator associated with $k$ and a uniform measure. Then, for any $\varepsilon > 0$ and $n_0 > C_\varepsilon \ell^\varepsilon$,*

$$\sum_{n > n_0} \lambda_n \leq A_\varepsilon \ell^{-2\nu+\varepsilon} n_0^{-\frac{2\nu}{d}}.$$

Theorem 23 follows by noting that a Matérn kernel $k(x, y)$ depends on the inputs $x, y$ and the lengthscale $\ell$ only through $\|x - y\|_2/\ell$, and so the effect of scaling the domain corresponds exactly to the inverse change in the lengthscale. A formal proof follows by writing this as a change of variables:

*Proof of theorem 23.* Suppose $(\lambda, \vartheta)$ satisfy

$$\lambda\vartheta(x') = \int_{[0,1]^d} k((x - x')/\rho)\vartheta(x)dx.$$

That is, $(\lambda, \vartheta)$ is an eigenpair of the integral operator associated with a Matérn kernel with lengthscale $\rho$ and a uniform measure on $[0, 1]^d$. Define $z = \rho x$, $z' = \rho x'$, then $dz/\rho^d = dx$. Making this substitution,

$$\lambda\vartheta(z'/\rho) = \int_{[0,\rho]^d} k(z - z')\vartheta(z/\rho)dz/\rho^d.$$

Define $\vartheta'(x) = \vartheta(x/\rho)$ and let $p_\rho(z) = 1/\rho^d$ denote the density of a uniform measure on $[0, \rho]^d$ with respect to the Lebesgue measure. Then,

$$\lambda\vartheta'(z') = \int_{[0,\rho]^d} k(z - z')\vartheta'(z)p_\rho(z)dz,$$

and so $(\lambda, \vartheta')$ is also an eigenpair for the operator with lengthscale 1 and a uniform measure on $[0, \rho]^d$. Thus the effect on the eigenvalues of changing lengthscale $\rho \mapsto 1$ is the same as that of changing the diameter of the domain $1 \mapsto \rho$, and the claim then follows from lemma 24 and noting the result is translation invariant. □

## 4.2 Bounds on information gain

We seek to bound the quantity $\log\det(I+\lambda^{-1}K_{XX})$ for all possible sampling locations $X = \{X_1, \dots, X_t\} \subset [0,1]^d$. Using a continuity argument, M. W. Seeger et al. (2008) show that for $\{(\lambda_n, \vartheta_n)\}$ the sequence of eigenvalues and eigenfunctions for the operator associated with a kernel $k$ and with a Gram matrix $K_{XX}$, we have

$$\log\det(I + \lambda^{-1}K_{XX}) \leq \sum_{n>0} \log\left(1 + \lambda^{-1}\lambda_n \sum_{i=1}^{t} \vartheta_n^2(X_i)\right). \tag{4.1}$$

From here, there are two standard ways of proceeding, both of which focus on eliminating the term $\sum_{i=1}^{t} \vartheta_n^2(X_i)$. One follows from a discretisation argument (Srinivas, Krause, S. M. Kakade et al., 2009), while the other uses an additional assumption of uniform boundedness of the eigenfunctions (Vakili et al., 2021). We now integrate our results on tail sums of eigenvalues with the two methods in turn.

### 4.2.1 Using a discretisation argument

The original bound by Srinivas, Krause, S. M. Kakade et al. (2009) was derived using a discretisation argument. Adapted to our notation, it reads as follows.

**25 Theorem.** *(Srinivas, Krause, S. Kakade et al., 2010, Theorem 8) Suppose that $A \subset \mathbb{R}^d$ is compact and $k$ is kernel continuously differentiable in a neighbourhood of $A$. Write $\{\lambda_n\}$ for the eigenvalues of $k$ with respect to the uniform distribution over $A$. Pick $\zeta > 0$ and let $F_t = (4\zeta+2)\mathcal{V}_A t^\zeta \log t$, where $\mathcal{V}_A$ is the volume of $A$. Then the information gain on $A$ after observing $t$ points is bounded by*

$$C \max_{r=1,\dots,t} \left[ n_0 \log\frac{rF_t}{\lambda} + (4\zeta+2)\,\mathcal{V}_A \log\frac{t}{\lambda}\left(1-\frac{r}{t}\right)\left(t^{\zeta+1}\sum_{n>n_0}\lambda_n + 1\right)\right] + C' t^{1-\zeta/d}, \tag{4.2}$$

*for any $n_0 \in \mathbb{N} \cap [1, F_t]$.*

The proof of the theorem is lengthy. Very briefly:

1. For points $X_1, \dots, X_t$ sampled from a uniform measure $\mu$ on the domain, applying Jensen's inequality and noting that $\mathbf{E}\vartheta_n^2(X_i) = 1$ for all $n$ by orthonormality of eigenfunctions, and then using the linear bound $\log(1+x) \leq x$ for $x \geq 0$,

$$\mathbf{E}\sum_{n>n_0} \log\left(1 + \lambda^{-1}\lambda_n \sum_{i=1}^{t} \vartheta_n^2(X_i)\right) \leq \sum_{n>n_0} \log\left(1 + \lambda^{-1}t\lambda_n\right) \leq \lambda^{-1}t \sum_{n>n_0} \lambda_n. \tag{4.3}$$

This bound on the tail of the expected information gain was established in M. W. Seeger et al. (2008), and eventually contributes to the second term in eq. (4.2); the first term is related to a maximal bound on the head of this sum.

2. The bound now depends on the maximum of the tail sum of the eigenvalues $\{\lambda_n\}$ with respect to an operator with a uniform measure. This is bounded by the maximum tail sum of eigenvalues on a regular discretisation of the domain, making use of use Lipschitz continuity, which follows from the assumption that $k$ is continuously differentiable. This bound is responsible for the $C't^{1-\zeta/d}$ additive term at the end of eq. (4.2), where $\zeta > 0$ controls how finely we discretise.

3. Finally, we compute a sequential greedy approximation of the maximum of the tail sum on the discretisation and use the submodularity of information gain to bound the true maximum in terms of the greedy approximation. This contributes at most a constant factor, the $C$ at the front of eq. (4.2).

Helpfully, because of the use of a discretisation argument, the result and its proof carefully track dependencies of the bound on the volume of the domain. We can therefore straightforwardly incorporate our new bounds:

**26 Theorem.** *Pick $t \in \mathbb{N}$ and let $A \subset \mathbb{R}^d$ be an interval of diameter $\rho$ satisfying $t^{-a} \leq \rho \leq t^{-b}$ for some $0 < b \leq a < \infty$. Then $\zeta > 0$ chosen independently of $t$, the information gain on $A$ for a conjugate Gaussian process regressor equipped with a Matérn-$\nu$ kernel with $\nu > 1$ is bounded as*

$$\gamma_t^A \leq C_\varepsilon t^\varepsilon \left( n_0 + n_0^{-\frac{2\nu}{d}} t^{\zeta+1-b(d+2\nu)} + t^{1-\zeta/d} \right) \quad (\forall \varepsilon > 0, \forall n_0 > C'_\varepsilon t^\varepsilon).$$

*In particular, if $b \geq \frac{d+1}{d+2\nu}$, choosing $\zeta = d$ we obtain $\gamma_t^A \leq C'_\varepsilon t^\varepsilon$ for all $\varepsilon > 0$.*

*Proof.* First, take the maximum for both terms in eq. (4.2) individually, substitute $\mathcal{V}_A = \rho^d$ and bound all constants and logarithmic terms with $C_\varepsilon t^\varepsilon$. With that, we have

$$\gamma_t^A \leq C_\varepsilon t^\varepsilon \left( n_0 + \rho^d t^{\zeta+1} \sum_{n>n_0} \lambda_n + t^{1-\zeta/d} \right).$$

Now use $\sum_{n>n_0} \lambda_n \leq A_\varepsilon \rho^{2\nu-\varepsilon} n_0^{-\frac{2\nu}{d}}$ from theorem 23, then $t^{-a} \leq \rho \leq t^{-b}$ and relabel $a\varepsilon \mapsto \varepsilon$. □

We have some remarks on the applicability of this result:

1. The proof of theorem 25 requires Lipschitz continuity and therefore that $\nu > 1$. However, the proof could be relaxed to use Hölder continuity, and with that extended to the case $\nu \leq 1$. While the exact form of the result would be different for $\nu \leq 1$—in particular, $\nu$ would feature in the exponent of the additive discretisation term—following through we would find that for all $\nu > 0$ there exists a $b < \infty$ such that $\rho \leq t^{-b} \implies \gamma_t^A \leq C_\varepsilon t^\varepsilon$. This weaker statement suffices for the results on regret of the hierarchical algorithm presented in chapter 5 (see theorem 42).

2. Theorem 25 is stated directly in terms of volumes rather than lengthscales. For that reason, the proof of bounds on information gain with respect to domain diameter, presented in theorem 26, is very straightforward. On the other hand, a formal proof that an equivalent result holds for lengthscale would require replicating much of the original proof of theorem 25. Due to this, we forgo presenting a formal proof; it is the result on diameter and not lengthscale that will form part of our core contributions in chapter 5.

3. Theorem 25 as stated in Srinivas, Krause, S. Kakade et al. (2010) assumes strong conditions on the sample paths of the Gaussian process associated with the kernel $k$. These are not necessary for the proof of the theorem itself and are there only for the benefit of a later application of the result to Bayesian optimisation.

### 4.2.2 Using assumption of uniformly bounded eigenfunctions

A recent result by Vakili et al. (2021) provides a cleaner, stronger information gain bound for kernels with uniformly bounded eigenfunctions.

**27 Theorem** (Vakili et al. (2021), Theorem 3)**.** *Let $k$ be a continuous kernel on $\mathcal{X}$ with uniformly bounded eigenfunctions and $\lambda_1 \geq \lambda_2 \geq \dots$ the associated operator eigenvalues with respect to a finite Borel measure. Then there exists a constant $C > 0$ such that for all $n_0, t \in \mathbb{N}$ and $n_0 < t$,*

$$\gamma_t \leq C \left( n_0 \log t + t \sum_{n > n_0} \lambda_n \right).$$

Comparing the bound in theorem 27 with eq. (4.3) on page 70, the result implies that for kernels satisfying the uniform boundedness assumption, maximal information gain is of the same order as the expected information gain for uniform covariates.

We present a simple proof of theorem 27. The full proof in Vakili et al. (2021) derives a better dependence on some of the constants than that sketched here.

*Proof.* Using eq. (4.1) on page 70 and a linear bound $\log(1+x) \leq x$ for $x \geq 0$ for the tail eigenvalues, we have

$$\gamma_t \leq \frac{1}{2} \sum_{n \leq n_0} \log \left( 1 + \lambda^{-1} \lambda_n \sum_{i=1}^{t} \vartheta_n^2(X_i) \right) + \frac{1}{2} \lambda^{-1} \sum_{n > n_0} \lambda_n \sum_{i=1}^{t} \vartheta_n^2(X_i).$$

By assumption, there exists an $A_1 > 0$ such that $\vartheta_n^2(x) \leq A$ for all $n$ and all $x \in \mathcal{X}$. Moreover, since the integral operator associated with $k$ is trace class,

$$\lambda_n \leq \sum_{n>0} \lambda_n = \int_{\mathcal{X}} k(x,x) dx \leq \sup_{x \in \mathcal{X}} k(x,x) \leq A_2$$

for some $A_2 > 0$ dependent on $k$. And so,

$$\gamma_t \leq \frac{n_0}{2} \log(1 + \lambda^{-1} t A_1 A_2) + \frac{1}{2} \lambda^{-1} t A_1 \sum_{n > n_0} \lambda_n,$$

as claimed. □

Vakili et al. claim that this closes the gap between the lower and upper bounds on information gain for the Matérn kernel, proposing a bound of

$$\gamma_t \leq C t^{\frac{d}{d+2\nu}} \log^{\frac{2\nu}{2\nu+d}} t$$

for all $d \geq 1$ and $\nu > 1/2$. We question this specific application of the result in section 4.4, but for now proceed to combine it with our new bounds on the eigenvalues. To combine theorem 27 with changing lengthscale or the diameter of the domain, we will need a stronger form of the uniform boundedness assumption:

**28 Definition.** *Let $k$ be a kernel. We say the eigenfunctions of $k$ are uniformly bounded independently of scale if there exists an $A_k > 0$ such that for all $a \in (0,1]$ the operator eigenfunctions of the kernel $k(ax, ay)$ are uniformly bounded by $A_k$.*

By examining the change of variables used in the proof of theorem 23, it is immediate that definition 28 is sufficient both in cases of changing lengthscale and diameter. Indeed, we could even relax the assumption to allow for a polylogarithmic dependence of $A_k$ on $a$. Either way:

**29 Theorem.** *Let $k$ be a Matérn-$\nu$ kernel on $A \subset \mathbb{R}^d$, an interval of diameter $\rho$. Assume that the eigenvalues of $k$ are uniformly bounded independently of scale (definition 28).*

*Then for all $t \in N$ and $\rho \leq t^{-b}$ the information gain on $A$ for a conjugate Gaussian process regressor with kernel $k$ is bounded as*

$$\gamma_t^A \leq C_\varepsilon t^\varepsilon \left( n_0 + t^{1-2\nu b} n_0^{-\frac{2\nu}{d}} \right) \quad (\forall \varepsilon > 0,\ \forall n_0 > C'_\varepsilon t^\varepsilon).$$

*In particular, for $b \geq \frac{1}{2\nu}$, $\gamma_t^A \leq C'_\varepsilon t^\varepsilon$ for all $\varepsilon > 0$.*

This result and an equivalent for the lengthscale dependence are immediate. It remains to check that the additional assumption (definition 28) holds for particular values of $\nu$ and $d$. We look at this in section 4.4, following a brief diversion.

## 4.3 Improved regret bounds for GP-UCB

Our bound on information gain with respect to kernel lengthscale can be used to improve the results on the regret of GP-UCB, when using a lengthscale chosen as a function of the horizon. Recall that the regret incurred by GP-UCB can be written in the form

$$R_T \leq C\sqrt{T\gamma_T}(\|f\|_{k_\nu} + \sqrt{\gamma_T}),$$

where we include a direct dependence on the RKHS norm $\|f\|_{k_\nu}$ rather than on its upper bound $\mathfrak{R}$. From theorems 26 and 29 the information gain term can be reduced by increasing lengthscale. On the other hand, this increases $\|f\|_{k_\nu}$:

**30 Lemma.** *Let $k_\nu^1$ and $k_\nu^\ell$ be Matérn-$\nu$ kernels with lengthscales 1 and $\ell$ respectively. Then for any $f \in H_{k_\nu^1}([0,1]^d)$, $\|f\|_{k_\nu^\ell} \leq \ell^\nu \|f\|_{k_\nu^1}$.*

*Proof.* Denote by $S^1$ and $S^\ell$ the respective spectral densities. We have,

$$S^\ell(\omega) = C\ell^d(C' + \ell^2|\omega|^2)^{-2\nu-d} \geq C\ell^{-2\nu}(C' + |\omega|^2)^{-2\nu-d} = \ell^{-2\nu} S^1(\omega),$$

and therefore we can bound the change in norm as

$$\|f\|_{k_\nu^\ell}^2 = \int \frac{(\mathcal{F}Ef)^2(\omega)}{S^\ell(\omega)} d\omega \leq \ell^{2\nu} \int \frac{(\mathcal{F}Ef)^2(\omega)}{S^1(\omega)} d\omega = \ell^{2\nu} \|f\|_{k_\nu^1}^2,$$

where $E\colon [0,1]^d \mapsto \mathbb{R}^d$ is a suitable extension operator, per theorem 5 on page 39. □

By selecting $\ell$ as a function of $T$ such that $\|f\|_{k_\nu^\ell}$ and $\sqrt{\gamma_T}$ are asymptotically equivalent

up to constant factors, we obtain the following stronger bound.[1]

**31 Theorem.** *Under the usual assumptions, GP-UCB with a Matérn-$\nu$ kernel, $\nu > 1$, and*

$$\ell = T^b \quad \textit{for} \quad b = \frac{d(d+1)}{d^2(2\nu+1) + 4\nu(d+\nu)}$$

*has regret bounded as*

$$R_T \leq C_\varepsilon T^{y+\varepsilon} \quad \textit{with} \quad y = \frac{d^2(6+1/\nu) + 8d + 4\nu}{d^2(4+2/\nu) + 8d + 8\nu}$$

*for all $\varepsilon > 0$.*

This bound is sublinear for all $\nu$ when $d = 1$ and for all $\nu > 1/4d^2 + d\sqrt{d^2-4}$ for $d \geq 2$. This is an improvement over the previous results, which required $\nu > d^2/2$ for all $d \geq 1$ (Srinivas, Krause, S. M. Kakade et al., 2009; Chowdhury et al., 2017).

Using the additional assumption of uniformly bounded eigenfunctions, we obtain the following result. Proofs of this and the previous result are included in appendix 4.B.

**32 Theorem.** *For any $\nu > 0$ such that $k_\nu$ has eigenfunctions that are uniformly bounded independently of scale (definition 28), setting $\ell = T^b$ for $b = \frac{d}{4\nu(d+\nu)}$, the regret incurred by GP-UCB can be bounded as*

$$R_T \leq C_\varepsilon T^{\frac{2d+\nu}{2d+2\nu}+\varepsilon} \quad (\forall \varepsilon > 0).$$

This result gives sublinear regret for all $\nu, d$ for which the stronger uniform boundedness condition holds. It is not, however, near-minimax-optimal.

## 4.4 On uniformly bounded eigenfunctions

The uniform boundedness assumption made in Vakili et al. (2021) is very strong. D.-X. Zhou (2002) constructs an example showing that neither compact support nor smoothness conditions allow us to infer uniform boundedness of eigenfunctions (see also the discussion in section 3 of Minh et al. (2006)). Here, we outline a general derivation of the eigenfunctions for the Matérn kernel and discuss the assumption.

[1] Strictly speaking, theorem 31 requires proving that the expected relationship between lengthscale and information gain holds under the proof technique of Srinivas, Krause, S. M. Kakade et al. (2009). That it does is clear. However, due to the length of the original proof, we have not shown this formally.

The eigenfunctions associated with a kernel are the eigenfunctions of the Fredholm homogenous equation of the second kind, given in the context of the relevant literature on integral equations in the form

$$\vartheta(x) = \lambda' \int_{[0,1]^d} k(x,y)\vartheta(x)dy.$$

It is one of the most well studied integral equations. In the case where the kernel $k$ is stationary and its spectral density can be written as a ratio of two rational polynomials, that is $S(\omega) = \frac{N(\omega)}{D(\omega)}$ for polynomials $N, D$, the general solution is given by

$$D(\nabla^2)\vartheta(x) = \lambda' N(\nabla^2)\vartheta(x),$$

where $\nabla^2$ denotes the Laplace operator and its exponentiation corresponds to repeated application of the operator (Le Maître et al., 2010, chapter 2). In particular, for a Matérn-$\nu$ kernel in $d$-dimensions, the eigenfunctions $\vartheta$ satisfy

$$(\nabla^{2\nu+d}\vartheta)(x) = \lambda'\vartheta(x). \tag{4.4}$$

From this, it is clear that the eigenfunctions are of the form

$$\vartheta(x) = \sum_{r \leq R} A_r e^{-B_r x} \cos(C_r x + D_r), \tag{4.5}$$

where $R$ depends on $\nu, d$ and the constants $A_r, B_r, C_r, D_r > 0$; these constants need to be solved for using boundary conditions.

Vakili et al. (2021) cites Riutort-Mayol et al. (2020) for uniform boundedness of the Matérn kernel eigenfunctions. There, the authors solve eq. (4.4) under the homogenous Dirichlet boundary conditions,

$$\vartheta(x) = 0 \text{ for all } x \in \partial[0,1]^d. \tag{4.6}$$

These lead to harmonic sinusoidal eigenfunctions. However, any RKHS with a basis of functions that satisfy eq. (4.6) contains only functions $f$ that satisfy $f(x) = 0$ for all $x \in \partial[0,1]^d$, and therefore only a strict subset of the Matérn kernel RKHS. While using the solution with homogenous Dirichlet boundary conditions to approximate stationary kernels has been studied formally in Solin et al. (2020), it is not shown or implied that the uniform boundedness of eigenfunctions under this approximation implies the uniform

boundedness of eigenfunctions for the true kernel. If it were so, it would immediately imply that all stationary kernels with compact support have uniformly bounded eigenfunctions.

From theorem 12 on page 42, it is apparent that the correct definition of a Matérn kernel on $[0,1]^d$ is given by the restriction of the kernel defined on $\mathbb{R}^d$. The boundary conditions to recover the restricted kernel then need to capture that each function $f$ in the RKHS of the kernel on $[0,1]^d$ is the restriction of a suitable function on $\mathbb{R}^d$. This, in effect, introduces a discontinuity at the boundary $\partial[0,1]^d$. A method for solving for the eigenfunctions under such boundary conditions is considered for the case $d = 1$ in Youla (1957). There, the author extends the ratio of polynomials defining the spectrum of the kernel to the complex plane and treats the boundary as a removable singularity (the Weiner-Hopf method). The solution for the coefficients defining the eigenfunctions (those in eq. (4.5)) are then given as the solution to a set of transcendental equations, which grow in complexity with the degree of the polynomials $N, D$. For the simplest case where the degree of $N$ is 0 and $D$ is 1, which corresponds to the Matérn-1/2 kernel, Youla (1957) finds that the eigenfunctions take the form

$$\vartheta(x) = A_1 \cos(C_1 x + D_1). \tag{4.7}$$

Given the orthonormalisation constraint in the definition of eigenfunctions, these are therefore clearly uniformly bounded independently of scale (and thus also uniformly bounded). For higher values of $\nu$, the solution becomes significantly more complicated and the properties of the eigenfunctions more difficult to infer. Moreover, extending the Weiner-Hopf method to $d > 1$ is very unpleasant. However, since for all values of $\nu$ and $d$, the constants $A_r, B_r, C_r, D_r$ are given by the solution of a transcendental equation, they do not correspond to harmonic frequencies on $[0,1]^d$ and thus differ from the result obtained under the Dirichlet boundary conditions.

It may be tempting to extend the result for the one-dimensional Matérn-1/2 kernel case to $d > 1$ by taking a $d$-product of one dimensional Matérn-1/2 kernels. This is sometimes taken as the definition of the $d$-dimensional Matérn kernel in the literature, for example in Mutný et al. (2019) and Riutort-Mayol et al. (2020). The two definitions are not, however, equivalent. Using the simplified spectral density $\mathcal{F}k_{\nu,d} = (1 + \|\omega\|_2^2)^{-\nu-d/2}$ for the Matérn-$\nu$ kernel and considering points on the line $\omega_1 = \ldots = \omega_d$, for all $d > 1$,

$$\mathcal{F}\prod_{i=1}^{d} k_{\nu,1} = \prod_{i=1}^{d} \mathcal{F}k_{\nu,1} = \left(1 + d\|\omega\|^2\right)^{-d\nu-d/2} < C_d \left(1 + \|\omega\|^2\right)^{-\nu-d/2} = C_d \mathcal{F}k_{\nu,d}.$$

We see therefore that the $d$-product kernel has a much faster rate of decay, and so higher smoothness of functions in the corresponding RKHS, than the correctly defined $d$-dimensional Matérn kernel. A formal analysis of the $d$-product kernel is given in Ritter et al. (1995).

# Appendix 4.A Proof of results on the tail sums of Matérn kernel eigenvalues

This appendix develops the results necessary to prove lemma 24 on page 69, with the proof itself contained within section 4.A.6. The intermediate results are based heavily on the structure and proof techniques of Widom (1963), with our labour being in establishing explicit, quantitative bounds in place of results on asymptotic equivalences.

## 4.A.1 A symmetrised integral operator

Widom's results are for a symmetrised version of the integral operator: that of the form $M_{V^{1/2}} T M_{V^{1/2}}$. Before proceeding, we establish that, for the case where $V$ corresponds to the Lebesgue density of a measure with support equal to the domain of the kernel, the eigenvalues of the symmetrised operator are equal to those of $TM_V$. Henceforth we work with exclusively with the symmetrised operator.

**33 Lemma.** *For $\mathcal{X}$ a compact set, let $k\colon \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ be a kernel and $T : L^2(\mathcal{X}) \mapsto L^2(\mathcal{X})$ the associated integral operator with respect to the Lebesgue measure on $\mathcal{X}$. Let $V\colon \mathcal{X} \mapsto \mathbb{R}$ be a bounded positive function and denote by $M_V$ the operator corresponding to a multiplication by $V$. Then both $M_{V^{1/2}} T M_{V^{1/2}}$ and $TM_V$ are maps $L^2(\mathcal{X}) \mapsto L^2(\mathcal{X})$ and these have the same eigenvalues.*

In the proof of the lemma, we make use the multiplicative inverse operator $M_{V^{-1/2}}$. While this is not a bounded operator, it only ever appears in conjunction with its inverse $M_{V^{1/2}}$.

*Proof.* The integrability results follow trivially from the boundedness of $V$. For the result on eigenvalues, we prove the two inclusions in turn:

1. Suppose $(\vartheta, \lambda)$ satisfy $M_{V^{1/2}} T M_{V^{1/2}} \vartheta = \lambda \vartheta$ and take $\vartheta'$ such that $M_{V^{1/2}} \vartheta' = \vartheta$. Then
$$M_{V^{1/2}} T M_V \vartheta' = M_{V^{1/2}} \lambda \vartheta'.$$
Left-applying $M_{V^{-1/2}}$ on both sides we see that $\lambda$ is an eigenvalue of $TM_V$.

2. Suppose $(\vartheta, \lambda)$ satisfy $TM_V \vartheta = \lambda \vartheta$. Let $\vartheta' = M_{V^{1/2}} \vartheta$. Then
$$T M_{V^{1/2}} \vartheta' = \lambda M_{V^{-1/2}} M_{V^{1/2}} \vartheta'$$
Left-applying $M_{V^{1/2}}$ on both sides, we have that $\lambda$ is an eigenvalue of $M_{V^{1/2}} T M_{V^{1/2}}$.

Therefore the eigenvalues of $M_{V^{1/2}} T M_{V^{1/2}}$ and $T M_V$ are equal. □

### 4.A.2 Notation and assumptions (I)–(III)

We will write $k_p$ for a periodic kernel on $I_d = [-\pi, \pi]^d$, $T_p$ for the associated integral operator and $\{c_n \colon n \in \mathbb{N}^d\}$ for the multiset of the corresponding Fourier coefficients. We refer to $n \in \mathbb{N}^d$ as lattice points. For $N \subset \mathbb{N}^d$, we write $|N|$ for the cardinality of $N$; for $A \subset \mathbb{R}^d$, $|A|$ denotes its Lebesgue measure. For $\varepsilon > 0$, we write $\{c_n > \varepsilon\}$ as shorthand for $\{n \colon c_n > \varepsilon\}$ and denote by $\Psi_c(\varepsilon) = |\{n \colon c_n > \varepsilon\}|$.

Where specified, the coefficients $\{c_n\}$ satisfy:

(I) $c_n \geq 0$ for all $n \in \mathbb{Z}^d$,

(II) $\lim_{\alpha \to \infty} c_{\alpha n} = 0$ for $n \neq 0$, and

(III) there exists an $R$ such that for all $n$ with $\|n\| > R$,

$$\|n'\| > \|n\| + 1 \implies c_{n'} < c_n.$$

We will refer to these assumptions by their corresponding Roman numerals.

For a trace class operator $T$, we write $N^+(\varepsilon, T)$ and $N^-(\varepsilon, T)$ for the number of eigenvalues of $T$ which are greater than $\varepsilon$ and smaller than $-\varepsilon$ respectively. For $\Omega \subset \mathbb{R}^d$, we write $P_\Omega$ for the operator corresponding to multiplication by the characteristic function of $\Omega$.

### 4.A.3 A lemma by Widom, modified

Our bounds for tail eigenvalues of the Matérn kernel will require a modified version of the Main Lemma from Widom (1963). For a periodic kernel, the lemma relates the eigenvalues of its integral operator restricted to a subset of the domain to the Fourier coefficients of the kernel. With the established notation:

**34 Lemma** (Widom (1963), Main Lemma, modified)**.** *Let $\Omega_1$ and $\Omega_2$ be non-overlapping intervals contained in $I_d$. Then if $\{c_n\}$ satisfies (I)–(III) we have that for a constant $A > 0$ and all $\varepsilon < \max\{c_{n_0} \colon \|n_0\| > R + 3\}$,*

$$N^{\pm}(\varepsilon, P_{\Omega_1} T_p P_{\Omega_2} + P_{\Omega_2} T_p P_{\Omega_1}) \leq AR\Psi_c(\varepsilon).$$

Lemma 34 differs from Widom's Main Lemma in that it provides an upper bound with an

explicit dependence on $R$; Widom's result was purely asymptotic in nature and assumed $R$ to be a constant. This will be key for our application.

Our modification can be isolated to a single intermediate result, lemma 35. We now state and prove the modified intermediate result and then sketch how to propagate the change through the proof of Main Lemma. For the rest of this subsection, assume any coefficients $\{c_n\}$ satisfy (I)-(III).

**35 Lemma** (Widom (1963), Lemma 2, modified)**.** *For $R$ sufficiently large, there exists a constant $A_2 > 0$ such that the set $\{n\colon c_n > \varepsilon\}$ is contained within a ball of volume $A_2 R^d \Psi_c(\varepsilon)$ for all $\varepsilon > 0$.*

The change from Lemma 2 as stated by Widom is the introduction of an explicit dependence on the radius $R$ from (III), which was previously assumed to be constant.

*Proof.* Let

$$r_0 = \max\{r\colon \exists n \text{ with } \|n\| = r \text{ and } c_n > \varepsilon\}$$

and associate with $r_0$ a point $n_0$ satisfying $\|n_0\| = r_0$.

We consider two cases:

1. Case $r_0 > R + 1$. Suppose that there exists a point $b$ such that

$$\|b\| + 1 < r_0 \quad \text{and} \quad c_b < \varepsilon.$$

   By assumption III, $r_0 > \|b\| + 1 \implies c_{n_0} < c_b$ and therefore $c_{n_0} < \varepsilon$. But by definition of $r_0$ and $n_0$, $c_{n_0} > \varepsilon$. Hence no point satisfying the definition of $b$ exists. And so we have that for all $b \in \mathbb{Z}^d$ with $\|b\| + 1 < r_0$, $c_b > \varepsilon$. Now take

$$r_1 = \max\{r\colon \exists b \text{ with } \|b\| = r \text{ and } r + 1 < r_0\}.$$

   Since for any $x > 0$ there exists an $n \in \mathbb{Z}^d$ with $\|n\| \in [x, x+1)$, we have that $r_1 \geq r_0 - 2$. From the definition of $r_0$, $\{n\colon c_n > \varepsilon\} \subset B(r_0)$. Since $r_1 \geq 1$, we have $r_1 \geq r_0/3$, and so $|B(r_0)| \leq C|B(r_1)|$. Also, from the definition of $r_1$, for any $n \in B(r_1)$, $c_n > \varepsilon$, and so

$$B(r_1) \cap \mathbb{Z}^d \subset \{n\colon c_n > \varepsilon\},$$

   leading to $|B(r_0)| \leq C|B(r_1)| \leq C_1|\{n\colon c_n > \varepsilon\}|$.

2. Case $r_0 \leq R+1$. Then $r_1 \leq r_0 \leq R+1$, and so $|B(r_1)| \leq C_2 R^d$.

Defining $A_2 = (C_1 \vee 1)C_2$ we obtain the result. □

We will also need the following lemma used in the proof of the original statement.

**36 Lemma** (Widom (1963), Lemma 1)**.** *Assume that* $\{n\colon c_n \neq 0\} \subset B(r)$ *with* $r > 2$ *Then*

$$\int_{I_d} \|x\| \left|\sum_n c_n e^{i\langle n,x\rangle}\right|^2 dx \leq A_1 r^{d-1} \log r \max_n c_n^2,$$

*where* $A_1$ *depends on* $d$ *only.*

With that, we now sketch the proof of the modified main lemma.

*Sketch proof of lemma 34.* Widom's approach to proving the main lemma is to decompose $T_p$ into a sum $T_1 + T_2 + T_3$, with each operator in turn given by the following Fourier coefficients:

$$c_{n,1} = \begin{cases} c_n & \text{if } c_n \leq \varepsilon \\ \varepsilon & \text{otherwise} \end{cases}, \tag{4.8}$$

$$c_{n,2} = \begin{cases} c_n - \delta & \text{if } c_n > \delta \leq \varepsilon \\ 0 & \text{otherwise} \end{cases}, \tag{4.9}$$

$$c_{n,3} = \begin{cases} c_n - \varepsilon & \text{if } \varepsilon < c_n \leq \delta \\ \delta - \varepsilon & \text{if } c_n > \delta \\ 0 & \text{if } c_n \leq \varepsilon \end{cases}, \tag{4.10}$$

for $\delta, \varepsilon > 0$ with $\delta < \varepsilon$. $T_1$ has norm bounded by $\varepsilon$, and $T_2$ has rank at most $\Psi_c(\delta)$, making these simple to bound. The crux of the argument is then bounding $T_3$.

Widom shows that the number of eigenvalues of $P_{\Omega_1} T_3 P_{\Omega_2} + P_{\Omega_2} T_3 P_{\Omega_1}$ that exceed $\varepsilon > 0$, which we denote $N_3^+(\varepsilon)$, can be bounded as

$$\varepsilon^2 N_3^+(\varepsilon) \leq A \int_{I_d} \|x\| \left|\sum_n c_{n,3} e^{i\langle n,x\rangle}\right|^2 dx.$$

Now by lemma 36, we can bound this integral by $A_1 r^{d-1} \log r \max_n c_n^2$, where $r > 2$ is to be chosen such that $\{c_{n,3} \neq 0\} \subset \mathcal{B}(r)$. Since $\{c_{n,3} \neq 0\} = \{c_n > \varepsilon\}$, lemma 35 shows that we can choose $r$ to be the radius of a sphere of volume $A_2 R^d \Psi_c(\varepsilon)$, whenever that

quantity is greater than 2. From (III), this is guaranteed to hold for all $\varepsilon < c_{n_0}$ with $n_0$ satisfying $\|n_0\| > R + 3$. Widom completes his proof by showing that the $N_3^+(\varepsilon)$ term dominates the other two in terms of dependence on $\varepsilon$. A tedious but trivial modification of that argument confirms the same holds for the dependence on $R$. □

### 4.A.4 Bounding tail sums of eigenvalues for a restriction of a periodic kernel

We now use the modified main lemma to bound the tail sum of the eigenvalues for the restriction of a periodic kernel $k_p$, in terms of the tail sum of the Fourier coefficients of $k_p$. To see how and why this will work, consider that we can write any stationary kernel on a compact domain as the restriction of a periodic kernel on a larger domain, and that the eigenvalues of a periodic kernel are precisely its Fourier coefficients. This establishes the required link.

We begin with a technical lemma.

**37 Lemma.** *Let $A > 0$ be a constant and $\{h_i > 0\}$ a non-increasing sequence satisfying $|\{h_i > \varepsilon\}| \leq A\Psi_c(\varepsilon)$ for all $\varepsilon < \max\{c_{n_0} \colon \|n_0\| > R + 3\}$ and*

$$m(x) = \min\{c_n \colon \lceil x \rceil \leq \|n\| < \lceil x \rceil + 1\}.$$

*Then, for all $i_0 > R$,*

$$\sum_{i > i_0} h_i \leq C_d(A+1) \sum_{i > i_0} m\Big(i^{1/d}\Big).$$

*Proof.* Some definitions:

- Let $g = \{g_j\}$ be a nonincreasing sequence and denote $\Psi_g(\varepsilon) = |\{g_j > \varepsilon\}|$.
- For a fixed $j_0 \in \mathbb{N}$, let $\mathcal{A}$ be the set of sequences such that for all $a \in \mathcal{A}$, $|\{a_i > \varepsilon\}| \leq A\Psi_g(\varepsilon)$ for all $\varepsilon < g_{j_0}$.

We begin by constructing a sequence $\beta$ that is an elementwise upper bound on sequences in $\mathcal{A}$. Let $M_j = \lfloor A\Psi_g(g_{j+1}) \rfloor - \lfloor A\Psi_g(g_j) \rfloor$ and take

$$\beta = \{\underbrace{g_0, \ldots, g_0}_{M_0 \text{ terms}}, \underbrace{g_1, \ldots, g_1}_{M_1 \text{ terms}}, \ \ldots, \ \underbrace{g_j, \ldots, g_j}_{M_j \text{ terms}}, \ \ldots\}.$$

Then $\beta$ has the following properties:

1. For all $\beta' \in \mathcal{A}$ and all $i \in \mathbb{N}$, $\beta_i \geq \beta'_i$. Suppose otherwise: then there exists a $\beta' \in \mathcal{A}$ and $i' \in \mathbb{N}$ such that $\beta'_{i'} > \beta_{i'}$. Let $j' = \max\{j \colon \beta_{i'} = g_j\}$. Then

$$|\{\beta'_i > g_{j'}\}| \geq |\{\beta_i > g_{j'}\}| + 1 = \lfloor A\Psi_g(g_{j'})\rfloor + 1 \geq A\Psi_g(g_{j'}),$$

   and therefore $\beta' \notin \mathcal{A}$.

2. For all intervals of the form $[g_j, g_{j'})$,

$$|\beta \cap [g_j, g_{j'})| \leq (A+1)|\{g_i\} \cap [g_j, g_{j'})|.$$

   This follows from bounding

$$|\beta \cap [g_j, g_{j'})| \leq \lfloor A\Psi_g(g_j)\rfloor - \lfloor A\Psi_g(g_{j'})\rfloor \leq A(\Psi_g(g_j) - \Psi_g(g_{j'})) + 1,$$

   and using $1 \leq \Psi_g(g_j) - \Psi_g(g_{j'}) = |\{g_i\} \cap [g_j, g_{j'})|$ for any non-empty interval.

With that construction in place, we identify $\{g_j\}$ with a nonincreasing ordering of $\{c_n \colon n \in \mathbb{N}^d\}$. Thus $\Psi_g = \Psi_c$, and so by property 1 of $\beta$, $\sum_{i>i_0} h_i \leq \sum_{i>i_0} \beta_i$ for all $i_0 \geq j_0$ where $j_0$ is such that $g_{j_0} = c_{n_0}$. We now proceed to upper bound $\sum_{i>i_0} \beta_i$.

Let $a(j) = \min\{c_n \colon j \leq \|n\| < j+1\}$ and $b(j) = \max\{c_n \colon j \leq \|n\| < j+1\}$ and define $D_j = [a(j+1), a(j))$. We bound the tail sum of $\beta$ using $\{D_j\}$ as a partition of its values:

$$\sum_{i>i_0} \beta_i \leq \sum_{j>j'_0} |\beta \cap D_j| \max\{\beta_i \colon \beta_i \in \beta \cap D_j\} \leq \sum_{j>j'_0} |\beta \cap D_j|\, a(j)$$

for $j'_0$ such that $\beta_{i_0+1} \in D_{j'_0+1}$, that is $j'_0 = \min\{j \colon \beta_{i_0+1} \leq a(j+2)\}$. Now observe that the intervals $D_j$ are of the form considered in property 2 of $\beta$. Therefore, by 2, $|\beta \cap D_j| \leq (A+1)|\{c_n\} \cap D_j|$ for all $j$ sufficiently large. This with the previously derived inequalities yield

$$\sum_{i>i_0} h_i \leq \sum_{i>i_0} \beta_i \leq \sum_{j>j'_0} |\beta \cap D_j|\, a(j) \leq (A+1) \sum_{j>j'_0} |\{c_n\} \cap D_j|\, a(j),$$

and so our bound now depends on $\{c_n\}$ only.

We now bound $|\{c_n\} \cap A_j|$. Consider the two constraints in turn:

1. Since $c_n \geq a(j+1)$, applying (III) twice, we have that $b(j+2) < a(j) < c_n$, and therefore $\|n\| < j+3$.

2. Also, $c_n < a(j)$. By (III), $a(j) < a(j-2)$, and therefore $\|n\| > j-2$.

Hence $c_n \cap A_j \subset \mathcal{B}(j+3) \setminus \mathcal{B}(\max\{j-2,0\})$, a fixed width annulus, and so

$$|c_n \cap A_j| \leq C_d \, |\mathcal{B}(j+3) \setminus \mathcal{B}(\max\{j-2,0\})| \leq C'_d j^{d-1}.$$

Here, the counting measure on the left hand side is bounded by the Lebesgue measure of the annulus containing that set on the right hand side. This gives,

$$\sum_{j>j'_0} |\{c_n\} \cap A_j| \, a(j) \leq C''_d \sum_{j>j'_0} j^{d-1} a(j).$$

for some $C''_d > 0$.

The stated result follows by a change of variables. Observe that the number of $d$-roots between any two consecutive integers $j$ and $j+1$ grows as $j^{d-1}$. So for any constant $C > 0$ and some $C_d > 0$, the sequence $\{\lceil n^{1/d}/C \rceil : n \in \mathbb{N}\}$ takes on each integral value $j \in \mathbb{N}$ at least $CC_d j^{d-1}$ times. Therefore,

$$\sum_{i>i_0} h_i \leq C_d(A+1) \sum_{j>j'_0} j^{d-1} a(j) \leq CC'_d(A+1) \sum_{n>n_0} a(\lceil n^{1/d} \rceil)$$

with $n_0 = \min\{n : \beta_{i_0+1} \leq a(\lceil n^{1/d}/C \rceil + 2)\}$. Now observe that $i_0$ counts the number of points excluded from the sum on the left, whereas $n_0$ is, up to constant factors, the volume of a ball in $\mathbb{N}^d$ that we exclude from the sum on the right. Together with (III), this shows that $n_0$ needs to scale as $C_d i_0$ for some $C_d > 0$. We absorb $C_d$ into the constant in the definition of $n_0$. Noting $a(\lceil \cdot \rceil) = m(\cdot)$, we obtain the stated result. □

The following lemma is the main result of this subsection.

**38 Lemma.** *Let $\Omega$ be the interval $|x_i| \leq \pi/3$. Let $\{c_n\}$ be the Fourier coefficients of a periodic operator $T_p$ and let*

$$m(x) = \min\{c_n : \lceil x \rceil \leq \|n\| < \lceil x \rceil + 1\}.$$

*Denote by $\{\lambda_n\}$ the eigenvalues of $P_\Omega T_p P_\Omega$. Then for all $n_0$ with $\|n_0\| > R$,*

$$\sum_{n>n_0} \lambda_n \leq C(R+1) \sum_{n>n_0} m(C' n^{1/d}).$$

To prove lemma 38 we need the following standard result.

**39 Min-max theorem** (Widom (1963))**.** *Let $A_i$ $(i = 1, \ldots, i_0)$ be self-adjoint and completely continuous. Then if $\varepsilon = \sum \varepsilon_i$ we have $N^{\pm}(\varepsilon, \sum A_i) \leq \sum N^{\pm}(\varepsilon_i, A_i)$.*

*Proof of lemma 38.* Let $\Omega_1, \Omega_2, \ldots, \Omega_{3^d}$ be non-overlapping translates of $\Omega$ that cover $I_d$. Then we can decompose $T_p$ as $T_p = \sum_{j,k} P_{\Omega_j} T_p P_{\Omega_k}$ where $P_{\Omega_j}$ is the projection operator corresponding to multiplying by the characteristic function of the set $\Omega_j$. Then, by the min-max theorem, for any $0 < \delta < 1$,

$$N^+((1-\delta)\varepsilon, T_p) \geq N^+(\varepsilon, \sum_j P_{\Omega_j} T_p P_{\Omega_j}) - N^+(\delta\varepsilon, -\sum_{j \neq k} P_{\Omega_j} T_p P_{\Omega_k}) \tag{4.11}$$

$$= N^+(\varepsilon, \sum_j P_{\Omega_j} T_p P_{\Omega_j}) - N^-(\delta\varepsilon, \sum_{j \neq k} P_{\Omega_j} T_p P_{\Omega_k}). \tag{4.12}$$

Since $T_p$ is periodic, we have $N^+((1-\delta)\varepsilon, T_p) = \Psi_c((1-\delta)\varepsilon)$. And since $\sum_j P_{\Omega_j} T_p P_{\Omega_j}$ is a direct sum,

$$N^+(\varepsilon, \sum_j P_{\Omega_j} T_p P_{\Omega_j}) = 3^d N^+(\varepsilon, P_\Omega T_p P_\Omega).$$

Applying the min-max theorem, we also have

$$N^-(\delta\varepsilon, \sum_{j \neq k} P_{\Omega_j} T_p P_{\Omega_k}) \leq \sum_{j<k} N^-(\delta\varepsilon/3^{2d}, P_{\Omega_j} T_p P_{\Omega_k} + P_{\Omega_k} T_p P_{\Omega_j}).$$

By lemma 35, each term in the above sum is bounded by $AR\Psi_c(\delta\varepsilon/3^{2d})$ whenever $\delta\varepsilon/3^{2d} < c_R$. Therefore,

$$N^+(\varepsilon, P_\Omega T_p P_\Omega) \leq 3^{-d}\Psi_c((1-\delta)\varepsilon) + 3^d CR\Psi_c(\delta\varepsilon/3^{2d}),$$

for $\varepsilon$ sufficiently small. Since $\Psi_c$ is monotonically decreasing,

$$|\{\lambda_n > \varepsilon\}| \leq C_d R |\{c_n > C_{\delta,d}\varepsilon\}| \quad (\forall \varepsilon < c_R),$$

and the result follows by applying lemma 37. □

## 4.A.5 Relating Fourier coefficients to spectral density

Next, in lemma 40, we relate the spectral density of a stationary kernel $k$ with the Fourier coefficients $\{c'_n\}$ of a periodic kernel $k_p$ equal to $k$ on a subset of its domain.

**40 Lemma.** *Let $S$ be the Fourier transform of a stationary kernel $k$ with an integral operator $T$, and let $k_p$ be a periodic kernel on $I_d$ such that $M_V T_p M_V = M_V T M_V$. Let $\{c'_n\}$ be the Fourier coefficients of $k_p$. Then for all $n, \varepsilon, \alpha > 0$,*

$$S(n) - C_{\varepsilon,\alpha} S(0) \|n\|^{-\alpha} < c'_n \leq (1+\varepsilon) S(n) + C_{\varepsilon,\alpha} S(0) \|n\|^{-\alpha}.$$

*Proof of lemma 40.* Let $u(x)$ be an infinitely differentiable function on $\mathbb{R}^d$ satisfying $u(x) = 1$ for $|x_i| \leq \pi/2$ and $u(x) = 0$ where $|x_i| > \pi$ for any $i$, and let $U$ be its Fourier transform. Then $M_V T M_V$ is the integral operator on $L_2(I_d)$ with the kernel

$$V(x)k(x-y)V(y) = V(x)k(x-y)u(x-y)V(y)$$

since $u(x-y) = 1$ whenever $V(x)V(y) \neq 0$. Importantly, this is also equal to

$$V(x)k_p(x-y)V(y)$$

where $k_p(x)$ is a function of period $2\pi$ equal to $k(x)u(x)$ for $|x_i| \leq \pi$. Therefore $k_p$ is the kernel of the operator $T$ on $L_2(I_d)$ associated with the sequence $\{c'_n\}$ given by

$$c'_n = \int_{I_d} k_p(x) e^{-in\cdot x} dx = \int_{E_d} u(x)k(x) e^{-in\cdot x} dx.$$

Noting the relationship between multiplication and convolution for Fourier transforms, we have

$$c'_n = (2\pi)^{-d} \int U(\omega) S(n-\omega) d\omega = S(n) + (2\pi)^{-d} \int U(\omega) \left(S(n-\omega) - S(n)\right) d\omega,$$

since $(2\pi)^{-d} \int U(\omega) d\omega = u(0) = 1$. Then for any $\delta > 0$,

$$\begin{aligned} c'_n = S(n) &+ (2\pi)^{-d} \int_{\|\omega\| \leq \delta \|n\|} U(\omega) \left(S(n-\omega) - S(n)\right) d\omega \\ &+ (2\pi)^{-d} \int_{\|\omega\| > \delta \|n\|} U(\omega) \left(S(n-\omega) - S(n)\right) d\omega. \end{aligned}$$

Looking at the first integral, we have

$$\int_{\|\omega\| \leq \delta \|n\|} U(\omega) \left(S(n-\omega) - S(n)\right) d\omega \leq \sup_{\|\omega\| \leq \delta \|n\|} |S(n-\omega) - S(n)|.$$

Since $S$ is continuous, for all $\varepsilon > 0$ there exists a $\delta > 0$ such that

$$0 < \sup_{\|\omega\| \le \delta \|n\|} |S(n-\omega) - S(n)| \le \varepsilon S(n).$$

Now consider the second integral. Since $S$ decreases away from 0 and is positive, and $u$ is smooth,

$$-C_{\varepsilon,\alpha} S(0) \|n\|^{-\alpha} < (2\pi)^{-d} \int_{\|\omega\| > \delta\|n\|} U(\omega) \left(S(n-\omega) - S(n)\right) d\omega \le C_{\varepsilon,\alpha} S(0) \|n\|^{-\alpha}$$

for all $\alpha > 0$. Using bounds from section 4.A.5 in the expression for $c'_n$ gives

$$S(n) - C_{\varepsilon,\alpha} S(0) \|n\|^{-\alpha} < c'_n \le (1+\varepsilon) S(n) + C_{\varepsilon,\alpha} S(0) \|n\|^{-\alpha} \quad (\forall \varepsilon, \alpha > 0). \qquad \square$$

Next, we confirm that the derived bounds on $\{c'_n\}$ imply that when $k$ is a Matérn family kernel, its Fourier coefficients satisfy (III), and determine the dependence of the corresponding radius $R$ on the kernel lengthscale; that they satisfy (I) and (II) is immediate. For this result, we will use a simplified form for the spectral density of a Matérn-$\nu$ kernel, writing $S^\ell(\omega) = \ell^d (1 + \ell^2 |\omega|^2)^{-p}$ where $p = \nu + d/2$. The omitted constant factors affect the result by a constant factor and will be accounted for later.

**41 Lemma.** *Let $S^\ell$ be the simplified spectral density of a Matérn kernel with lengthscale $\ell$ and let $\{c'_n\}$ be a sequence satisfying*

$$S^\ell(n) - C_{\varepsilon,\alpha} S^\ell(0) \|n\|^{-\alpha} < c'_n \le (1+\varepsilon) S^\ell(n) + C_{\varepsilon,\alpha} S^\ell(0) \|n\|^{-\alpha} \quad (\forall n, \alpha > 0). \quad (4.13)$$

*Then for all $\varepsilon > 0$ and $\|n\| > (1+\varepsilon)\ell^\varepsilon$, we have $c'_n > c'_{n+1}$.*

The lemma shows that such $\{c'_n\}$ satisfies (III) with $R = C_\varepsilon \ell^\varepsilon$ for all $\varepsilon > 0$ and $\ell > 1$.

*Proof.* From eq. (4.13), a sufficient condition for $c'_n > c'_{n+1}$ is that

$$S^\ell(n) - C_{\varepsilon,\alpha} S^\ell(0) n^{-\alpha} > (1+\varepsilon) S^\ell(\|n\| + 1) + C_{\varepsilon,\alpha} S^\ell(0) (\|n\| + 1)^{-\alpha}$$

Substituting in the expression for $S^\ell$, dividing through by $\ell^d$ that is

$$(1 + \ell^2 \|n\|^2)^{-p} - (1+\varepsilon)(1 + \ell^2 (\|n\| + 1)^2)^{-p} > C_{\varepsilon,\alpha} (\|n\|^{-\alpha} + (\|n\| + 1)^{-\alpha}).$$

Since $\|n\|^{-\alpha} > (\|n\| + 1)^{-\alpha}$, it suffices that the left hand side is greater than $C_{\varepsilon,\alpha} \|n\|^{-\alpha}$.

Multiplying through, it suffices that $n$ satisfies

$$(1+\ell^2(\|n\|+1)^2)^p - (1+\varepsilon)(1+\ell^2\|n\|^2)^p > 2C_{\varepsilon,\alpha}\|n\|^{-\alpha}(1+\ell^2(\|n\|+1)^2)^{2p}.$$

For $\|n\|$ sufficiently large independent of $\ell$, the right hand side can be upper bounded by $C'_{\varepsilon,\alpha'}\ell^4 p\|n\|^{-\alpha'}$ with $\alpha' = \alpha - 4p$. Similarly, for $\|n\|$ sufficiently large independent of $\ell$, the left hand side can be lower bounded by $\ell^{2p}((\|n\|+1)^{2p} - (1+\varepsilon)^2\|n\|^2 p$, which is in turn eventually lower bounded by $\ell^{2p}C$ for any $C$. Choose $C = C'_{\varepsilon,\alpha'}$, and denote by $A_{\varepsilon,\alpha'} > 0$ the cut-off independent of $\ell$ beyond which both these bounds hold. Then we require that $\|n\|^{\alpha'} > A_{\varepsilon,\alpha'} \wedge \ell^{2p}$. Since $\ell > 1$, it suffices that $\|n\|^{\alpha'} > A_{\varepsilon,\alpha'}\ell^{2p}$, and since we can pick $\alpha$ arbitrarily large, it suffices that $\|n\| > (1+\varepsilon)\ell^\varepsilon$ for some $\varepsilon > 0$. □

### 4.A.6 Bound on tail eigenvalue sum with respect to lengthscale

We now prove the result we set out to in this appendix: lemma 24 on page 69.

*Proof of lemma 24.* Write $k$ for a Matérn kernel with lengthscale $\ell \geq 1$ and use the simplified expression $S(\omega) = \ell^d(1+\ell^2\|\omega\|^2)^{-p}$ with $p = \nu + d/2$ for its spectral density. Associate with $k$ a periodic kernel $k_p$ such that $k_p(x) = k(x)$ on the interval $|x_i| \leq 1$ and let $\{c_n\}$ denote the Fourier coefficients of $k_p$. By lemma 41, $\{c_n\}$ satisfy (III) with constant $R_\varepsilon = (1+\varepsilon)\ell^\varepsilon$ for all $\varepsilon > 0$. Therefore by lemma 38, for all $n_0 > R_\varepsilon$ we have

$$\sum_{n>n_0}\lambda_n \leq C\sum_{n>z_\varepsilon} b(\lceil C' n^{1/d}/R_\varepsilon\rceil)$$

for some constants $C, C' > 0$. By lemma 40, for all $\varepsilon, \alpha > 0$,

$$c_n \leq (1+\varepsilon)S(n) + C_{\varepsilon,\alpha}S(0)\|n\|^{-\alpha}. \tag{4.14}$$

Since $S(n)$ depends on $n$ only through its norm $\|n\|$, let us write $\kappa(\|n\|) = S(n)$, and note that $\kappa(0) = \ell^d$. Note that $\kappa$ is monotonically decreasing. Now, recalling that $b(z) = \max\{c_n : z \leq \|n\| \leq z+1\}$ and using eq. (4.14), we have

$$b(z) \leq (1+\varepsilon)\kappa(z) + C_{\varepsilon,\alpha}\ell^d z^{-\alpha},$$

and therefore the sum in section 4.A.6 can be bounded as

$$\sum_{n>n_0}\lambda_n \leq \sum_{z>n_0} C(1+\varepsilon)\kappa(\lceil C' z^{1/d}/R_\varepsilon\rceil) + C'_{\varepsilon,\alpha}\ell^d\lceil C' z^{1/d}/R_\varepsilon\rceil^{-\alpha} \tag{4.15}$$

Since we can choose $\alpha$, we can always make the second term lower order. To handle the first term, consider $\sum_{z>n_0} \kappa(\lceil C' z^{1/d}/R_\varepsilon \rceil)$. We will relax the $\lceil \cdot \rceil$ and set $C' = 1$, as both contribute at most a constant factor to the sum. Since $\kappa(z)$ is monotonic, we can then bound the relaxed sum using an integral. That is,

$$\begin{aligned}\sum_{z>n_0} \kappa(z^{1/d}/R_\varepsilon) &\leq \int_{z>n_0} \kappa(z^{1/d}/R_\varepsilon) dz = \ell^d \int_{n>n_0} \left(1 + (\ell^2/R_\varepsilon^2) z^{2/d}\right)^{-p} dz \\ &\leq \ell^{-2\nu} R_\varepsilon^{2p} \int_{z>n_0} z^{-\frac{2\nu+d}{d}} dz \leq C \ell^{-2\nu+2p\varepsilon} n_0^{-\frac{2\nu}{d}}.\end{aligned}$$

The result as stated follows by substituting this bound into eq. (4.15), relabelling $2p\varepsilon \mapsto \varepsilon$, and inserting a constant term into the condition on $n_0$ to account for the use of the simplified spectral density. □

# Appendix 4.B Proofs of improved regret bounds for the GP-UCB algorithm

Both theorems 31 and 32 on page 75 follow from choosing specific values for parameters in previous results and simplifying. For both, recall that the regret of GP-UCB can be bounded as

$$R_T \leq \sqrt{T} \left(\sqrt{\gamma_T} \|f\|_{k^1} \ell^\nu + \gamma_T\right). \tag{4.16}$$

*Proof of theorem 31 (discretisation).* From theorem 26 on page 71 (and see remarks afterwards discussing the relationship with lengthscale), we have

$$\gamma_T^A \leq C_\varepsilon T^\varepsilon \left(n_0 + n_0^{-\frac{2\nu}{d}} T^{\zeta+1-b(d+2\nu)} + T^{1-\zeta/d}\right) \quad (\forall \varepsilon > 0).$$

Let $n_0 = T^a$ and $\ell = T^b$, with

$$a = \frac{d(d+1-b(d+2\nu))}{d(d+1)+2\nu} \quad \text{and} \quad b = \frac{d(d+1)}{d^2(2\nu+1)+4\nu(d+\nu)},$$

and choose

$$\zeta = \frac{2a\nu + db(d+2\nu)}{d+1}.$$

Substituting into our bound on $\gamma_T$, we have

$$\gamma_T \leq C_\varepsilon T^{x+\varepsilon} \quad (\forall \varepsilon > 0) \quad \text{where} \quad x = \frac{d(d+1)}{d^2(1+\frac{1}{2\nu})+2d+2},$$

leading to an overall bound on regret of the stated form. □

*Proof of theorem 32 (uniformly bounded eigenfunctions).* From theorem 29 on page 73,

$$\gamma_T \leq C_\varepsilon \left( n_0 \log T + \ell^{-2\nu+\varepsilon} T n_0^{-\frac{2\nu}{d}} \right) \quad (\forall \varepsilon > 0).$$

Choosing, $n_0 = T^{\frac{d}{d+\nu}}$ and $\ell = T^{\frac{d}{4\nu(d+\nu)}}$ we obtain $\gamma_T \leq C_\varepsilon T^{\frac{d}{d+\nu}+\varepsilon}$ for all $\varepsilon > 0$. Substituting these into eq. (4.16) yields the result. □

# Chapter 5

# Hierarchical GP Optimisation

We now introduce the main algorithmic contribution of part I, a hierarchical GP-UCB method that uses the bound on information gain developed in chapter 4 to attain strong bounds on regret and computational complexity without sacrificing practical performance.

## 5.1 The Partitioned GP-UCB algorithm

Our base algorithm is detailed in fig. 5.1. We will use it to explain the main idea and derive regret bounds. Later, we will provide modifications to handle discretisation and acquisition costs and make the algorithm adaptive.

The key element of the algorithm is a sequence of covers $\{\mathcal{A}_t\}$, sets consisting of axis-aligned hypercubes overlapping at their boundaries only, such that for all $t \geq 1$,

$$\bigcup \{A \in \mathcal{A}_t\} = [0,1]^d \quad \text{and} \quad \rho(A) \leq (N(A)+1)^{-b} \quad (\forall A \in A_t), \tag{5.1}$$

where $N(A)$ denotes the number of observations in the set $A \subset [0,1]^d$, $\rho(A)$ its diameter (defined with respect to the infinity norm) and $b \in (0,\infty)$ is the splitting parameter, which needs to be specified a priori (determined shortly as a function of $\nu$ and $d$ only). The construction itself is simple: take $\mathcal{A}_1 = \left\{[0,1]^d\right\}$; for $t > 1$, split any $A \in \mathcal{A}_t$ that does not satisfy eq. (5.1) into $2^d$ elements by cutting it in half across each dimension.

Comparing the property in eq. (5.1) with the statements of theorems 26 and 29 on page 71 and on page 73, we see that for a sufficiently high splitting parameter $b$, the information gain associated with the observations on each element $A \in \mathcal{A}_t$ at time step $t$, denoted

**Algorithm:** Partitioned GP-UCB
**Parameters:** $\delta \in (0,1)$, splitting factor $b \in (0,\infty)$
**Initialisation:** let $\mathcal{A}_1 = \left\{[0,1]^d\right\}$ and $\mathcal{H}_1 = \emptyset$

For each $t \in \mathbb{N}$:

1. For each $A \in \mathcal{A}_t$, let $(\mu_t^A, \sigma_t^A) = \mathrm{GPR}(1 + 1/t^2, \mathcal{H}_t^A)$ with

$$\mathcal{H}_t^A = \{(x,y) \in \mathcal{H}_t \colon x \in A\}$$

   and compute the maximum of

$$u_t^A(x) = \mu_t^A(x) + \beta_t^A(\delta)\sigma_t^A(x),$$

   denoting by $x_t^A \in A$ the maximiser, where $\beta_t^A = \beta_t(\delta/\eta_T)$ with

$$\beta_t(\delta) = \mathfrak{R} + \mathfrak{B}\sqrt{2(\gamma_t + 1 + \log(1/\delta))} \quad \text{and} \quad \eta_T = 4(T+1)^{bd}.$$

2. Select $X_t$ as an $x_t^A$ corresponding to a maximal $u_t^A(x_t^A)$ for $A \in \mathcal{A}_t$. Observe $Y_t$ corresponding to $X_t$ and store the observation:

$$\mathcal{H}_{t+1} = \mathcal{H}_t \cup \{(X_t, Y_t)\}.$$

3. Denote by $A_t$ the element of $\mathcal{A}_t$ such that $X_t \in A_t$. Let

$$N(A_t) = |\mathcal{H}_{t+1}^A|,$$

   and denote by $\rho(A_t)$ the diameter of element $A_t$. If

$$\rho(A_t) > (N(A_t) + 1)^{-b},$$

   let $\mathcal{A}_{t+1} = (\mathcal{A}_t \setminus \{A_t\}) \cup \mathrm{split}(A_t)$, else $\mathcal{A}_{t+1} = \mathcal{A}_t$.

Here, the split operation in step 3 takes a hypercube $A$ and returns a set of $2^d$ hypercubes by cutting $A$ in half across each dimension.

Figure 5.1: Partitioned GP-UCB algorithm.

$\gamma_t^A$, satisfies

$$\gamma_t^A \leq C_\varepsilon N(A)^\varepsilon \leq C_\varepsilon t^\varepsilon \quad (\forall \varepsilon > 0).$$

Therefore, using the standard GP-UCB concentration inequality (theorem 14 on page 45), for any $\delta' \in (0,1)$, $A \in \mathcal{A}_t$ and all $x \in A$,

$$|f(x) - \mu_t^A(x)| \leq \sigma_t^A(x) \left( \mathfrak{R} + \mathfrak{B}\sqrt{2\left(\gamma_t + \log(1/\delta')\right)} \right) \leq C_{\varepsilon,\delta} N(A)^{-\frac{1}{2}+\varepsilon},$$

for all $\varepsilon > 0$, where $(\mu_t^A, \sigma_t^A) = \mathrm{GPR}(\,\cdot\,, \mathcal{H}_t^A)$ is a GP regressor constructed with just the observations contained within $A$. *This concentration result matches up to the arbitrarily small $\varepsilon$ the result we would obtain through Azuma-Hoeffding if the observations in each cover element $A$ were independent.*

To obtain a bound holding with probability $1-\delta$ for all $t \leq T$, $A \in \mathcal{A}_t$ and $x \in A$, we first find a set $B_T$ such that $\mathcal{A}_t \subset B_T$ for all $t \leq T$ with probability one and bound its size. In lemma 48 on page 106, we show that for such $B_T$ with $|B_T| \leq \eta_T = 4(T+1)^{bd}$ exists. We then take a union bound over the elements of $B_T$ with confidence parameter $\delta' = \delta/\eta_T$.[1]

## 5.2 Regret analysis

We now state, discuss and provide a proof of our main result on the regret incurred by Partitioned GP-UCB. We show that the regret is sublinear under very weak smoothness assumptions and near-minimax-optimal when the eigenfunctions of the reproducing kernel are assumed to be uniformly bounded independently of scale.

### 5.2.1 Overview of results

Our main result concerning the regret incurred by Partitioned GP-UCB is:

**42 Theorem.** *Let the splitting parameter $b \geq 0$ be chosen such that for the kernel $k_\nu$ on $[0,\rho]^d$, $\rho < n^{-b}$ implies $\gamma_n \leq C_\varepsilon n^\varepsilon$ for all $\varepsilon > 0$. Then the regret incurred running Partitioned GP-UCB for $T$ steps is bounded as*

$$R_T \leq C T^{\frac{2db+1}{2db+2}+\varepsilon} \quad (\forall \varepsilon > 0).$$

[1] We can actually do slightly better. In Janz, Burt et al. (2020) we show through a martingale argument that using $\eta_t$ ($\leq \eta_T$) suffices. However, using $\eta_t$ complicates both the proofs and implementation while providing no perceptible improvement in empirical performance.

Examining the bound in theorem 42, it is clear that Partitioned GP-UCB converges to an optimum as long as there exists a $b$ such that reducing the diameter of a set at a rate of $n^{-b}$ leads to information gain for the corresponding kernel that is bounded by an arbitrarily small polynomial of $n$. The higher the splitting parameter $b$, the more frequently the algorithm splits and the higher the resulting regret.

We can use the bounds derived in chapter 4 to select suitable values of $b$. First, using the discretisation bound on information gain from theorem 26 on page 71, we have:

**43 Corollary.** *Under the usual assumptions, the regret after $T$ steps of running Partitioned GP-UCB on $H_{k_\nu}([0,1]^d)$ for $\nu > 1$ with $b = \frac{d+1}{d+2\nu}$ is bounded as*

$$R_T \leq C_\varepsilon T^{\frac{d(2d+3)+2\nu}{d(2d+4)+4\nu}+\varepsilon} \quad (\forall \varepsilon > 0).$$

The result in corollary 43 is significantly stronger than those for standard GP-UCB algorithms, in that it is sublinear (and therefore guarantees convergence in a global optimisation sense) for all $d$ and $\nu > 1$. As discussed at the end of section 4.2.1, a slightly more careful analysis can extend this result to hold for all $\nu > 0$. In contrast, the regret of GP-UCB is only guaranteed to be sublinear for $\nu$ and $d$ satisfying $\nu > d^2/2$.

Alternatively, using theorem 29 on page 73, we obtain:

**44 Corollary.** *Suppose $k_\nu$, $\nu > 0$, has eigenfunctions that are uniformly bounded independently of scale (definition 28). Then, under the usual assumptions, the regret after $T$ steps of running Partitioned GP-UCB on $H_{k_\nu}([0,1]^d)$ with $b = \frac{1}{2\nu}$ is bounded as*

$$R_T \leq C_\varepsilon T^{\frac{d+\nu}{d+2\nu}+\varepsilon} \quad (\forall \varepsilon > 0).$$

The regret bound given in corollary 44 matches the algorithm agnostic lower bound on worst-case expected regret given by Scarlett, Bogunovic et al. (2017a) to within a factor of $T^\varepsilon$. Furthermore, using the standard trick for converting bandit algorithms to global optimisation algorithms (Bubeck et al., 2011), under the assumptions of corollary 44, the time to simple regret $SR_T \leq \Delta$, denoted $\tau_\Delta$, is bounded as

$$\tau_\Delta \leq C_\varepsilon \left(\frac{1}{\Delta}\right)^{2+\frac{d}{\nu}+\varepsilon} \quad (\forall \varepsilon > 0).$$

This matches the respective algorithm agnostic lower bounds up to a $\Delta^{-\varepsilon}$ factor. However,

since corollary 44 relies on uniform boundedness of eigenfunctions, we can only claim this result for the case $d = 1$, $\nu = \frac{1}{2}$ (as discussed in section 4.4).

### 5.2.2 Proof of main result

The proof of theorem 42 relies on the following two lemmas:

**45 Lemma.** *The number of all cover elements used running Partitioned GP-UCB for $T$ steps can be bounded as*

$$|\cup_{t\leq T} \mathcal{A}_t| \leq C_{d,\nu} T^{\frac{db}{db+1}} + |\mathcal{A}_1|,$$

*where $C_{d,\nu} > 0$ depends on $d$ and $\nu$ only.*

**46 Lemma.** *Given $\delta \in (0,1)$, with probability $1-\delta$, for all $t \leq T$, all $A \in \bigcup_{t\leq T} \mathcal{A}_t$ and all $x \in A$,*

$$|\mu_t^A(x) - f(x)| \leq \beta_t^A(\delta)\sigma_t^A(x),$$

*where $\beta_t^A = \beta_t(\delta/\eta_t)$,*

$$\beta_t(\delta) = \mathfrak{R} + \mathfrak{B}\sqrt{2(\gamma_t + \log(1/\delta))} \quad \textit{and} \quad \eta_t = 4(t+1)^{bd}.$$

The first lemma bounds the number of cover elements used by the algorithm. We leave in an explicit dependence on $|\mathcal{A}_1|$. While the base algorithm always uses $\mathcal{A}_1 = [0,1]^d$, we will relax this shortly. The second lemma is our main concentration result.

The proof of the main result is a minor modification of that for GP-UCB, as given in Srinivas, Krause, S. M. Kakade et al. (2009) and Chowdhury et al. (2017).

*Proof of theorem 42.* For each $x \in [0,1]^d$ let $A_t(x)$ be an element of $\mathcal{A}_t$ such that

$$A_t(x) \in \operatorname*{arg\,max}_{A\in\mathcal{A}_t : x\in A} \mu_t^A(x) + \beta_t^A \sigma_t^A(x).$$

That is, $A_t(x)$ is an element of $\mathcal{A}_t$ on which the upper confidence bound associated with $x$ is the highest. This is a technicality to deal with points on the overlapping boundaries of cover elements.

For any $x \in [0,1]^d$, we have that $u_t^{A_t(x)}(x) \leq u_t^{A_t(X_t)}(X_t)$ by the definition of $X_t$. Therefore this also holds for $x^\star \in \arg\max_{x\in[0,1]^d} f(x)$. Expanding this expression and applying

lemma 46, we bound the per-step regret $r_t$ as

$$r_t = f(x^\star) - f(X_t) \leq 2\beta_t^{A_t(X_t)}\sigma_t^{A_t(X_t)}(X_t)$$

with probability $1-\delta$ for all $t \geq 1$. This is the same construction as in theorem 21, bounding per-step regret with twice the confidence width at $X_t$.

We now sum the squares of per-step regrets up to the horizon. We have

$$\sum_{t=1}^{T} r_t^2 \leq 4\sum_{t=1}^{T}(\beta_t^{A_t(X_t)})^2(\sigma_t^{A_t(X_t)}(X_t))^2 \leq 4\sum_{t=1}^{T}\sum_{A\in\mathcal{A}_t}\mathbb{1}\{X_t \in A\}(\beta_t^A)^2(\sigma_t^A(X_t))^2,$$

where the second inequality accounts for points that fall into multiple cover elements. Now, denote by $\widetilde{\mathcal{A}}_T = \bigcup_{t\leq T}\mathcal{A}_t$, the set of all cover elements created until time $T$, and define the initial time for an element $A \in \widetilde{\mathcal{A}}_T$, as $\tau(A) = \min\{t\colon A \in \mathcal{A}_t\}$ and the terminal time as $\tau'(A) = \max\{t\colon A \in \mathcal{A}_t\}$. Then,

$$\sum_{t=1}^{T}\sum_{A\in\mathcal{A}_t}\mathbb{1}\{X_t \in A\}(\beta_t^A)^2(\sigma_t^A(X_t))^2 = \sum_{A\in\widetilde{\mathcal{A}}_T}\sum_{t=\tau(A)}^{\tau'(A)}\mathbb{1}\{X_t \in A\}(\beta_t^A)^2(\sigma_t^A(X_t))^2$$

Bounding the sum of up to $\tau'(A)$-many variances on $A$ with $\gamma^A_{\tau'(A)}$ and using that, by the construction of the cover, $\gamma^A_{\tau'(A)} \leq C_\varepsilon(\tau'(A))^\varepsilon \leq C_\varepsilon T^\varepsilon$ for all $\varepsilon > 0$, both for the explicit $\gamma^A_{\tau'(A)}$ term and that contained within $\beta^A_{\tau'(A)}$,

$$\sum_{t=1}^{T} r_t^2 \leq C\sum_{A\in\widetilde{\mathcal{A}}_T}(\beta^A_{\tau'(A)})^2\gamma^A_{\tau'(A)} \leq C_{\varepsilon,\delta}T^\varepsilon\sum_{A\in\widetilde{\mathcal{A}}_T}1 = C_{\varepsilon,\delta}T^\varepsilon\left|\bigcup_{t=1}^{T}\mathcal{A}_t\right|.$$

Using lemma 45 to bound the cardinality of the set on the right hand side and applying Cauchy-Schwarz, we conclude that

$$R_T \leq \sqrt{T\textstyle\sum_{t=1}^{T} r_t^2} \leq C'_{\varepsilon,\delta}T^{\varepsilon/2}T^{\frac{db+1}{db+2}}. \qquad \square$$

## 5.3 Computational complexity

While Partitioned GP-UCB uses an independent Gaussian process regressor for each $A \in \mathcal{A}_t$ and $t \leq T$, at any given time-step $t$, we only need to fit up $2^d$ regressors:

1. Either the element $A_t$ such that $X_t \in A_t$ split, creating $2^d$ child elements that each

require the fitting of a Gaussian process.

2. Or the element $A_t$ did not split and therefore we require only an online Cholesky update to up to $2^d$ regressor to account for the new observation $(X_t, Y_t)$.

The regressors on all other elements can be cached. By the design of the algorithm, for any $A \in \mathcal{A}_t$, we have $\gamma_t^A \leq C_\varepsilon t^\varepsilon$. Recalling that standard sketching techniques allow the computation of the GP regressor and prediction in time that scales only with information gain, the overall runtime of Partitioned GP-UCB running on a finite domain $D \subset [0,1]^d$ can be bounded on the order of $T^{1+\varepsilon}|D|$. Therefore, for finite domains with functions of any smoothness (for which a suitable splitting factor $b$ can be derived), the runtime of Partitioned GP-UCB is near-linear. This extends the result from Calandriello, Carratino et al. (2020), which applies to smooth functions only.

To run on the convex domain $[0,1]^d$, Partitioned GP-UCB can be used with the same discretisation scheme as GP-UCB (theorem 22 on page 60). For functions that are not strongly differentiable, we have the following stronger result, proven in appendix 5.B.

**47 Theorem.** *For the Partitioned GP-UCB algorithm on $H_{k_\nu}([0,1]^d)$ for $\nu \leq 1$ with a splitting factor $b = \frac{1}{2\nu}$, the regret incurred evaluating the UCB only at the centre-point of each $A \in \mathcal{A}_t$ is within a constant factor of that incurred by considering all $x \in A$.*

With these choices of discretisation and using the sketching and batching methods of Calandriello, Carratino et al. (2020), we obtain an overall runtime on the order of

$$T^{1+\varepsilon} \text{ when } \nu \leq 1 \quad \text{and} \quad T^{1+\frac{d\nu}{d+2\nu}+\varepsilon} \text{ otherwise,} \tag{5.2}$$

both holding for all $\varepsilon > 0$. The result on $\nu \leq 1$, and indeed the construction, corresponds closely to the scheme used in the Hierarchical Optimistic Optimisation algorithm. We suspect that there may be a way to extend the adaptive discretisation we use for $\nu \leq 1$ to all values $\nu > 0$. However, our attempts to achieve this—by letting the number of points for which the UCB is evaluated on each element depend on the diameter of the element—have so far resulted in bounds that scale as $T^d$.

## 5.4 Practical considerations

We now discuss a series of details that, while not relevant to the worst-case regret bound, affect the practical performance of the algorithm.

**Adaptive construction** The algorithm as given in fig. 5.1 is is not adaptive. This is because the splitting condition used in step 3 depends explicitly on worst-case bounds for the information gain. This is easy to fix. Rather than splitting based on the number of points within the region, we can choose a constants $z, Z > 0$ and split whenever

$$\gamma_t^A = \frac{1}{2} \log \det(K_{X_t^A X_t^A} + \lambda^{-1} I) > Z N(A)^z. \tag{5.3}$$

It is clear that this appropriately controls the information gain on every cover element $A \in \mathcal{A}_t$. On the other hand, since for any $\varepsilon > 0$ the previous splitting lower bounded the number of points required for a threshold of the type in eq. (5.3) to be exceeded, the number of cover elements split by the adaptive algorithm is bounded to within a constant factor of the original construction. The worst-case regret of the adaptive algorithm is therefore within a $C_Z T^z$ factor of the original.

**Sequential tightening** A large part of the empirical regret incurred by Partitioned GP-UCB comes from newly created cover elements that contain few or no observations. When using a fixed discretisation, we can use the following *sequential tightening* trick to reduce this regret. For each $x \in [0,1]^d$ and $t \leq T$ denote by $A_t(x)$ an element of $\mathcal{A}_t$ containing $x$. Then, with probability at least $1 - \delta$,

$$f(x^\star) \leq u_t^{A_t(x)}(x) \quad (\forall t \leq T) \quad \text{and therefore also} \quad f(x^\star) \leq \min\left\{u_\tau^{A_\tau(x)}(x) \colon \tau \leq t\right\}$$

for all $x \in [0,1]^d$. Thus, we can obtain a tighter bound on $f$ for each point in the discretisation by only updating the upper confidence bound when a newly computed value is lower. With the use of this trick, splitting a cover element no longer increases the upper confidence bound on the points contained within it.

**Sketching** While kernel sketching can significantly decreases the computational complexity of Partitioned GP-UCB, it tends to be slower than online Cholesky updates when the number of observations is small. By construction, the number of observations used within each regressor when running Partitioned GP-UCB tends to remain small, and the benefits of sketching may be limited. We can combine the advantages of both online Cholesky updates and sketching by switching between the two in an adaptive manner: using either a threshold based on the number of data points in the region or on current per-step computation time.

**Initial cover and warm-starts** The algorithm thus far assumed $\mathcal{A}_1 = \{[0,1]^d\}$. This is, however, not necessary. For the regret bound to hold, we merely require that $|\mathcal{A}_1| \leq CT^{\frac{db}{db+1}}$ for some $C$ independent of $T$ (see lemma 45). Consequently, if using a double-and-guess approach for unknown parameters, say $T$ or $\mathfrak{B}$, when doubling occurs we can simply continue using the existing cover. Moreover, in the scenario where the algorithm is warm-started with some previously collected observations, we are free to construct $\mathcal{A}_1$ so as to minimise the computational cost of the initial iterations.

## 5.5 Experimental validation

We validate the proposed Partitioned GP-UCB algorithm with and without sequential tightening by benchmarking it on a number of standard two dimensional test functions. We compare performance against the standard GP-UCB algorithm and the near-minimax-optimal SupKernelUCB algorithm, all used with a fixed discretisation.[2]

**Experiment setup** We use standard two-dimensional test functions (Surjanovic et al., 2013), with domains scaled to $[0,1]^2$ and range scaled to $[-1,1]$. These are discretised into a $100 \times 100$ regular grid. We add independent centred Gaussian noise to each observation, with variance $1/20$ (low noise) and $1/2$ (high noise). In table 5.1, we report regret after 1000 interactions estimated using 10 seeds, normalised as a fraction of expected regret incurred by uniform sampling; and in table 5.2, the average runtime.

**Baselines** We implement GP-UCB, SupKernelUCB, Partitioned GP-UCB (PGP-UCB) and Partitioned GP-UCB with Sequential Tightening (PGP-UCB-ST). All use:

- A Matérn-1/2 kernel with lengthscale $\ell = 1.0$. The lengthscale was chosen as maximising performance for the standard GP-UCB algorithm from the set $\{0.1, 0.25, 1.0\}$.
- Confidence parameter $\delta = 0.05$, the exact subGaussianity constant $\mathfrak{B} \in \{1/20, 1/2\}$ and an (arbitrarily chosen) RKHS norm bound $\mathfrak{R} = 1.0$.
- Online Cholesky updates.

We chose not to include BKB and BBKB (Calandriello, Carratino et al., 2019; Calandriello, Carratino et al., 2020) in our comparisons. Our preliminary experiments suggested that these are considerably slower for the horizon lengths considered here and suffer higher

[2]Code for experiments: djanz.org/pgp_ucb/code

regret. Since they can be combined with each of the methods considered, any benefits they may provide are orthogonal to our work. We also do not benchmark against HOO. The assumptions and parameters within HOO are not easily comparable with those used here, particularly seeing as HOO does not use a fixed discretisation.

Table 5.1: Regret ± standard error for the benchmarked algorithms on standard test functions, expressed as a fraction of regret incurred by uniform sampling. 10 seeds.

| Noise | Function | GP-UCB | SupKernelUCB | PGP-UCB | PGP-UCB-ST |
|---|---|---|---|---|---|
| Low | Bukin N. 6 | $\mathbf{0.13 \pm 0.00}$ | $1.01 \pm 0.00$ | $0.19 \pm 0.01$ | $0.22 \pm 0.02$ |
| | 6-Hump Camel | $0.18 \pm 0.00$ | $1.00 \pm 0.02$ | $0.21 \pm 0.01$ | $\mathbf{0.14 \pm 0.01}$ |
| | Eggholder | $\mathbf{0.11 \pm 0.00}$ | $1.04 \pm 0.00$ | $0.29 \pm 0.06$ | $0.14 \pm 0.01$ |
| | Rosenbrock | $0.17 \pm 0.00$ | $1.01 \pm 0.02$ | $0.25 \pm 0.01$ | $\mathbf{0.14 \pm 0.00}$ |
| | Schaffer N. 2 | $\mathbf{0.07 \pm 0.02}$ | $1.01 \pm 0.01$ | $0.11 \pm 0.01$ | $0.12 \pm 0.01$ |
| High | Bukin N. 6 | $\mathbf{0.57 \pm 0.01}$ | $1.01 \pm 0.00$ | $0.66 \pm 0.01$ | $0.63 \pm 0.01$ |
| | 6-Hump Camel | $0.61 \pm 0.01$ | $1.00 \pm 0.02$ | $0.62 \pm 0.01$ | $\mathbf{0.54 \pm 0.01}$ |
| | Eggholder | $\mathbf{0.69 \pm 0.01}$ | $1.04 \pm 0.00$ | $0.89 \pm 0.01$ | $0.82 \pm 0.00$ |
| | Rosenbrock | $0.52 \pm 0.01$ | $1.01 \pm 0.02$ | $0.49 \pm 0.01$ | $\mathbf{0.47 \pm 0.01}$ |
| | Schaffer N. 2 | $\mathbf{0.41 \pm 0.01}$ | $1.01 \pm 0.01$ | $0.54 \pm 0.02$ | $0.50 \pm 0.01$ |

**Results** The results, shown in table 5.1, suggest that sequential tightening generally improves the performance of Partitioned GP-UCB. The performance of GP-UCB and PGP-UCB-ST is similar, with relative performance depending more on the function than on the level of noise. SupKernelUCB failed to significantly outperform the uniform random baseline across all of the problems—this is unsurprising, Shawe-Taylor et al. (2010) (slide 122) show similarly poor performance for the closely related SupLinRel algorithm on a linear bandit problem with $T = 5000$. Runtime results in table 5.2 show that Partitioned GP-UCB algorithms are much faster than GP-UCB and SupKernelUCB.

Table 5.2: Average runtime in seconds for each benchmarked algorithm.

| GP-UCB | SupKernelUCB | PGP-UCB | PGP-UCB-ST |
|---|---|---|---|
| 3300 | 3500 | 9.3 | **8.3** |

## 5.6 Discussion

We are now at the end of part I of this thesis. In this chapter, we have outlined our main algorithmic contribution, a hierarchical GP-UCB algorithm that uses our new bounds on the scaling of information gain with the diameter of the domain together with regressors fitted to carefully constructed subsets of the observations to obtain strong guarantees on regret for a large set of continuous functions while retaining empirical effectiveness.

Our method can be seen to combine ideas from SupKernelUCB and HOO to improve the classic GP-UCB algorithm. While, unlike GP-UCB, it only uses nearby observations when computing the upper confidence bound for a given point, this works well for the Matérn kernel, where the influence of observations rapidly decreases with Euclidean distance. Indeed, the algorithm can be seen as a generalisation of GP-UCB: as $\nu \to \infty$, the optimal cover size in Partitioned GP-UCB tends to one for all time steps, recovering the base GP-UCB algorithm. Our main technical results are:

- We improve over the worst-case regret of GP-UCB for all $\nu, d > 0$ while, unlike SupKernelUCB, remaining competitive with GP-UCB in practice.
- Under an additional assumption on the Matérn kernel used, we derive near-optimal bounds for both regret and simple regret. We have previously verified that this assumption holds for the case $\nu = 1/2$, $d = 1$.
- We show that for $\nu \leq 1$ on $[0, 1]^d$ and for $\nu > 0$ on a finite subset of $[0, 1]^d$, our algorithm attains near-linear (and therefore near-optimal) runtime. This extends the results for the BBKB algorithm, which attains that rate for $\nu = \infty$. We show that, in practice, Partitioned GP-UCB is much faster than the standard GP-UCB algorithm even without using sketching methods.

Our work leaves a number of important questions unresolved:

- *Uniform boundedness.* Our strongest results on regret rely on the assumption of uniform boundedness of the kernel eigenfunctions. While this assumption is common in the literature, it has only been confirmed to hold in a number of special cases. Confirming when this holds more generally is an interesting open question.
- *Computational complexity.* In the case of $H_{k_\nu}([0, 1]^d)$ with $\nu \leq 1$, we derived a method of adaptive discretisation for Partitioned GP-UCB that attains near-optimal bounds on computational complexity without affecting the regret bounds. However, we failed to extend this result to the case $\nu > 1$. Such an extension would

significantly strengthen the case for hierarchical GP-UCB algorithms.

- *Hyperparameter tuning.* In practical applications, lengthscale and other algorithm parameters are often chosen online based on the observations. Can we incorporate this into hierarchical GP-UCB methods while retaining our regret bounds?
- *Confidence intervals for kernel ridge regression with online design.* Finally, it remains an open question whether the confidence intervals presented in theorem 14 can be improved.

We find the last of these particularly interesting—a positive answer may make the base GP-UCB algorithm near-optimal.

# Appendix 5.A Proofs of theoretical results

*Proof of lemma 45.* First, any element $A \in \widetilde{\mathcal{A}}_T$ was either in the original cover $\mathcal{A}_1$ or was created by the splitting of a cover element in $\widetilde{\mathcal{A}}_T \setminus \mathcal{A}_T$ into $2^d$ elements. Therefore

$$|\widetilde{\mathcal{A}}_T| = |\mathcal{A}_1| + 2^d |\widetilde{\mathcal{A}}_T \setminus \mathcal{A}_T|.$$

Denote $\Theta_T = \widetilde{\mathcal{A}}_T \setminus \mathcal{A}_T$. Writing $\rho_0 \in (0, 1]$ for the diameter of any element $A \in \mathcal{A}_1$ (e.g. $\rho_0 = 1$ if $\mathcal{A}_1 = \{\mathcal{X}\}$). Then, for any given $T$,

$$|\Theta_T| \leq \max_{X_1,\ldots,X_T} |\Theta_T| = \max_{X_1,\ldots,X_T} \sum_{i=0}^{\infty} \sum_{A \in \Theta_T} \mathbb{1}\{\rho_A = 2^{-i}\rho_0\},$$

where the maximisation problem can be understood as arranging the points $X_1, \ldots, X_T$ so as to maximise the number of splits that occur. We now proceed to upper bound the solution to this maximisation problem by considering only a subset of the constraints imposed by the splitting procedure.

First constraint: we have a *budget constraint* derived from placing only $T$ points. Write $N_t^A$ for the number of points in $A \subset \mathbb{R}^d$ at time $t$. Let $\tau(A) = \min\{t \colon A \in \mathcal{A}_t\}$ and $\tau'(A) = \max\{t \colon A \in \mathcal{A}_t\}$. Suppose there exists a lower bound $M(\cdot)$ such that

$$M(\rho_A) \leq N^A_{\tau'(A)} - N^A_{\tau(A)}$$

for all $A \in \Theta_T$. Then

$$\begin{aligned}
&\sum_{i=0}^{\infty} M(2^{-i}\rho_0) \sum_{A \in \Theta_T} \mathbb{1}\{\rho_A = 2^{-i}\rho_0\} \\
&\quad \leq \sum_{i=0}^{\infty} \sum_{A \in \Theta_T} (N^A_{\tau'(A)} - N^A_{\tau(A)}) \mathbb{1}\{\rho_A = 2^{-i}\rho_0\} \\
&\quad = \sum_{A \in \Theta_T} N^A_{\tau'(A)} - N^A_{\tau(A)} \leq \sum_{A \in \widetilde{\mathcal{A}}_T} N^A_{\tau'(A)} - N^A_{\tau(A)} \\
&\quad = \sum_{A \in \widetilde{\mathcal{A}}_T} \sum_{t=\tau(A)}^{\tau'(A)} \mathbb{1}\{X_t \in A\} = \sum_{t=1}^{T} \sum_{A \in \mathcal{A}_T} \mathbb{1}\{X_t \in A\} \\
&\quad = \sum_{t=1}^{T} |\{A \in \mathcal{A}_t \colon X_t \in A\}| \leq 2^d T.
\end{aligned}$$

Now we verify that a suitable $M(\cdot)$ exists; this can be interpreted as the minimum *cost* of splitting an element $A \in \Theta_T$. Because $A$ split,

$$N^A_{\tau'(A)} + 1 > \rho_A^{-1/b} \geq N^A_{\tau'(A)}$$

Suppose that $A'$ is the element that split to create $A$. Then $\rho_{A'} = 2\rho_A$ and $\tau'(A') + 1 = \tau(A)$. Therefore,

$$N^A_{\tau'(A)} - N^A_{\tau(A)} \geq N^A_{\tau'(A)} - N^{A'}_{\tau'(A')} - 1 \geq \rho_A^{-1/b}(1 - 2^{-1/b}) - 2 = M(\rho_A).$$

Second constraint: a *supply constraint*. There are at most $\lceil \rho_0^{-d} \rceil$ elements of diameter $\rho_0$, and therefore at most $\lceil \rho_0^{-d} \rceil 2^{di}$ elements of diameter $2^{-i}\rho_0$ can be split, leading to

$$\sum_{A \in \Theta_T} \mathbb{1}\{\rho_A = 2^{-i}\} \leq \left\lceil \rho_0^{-d} \right\rceil 2^{di}.$$

Since $M(2^{-i}\rho_0)$ increases with $i$, the solution to the relaxed optimisation problem will be to buy all the available $A$ with smallest diameter, subject to the supply and budget constraints. Suppose the smallest $A$ split with this strategy has a diameter $2^{-z}\rho_0$ for some $z \in \mathbb{N}$. Then, since the supply constraint is binding and budget constraint is satisfied,

$$\sum_{i=0}^{z-1} M(2^{-i}\rho_0) 2^{di} \left\lceil \rho_0^{-d} \right\rceil \leq 2^d T.$$

Writing $\rho_0 = CT^\alpha$ for some $C > 0$ and using the geometric series formula to solve for $2^z$, we obtain $2^z = CT^{\frac{b}{bd+1}}$, a quantity independent of $\alpha$. Counting all the cover elements of diameters $2^0\rho_0, \ldots, 2^z\rho_0$, we have that $\sum_{i=0}^z 2^{di} = C2^{dz} \leq C'T^{\frac{db}{db+1}}$. Since this construction upper bounds $|\Theta_T|$, we have $|\widetilde{\mathcal{A}}_T| \leq CT^{\frac{db}{db+1}}$ as claimed. □

**48 Lemma.** *For each $t \in \mathbb{N}$ there exists a set $B_t$ with $|B_t| \leq 4(t+1)^{bd}$ such that $\mathcal{A}_t \subset B_t$ with probability one.*

*Proof.* Let $B^0 = \mathcal{A}_1$ and define recursively $B^{i+1}$ to be the set of the hypercubes created by splitting each element in $B^i$ into $2^d$ hypercubes. Let $\rho_0 \in (0, 1]$ be the diameter of elements in $\mathcal{A}_1$. Trivially,

$$\mathcal{A}_t \subset \bigcup_{i \geq 0} B^i \implies \mathcal{A}_t \subset \bigcup_{i \geq 0} \left( B^i \cap \mathcal{A}_t \right).$$

Suppose that $Z \in B^i \cap \mathcal{A}_t$ for some $i \geq 0$. By the splitting condition,

$$Z \in \mathcal{A}_t \setminus B^0 \implies \rho_Z > (N_t^Z + 1)^{-b} \geq (t+1)^{-b}.$$

Also, $Z \in B^i$ implies $\rho_Z \leq 2^{-i}\rho_0$. Therefore,

$$(t+1)^{-b} \leq \rho_Z \leq 2^{-i}\rho_0.$$

From this we conclude that $i \leq J_t \doteq \lfloor \log_2 \left((t+1)^b \rho_0\right) \rfloor$. Let $B_t = \bigcup_{i \leq J_t} B^i$. Since for all $i > J_t$, $B^i \cap \mathcal{A}_t = \emptyset$, we have $\mathcal{A}_t \subset B_t$. Now we need to bound the cardinality of $B_t$. We have $|B^i| = \lceil \rho_0^{-d} \rceil 2^{di}$, so

$$|B_t| = \sum_{i=0}^{J_t} |B^i| = \lceil \rho_0^{-d} \rceil \sum_{i=0}^{J_t} 2^{di} \leq \lceil \rho_0^{-d} \rceil 2^{dJ_t+1} \leq 4(t+1)^{db}. \qquad \square$$

# Appendix 5.B Proof of discretisation result

We prove the constant-size discretisation result claimed in theorem 47. This proof uses the same counting argument and monotonicity argument as the proof of lemma 45, and the same Hölder continuity argument as theorem 22. We recommend reading those first.

*Proof of theorem 47.* On any cover element $A$, by the same Hölder continuity argument as in theorem 22, the excess per-step regret associated with the discretisation is bounded as $C_d \rho_A^\nu$. We can therefore bound the excess regret $R'_T$ as

$$R'_T \leq C_d \sum_{t=1}^{T} \sum_{A \in \mathcal{A}_t} \rho_A^\nu \mathbb{1}\{X_t \in A\} = \sum_{A \in \widetilde{A}_T} (\tau'(A) - \tau(A))\, \rho_A^\nu,$$

where as before, $\widetilde{A}_T = \bigcup_{t \leq T} \mathcal{A}_T$, and $\tau(A)$, $\tau'(A)$ are the initial and final times for an element $A \in \widetilde{A}_T$ respectively.

Now, since with the constant size discretisation scheme we contribute at most one point to any cover element per time step, we can bound $\tau'(A) - \tau(A)$ by the number of points needed to split $A$. By the splitting rule, this is at most $\rho_A^{-1/b}$. With that,

$$R'_T \leq \sum_{A \in \widetilde{A}_T} \rho_A^{-1/b+\nu} = \sum_{i=0}^{\infty} 2^{-i(\nu-1/b)} \mathbb{1}\{A \in \widetilde{\mathcal{A}}_T\}, \tag{5.4}$$

where we used that all $A \in \widetilde{\mathcal{A}}_T$ have diameters of the form $2^i$ for $i \in \mathbb{N}$. To upper bound the excess regret, it remains to construct a set $\widetilde{\mathcal{A}}_T$ that maximises the right hand side.

As in the proof of lemma 45, the weight factor of each term in the summation of eq. (5.4), in this case $2^{-i(\nu-1/b)}$, decreases with $i$. Therefore, to maximise the sum, we place each point in turn in the largest cover element that has not yet split. Suppose the smallest $A$ split with this strategy after placing $T$ points has a diameter $2^{-z}$ for some $z \in \mathbb{N}$. Then, since there are $2^{di}$ elements at each depth $i$ (supply constraint), we can bound the excess regret as

$$R'_T \leq \sum_{i=0}^{z-1} 2^{id} 2^{-i(\nu-1/b)}.$$

In the proof of lemma 45, we showed that for this selection strategy, $2^z = CT^{\frac{b}{db+1}}$. Together with the geometric sum formula, this gives

$$R'_T \leq CT^{\frac{db+1-b\nu}{db+1}}.$$

Now compare this with the result for the regret of Partitioned GP-UCB as a function of the splitting parameter $b$, given in theorem 42. We have that the discretisation regret is dominated whenever $2(1 - b\nu) \leq 1$. This is satisfied by our choice $b = \frac{1}{2\nu}$. □

# Part II

# Reinforcement learning with neural-linear models

# Overview of part II

Within part II, we tackle the problem of exploration in deep reinforcement learning. Our focus here shifts from theory to practice, with our overall goal being that of developing a simple model-free reinforcement learning algorithm with demonstrably strong performance on both hard tabular exploration tasks and on the Atari Learning Environment benchmark (ALE, M. G. Bellemare et al., 2013).

While exploration is the main focus of the theoretical bandit-based perspective on reinforcement learning, it is largely sidelined within applied reinforcement learning research: TD-Gammon, a notable early application of reinforcement learning to the game of Gammon, succeeded in large part because the stochasticity inherent in the game provided for adequate exploration (Tesauro et al., 1995; Pollack et al., 1997); recent breakthroughs combining reinforcement learning with convolutional neural networks to tackle classic Atari 2600 games made very little progress on games that require exploration (Mnih et al., 2015); modern reinforcement learning methods that have been used to tackle Go, Chess and other complex board games still use naïve exploration methods and rely instead on enormous quantities of computation and data (Silver, Huang et al., 2016; Silver, Hubert et al., 2017; Schrittwieser et al., 2020).

As the combination of deep learning and reinforcement learning starts to see real-life use, with applications ranging from the design of computer programs, novel materials, chemicals and drugs, through to the protein folding problem, robotics and autonomous driving (Janz, Westhuizen et al., 2018; Jumper et al., 2021; Kendall et al., 2019), finding robust, scalable exploration methods remains a key challenge. In this part, we focus on particular this challenge, using insights from the bandit-based regret-oriented perspective on reinforcement learning to develop effective and scalable algorithms, and combine these heuristically with deep learning.

## Contributions

In part II we develop Successor Uncertainties (SU), a randomised value function model (Osband, Van Roy, D. J. Russo et al., 2019) that when combined with posterior sampling and neural network function approximation gives rise to an effective deep reinforcement learning algorithm. Successor Uncertainties models the value for a state-action pair $(s, a)$ of an unknown MDP as a linear product of the form $Q(s, a) = \langle w, \varphi(s, a) \rangle$, where $w \in \mathbb{R}^d$ are some weights, inferred using standard conjugate Gaussian linear regression, and $\varphi(s, a) \in \mathbb{R}^d$ is an embedding learnt using a neural network. We refer to models of this form as *neural-linear*.

Neural-linear models have the appeal of simplicity, in that they involve only minor modifications of the well-studied Deep Q-Networks framework (DQN, Mnih et al., 2015). Our key observation within part II is that many previous algorithms based on neural-linear models are flawed (e.g. Moerland et al., 2017; Azizzadenesheli et al., 2018; Touati et al., 2019; O'Donoghue et al., 2018; Lipton et al., 2018), and especially so when combined with posterior-sampling-based exploration. We show that these and other similar algorithms fail to outperform uniform random exploration on standard sparse-reward hard-exploration tasks, examine their shortcomings, and look at how erroneous experimental methodology had obscured these.

Our proposed model, Successor Uncertainties, introduces an additional constraint in the neural network embeddings $\varphi$ that addresses the issues present within previous neural-linear methods. The basis of the method is the observation that if the quantity predicted, the Q-function, obeys a temporal difference relationship, the uncertainty about it ought to as well. And since the uncertainty estimates given by a linear Gaussian model are based solely on input locations (the neural network embeddings), these too need to obey a temporal difference structure. We show that the appropriate structure is that of a successor representation (Dayan, 1993), and that it can be computed with only minor modifications to the standard DQN model and optimisation framework.

We show that our Successor Uncertainties model can provide efficient exploration on the challenging Deep Sea tabular exploration benchmark (Osband, Van Roy, D. J. Russo et al., 2019) and present strong results on Atari 2600 games, demonstrating that our method can be scaled to tasks that require neural network function approximation.

## Structure and attribution

While part II is written to be self-contained, it uses concepts and basic results from bandit algorithm and linear modelling literature throughout. Readers unfamiliar with these topics may wish to consult chapters 1 and 2 respectively. Part II comprises three chapters:

1. Chapter 6 introduces reinforcement learning, including: Markov Decision Processes, planning, online episodic reinforcement learning, corresponding optimistic algorithms and the basics of deep reinforcement learning.

2. Chapter 7 provides an overview of feature-linear reinforcement learning algorithms, with a particular focus on those previously combined with deep reinforcement learning. These are evaluated on the Deep Sea baseline, with insights obtained later used within the design of our Successor Uncertainties.

3. Chapter 8 describes the Successor Uncertainties model, first in a tabular setting, with experiments on the Deep Sea baseline, and then as a deep reinforcement learning algorithm, with corresponding results on the Atari Learning Environment.

The core idea presented here was published as Janz, Hron, Mazur, Hofmann, Hernández-Lobato & Tschiatschek (2019), and was the result of an internship at Microsoft Research Cambridge (MSR). We thank MSR for the computational resources used for the Atari Learning Environment experiments. Much of the presentation of the material presented here was influenced by later discussions with Ian Osband and André Barreto at Deepmind, as well as Laurence Aitchison and Adam Yang. Adam Yang generated all experimental data for the Deep Sea experiments featuring in chapter 7. José Miguel Hernández-Lobato provided feedback on the writing.

# Chapter 6

# An introduction to reinforcement learning

In this introductory chapter we provide a basic overview of reinforcement learning. We begin by providing a terse introduction to Markov decision processes, the standard model for a stateful environment used within reinforcement learning. We then introduce the core problem tackled in this thesis, that of online episodic reinforcement learning, together with standard methods for solving it, based on optimism and posterior sampling. We finish by looking at the basics of deep reinforcement learning and at the Atari Learning Environment.

## 6.1 Markov decision processes

A Markov decision process (MDP) is a mathematical model for sequential decision making in a stateful environment with stochastic state transitions. Within this section, we cover the standard formalism around MDPs together with the basics of value functions, policy evaluation and planning. We recommend Szepesvári (2021) for a more thorough introduction to these topics, and the theory of reinforcement learning in general.

### 6.1.1 The MDP formalism

We define a finite MDP $M$ to a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \mathcal{P}_0, H, \gamma)$, where: $\mathcal{S}$ and $\mathcal{A}$ are finite sets of states and actions respectively; $\mathcal{R}$ is a conditional reward distribution, a map from $\mathcal{S} \times \mathcal{A}$ to the set of probability measures on $\mathbb{R}$; $\mathcal{P}$ is a conditional transition distribution,

a map from $\mathcal{S} \times \mathcal{A}$ to the set of probability measures on $\mathcal{S}$; $\mathcal{P}_0$ is a measure on $\mathcal{S}$ defining the initial state distribution; $H \in \mathbb{N}$ is the interaction horizon; and $\gamma \in [0, 1)$ is the discount factor. We write $\mathcal{M}$ for the set of all finite MDPs.

The interaction between a reinforcement learning agent and an MDP is as follows. The agent starts with an initial state $S_0$ sampled from $\mathcal{P}_0$. Thereafter, at each step $h \in [H]$, the agent finds itself in a state $S_h \in \mathcal{S}$, and selects an action $A_h \in \mathcal{A}$. In turn, the environment returns the next state $S_{h+1} \sim \mathcal{P}(S_h, A_h)$ and a reward $R_h \sim \mathcal{R}(S_h, A_h)$, where we assume $R_h = r(S_h, A_h) + \varepsilon_t$ for some $r \colon \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$ and $\varepsilon_h$ an independent, subGaussian random variable. We call $r$ the (mean) reward function.

A reinforcement learning agent is specified by a policy $\pi$, which is a map from $\mathcal{S}$ to the set of measures over $\mathcal{A}$ such that $\mathbf{P}(A_h|S_h) = \pi(S_h)(A_h)$ for all $h \in [H]$. We denote the set of all policies by $\Pi$. With each policy $\pi \in \Pi$ in an MDP $M \in \mathcal{M}$, we associated a *value*, denoted $\mathcal{V}_M^\pi$, given by

$$\mathcal{V}_M^\pi \doteq \mathbf{E}\left[R_0 + \gamma R_1 + \gamma^2 R_2 + \ldots + \gamma^H R_{H-1}\right] = \mathbf{E} \sum_{h<H} \gamma^h r(S_h, A_h),$$

where the expectation is with respect to the measure induced by the interaction between the environment and the agent. For a given MDP $M \in \mathcal{M}$, any policy $\pi^\star \in \Pi$ satisfying

$$\mathcal{V}_M^{\pi^\star} = \sup_{\pi \in \Pi} \mathcal{V}_M^\pi \doteq \mathcal{V}_M^\star$$

is said to be optimal in $M$. We call $\mathcal{V}_M^\star$ the (optimal) value of $M$. The existence of an optimal policy is easy to verify, but it need not be unique. To see that, construct an MDP where each action is duplicated. The task of computing or approximating an optimal policy is called planning; planning is the problem of reinforcement learning.

For ease, some of our later analysis will assume that the MDPs considered are tree-structured. That is, that $\mathcal{S} = \prod_{h<H} \mathcal{S}_h$ with $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset$ for all $i \neq j$ and where $s \in \mathcal{S}_h$ is reachable only within step $h$ of interaction. We can convert any finite MDP with state space $\mathcal{S}$ into a tree-structured MDP with state space $\mathcal{S}' = \mathcal{S} \times [H]$, mapping each $s \in \mathcal{S}$ to $(s, h) \in \mathcal{S}'$. We denote by $\mathcal{M}_T$ the set of tree-structured MDPs.

Somewhat unusually, we assume a discounted setting, where the value of future reward is discounted geometrically by the factor $\gamma \in [0, 1)$—a common assumption in applied RL literature—but also a fixed finite horizon $H$—more standard in theoretical work.

Each of these assumptions individually ensures that $\mathcal{V}_M^\star$ is bounded for all $M \in \mathcal{M}$ and, consequently, that an optimal policy exists. We include both to more easily encompass methods from across the two areas of literature within one framework, and will adjust the methods as needed to accommodate this.

### 6.1.2 Value functions, policy evaluation and greedy policies

We now introduce value functions, policy evaluation and greedy policies. These concepts underpin planning in MDPs.

We define value functions as follows:

- For a given MDP $M \in \mathcal{M}$ and policy $\pi \in \Pi$, we define the state-action value function $Q_M^\pi \colon \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ and the state value function $V_M^\pi \colon \mathcal{S} \mapsto \mathbb{R}$ by

$$Q_M^\pi(s,a) = \mathbf{E}\left[\sum_{h<H} \gamma^h r(S_h, A_h) \mid S_0 = s, A_0 = a\right] \text{ and } V_M^\pi(s) = \sum_{b \in \mathcal{A}} \pi(s)(b) Q_M^\pi(s,b),$$

  where we take $0/0 = 1$ as needed to define the conditional expectation for all $(s,a) \in \mathcal{S} \times \mathcal{A}$ and where that expectation is taken with respect to the measure given by the interaction of the policy $\pi$ and the transition dynamics $\mathcal{P}$, $\mathcal{P}_0$ of $M$.

- For a given MDP $M \in \mathcal{M}$, we define the optimal state-action value function $Q_M^\star \colon \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ and optimal state value function $V_M^\star \colon \mathcal{S} \mapsto \mathbb{R}$ by

$$Q_M^\star(s,a) = \max_{\pi \in \Pi} Q_M^\pi(s,a) \quad \text{and} \quad V_M^\star(s) = \max_{b \in \mathcal{A}} Q_M^\star(s,b).$$

We refer to state-action value functions as Q-functions.

For planning, we will need methods of computing $Q_M^\star$ for a given MDP $M \in \mathcal{M}$ or $Q_M^\pi$ for a given MDP $M$ and policy $\pi \in \Pi$—the latter of these is known as *policy evaluation*. A standard approach to both tasks is through the use of Bellman operators, defined:

- For an MDP $M \in \mathcal{M}$, the Bellman operator for a policy $\pi \in \Pi$ is the map $T_M^\pi \colon \mathbb{R}^{\mathcal{S}\times\mathcal{A}} \mapsto \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ given by

$$T_M^\pi Q(s,a) = r(s,a) + \gamma \mathbf{E}_{s' \sim \mathcal{P}(s,a)} \mathbf{E}_{a' \sim \pi(s')} Q(s',a') \quad (\forall (s,a) \in \mathcal{S} \times \mathcal{A}).$$

- For an MDP $M \in \mathcal{M}$, the associated Bellman optimality operator is the map

$T^\star_M \colon \mathbb{R}^{\mathcal{S}\times\mathcal{A}} \mapsto \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ given by

$$T^\star_M Q(s,a) = r(s,a) + \gamma \mathbf{E}_{s'\sim\mathcal{P}(s,a)} \max_{b\in\mathcal{A}} Q(s',b) \quad (\forall (s,a) \in \mathcal{S}\times\mathcal{A}).$$

Both $T^\star_M$ and $T^\pi_M$ are $\gamma$-contractions on $\mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ under the metric induced by the infinity norm. Therefore, by Banach's fixed-point theorem, each operator has a unique fixed-point. By inspection, these can be identified as $Q^\star_M$ and $Q^\pi_M$ respectively; that is, $T^\star_M Q^\star_M = Q^\star_M$ and $T^\pi_M Q^\pi_M = Q^\pi_M$. Furthermore, for all $q_0 \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$, $(T^\star_M)^k q_0 \to Q^\star_M$ and $(T^\pi_M)^k q_0 \to Q^\pi_M$ as $k \to \infty$ uniformly at a rate geometric in $\gamma$.[1] Evaluating these for a finite $k$ then gives the standard iterative approximation approaches for $Q^\star_M$ and $Q^\pi_M$ respectively. For tree-structured MDPs, these are exact after $H$ iterations. Approaches to planning based on such Bellman operator iteration and resulting reinforcement learning algorithms are referred to as *model-free*.

Policy evaluation within finite MDPs can also be performed more directly by solving a linear system of equations. Consider the Q-function and mean reward function as vectors $Q^\pi_M, r \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ and write $P^\pi_M \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|\times|\mathcal{S}||\mathcal{A}|}$ for a matrix with entries given by

$$[P^\pi_M]_{(s,a),(s',a')} = \mathbf{P}(S_{t+1} = s', A_{t+1} = a' | S_t = s, A_t = a) \tag{6.1}$$

with respect to the measure induced by the interaction of $\pi$ and $M$. Then,

$$Q^\pi_M = \sum_{h<H} (\gamma P^\pi_M)^h r \approx \sum_{h=0}^{\infty} (\gamma P^\pi_M)^h r = (I - \gamma P^\pi_M)^{-1} r.$$

The approximation becomes accurate for large $H$ and low $\gamma$. It can be made exact in a tree-structured MDP by augmenting the state-space with an absorbing terminal state with zero reward for all actions. Methods such as this, based on explicit models of the transition and reward functions, are referred to as *model-based*.

We now define greedy policies. For a Q-function $Q \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$, we say that $\tilde{\pi}$ is greedy with respect to $Q$ if

$$\operatorname{supp} \tilde{\pi}(s) \subset \arg\max_{b\in\mathcal{A}} Q(s,b) \quad (\forall s \in \mathcal{S}).$$

We denote by $GQ$ the set of all policies greedy with respect to $Q$. For any $\tilde{\pi} \in GQ^\star_M$ we have $V^{\tilde{\pi}}_M = V^\star_M$, and thus any policy greedy with respect to $Q^\star_M$ is optimal in $M$.

[1] See appendix A of Szepesvári (2010) for a proof of Banach's fixed-point theorem and its consequences in the context of Bellman operators.

This reduces planning to the task of computing $Q^\star_M$.

### 6.1.3 Planning in MDPs

Planning is the problem of computing or approximating the optimal policy within an MDP. It is the element differentiating reinforcement learning from bandit problems. Algorithms considered in this thesis will be based on two basic approaches.

The first, *value iteration* (VI), uses the Bellman operator iteration to estimate $Q^\star_\pi$ and returns a policy greedy with respect to that estimate. Specifically, for a starting point $q_0 \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ and stopping time $k \in \mathbb{N}$, VI returns $\pi \in G(T^\star_M)^k q_0$. Since the iterative estimate of the optimal Q-function converges uniformly at a geometric rate, for sufficiently large $k$ the value of the returned policy is close to optimal. VI is exact within tree structured MDPs after $H$ steps.

The second, *policy iteration* (PI), starts with an arbitrary $\pi_0 \in \Pi$, defines a recursive sequence of estimates by $\pi_{i+1} \in GQ^{\pi_i}_M$ and returns $\pi_k$ for some stopping time $k \in \mathbb{N}$. PI is based on the policy improvement theorem, which gives that for this construction, $\mathcal{V}^{\pi_{i+1}}_M \geq \mathcal{V}^{\pi}_M$, and is exact is in any finite MDP after a finite number of steps. However, since each iteration requires solving policy evaluation, policy iteration is expensive. Many modern RL algorithms perform what is known as *generalised policy iteration* (GPI, Sutton, Barto et al., 1998), where each $Q^{\pi_i}_M$ is approximated using only a few iterations of the Bellman operator–based procedure, warm-started using $Q^{\pi_{i-1}}_M$.

## 6.2 Online, episodic reinforcement learning

We frame the problem tackled in part II as online episodic reinforcement learning, where a reinforcement learning agent interacts with the same unknown MDP over multiple episodes, and aims to minimise the regret it incurs over the course of the interactions. In this section we formalise the relevant notions of regret and present the standard optimistic and posterior sampling strategies for solving this problem. We end on the framework of Randomised Value Functions and, in particular, the Randomised Least Squares Value Iteration algorithm (RVF & RLSVI, Osband, Van Roy and Wen, 2016). These will provide much of the theoretical basis for our Successor Uncertainties model.

### 6.2.1 Problem definition

We consider minimising regret incurred over a series of $L \in \mathbb{N}$ episodes of interaction with an unknown MDP $M$. At the start of episode $l \in [L]$, the agent selects a policy $\pi_l \in \Pi$ and, with some abuse of notation around the indexing, receives an initial state $S_{lH} \sim \mathcal{P}_0$, with the following states, actions and rewards indexed as $S_{lH+h}, A_{lH+h}, R_{lH+h}$ for $h \in [H]$. The choice of policy $\pi_l$ is allowed to depend on the history up to episode $l$,

$$\mathcal{H}_{lH} = (S_0, A_0, R_0, S_1, \ldots, S_{lH}).$$

We call an algorithmic construction for $\{\pi_l\}$ a reinforcement learning algorithm, and measure its performance on an MDP $M \in \mathcal{M}$ over $L$ episodes by the expected regret,

$$\mathbf{E}R(L, \{\pi_l\}, M) = \mathbf{E}\sum_{l<L}\sum_{h<H} \gamma^h \left(r_h^\star - r(S_{lH+h}, A_{lH+h})\right) = \mathbf{E}\sum_{l<L} \mathcal{V}_M^\star - \mathcal{V}_M^{\pi_l},$$

where $r_h^\star$ denotes the optimal reward attainable at step $h$ in $M$. The goal of designing a reinforcement learning algorithm is to minimise regret over some set of MDPs, which we take to be the set of tree-structured MDPs, $\mathcal{M}_T$. This can be done either in a frequentist setting, where we minimise the worst-case regret $\hat{R}$, or a Bayesian setting, where we are given a prior measure over $\mathcal{M}_T$, $P_M$, and minimise the Bayesian regret $BR$. These are defined as in part I:

$$\hat{R}(L, \{\pi_l\}) = \sup_{M \in \mathcal{M}_T} \mathbf{E}R(L, \{\pi_l\}, M) \quad \text{and} \quad BR(L, \{\pi_l\}) = \mathbf{E}_{M \sim P_M}\mathbf{E}R(L, \{\pi_l\}, M).$$

We focus on the Bayesian setting, assuming a Dirichlet prior over the transition probabilities and a Beta prior over mean rewards. Together with multinomial and Beta likelihood (with rewards assumed to lie in the set $\{0, 1\}$), this forms the *true or underlying posterior* for $M$, which in turn induces a *true or underlying posterior* for $Q_M^\star$ and, for a given policy $\pi \in \Pi$, for $Q_M^\pi$.

For finite MDPs, the lower bound on regret for a $k$-armed bandit, $\sqrt{kT}$ (discussed in chapter 1), immediately gives a lower bound on frequentist regret of the form $\sqrt{|\mathcal{A}|T}$. Osband and Van Roy (2016) extend this to show that frequentist regret on $\mathcal{M}_T$ is lower bounded as $\sqrt{H|\mathcal{S}||\mathcal{A}|T}$. We have not seen a specific lower bound for the Bayesian setting considered here, but it too will be of the form $\text{poly}(H, |\mathcal{S}|, |\mathcal{A}|)\sqrt{T}$. The broad goal for finite MDPs is then to design practical algorithms with regret upper bounded by a polynomial of the same form. Algorithms achieving this criterion, and in particular

those polynomial in $H$, are said to perform *deep exploration* (Kearns et al., 2002). This is in contrast with *naïve exploration methods*, such as selecting actions uniformly, which lead to regret exponential in $H$.

The presented lower bound implies a certain hardness for finite MDPs. Substituting in the appropriate cardinalities for any real-life task, say a computer game, leads to lower bounds that make the task practically impossible (how many different possible states and actions does StarCraft II have?); further structure is needed. For the remainder of this section, we focus on methods for problems that we call *tabular*—problems that can be solved, at least in principle, using just pen and paper—and consider only briefly heuristic extensions to a linear function approximation setting, where we hope for regret that scales with $d$, the dimension of some state-action embedding, independently of $|\mathcal{S}|$ and $|\mathcal{A}|$. In the next section, we will consider neural network function approximation.

### 6.2.2 Algorithm design principles

We now outline optimism and posterior sampling, two standard approaches for tackling online reinforcement learning problems. The first involves the construction of upper confidence bounds much like those in part I, and is usually seen in the context of frequentist regret. The second, posterior sampling, is more often introduced in the Bayesian setting. However, both can be used and analysed in either setting.

**Optimism** Optimistic methods rely on the construction of time-uniform upper confidence bounds for the state-action value functions, a sequence $\{U_l\}$ with each $U_l$ mapping $\mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ such that for some given $\delta \in (0, 1)$ there exists an event $\mathcal{E}$ with $\mathbf{P}(\mathcal{E}) \geq 1 - \delta$ on which $U_l \geq Q^\star_M$ for all $l \in [L]$. Then, for each episode $l$, we take $\pi_l \in GU_l$, and incur regret bounded as

$$\mathbf{E}R(L, \{\pi_l\}, M) \leq \sum_{l<L} \mathbf{E}\Big[ \underbrace{\mathcal{V}^\star_M - \max_{a \in \mathcal{A}} U_l(S_0, a)}_{\Delta^l_{\text{opt}}(S_0)} \Big] + \mathbf{E}\Big[ \underbrace{\max_{b \in \mathcal{A}} U_l(S_0, b) - \mathcal{V}^{\pi_l}_M}_{\Delta^l_{\text{conc}}(S_0)} \Big].$$

on MDP $M \in \mathcal{M}_T$. Examining the two terms:

1. $\mathbf{E}\Delta^l_{\text{opt}}(S_0)$ is an optimism term. By construction, $\mathbf{E}[\Delta^l_{\text{opt}} \mid \mathcal{E}] \leq 0$ for all $l < L$.
2. $\mathbf{E}\Delta^l_{\text{conc}}(S_0)$ is the difference between the optimistic estimate of the value of the policy $\pi_l$, $\mathbf{E} \max_{b \in \mathcal{A}} U_l(S_0, b)$ and its true value $\mathcal{V}^{\pi_l}_M$.

Combining the two, and bounding regret on $\mathcal{E}^C$ linearly,

$$\mathbf{E}(L, \{\pi_l\}, M) \leq \delta LH + (1-\delta) \sum_{l<L} \mathbf{E}\left[\Delta^l_{\text{conc}}(S_0) \mid \mathcal{E}\right],$$

and it remains to choose a suitable $\delta$ (as a function of $L$ and $H$). With that, if we can show that $\mathbf{E}[\max_{b\in\mathcal{A}} U_l(S_0, b) \mid \mathcal{E}] \to \mathcal{V}^\star_M$ as $l \to \infty$, then $\mathbf{E}[\mathcal{V}^{\pi_l}_M \mid \mathcal{E}] \to \mathcal{V}^\star_M$ and $\mathbf{E}[\Delta^l_{\text{conc}}(S_0) \mid \mathcal{E}] \to 0$ on $\mathcal{E}$ as well; that is, if our upper confidence bounds are consistent for the optimal value function, we recover sublinear regret.

Optimism thus converts the problem of online reinforcement learning to that of computing tight constant-probability optimistic estimates of the optimal value function. It is at the core of many provably efficient reinforcement learning algorithms, for example Kearns et al. (2002), Brafman et al. (2002), Strehl et al. (2006) and Dann et al. (2017), and gives a near-minimax-optimal algorithm for the set of finite MDPs under frequentist regret (Azar et al., 2017). Optimistic algorithms can be either model-based, maintaining a confidence set for the MDP $M$ and maximising value over that set, or model-free, working directly with upper confidence bounds for Q-functions.

**Posterior sampling** A Bayesian alternative to optimism, posterior sampling (PS) maintains a posterior over MDPs and uses a sequence of policies greedy with respect to samples from this posterior. We formalise *exact* posterior sampling in episodic reinforcement learning as follows. At the start of the $l$th episode, we compute a posterior over $M$, $P^l_M = \mathbf{P}(M \mid H_{lH})$, starting with $P^0_M = P_M$, the prior. We then sample $M_l \sim P^l_M$ and take a $\pi_l \in GV^\star_{M_l}$. For that construction,

$$BR(L, \{\pi_l\}) = \sum_{l=0}^{L-1} \mathbf{E} \underbrace{\mathcal{V}^\star_M - \mathcal{V}^{\pi_l}_{M_l}}_{\Delta^l_{\text{opt}}} + \mathbf{E} \underbrace{\mathcal{V}^{\pi_l}_{M_l} - \mathcal{V}^{\pi_l}_M}_{\Delta^l_{\text{conc}}},$$

where the expectations are over $M$ and $M_l$. Now:

1. $\mathbf{E}\Delta^l_{\text{opt}} = 0$ for all $l \in [L]$. To see this, note that $\mathcal{V}^{\pi_l}_{M_l} = \mathcal{V}^\star_{M_l}$ and let $f\colon \mathcal{M} \mapsto \mathbb{R}$ be the function $f\colon M \mapsto \mathcal{V}^\star_M$. By the tower property,

$$\mathbf{E}\Delta^l_{\text{opt}} = \mathbf{E}\mathbf{E}[\mathcal{V}^\star_M - \mathcal{V}^\star_{M_l} | \mathcal{H}_{lH}] = \mathbf{E}\left[\mathbf{E}[f(M)|\mathcal{H}_{lH}] - \mathbf{E}[f(M_l)|\mathcal{H}_{lH}]\right].$$

   The distribution of $f(M)|\mathcal{H}_{lH}$ is then the pushforward by $f$ of the true posterior for $M$, and of $f(M_l)|\mathcal{H}_{lH}$ the pushforward by $f$ of $P^l_M$. In the exact setting, these

two conditional measures are equal for all $l$, giving the claim.

2. $\Delta^l_{\text{conc}}$ is the difference in the value of $\pi_l$ in $M_l$ and $M$. A similar tower property expansion combined with a continuity argument shows that if the posterior $P^l_M$ concentrates suitably as $l \to \infty$, then $\mathbf{E}\Delta^l_{\text{conc}} \to 0$ in the same limit.

Combining these, we have $BR(L, \{\pi_l\}) \leq \sum_{l<L} \mathbf{E}\Delta^l_{\text{conc}}$, and it remains to quantify the rate of convergence. Under standard conjugate exponential family priors and likelihoods for the transition dynamics and mean rewards, this is simple.

This exact posterior sampling algorithm and its analysis was introduced by Osband, D. Russo et al. (2013) as Posterior Sampling for Reinforcement Learning (PSRL). The authors have shown PSRL to be very effective in practice on tabular MDPs. Prior to this, posterior sampling has been used as a heuristic in a bandit-like setting by Thompson (1933). Therein, posterior sampling is introduced as the procedure of selecting arms according to the posterior probability that they are optimal. This is entirely to the procedure described here of sampling an environment from the posterior and playing an arm optimal in that environment.

### 6.2.3 Randomised value functions and stochastic optimism

Randomised value functions (RVF, Osband, Van Roy and Wen, 2016) is a framework for model-free approximate posterior sampling where, rather than computing a posterior over MDPs, we maintain an approximate posterior distribution $P^l_Q$ over value functions. Then, at the start of the $l$th episode, an RVF agent samples a *plausible value function* $Q_l \sim P^l_Q$ and follows a policy $\pi_l \in GQ_l$.

To design an RVF method, we need to define a construction for $P^l_Q$. We call this the Q-function model. In choosing this construction, we look for two qualities: that the posterior is easy to update and sample from, and that the induced distribution over the policies $\{\pi_l\}$ leads to low regret. The first is a matter of computational complexity, and needs to be assessed on a per-algorithm basis. The latter can be analysed using the framework of *stochastic optimism*. Here, we ask that:

1. $P^l_Q$ is *stochastically optimistic* for the distribution of $Q^\star_M$, in that for $Q_l \sim P^l_Q$,

$$\mathbf{E}\left[\max_{b\in\mathcal{A}} Q^\star_M(s,b) - \max_{a\in\mathcal{A}} Q_l(s,a)\right] \leq 0 \quad (\forall s \in \mathcal{S},\ \forall l \in [L]).$$

2. $Q_l$ converges to $Q^\star_M$ in a suitable sense as $l \to \infty$.

With these two conditions, we can use analysis much like that for exact posterior sampling to bound the regret of RVF methods. It will be useful to observe that stochastic optimism is transitive. When designing a new RVF algorithm for a setting considered by an existing RVF method, it suffices to prove that the new algorithm is stochastically optimistic for the existing method. This is often significantly easier.

### 6.2.4 Randomised least squares value iteration

Randomised Least Squares Value Iteration (RLSVI) is the original RVF algorithm, proposed and analysed in the Bayesian setting by Osband, Van Roy and Wen, 2016. Zanette et al. (2020) have since extended its analysis to a frequentist setting. We examine the algorithm from the Bayesian perspective, where the crux of the technical analysis is in showing *Gaussian-Dirichlet stochastic optimism*: that a Gaussian distribution with a sufficiently high mean and variance is stochastically optimistic for a given Dirichlet distribution (recall that $Q^\star_M$ is a Dirichlet random variable under our assumptions). We take this stochastic optimism result for granted and focus on the RLSVI Q-function model.

In each episode $l \in [L]$, RLSVI partitions the plausible Q-function $Q_l$ into $\{Q_l|_h \colon h \in [H]\}$, where $Q_l|_h \colon \mathcal{S}_h \times \mathcal{A} \mapsto \mathbb{R}$ is the restriction of $Q_l$ to $\mathcal{S}_h \times \mathcal{A}$, and samples $Q_l|_h(s,a)$ for $(s,a) \in \mathcal{S}_h \times \mathcal{A}$ conditionally on $Q_l|_{h+1}$, with the boundary condition $Q_l|_H(s,a) = 0$ for all $(s,a) \in \mathcal{S}_0 \times \mathcal{A}$. The conditional distribution of $Q|_h(s,a)$ given $Q|_{h+1}$ and the history $\mathcal{H}_{lH}$ is modelled by conjugate Gaussian linear regression with prior mean $m \in \mathbb{R}$, prior variance $\alpha > 0$, likelihood variance $\beta > 0$ and 'observations' of the form

$$\{R_{iH+h} + \gamma \max_{b \in \mathcal{A}} Q_l|_{h+1}(S_{iH+h+1}, b) \colon S_{iH+h} = s, A_{iH+h} = a, i \in [l]\}.$$

The resulting conditional distribution for each step $h \in [H]$ at the start of episode $l$ is then $\mathcal{N}(\mu_h, \mathrm{diag}(\nu_h))$ where, writing $\chi_t(s,a) = \mathbb{1}\{S_t = s, A_t = a\}$,

$$\mu_h(s,a) = \nu_h(s,a) \left[\alpha^{-1} m + \beta^{-1} \sum_{i<l} \chi_{iH+h}(s,a) \left(R_{iH+h} + \gamma \max_{b \in \mathcal{A}} Q_l|_{h+1}(S_{iH+h+1}, b)\right)\right] \tag{6.2}$$

$$\text{and} \quad \nu_h(s,a) = \left(\alpha^{-1} + \beta^{-1} n_{lH}(s,a)\right)^{-1} \quad \text{with} \quad n_{lH}(s,a) = \sum_{\tau < lH} \chi_\tau(s,a)$$

for all $(s,a) \in \mathcal{S}_h \times \mathcal{A}$. Choosing $m, \alpha, \beta$ appropriately, each conditional distribution $Q_l|_h$ given $Q_l|_{h+1}$ can be made stochastically optimistic for the true conditional distribution $Q^\star_M|_h$ given $Q^\star_M|_{h+1}$ (using Gaussian-Dirichlet stochastic optimism). A recursive argument then shows that the resulting sample $Q_l$ is stochastically optimistic for $Q^\star_M$. Note that while each conditional is Gaussian, $P^l_Q$ is *not* jointly Gaussian.

Osband, Van Roy and Wen (2016) generalise RLSVI to a linear function approximation setting by taking $Q|_h(s,a) = \langle w_h, \varphi(s,a)\rangle$ for some given embedding function $\varphi\colon \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}^d$ and weights $\{w_h \in \mathbb{R}^d\}$. Each conditional distribution is then obtained by performing conjugate linear Gaussian regression for the corresponding weights (the tabular version may be recovered by taking $\varphi(s,a) = e_{(s,a)}$, a one-hot encoding of $\mathcal{S} \times \mathcal{A}$). The authors present experiments that suggest this heuristic extension leads to empirical regret that scales as poly$(d)$ on linearly-embedded tabular MDPs. Since then, Jin et al. (2020) have proven a bound of that form for a similar algorithm based on linear function approximation with OFUL confidence intervals and least squares value iteration under certain linearity assumptions. We will take a similar approach in our work to that of the authors of RLSVI: we will analyse a tabular version of our algorithm and extend it heuristically to a linear setting.

## 6.3 Deep reinforcement learning

We now turn to deep reinforcement learning, the empirically-driven combination of reinforcement learning with neural network function approximation. In this section, we outline the key components of the seminal deep reinforcement learning algorithm, Deep Q-Networks (DQN, Mnih et al., 2015), and some now-standard tweaks. We also look at the Atari Learning Environment and the neural network architecture used to tackle it.

### 6.3.1 The empirical MDP model

Throughout our exposition we will find it useful to refer to empirical MDP models: those resulting from using simple empirical averages of the observed quantities. Writing $n_t(s,a) = \sum_{\tau<t} \mathbb{1}\{S_\tau = s, A_\tau = a\}$ and taking $\frac{1}{n_t(s,a)} = 0$ when $n_t(s,a) = 0$, we refer to $\bar{\mathcal{R}}$ mapping $\mathcal{S} \times \mathcal{A}$ to a set of measures on $\mathbb{R}$ as an empirical reward model if

$$\mathbf{E}\bar{\mathcal{R}}(s,a) = \frac{1}{n_t(s,a)} \sum_{\tau<t} R_\tau \mathbb{1}\{S_\tau = s, A_\tau = a\}$$

and to $\bar{\mathcal{P}}$ mapping $\mathcal{S} \times \mathcal{A}$ to the set of measures on $\mathcal{S}$ as an empirical transition model if

$$\mathbf{E}\bar{\mathcal{P}}(s,a)(s') = \frac{1}{n_t(s,a)} \sum_{\tau<t} \mathbb{1}\{S_\tau = s, A_\tau = a, S_{\tau+1} = s'\}.$$

We denote these expectations as $\bar{r} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ and $\bar{P} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|\times|\mathcal{S}|}$ respectively. Also, for a policy $\pi \in \Pi$, we will write $\bar{P}^\pi \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|\times|\mathcal{S}||\mathcal{A}|}$ for the state-action transition matrix given elementwise by

$$[\bar{P}^\pi]_{(s,a),(s',a')} = \frac{\pi(s')(a')}{n_t(s,a)} \sum_{\tau<t} \mathbb{1}\{S_\tau = s, A_\tau = a, S_{\tau+1} = s'\}.$$

We refer to $\bar{M} = (\mathcal{S}, \mathcal{A}, \bar{\mathcal{R}}, \bar{\mathcal{P}}, \bar{\mathcal{P}}_0, H, \gamma)$ as an empirical MDP model, with the empirical initial state distribution $\bar{\mathcal{P}}_0$ defined accordingly.

### 6.3.2 Generalised policy iteration

The deep reinforcement learning algorithms we consider are built around gradient-based generalised policy iteration planning methods. A classic algorithm that has been adapted for this purpose is *Q-learning* (Watkins, 1989; Watkins and Dayan, 1992). Q-learning begins by initialising some Q-function $Q_0 \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$, for example $Q_0(s,a) = 0$ for all $(s,a) \in \mathcal{S} \times \mathcal{A}$. For each time-step thereafter, it constructs $Q_t$ from $Q_{t-1}$ and the tuple $(S_{t-1}, A_{t-1}, R_{t-1}, S_t)$ by computing the *target*

$$Y_t = R_{t-1} + \gamma \max_{b\in\mathcal{A}} Q_t(S_t, b),$$

and setting

$$Q_t(A_{t-1}, S_{t-1}) = (1-\alpha)Q_{t-1}(A_{t-1}, S_{t-1}) + \alpha Y_t$$

for $\alpha \in (0,1)$ a learning rate and $Q_t(s,a) = Q_{t-1}(s,a)$ for all $(s,a) \neq (S_{t-1}, A_{t-1})$. Q-learning can be interpreted as computing the Q-function for a weighted empirical MDP $\bar{M}$. Under mild assumptions on the stream of observations and the learning rate, the Q-learning estimator $Q_t$ converges to $Q^\star_M$ as $t \to \infty$.

Q-learning is an example of an off-policy reinforcement learning algorithm, in that the Q-function estimate does not explicitly depend on the interaction policy used by the algorithm. If instead we use the targets

$$Y_t = R_{t-1} + \gamma Q_{t-1}(S_t, A_t),$$

we obtain the *SARSA* algorithm, so named for the $(S_{t-1}, A_{t-1}, R_{t-1}, S_t, A_t)$ quintuple used therein (Rummery et al., 1994). This is an on-policy algorithm: the resulting Q-function depends on the policy used for interaction. Taking the interaction policy to be greedy with respect to $Q_t$ at each step, SARSA and Q-learning become equivalent.

Another related algorithm is Expected SARSA (ESARSA, Sutton, Barto et al., 1998). Here, together with a tuple $(S_{t-1}, A_{t-1}, R_{t-1}, S_t)$, we take policy $\pi_t$ and use the targets

$$Y_t = R_{t-1} + \gamma \langle \pi_t(S_t), Q_t(S_t, \cdot) \rangle.$$

When $\pi_t$ is the interaction policy, such that $A_t \sim \pi_t(S_t)$, ESARSA is on-policy and the targets used are equal in expectation to those of SARSA (but lower variance). With a general policies, ESARSA is an off-policy method. When the policy is fixed throughout, that is $\pi_t = \pi$ for some $\pi \in \Pi$ and all $t < T$, ESARSA computes $Q^\pi_{\tilde{M}}$ for a weighted empirical MDP $\tilde{M}$. Convergence of $Q_t$ to $Q^\pi_M$ is then guaranteed under standard conditions on $\pi$, $\mathcal{P}_0$ and $\alpha$ (Van Seijen et al., 2009).

### 6.3.3 Neural network function approximation

We now look at deep reinforcement learning: the combination of reinforcement learning with neural network function approximation. The core idea behind deep (model-free) reinforcement learning, and in particular Deep Q-Networks (DQN) models, is to take $Q_t(s, a) = Q(s, a; \theta_t)$, where $Q$ is a neural network and $\theta_t$ are the parameters of that neural network, compute the standard Q-learning targets $Y_t$, and use gradients of the Bellman loss

$$g_t = \partial_{\theta_{t-1}} (Q(s, a; \theta_{t-1}) - Y_t)^2$$

with a gradient-based optimisation method to obtain updated parameters $\theta_t$. This basic idea is augmented with several heuristics to improve the performance of the agent.

First, Deep Q-Networks introduces a replay buffer (Lin, 1992), which stores the last $\tau$-many observations for some $\tau \in \mathbb{N}$ and is used to compute an averaged gradient

$$\bar{g}_t = \frac{1}{B} \sum_{b=1}^{B} g_{i(b)}$$

for a batch size $B$ and for $i(b)$ a uniform sample from $\{t - \tau, \ldots, t\}$. This acts to reduce the variance of the gradients and decorrelate the updates, making training more stable—if

were we to use sequential observations for updates, the strong autocorrelation between the Q-function at consecutive time steps may lead to instability in the neural network training procedure. We can interpret the replay buffer to be a biased non-parametric empirical model of the rewards and transition dynamics.

Another feature introduced in the DQN algorithm is the use of delayed weights in the computation of the targets. That is, the targets $Y_t$ are computed using weights $\theta'_t$, a copy of $\theta_t$ updated periodically (say every $\tau$ steps). That is,

$$Y_t = R_{t-1} + \max_{a \in \mathcal{A}} Q(S_t, a; \theta'_{t-1}).$$

This acts to slow down and stabilise the learning of the neural network weights. A further now-standard modification of the targets is given by double Q-learning (Van Hasselt et al., 2016), where the weights $\theta_{t-1}$ are used to estimate the optimal action but the delayed weights $\theta'_{t-1}$ to compute its value. The targets are then given by

$$Y_t = R_{t-1} + Q(S_t, a; \theta'_{t-1}) \quad \text{where} \quad a \in \arg\max_{b \in \mathcal{A}} Q(S_t, b; \theta_{t-1}).$$

Double Q-learning tackles the interaction of the bias of Q-learning and the neural network training dynamics. Specifically, when $Q_t = Q(\cdot, \cdot; \theta_t)$ is seen as a random variable distributed conditionally on $\mathcal{H}_t$, standard Q-learning, while consistent, is positively biased for the underlying Q-function, which can cause the neural network Q-function estimate to diverge. Using delayed weights $\theta'_t$ acts to reduce the bias. Indeed, if $\theta'_t$ were obtained from an independent stream of data $\mathcal{H}'_t$, double Q-learning would have negative bias.

In this thesis, we will not explicitly differentiate between $\theta_t$ and $\theta'_t$; delayed targets may be combined with any of the methods as needed. Furthermore, when working with deep reinforcement learning algorithms, we will express the Q-function and other cumulants as infinite sums of the form $\mathbf{E}[\sum_{\tau=0}^{\infty} \gamma^\tau R_\tau \mid S_0 = s, A_0 = a]$. This is entirely equivalent to the discounted episodic view, in that we simply take $R_\tau = 0$ for all $\tau > H$ within the expectation. Practically, this is achieved by setting $Y_{lH} = 0$ for all $l \in [L]$.

### 6.3.4 Exploration within deep reinforcement learning

To construct a deep reinforcement learning algorithm, it remains to specify how the agent interacts with the environment. This is referred to as an *exploration method*. The standard exploration method within deep reinforcement learning is epsilon-greedy, where

for some choice of $\varepsilon_t \in (0, 1)$, we take

$$A_t \in \arg\max_{a \in \mathcal{A}} Q(S_t, a; \theta_{t-1}) \text{ with probability } 1 - \varepsilon_t,$$

and $A_t \sim \text{Uniform}(\mathcal{A})$ otherwise (used in Mnih et al., 2015). Epsilon-greedy is simple, and for $\varepsilon_t$ bounded away from zero for all $t < T$, it satisfies the conditions required for the convergence of Q-learning and ESARSA (Van Seijen et al., 2009). However, epsilon-greedy is a naïve exploration method. It does not perform deep exploration.

### 6.3.5 Network architecture

The network architecture used in deep reinforcement learning is problem-specific. Here, we describe the architecture used by Mnih et al. (2015) to tackle the Atari Learning Environment (ALE). Our work will use an extension of this standard architecture. First, however, we briefly describe the ALE interface.

The Atari Learning Environment takes integer input that represents an action, and returns an RBG image screen observation and a scalar reward. The following preprocessing is taken as standard:

- The image is scaled down and converted to grayscale, resulting in an $84 \times 84$ `uint8` array. The larger resolution and colour are unnecessary to solve the problems, and this greatly reduces the computational cost of training.
- Only every fourth frame is used, with a maximum taken pixel-wise over the skipped frames and the same action repeated for each. The granularity of selecting actions every frame is too high for standard deep reinforcement learning algorithms.
- Sequences of four consecutive un-skipped frames are stacked together to give the state. This is because the images alone are insufficient to give a Markov state as, for example, a single image may not capture movement.

With that, the architecture of Mnih et al. (2015) consists of:

- A *body* element takes the $4 \times 84 \times 84$ stacked frames, which we take to be a state $s \in \mathcal{S}$, puts these through a three-layer convolutional network, flattens the output and applies a densely connected layer followed by a ReLU activation. This gives an embedding $\varphi(s) \in \mathbb{R}^d$.
- A *head*, a set of weight vectors $w_1, \ldots, w_{|\mathcal{A}|} \in \mathbb{R}^d$, giving the Q-value estimates as

linear combinations $Q(s, a; \theta_t) = \langle \varphi(s), w_a \rangle$.

Here, the parameters $\theta_t$ are the weights $\{w_a\}$ and the trainable parameters of the neural network $\varphi$. Moving forward, we will suppress the dependence of neural network Q-function estimates on $\theta_t$ within our notation and write instead $Q_t(s, a)$, or just $Q(s, a)$ when the relevant time-step is clear from context.

## 6.4 Discussion

We introduced reinforcement learning from two perspectives:

1. One focused on regret, much in line with part I of this thesis, along with solutions based on optimism and posterior sampling. Under this perspective, reinforcement learning is about an online trade-off between exploration and exploitation.
2. A practical perspective, focused on the combination of planning, based on generalised policy iteration, and the training of neural networks. This perspective places little importance on the online interaction aspect of reinforcement learning.

The uneasy fit between problem and solution apparent within the practical perspective has led to most applied reinforcement learning work to be constrained to areas where data is cheap, for example computer games (where interactions require only minimal computation) and biology or chemistry problems (where data is generated synthetically).

In the reminder of this thesis, we look to bridge the gap between the two perspectives, using theoretical insights to develop strong deep reinforcement learning algorithms.

# Chapter 7

# Feature-linear reinforcement learning algorithms

We now look at a number of methods that combine optimism or posterior sampling with linear function approximation and evaluate these on the Deep Sea benchmark (Osband, Aslanides et al., 2018), a set of MDPs with sparse rewards that test for deep exploration. Where relevant, we discuss any challenges in scaling these methods to the deep reinforcement learning setting.

## 7.1 The benchmark: Deep Sea MDPs

The *Deep Sea* benchmark is a set of tree-structured tabular problems with sparse reward, described by Osband, Aslanides et al. (2018) as:

> The environments are indexed by problem scale $H \in \mathbb{N}$ and action mask $W \sim \text{Bernoulli}(0.5)^{H \times H}$, with $\mathcal{S} = \{0,1\}^{H \times H}$ and $\mathcal{A} = \{0,1\}$. The agent begins each episode in the upper left-most corner of a $H$ by $H$ grid and deterministically falls one row per time step. The state encodes the agent's row and column as a one-hot vector. The actions $\{0,1\}$ move the agent left or right depending on the action mask $W$ at state $s \in \mathcal{S}$, which remains fixed throughout interaction. The agent incurs a cost of $0.01/H$ for moving right in all states except for the right-most, in which the reward is 1. The reward for action left is always zero. An episode ends after $H$ time steps so that the optimal policy is to move right each step and receive a total return of 0.99;

all other policies receive zero or negative return.

We consider an agent to have solved a size $H$ deep sea problem when its average per-episode regret drops below 0.9. We present some initial results on Deep Sea in fig. 7.1. The three plots therein depict:

- *Top.* The number of episodes taken by an agent using a uniformly random exploration policy to observe the first positive reward on each problem of size. The left hand side dotted line shows the analytic expected time for the first reward to be observed; this grows exponentially with $H$.

- *Middle.* The number of episodes for RLSVI to solve a Deep Sea MDP versus the problem size (henceforth the *performance*). The right hand side dotted line represents an empirical polynomial fit to the performance of RLSVI.

- *Bottom.* The performance of Q-learning with an epsilon greedy policy. Only crosses appear here: this naïve method failed to solve any of the problems within 2000 episodes. This is due to the small misleading negative rewards, which cause the algorithm to become stuck in a locally optimal left-always policy.

We use the reference implementation of the Deep Sea benchmark included in the *bsuite* benchmark suite (Osband, Doron et al., 2019). For all experiments, linear model parameters are set to $\alpha = 100$ and $\beta = 0.01$. These were found to be optimal in a coarse grid search for the upcoming 'optimism via reward bonuses' method; other methods tested were significantly less sensitive to choice of these parameters. Where available, the prior mean parameter $m$ is set to zero. We discuss this choice in section 7.4, as part of a broader commentary on experimental methodology within reinforcement learning.

## 7.2 Intrinsic motivation methods

Intrinsic motivation is an approach to exploration based on rewarding the agent for visiting novel states (Schmidhuber, 1991). While frequently introduced through the lens of neuroscience or psychology, with novel behaviour sought out of boredom, we will view these as a methods for achieving optimism. In this section, we describe a generic implementation of a reward bonus–based agent first, and then look at an improvement on that methodology: the Uncertainty Bellman Equation (UBE, O'Donoghue et al., 2018).

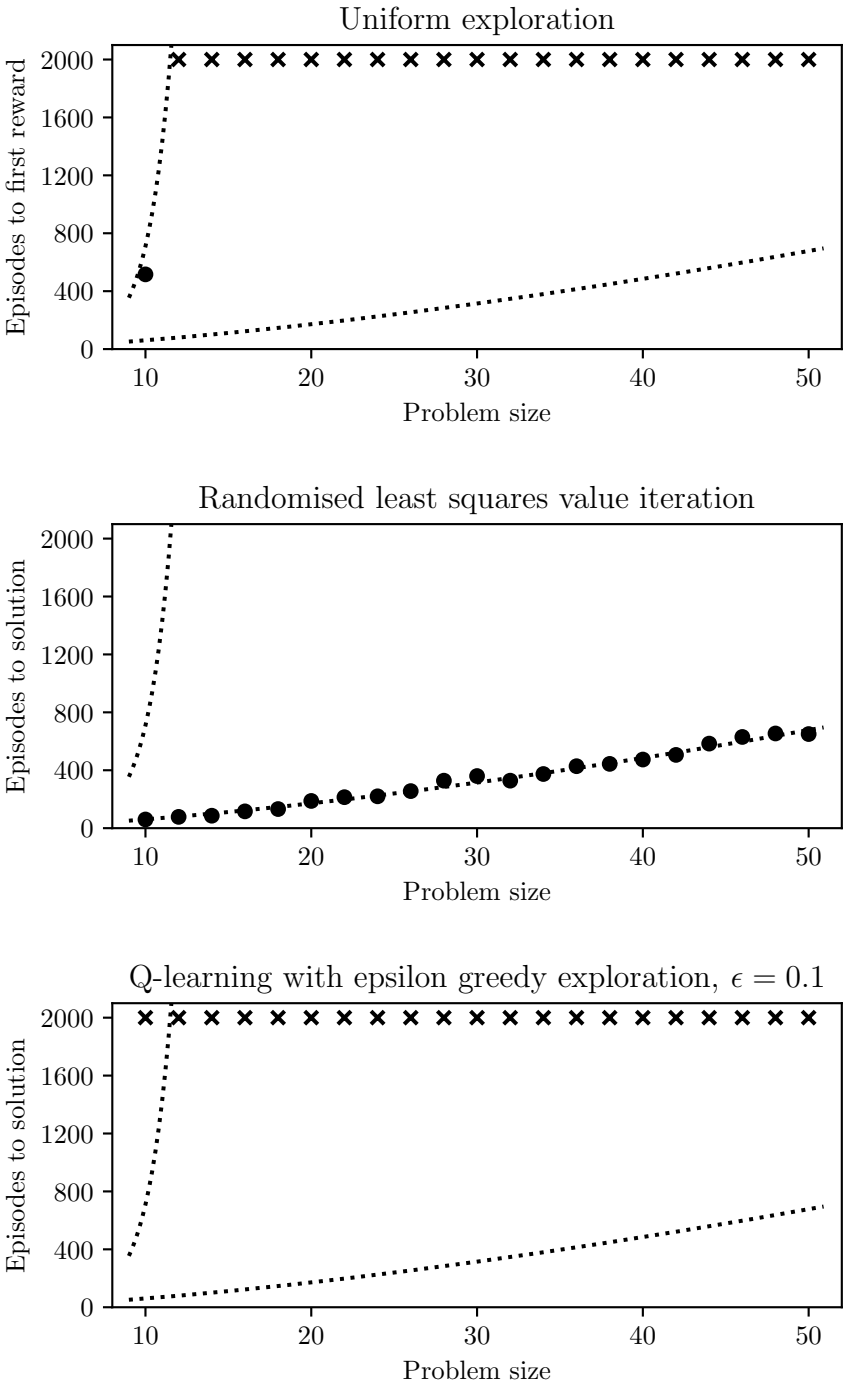



Figure 7.1: Deep Sea performance with a single seed for each method. Points denote number of episodes to first reward (top)/solution (middle, bottom) for each problem size; crosses denote runs that failed to find positive reward/solve problem within 2000 episodes. Dotted lines depict analytic time to first reward for a uniform policy and an empirical fit to the performance of RLSVI respectively.

### 7.2.1 Optimism via reward bonuses

Optimism via reward bonuses is a broad category that captures many intrinsic motivation methods where an agent performs planning on a combination of the extrinsic reward, given by the MDP, and an intrinsic reward, defined as part of the algorithm. That is, the agent has a secondary intrinsic reward stream $\tilde{R}_t = \tilde{r}(S_t, A_t)$ and acts greedily with respect to

$$U^\pi(s,a) = Q^\pi(s,a) + \mathbf{E}\left[\sum_{t=0}^{\infty} \gamma^t \tilde{r}(S_t, A_t) \mid S_0 = s, A_0 = a\right],$$

where the expectation is with respect to transitions given by an empirical transition model, the extrinsic and intrinsic rewards respectively, and a policy $\pi$. Here, in an attempt to establish optimism for $Q^\star_M$, we take $\pi$ to be a policy that maximising $U^\pi(s,a)$ across $\mathcal{S} \times \mathcal{A}$; the resulting $U^\pi$ can be computed using, for example, value iteration. In practice, Q-learning is often used for this purpose.

Methods in this category differ in the choice of the intrinsic reward signal. Within tabular problems, the choice of the intrinsic rewards is straightforward. We take $\tilde{r}(s,a) = \nu(s,a)^{1/2}$, for $\nu$ a *local variance* of the form

$$\nu(s,a) = (\alpha^{-1} + \beta^{-1} n_t(s,a))^{-1} \quad \text{for} \quad n_t(s,a) = \sum_{\tau<t} \mathbb{1}\{S_t = s, A_t = a\},$$

and parameters $\alpha, \beta > 0$.[1] We can extend this heuristically to the linear function approximation setting, where the Q-function is modelled as linear in some features $\varphi\colon \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}^d$, by taking

$$\nu(s,a) = [\Sigma]_{(s,a),(s,a)} \quad \text{for} \quad \Sigma = \left(\alpha^{-1} I + \beta^{-1} \sum_{\tau<t} \varphi(S_\tau, A_\tau)\varphi(S_\tau, A_\tau)^T\right)^{-1}.$$

When $\varphi(s,a) = e_{(s,a)}$, a one-hot encoding of $\mathcal{S} \times \mathcal{A}$, the two local variances are equivalent.

We present the performance of the tabular version of this algorithm in Figure 7.2 (top). While the method significantly outperforms uniform random exploration, it does not perform nearly as well as RLSVI. This is in line with results shown in the appendix of Osband, Van Roy and Wen (2016), which compare RLSVI with a reward bonus–based method with a different but similar intrinsic reward signal on a related set of problems.

[1] We ignore polylogarithmic terms throughout part II.

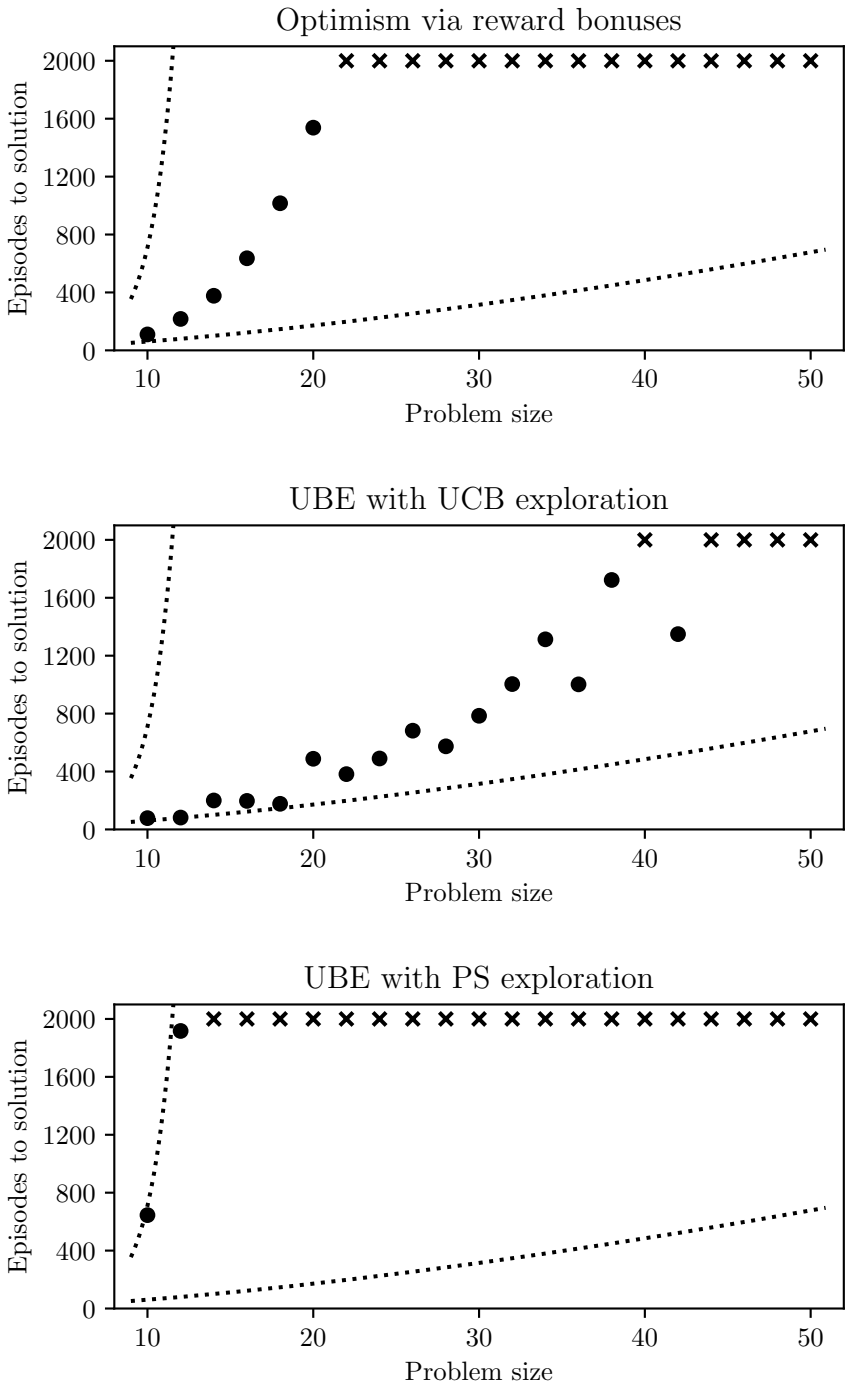



Figure 7.2: Deep Sea performance with a single seed for each method. Points denote number of episodes to solution for each problem size; crosses denote runs that failed to solve problem within 2000 episodes. Dotted lines depict analytic time to first reward for a uniform policy and an empirical fit to the performance of RLSVI respectively.

### 7.2.2 The Uncertainty Bellman Equation

The Uncertainty Bellman Equation (UBE, O'Donoghue et al., 2018) combined with UCB-style methodology explores using a policy greedy with respect to

$$U^\pi(s,a) = Q^\pi(s,a) + \sqrt{b^\pi(s,a)} \quad \text{for} \quad b^\pi(s,a) = \mathbf{E}\left[\sum_{\tau=0}^{\infty} \gamma^{2\tau} \nu(S_\tau, A_\tau) \mid S_0 = s, A_0 = a\right],$$

where the local variance $\nu$ and the policy $\pi$ are taken as before. The difference between the $U^\pi$ given by reward bonuses and by UBE is a change of the order of the square root operation with the summation and the expectation. O'Donoghue et al. (2018) justify this exchange of operations formally using the conditional Jensen's inequality.

To appreciate the consequences of the change introduced in UBE, consider the following two ways of establishing a constant probability bound for the sum of $R_0, \ldots, R_{H-1}$, each an independent standard normal random variable, using a Cramér-Chernoff bound:

1. For all $h < H$, with probability $1 - \frac{\delta}{H}$, $R_h \leq \sqrt{2 \log \frac{H}{\delta}}$. Thus, by the union bound, with probability no less than $1 - \delta$, $\sum_{h<H} R_h \leq H\sqrt{2 \log \frac{H}{\delta}}$.

2. On the other hand, since $\sum_{h<H} R_h$ is itself distributed as $\mathcal{N}(0, H)$, we can bound the sum directly. We obtain that with probability at least $1-\delta$, $\sum_{h<H} R_h \leq \sqrt{2H \log \frac{1}{\delta}}$.

These correspond in turn to the standard bonuses method and UBE when these are applied to a tree-structured MDP with $H$ states. The bound resulting from the UBE-type approach is on the order of $\iota\sqrt{H}$ tighter. Empirical evaluation of UBE with UCB bonuses on Deep Sea, shown in fig. 7.2 (middle), shows UBE performing better in practice.

However, when introducing UBE, O'Donoghue et al. (2018) combine the model not with UCB-style exploration but with posterior sampling. The resulting algorithm is an RVF-type method with $P_Q^l = \mathcal{N}\big(Q^\pi, \mathrm{diag}(\sqrt{b^\pi})\big)$ where $\pi$ is taken to be a policy maximising $Q^\pi$ (the Q-function derived from extrinsic rewards only). For each episode $l \in [L]$, the agent then interacts using a policy $\pi_l \in GQ$ for $Q \sim P_Q^l$. As shown in fig. 7.2 (bottom), this procedure *fails to outperform uniform random exploration.* This is unsurprising, considering that this UBE RVF method models the Q-function as independent across different state-action pairs. Indeed:

**49 Theorem.** *Suppose $|\mathcal{A}| > 1$ and let $P_Q$ be a factorised distribution. That is, for $Q \sim P_Q$, $Q(s,a)$ and $Q(s',a')$ are independent for all $(s,a) \neq (s',a')$. Assume further that for each $s \in \mathcal{S}$, the marginal distributions of $\{Q(s,a) \colon a \in \mathcal{A}\}$ are symmetric around*

*some $c_s \in \mathbb{R}$. Then the probability of executing any given sequence of $H$ actions under $\pi \in GQ$ is at most $2^{-H}$.*

This result shows UBE with posterior sampling will incur regret exponential in $H$ on a version of Deep Sea where all negative rewards are set to zero (this modification leads to the symmetry required in the theorem). The inclusion of negative rewards only makes the performance of UBE with posterior sampling worse.

*Proof of theorem 49.* For state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$, consider the events

$$E = \bigcap_{b \in \mathcal{A} \setminus \{a\}} \{Q \colon Q(s,a) > Q(s,b)\} \quad \text{and} \quad \tilde{E} = \bigcap_{b \in \mathcal{A} \setminus \{a\}} \{Q \colon Q(s,a) < Q(s,b)\}.$$

By symmetry, $\mathbf{P}(E) = \mathbf{P}(\tilde{E})$. Moreover, since $E$ and $\tilde{E}$ are disjoint, $\mathbf{P}(E) + \mathbf{P}(\tilde{E}) \leq 1$. Hence $\mathbf{P}(E) \leq \frac{1}{2}$. Now note $E$ is the event where a given action $a$ is selected in a state $s$. By assumption, $Q(s,a)$ and $Q(s',a')$ are independent for $(s,a) \neq (s',a')$. Thus the probability of executing a sequence of $H$ specific actions is at best (that is, under deterministic transitions) the product of the probabilities of executing a single desired action, and so upper bounded by $2^{-H}$. □

## 7.3 Bayesian-motivated approaches

Bayesian modelling offers an alternative perspective on the design of reinforcement learning algorithms, and has a history dating back to at least Dearden et al. (1998) and Strens (2000). Here, we look at the classic GP-SARSA algorithm (Engel et al., 2005), and Bayesian Deep Q-Networks (BDQN, Azizzadenesheli et al., 2018), an attempt at combining Bayesian modelling with the DQN neural network function approximation architecture. We end on a discussion of a broader class of methods that (unsuccessfully) combine Bayesian models with DQN, which includes BDQN and other popular heuristics.

### 7.3.1 GP-SARSA, an explicit Bayesian approach

GP-SARSA combines a Gaussian process prior for the Q-function with a Gaussian likelihood for rewards and SARSA-style targets to give an on-policy Q-function model that can be used with either posterior sampling or optimism (Engel et al., 2005). Here, we use the weight-space perspective on Gaussian process regression (discussed in chapter 2) to present a finite-dimensional feature-linear version of GP-SARSA. We also extend

this model to an off-policy setting, deriving a GP-ESARSA method, and examine the scalability issues that emerge.

The GP-SARSA model assumes that $Q(s,a) = \langle w, \varphi(s,a)\rangle$ for a given embedding function $\varphi\colon \mathcal{S}\times\mathcal{A}\mapsto\mathbb{R}^d$ and weights $w\in\mathbb{R}^d$ with a prior $\mathcal{N}(0,\alpha I)$, $\alpha>0$, on $w$. It then models the observed rewards as generated by the process

$$R_h = \begin{cases} Q(S_h,A_h) - \gamma Q(S_{h+1},A_{h+1}) + \varepsilon_h & h+1<H \\ Q(S_h,A_h)+\varepsilon_h & \text{otherwise,}\end{cases}$$

where each $\varepsilon_h$ is an independently and identically distributed $\mathcal{N}(0,\beta)$ random variable for some $\beta>0$. Writing $\psi_\tau = \varphi(S_\tau,A_\tau) - \gamma_\tau\varphi(S_{\tau+1},A_{\tau+1})$, where $\gamma_\tau\in\gamma,0$ takes value 0 on terminal steps, we have that $R_\tau = \langle w,\psi_\tau\rangle + \varepsilon_\tau$. We can thus interpret GP-SARSA as a conjugate linear Gaussian model for the observed rewards, and so $P_w^l = \mathcal{N}(\mu_w,\Sigma_w)$ with

$$\mu_w = \Sigma_w^{-1}\sum_{\tau<lH}\psi_\tau R_\tau \quad\text{and}\quad \Sigma_w = \left(\beta^{-1}\sum_{\tau<lH}\psi_\tau\psi_\tau^T + \alpha^{-1}I\right)^{-1}.$$

Notably, we can compute $P_w^{l+1}$ from $P_w^l$ using only order $d^2$ operations, independently of both $t$ and $|\mathcal{S}||\mathcal{A}|$. GP-SARSA is thus scalable. $P_w^l$ in turn induces a posterior over Q-functions, $P_Q^l = \mathcal{N}(\mu_Q,\Sigma_Q)$ where

$$\mu_Q(s,a) = \langle\mu_w,\varphi(s,a)\rangle \quad\text{and}\quad [\Sigma_Q]_{(s,a),(s',a')} = \langle\varphi(s,a)\varphi(s',a')^T,\Sigma_w\rangle_F,$$

with point-wise evaluation costs that scale with $d$ only.

On the Deep Sea benchmark, GP-SARSA with RVF exploration significantly outperforms uniform random exploration, but does not match the performance of RLSVI (fig. 7.3, top). This result is to be expected, given the previously outlined theoretical framework for RVF methods. Since GP-SARSA models $Q_M^{\pi'}$ for $\pi'$ the implicit policy induced by the combination of RVF exploration and SARSA targets, and not $Q_M^\star$, the optimal value function, it fails to be stochastically optimistic for $Q_M^\star$.

We may attempt to fix GP-SARSA by extending it to compute a posterior for any chosen policy $\pi\in\Pi$, taking $R_\tau = \langle w,\psi_\tau^\pi\rangle + \varepsilon_\tau$ with $\varepsilon_\tau$ as before and $\psi_\tau^\pi = \varphi(S_\tau,A_\tau) - \gamma_\tau\langle\pi(S_{\tau+1}),\varphi(S_{\tau+1},\cdot)\rangle$, and applying some form of policy iteration to compute a posterior over plausible optimal value functions. The resulting $P_w^l$ is then of the same form as in GP-SARSA, but with $\psi_\tau^\pi$ replacing $\psi_\tau$. We refer to this as GP-ESARSA.

However, policy evaluation in GP-ESARSA is too expensive to be useful. For a new $\pi$, each element $\psi_0^\pi, \ldots, \psi_{lH-1}^\pi$ summed over within the covariance needs to be updated, leading to an order $t$ cost. We can avoid this scaling with $t$ by working directly with the induced posterior over Q-functions, which we denote here as $P_{Q^\pi}^l$. Straightforward linear algebra shows that this is of the form $\mathcal{N}(\mu_Q^\pi, \Sigma_Q^\pi)$ for

$$\mu_Q^\pi = \Sigma_Q^{-1}(I - \gamma\bar{P}^\pi)^T\Phi^T R \quad \text{and} \quad \Sigma_Q^\pi = \left[\alpha^{-1}I + \beta^{-1}(I - \gamma\bar{P}^\pi)^T\Phi^T\Phi^\pi(I - \gamma\bar{P}^\pi)\right]^{-1}, \tag{7.1}$$

where $R = [R_0, \ldots, R_{lH-1}]^T$, $\Phi = [\varphi(S_0, A_0), \ldots, \varphi(S_{lH-1}, A_{lH-1})]$ and $\bar{P}^\pi$ is the empirical state-action transition matrix for the policy $\pi$, as defined in section 6.3.1. But this alternative approach has a computational complexity that grows with $|\mathcal{S} \times \mathcal{A}|$, and so still does not scale beyond the tabular setting.

### 7.3.2 Bayesian Deep Q-Networks

Bayesian Deep Q-Networks (BDQN, Azizzadenesheli et al., 2018) combines approximate conjugate Gaussian linear regression with Deep Q-Networks to give a neural-linear Q-function model. While it is not a good model (see results in fig. 7.3, middle & bottom), analysing it will provide useful insights for our later work, and reveal weaknesses common amongst other 'pseudo-Bayesian' deep reinforcement learning methods.

BDQN models the Q-function as $Q(s, a) = \langle w_a, \varphi(s)\rangle$ where $\varphi\colon \mathcal{S} \mapsto \mathbb{R}^d$ is a neural network (canonically the body of a DQN network) and $w_1, \ldots, w_{|\mathcal{A}|} \in \mathbb{R}^d$ are independent random vectors with $w_a \sim \mathcal{N}(\mu_a, \Sigma_a)$ for each $a \in \mathcal{A}$, with

$$\Sigma_a = \left(\alpha^{-1}I + \beta^{-1}\sum_{\tau<t} \mathbb{1}\{A_\tau = a\}\varphi(S_\tau)\varphi(S_\tau)^T\right)^{-1}$$

for parameters $\alpha, \beta > 0$ and with $\mu_a$ given by minimising the squared Bellman loss

$$\mathcal{L}(s, a, r, s') = (\langle w_a, \varphi(s)\rangle - r - \gamma Y(s'))^2 \quad \text{with} \quad Y(s') = \max_{b\in\mathcal{A}}\langle \mu_b, \varphi(s')\rangle$$

with respect to $\{\mu_a\}$ and the parameters of $\varphi$, in expectation over observations $(s, a, r, s')$ sampled from a replay buffer and $\{w_a\}$ sampled according to their respective models.

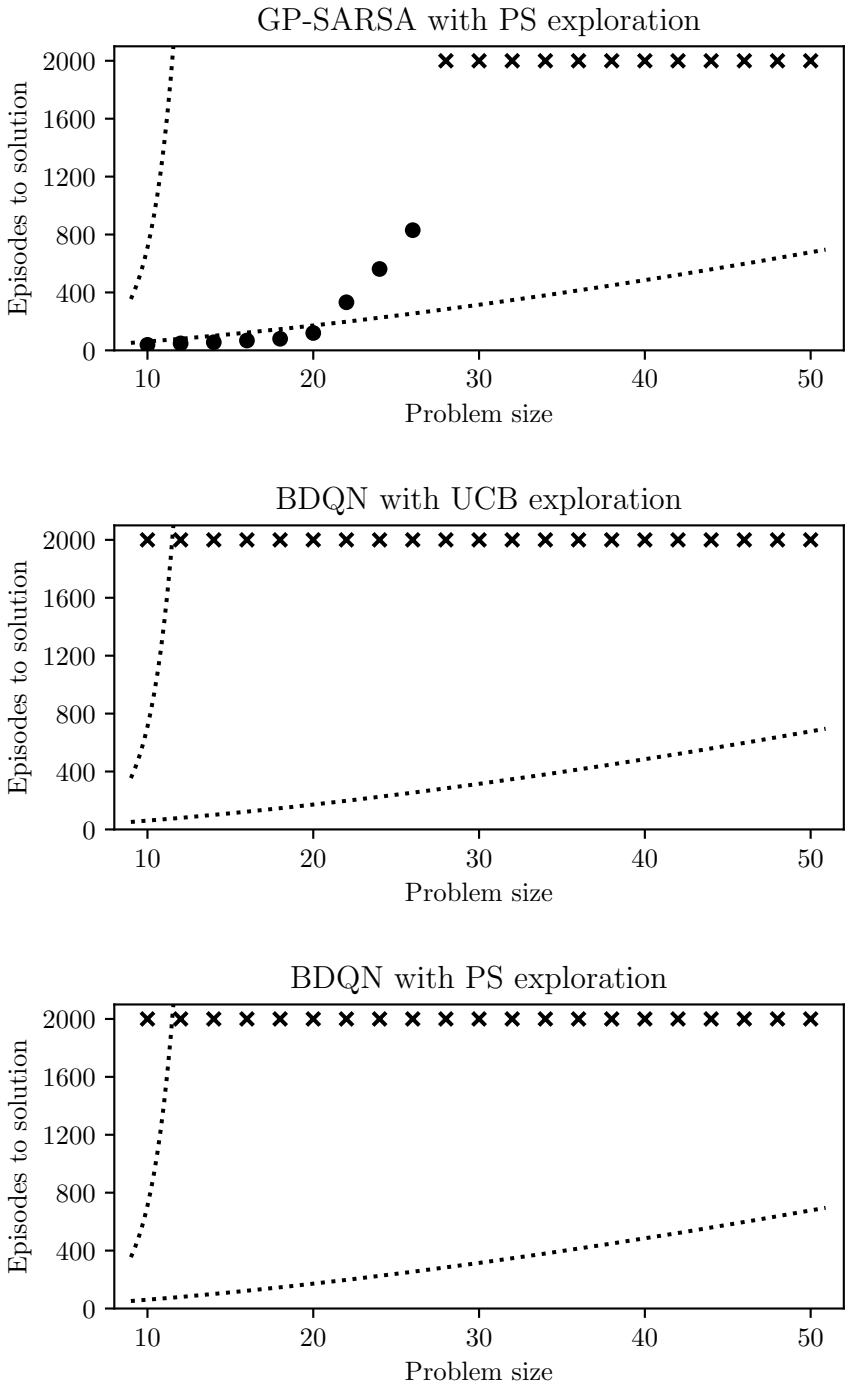



Figure 7.3: Deep Sea performance with a single seed for each method. Points denote number of episodes to solution for each problem size; crosses denote runs that failed to solve problem within 2000 episodes. Dotted lines depict analytic time to first reward for a uniform policy and an empirical fit to the performance of RLSVI respectively.

The resulting posterior is $P_Q = \mathcal{N}(\mu_Q, \Sigma_Q)$ with

$$\mu_Q(s,a) = \langle \mu_a, \varphi(s) \rangle \quad \text{and} \quad [\Sigma_Q]_{(s,a),(s',a')} = \begin{cases} \langle \varphi(s)\varphi(s')^T, \Sigma_a \rangle_F, & a = a' \\ 0, & \text{otherwise,} \end{cases}$$

and the overall model can be described as approximate conjugate Gaussian regression for the Q-function performed independently for each action.

We show the performance of BDQN with optimistic exploration and RVF-style posterior sampling in fig. 7.3 (middle & bottom respectively). Both fail to solve any of the Deep Sea MDP sizes tested. This result holds empirically irrespective of the network architecture or initialisation used for $\varphi$, and of any network or model hyperparameters. We now examine why.

Consider first using the BDQN model structure with the embedding function fixed to be the one-hot embedding $\varphi(s) = e_s$, combined with:

- *Posterior sampling.* The resulting $P_Q$ is diagonal, and so $Q(s,a)$ and $Q(s',a')$ are independent for $(s,a) \neq (s',a')$. The Q-function posterior is also symmetric and centred (ignoring the effects of the small negative rewards), and so an application of theorem 49 on page 136 gives regret exponential in $H$.
- *Optimism.* Here, we use a policy greedy with respect to

  $$U(s,a) = \mu_Q(s,a) + [\Sigma_Q]^{1/2}_{(s,a),(s,a)}.$$

  On Deep Sea problems, prior to finding the positive reward and ignoring negative rewards, $\mu_Q(s,a) = 0$ for all $(s,a) \in \mathcal{S} \times \mathcal{A}$. Thus the algorithm acts greedily with respect to $[\Sigma_Q]^{1/2}_{(s,a),(s,a)}$, which with tabular embeddings this is equal to $(\alpha^{-1} + \beta^{-1} n_t(s,a))^{-1/2}$. The agent therefore explores in a quasi-uniform manner, incurring regret exponential in $H$.

In both cases, the failure of the algorithm can be understood as a failure to propagate uncertainty information from each step $h$ to $h-1$ within the MDP, for each $h < H$. In particular, the optimistic version looks like the standard optimism via reward bonuses method but where reward bonuses are not summed over the time steps.

Now consider the full BDQN algorithm with a neural network $\varphi \colon \mathcal{S} \mapsto \mathbb{R}$ and weights $\{\mu_a\}$, both trained to minimise the given squared Bellman loss. Now:

- *Posterior sampling.* Take an MDP with $|\mathcal{S}| = |\mathcal{A}| > 1$. Assume each state has a unique optimal action,[2] and denote by $a^\star \colon \mathcal{S} \mapsto \mathcal{A}$ the map taking states to corresponding optimal actions. Then since $\{w_a\}$ are independent random vectors and each optimal action is unique, $Q(s, a^\star(s))$ is independent of $Q(s, a^\star(s'))$ for all $s \neq s'$, and so by theorem 49 the regret incurred is exponential in $H$. Looking back at our Deep Sea MDPs, the mask $W$ acts to randomise the assignments between states and optimal actions, which causes BDQN to fail.

- *Optimism.* The problem in here is in the loss function. Writing $\tilde{Y}(s', r) = r + \gamma Y(s')$, we can decompose expected loss into

$$\mathbf{E}\mathbf{E}_Q\left[(Q(s,a) - \tilde{Y}(s',r))^2\right] = \underbrace{\mathbf{E}[(\mu_Q(s,a) - \tilde{Y}(s',r))^2]}_{\text{mean term}} + \underbrace{\mathbf{E}\text{Var}_Q(Q(s,a))}_{\text{variance term}},$$

  where the outer expectations are over $(s, a, r, s')$. Examining the two resulting terms: the mean term encourages the posterior mean to match the targets $\tilde{Y}(s', r)$—this is reasonable; the variance term acts to penalise the posterior variance on the previously observed locations—this is undesirable. With sufficient optimisation, the variance term causes the exploration bonus for any previously observed state-action pair to drop to near zero independently of the number of times it had been observed. Failure on the Deep Sea MDPs follows: once the bonus for action right in a given state drops below $0.01/H$, the algorithm can no longer solve the problem.

Both of these failure modes are fixable. For the first, change the Q-function model so that $Q(s, a) = \langle w, \varphi(s, a)\rangle$ for $\varphi \colon \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}^d$ and $w \in \mathbb{R}^d$ with $P_w = \mathcal{N}(\mu_w, \Sigma_w)$ for $\Sigma_w$ an appropriately chosen dense matrix. For the second, discard the variance term in the loss. However, there remains a further issue. Any function $\mu_Q$ that can be realised by either model—with $\varphi \colon \mathcal{S} \mapsto \mathbb{R}^d$ and $\{\mu_a\}$, or with $\varphi \colon \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}^d$ and a single $\mu_w \in \mathbb{R}^d$—can be realised by infinitely many choices of $\varphi$. While many of these choices may lead to a low average Bellman loss, not all will be suitable for exploration—after all, $\varphi$ given by a one-hot encoding can achieve zero 'mean term' loss while exploring no better than a uniformly random policy. The model is under-constrained. In the next chapter, when we develop Successor Uncertainties, we will take the first two fixes for BDQN as our starting point. Our main contribution will then be in the introduction of additional structure to constrain $\varphi$, which we show induces effective exploration.

[2] We say an action $a$ is optimal in state $s$ if there exists a $\pi^\star \in GQ^\star_M$ such that $\pi^\star(s)(a) > 0$.

### 7.3.3 Other pseudo-Bayesian deep RL methods

BDQN is one of many methods that combine Bayesian modelling with the DQN framework in an attempt to improve exploration. For example, Gal (2016) uses the dropout-based heuristic of Gal and Ghahramani (2016) in the DQN network; Y. Tang et al. (2017) use 'powerful variational inference subroutines' for DQN network parameters under Gaussianity assumptions; Moerland et al. (2017) use conjugate Gaussian linear regression; Lipton et al. (2018) use a Bayes-by-Backprop neural network (Blundell et al., 2015); Touati et al. (2019) use multiplicative flows (Louizos et al., 2017). These pseudo-Bayesian methods do not lead to deep exploration. The specific pitfalls vary, but they emerge from the treatment of reinforcement learning as an application of supervised learning, rather than as a problem that happens to use regression within its solutions. These methods replace the regression element without considering the broader problem.[3]

## 7.4 Deep reinforcement learning that matters

The failure of UBE with posterior sampling and BDQN with both exploration methods considered raises the question of why these and other similar methods are popular within the literature. This may be in part due to the way in which deep reinforcement learning algorithms are evaluated, a topic that has seen much discussion since the publication of 'Deep reinforcement learning that matters' (Henderson et al., 2018). We add to that conversation with some observations regarding the Deep Sea benchmark and similar hard-exploration problems, and discuss the ALE benchmark.

**Deep Sea** We are strong proponents of Deep Sea MDPs and other such tabular edge-case problems, exemplified by the *bsuite* benchmarks (Osband, Doron et al., 2019), as a means of fault-finding for reinforcement learning algorithms. However, both UBE and BDQN have been tested on hard exploration benchmarks and passed. Why?

First, O'Donoghue et al. (2018) test UBE with optimism on tabular problems, but then proceed to use UBE with posterior sampling on ALE. It is thus important to ensure that the model and method tested on the tabular baselines is as close as possible to the one used on the full problem. In this particular case, posterior sampling is in some sense harder than optimism, since it relies on the entire covariance matrix and not just its diagonal entries.

[3]Some of these have been previously criticised in Osband, Aslanides et al. (2018).

As an aside, this desire for consistency between tabular and deep implementations explains our decision to set $m = 0$ for our experiments. While theoretical results for RLSVI require $m = 1$, the method solves the Deep Sea benchmark without this mean optimism. Meanwhile, methods that combine the RVF framework with deep reinforcement learning frequently use a standard Q-network for the mean value prediction, as in UBE, and have no mechanism for optimism therein. Since our focus is to develop algorithms that work effectively in the deep learning setting, we avoid relying on mechanisms that do not translate directly to that setting.

Second, results showing BDQN and other pseudo-Bayesian DQN methods solve hard tabular exploration tasks (as in, for example, figure 1 of Touati et al., 2019) almost invariably involve poorly designed or implemented benchmarks. In the cited figure, for example, the authors most likely fail to randomise the index of the optimal action in each state (that is, did not include the matrix $W$, as specified in the description of the Deep Sea MDPs). Not randomising optimal actions within tabular benchmarks (as also done in, for example, Osband, Blundell et al., 2016; Plappert et al., 2018) can lead to rogue solutions, exemplified by the following:

**50 Theorem.** *Let $Q(s, a) = \langle w_a, \varphi(s) \rangle$ with $\varphi(s) = \phi(Ue_s) \in \mathbb{R}^d$ for $\phi$ a strictly positive activation function (for example sigmoid) applied elementwise and any weights $U \in \mathbb{R}^{d \times |\mathcal{S}|}$. Then sampling $w_a \sim \mathcal{N}(0, \alpha I)$ for $\alpha > 0$ independently solves a size $H$ Deep Sea MDP without randomised optimal actions in $L \leq -\log_2(1 - 2^{-d})^{-1}$ median number of episodes.*

Thus, a one-layer BDQN model with a single neuron followed by a sigmoid activation function solves a size $H$ Deep Sea MDP without randomised optimal actions in a median time of one episode, independently of $H$. This highlights the importance of sanity-checking results—for example, establishing a priori the performance we might expect from a strong method, like RLSVI, and asking questions if the proposed method exceeds it by a large margin.

*Proof of theorem 50.* Assume action 0 is optimal in all states and define $\Delta \doteq w_0 - w_1$. Action 0 is selected in state $s \in \mathcal{S}$ if $Q(s, 0) - Q(s, 1) = \langle \varphi(s), \Delta \rangle > 0$. By the strict positivity of $\phi$, the probability that 0 is selected for all $s^\star_0, \ldots, s^\star_{H-1}$, the sequence of states needed to reach the rewarding state, is lower bounded as

$$\mathbf{P}\left[\bigcap_{h<H} \{\hat{Q}(s^\star_h, 0) > Q(s^\star_h, 1)\}\right] \geq \mathbf{P}(\Delta > 0)\, \mathbf{P}\left[\bigcap_{h<H} \{\langle \varphi(s^\star_h), \Delta \rangle > 0\} \mid \Delta > 0\right] = \mathbf{P}(\Delta > 0),$$

where $\Delta > 0$ is to be interpreted elementwise. Since $\Delta \sim \mathcal{N}(0, 2\alpha I)$, $\mathbb{P}(\Delta > 0) = 2^{-d}$ for all $H \in \mathbb{N}$. The result follows by using the expression for the median of a geometric distribution. □

**Atari Learning Environment** ALE is a near-inescapable part of benchmarking general deep reinforcement learning algorithms. We believe, however, that algorithms should not be benchmarked *just* on ALE, and indeed that ALE is not a particularly good benchmark altogether:

- The ALE benchmark is expensive. We estimate that running standard DQN for the 49 games included in the standard ALE benchmark requires approximately 8,000 GPU hours on an NVIDIA RTX 1080Ti GPU with a 3.2GHz CPU. On the other hand, the performance of deep reinforcement learning algorithms can vary greatly between runs and many random seeds are required to confidently estimate their performance (Henderson et al., 2018).[4] For this reason, ALE experiments are not widely reproducible.

- ALE experiments are complex and not well standardised. ALE has many settings and is available through multiple wrappers, each with different defaults, and the output of ALE is heavily pre-processed on the user-side before being passed to the algorithm.[5] Furthermore, the way in which hyperparameters are chosen (for example, which games are used in a grid search) and how scores reported vary between papers. These factors introduce room for error and make the comparison of ALE scores across the literature difficult.

- Strong scores on ALE mean only that the specific implementation, hyperparameter and model combination performs well on ALE; it is hard to disentangle the effects of the proposed contribution from the other aspects. Even where the effect is indeed due to the proposed contribution, it is difficult to establish whether the benefit occurs through the mechanism proposed by the authors. These issues are amplified by the cost of ALE experiments, which generally keep the use of any controls—for example, running same implementation of an exploration algorithm but with an

[4]We have had a reviewer ask for 10 seeds in place of 3. While we agree with the reviewer that using 3 seeds is insufficient, a back-of-the-envelope calculation puts the cost of running 10 seeds using Amazon Web Services EC2 at over £50,000.

[5]Even applying the pre-processing steps in the wrong order can lead to very different results—for example, on Space Invaders, 'lasers' are only visible every four frames (Mnih et al., 2015), and interchanging two of the standard preprocessing steps can result in the loss of this important signal.

epsilon-greedy policy—to a minimum.

In the context of exploration, we can alleviate some of these issues by using the classification of ALE games by Ostrovski et al. (2017), which splits these into categories including 'hard exploration with dense rewards' and 'hard exploration with sparse rewards', and focusing our efforts on these hard exploration tasks. However, such insights into the ALE benchmark are not readily available for other aspects of deep reinforcement learning.

## 7.5 Discussion

In this chapter, we reviewed a number of methods that use a feature-linear structure and conjugate Gaussian regression to drive exploration. We did not, however, provide a broad overview of the many approaches to exploration in deep learning. Our key omissions are:

- Other intrinsic bonuses/uncertainty quantification methods. Methods considered in the literature include density models applied to the raw image observation (M. Bellemare et al., 2016; Ostrovski et al., 2017), hashing methods applied to auto-encoder-based state representations (H. Tang et al., 2017), and bonuses based on prediction errors (Houthooft et al., 2016; Stadie et al., 2015; Pathak et al., 2017), including those based on the prediction of the output of a random, fixed Q-function network (Burda et al., 2018).

- Ensembling methods. These are RVF methods that define the sampling distribution for posterior sampling as uniform on a discrete set of Q-function estimators, each perturbed in some manner (Osband, Van Roy, D. J. Russo et al., 2019). The deep reinforcement learning implementation of these, Bootstrapped DQN (Osband, Blundell et al., 2016) and Prior Functions (Osband, Aslanides et al., 2018), are amongst the strongest practical exploration methods in the literature, with performances roughly matching those of GP-SARSA and RLSVI respectively. Ensembling methods are closely related to linear Q-function models; this connection is discussed in Osband, Van Roy, D. J. Russo et al. (2019).

- Information directed sampling (IDS, D. Russo et al., 2014). IDS is an exploration framework based on estimating the amount of information to be gained about $Q^\star_M$ by selecting each action, and explicitly trading it off against an estimate of the regret incurred by taking that choice. IDS is superior to optimism and posterior sampling methods for problems that include complex structure,[6] but we are not

aware of any reason to prefer IDS to optimism in the setting of this thesis. IDS has been previously evaluated on ALE using ensembling-based uncertainty estimates (Nikolov et al., 2018).

In the next chapter, we build on the methods described so far to construct a simple yet effective exploration method based on a feature-linear Q-function model.

[6]Consider a pure exploration problem with a set of $k \in \mathbb{N}$ *choices*, where all one choice has reward 1 and the rest reward 0. The set of arms is then the powerset of choices and the reward for each arm is the average of the rewards of the selected choices. An optimal strategy would perform binary search for an order $\log k$ time to optimal reward; IDS can recover this approach, at least in theory. In contrast, any arm containing more than a single choice has a priori zero probability of being optimal and will therefore never be selected under posterior sampling. Posterior sampling therefore takes on average $k/2$ time-steps to find the optimal arm. See D. Russo et al. (2014) for more such examples.

# Chapter 8

# Successor Uncertainties

This chapter introduces the main contribution of part II: Successor Uncertainties (SU), a neural-linear value function model that, when combined with posterior sampling, offers strong performance on both hard tabular problems and on the Atari Learning Environment. The SU model takes the form

$$Q_l^\pi(s,a) = \langle w_l, \varphi_l^\pi(s,a)\rangle \quad \text{with} \quad w_l \sim \mathcal{N}(\mu_w^l, \Sigma_w^l) \quad (\forall l < L, \forall \pi \in \Pi)$$

for weights $w_l \in \mathbb{R}^d$ and embeddings $\varphi_l^\pi(s,a) \in \mathbb{R}^d$ (which will be estimated by a neural network). Recall this means that at the start of the $l$th episode, the agent will sample a set of weights $w_l$ from the posterior $\mathcal{N}(\mu_w^l, \Sigma_w^l)$, computed using observations from all previous episodes, and follow a policy greedy with respect to the induced Q-function $Q_l^\pi$, where the policy $\pi$ (used to compute the embeddings $\varphi_l^\pi$) remains to be specified. The form of the SU model is motivated by our analysis of BDQN insection 7.3.2.

## 8.1 The successor feature structure

The key insight separating SU from previous works is the introduction of a successor representation/feature constraint for the embeddings $\varphi_l^\pi$ (Dayan, 1993; Barreto et al., 2017). We begin by showing the necessity of this constraint:

**51 Theorem.** *Under the SU model structure, and assuming that for each $l \in [L]$, (I) conditionally on the history $\mathcal{H}_{lH}$, the implied posterior over rewards is independent of*

*the policy $\pi$, and (II) $\Sigma_w^l$ is positive definite, we have that for some $\psi\colon \mathcal{S}\times\mathcal{A}\mapsto\mathbb{R}^d$,*

$$\varphi_l^\pi(x) = \mathbf{E}\left[\sum_{h<H}\gamma^h\psi(X_h) \mid X_0 = x\right]$$

*for all $x\in\mathcal{S}\times\mathcal{A}$, where $X_1,\dots,X_{H-1}\in\mathcal{S}\times\mathcal{A}$ are random variables with law given by the interaction of $\pi$ and some transition model $\mathcal{P}_l$.*

The embeddings $\varphi_l^\pi$ are therefore the expected discounted sum of some local state-action embeddings $\psi$. This is known as a successor representation (Dayan, 1993) or, in the context of deep reinforcement learning, successor features (Barreto et al., 2017).

*Proof.* With the chosen model structure, for all $l\in L$ and a policy $\pi\in\Pi$, the mean reward for a state-action pair $(s,a)\in\mathcal{S}\times\mathcal{A}$ given $\mathcal{H}_{lH}$ is

$$\begin{aligned} r_l(s,a) &= Q_l^\pi(s,a) - \gamma\mathbf{E}_{s'}\langle Q_l^\pi(s',\cdot),\pi(s')\rangle \\ &= \langle w_l, \varphi_l^\pi(s,a) - \gamma\mathbf{E}_{s'}\langle\varphi_l^\pi(s',\cdot),\pi(s')\rangle\rangle, \end{aligned}$$

for $s'\sim\mathcal{P}_l(s,a)$ for some transition model $\mathcal{P}_l$. Note therefore that $r_l(s,a)$ is a Gaussian random variable, and by (I), $r_l(s,a) = \langle w_l,\psi(s,a)\rangle$ for $\psi$ independent of $\pi$ and satisfying

$$\psi(s,a) = \varphi_l^\pi(s,a) - \gamma\mathbf{E}_{s'}\langle\varphi_l^\pi(s',\cdot),\pi(s')\rangle \quad (\forall(s,a)\in\mathcal{S}\times\mathcal{A}).$$

Now consider $Q_l^\pi$. From the model definition,

$$Q_l^\pi(x) = \mathbf{E}\left[\sum_{h<H}\gamma^h r_l(X_h) \mid w_l\right] = \mathbf{E}\left[\sum_{h<H}\gamma^h\langle w_l,\psi(X_h)\rangle \mid w_l\right]$$

for $X_0 = x$ almost surely and $X_1,\dots,X_{h-1}\in\mathcal{S}\times\mathcal{A}$ random state-action pairs distributed according to the measure induced by $\mathcal{P}_l$ and $\pi$. Then, by linearity and recalling that $Q_l^\pi(x) = \langle w_l,\varphi_l^\pi(x)\rangle$, we have

$$Q_l^\pi(x) = \langle w_l, \mathbf{E}\sum_{h<H}\gamma^h\psi(X_h)\rangle = \langle w_l,\varphi_l^\pi(x)\rangle$$

for $w_l$ in the support of $\mathcal{N}(\mu_w^l,\Sigma_w^l)$. From (II), $\mathcal{N}(\mu_w^l,\Sigma_w^l)$ has full support on $\mathbb{R}^d$ for all $l\in[L]$. With that, the above weak equality implies the claim. □

Assumption (I) differentiates our model from GP-ESARSA, presented in section 7.3.1,

and will allow it to scale beyond tabular examples. It is also sensible: rewards are a property of the MDP, inferred from the history, whereas the policy is chosen by the agent, again based on the history. Hence, given the history, the two variables ought to be independent. (II) allows us to uniquely determine $\varphi_l^\pi$ for a given choice of $\psi$; this makes the exposition simpler, but is not otherwise necessary. It is also trivial to satisfy.

## 8.2 The Successor Uncertainties model and algorithms

We now use insight from theorem 51 to develop our Successor Uncertainties Q-function model, and develop two tabular randomised value function algorithms based on it, SU-MAX and SU-ESARSA. For this section, we assume that $\psi\colon \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}^d$ is a one-hot encoding of the state-action space, $\psi(s,a) = e_{(s,a)}$.

### 8.2.1 SU Q-function model

For each $l \in [L]$, the Successor Uncertainties model is given by the following two choices:

1. For a given policy $\pi$, we use embeddings $\varphi_l^\pi$ that satisfy

$$\varphi_l^\pi(s,a) = \begin{cases} \psi(s,a) & s \in \mathcal{S}_{H-1} \\ \psi(s,a) + \gamma \mathbf{E}_{s' \sim \bar{\mathcal{P}}_l(s,a)} \langle \pi(s'), \varphi_l^\pi(s', \cdot) \rangle & \text{otherwise,} \end{cases}$$

   for $\bar{\mathcal{P}}_l$ an empirical transition model.

2. We take the posterior for $w_l$ to be given by combining a prior $\mathcal{N}(0, \alpha I)$, $\alpha > 0$, with a likelihood induced by the model $R_\tau = \langle w, \psi(S_\tau, A_\tau) \rangle + \varepsilon_\tau$ with $\varepsilon_\tau \sim \mathcal{N}(0, \beta)$ for $\beta > 0$, independent for each $\tau < T$. That is, $P_w^l = \mathcal{N}(\mu_w, \Sigma_w)$ with

$$\mu_w = \Sigma_w \Psi R \quad \text{and} \quad \Sigma_w = \left(\alpha^{-1} I + \beta^{-1} \Psi \Psi^T\right)^{-1},$$

   where $\Psi = [\psi(S_\tau, A_\tau)\colon \tau < lH] \in \mathbb{R}^{d \times lH}$ and $R = [R_\tau\colon \tau < lH]^T \in \mathbb{R}^{lH}$.

The posterior over the Q-function is then $P_Q^l = \mathcal{N}(\mu_Q^l, \Sigma_Q^l)$ with

$$[\mu_Q^l]_{(s,a)} = \langle \mu_w, \varphi_l^\pi(s,a) \rangle \quad \text{and} \quad [\Sigma_Q^l]_{(s,a),(s',a')} = \langle \varphi_l^\pi(s,a) \varphi_l^\pi(s',a')^T, \Sigma_w \rangle_F.$$

Note that the Q-function covariance under the SU model is, in general, a dense matrix. Also, all quantities involved can be evaluated without instantiating any object of size

that scales with either $|\mathcal{S}|$ or $|\mathcal{A}|$, and so the model can scale beyond tabular problems.

Examining the expressions for $\mu_Q^l$ and $\Sigma_Q^l$ for a given state-action pair, the mean Q-value is high if the successor features thereafter align with the mean reward vector and the variance is low if the outer product of those successor features aligns closely with the outer products of previously observed features. This is intuitively sensible.

### 8.2.2 SU algorithms

We now present two randomised value function algorithms based on the SU model, SU-MAX, which models $Q_M^\star$, and SU-ESARSA, which models $Q_M^{\bar{\pi}_l}$ for a sequence of policies $\{\bar{\pi}_l\}$ induced by the interaction policy. We plot the performance of both methods on the Deep Sea benchmark in fig. 8.1. Both methods use $\alpha = 100$ and $\beta = 0.01$, the same parameter setting as were used in all previous experiments.

**SU-MAX** This is a classic RVF-type algorithm: at the start of episode $l \in [L]$, we sample $w_l \sim P_w^l$ and construct a reward function $r_l(s,a) = \langle w_l, \psi(s,a)\rangle$ for all $(s,a) \in \mathcal{S} \times \mathcal{A}$. We then solve for $Q_{M_l}^\star$ for the MDP $M_l$ given by $r_l$ and an empirical average transition model—for example, by using value iteration—and use a policy $\pi_l \in GQ_{M_l}^\star$.

The performance of SU-MAX, shown in fig. 8.1 (top), closely matches that of RLSVI. Indeed, we now show that $P_Q^l$ induced by SU-MAX is similar to that of RLSVI and use this to argue that SU-MAX has sublinear regret. Assume $w_0 \sim \mathcal{N}(m, \alpha I)$ for $m \in \mathbb{R}$ and $\alpha > 0$—that is, introduce a prior mean parameter $m$. Then for $Q_l \sim P_Q^l$ partitioned into restrictions $\{Q_l|_h\}$, where $Q_l|_h \colon \mathcal{S}_h \times \mathcal{A} \mapsto \mathbb{R}$ is the restriction of $Q_l$ to $\mathcal{S}_h \times \mathcal{A}$, and taking $Q_l|_H = 0$, the conditional distribution for $Q_l|_h$ given $Q_l|_{h+1}$ is $\mathcal{N}(\tilde{\mu}_h, \mathrm{diag}(\tilde{\nu}_h))$ where, writing $\chi_t(s,a) = \mathbb{1}\{S_t = s, A_t = a\}$,

$$\tilde{\mu}_h(s,a) = \tilde{\nu}_h(s,a)\Big(\alpha^{-1}m + \beta^{-1}\sum_{i<l}\chi_{iH+h}(s,a)R_{iH+h}\Big) + \frac{1}{n_{lH}(s,a)}\sum_{i<l}\chi_{iH+h}(s,a)\max_{b\in\mathcal{A}} Q|_{h+1}(S_{iH+h+1}, b)$$

and $\tilde{\nu}_h(s,a) = (\alpha^{-1} + \beta^{-1}n_{lH}(s,a))$. Denoting by $\mathcal{N}(\mu_h, \mathrm{diag}(\nu_h))$ the corresponding RLSVI conditional distributions (as given in eq. (6.2) on page 124), and setting $\alpha, \beta, m$

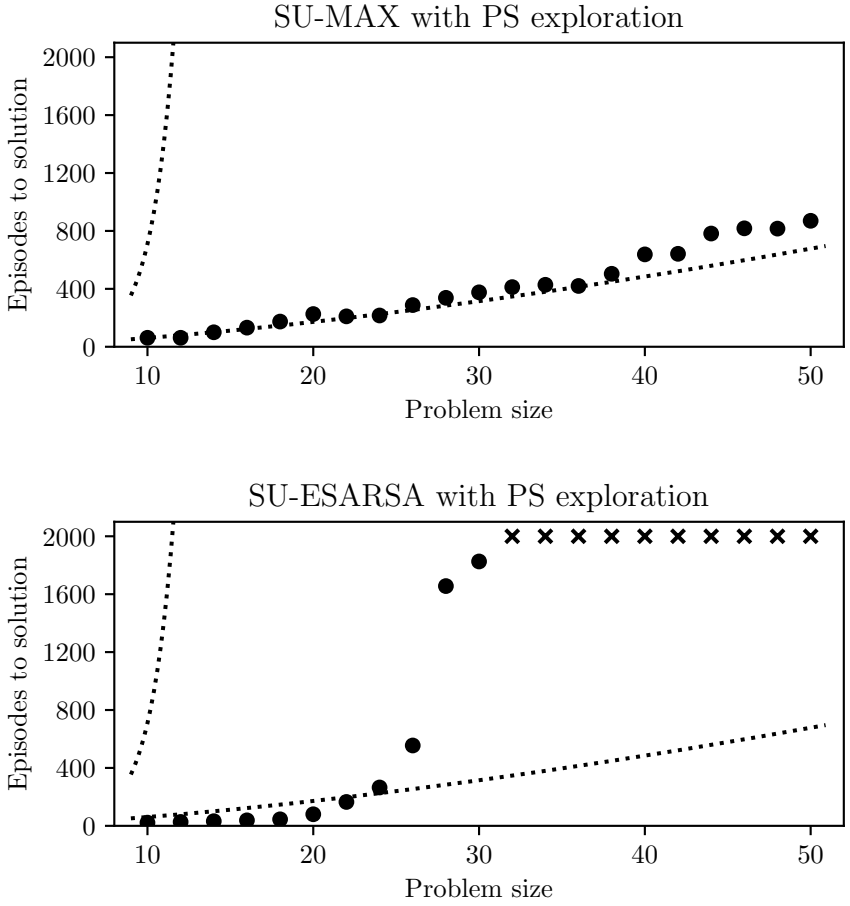



Figure 8.1: Deep Sea performance with a single seed for each method. Points denote number of episodes to solution for each problem size; crosses denote runs that failed to solve problem within 2000 episodes. Dotted lines depict analytic time to first reward for a uniform policy and an empirical fit to the performance of RLSVI respectively.

to the same values for both models, we have that

$$(\mu_h - \tilde{\mu}_h)(s,a) \leq \alpha H \nu_h(s,a) \quad \text{and} \quad \nu_h(s,a) = \tilde{\nu}_h(s,a) \quad (\forall (s,a) \in \mathcal{S} \times \mathcal{A}),$$

where we used $Q_l|_{h+1} \leq H$ for all $h \leq H$ to bound the difference of means. Therefore, since stochastic optimism is transitive and SU-MAX is stochastically optimistic for the RLSVI model, given appropriate $\alpha, \beta, m$ it is also stochastically optimistic for $Q^\star_M$. Noting also that the SU-MAX model concentrates at the same rate as RLSVI, we can infer that it has sublinear regret with a bound similar to that of RLSVI.

However, like RLSVI, SU-MAX requires the solving of a new planning task in each

episode and thus may be difficult to scale to the deep reinforcement learning setting. For this reason, we consider the following SU-ESARSA algorithm.

**SU-ESARSA** Here, we model the Q-function induced by the average of the posterior sampling exploration policies induced by our model $P_Q^l$. That is, we model the Q-function for the policy $\bar{\pi}_l = \mathbf{E}_{Q\sim P_Q^l} GQ$ or, more explicitly,[1]

$$\bar{\pi}_l(s)(a) = P_w^l(\langle w, \varphi_l^{\bar{\pi}_l}(s,a) - \varphi_l^{\bar{\pi}_l}(s,b)\rangle > 0 \text{ for all } b \neq a) \quad (\forall (s,a) \in \mathcal{S}\times\mathcal{A}).$$

We can solve for $\varphi_l^{\bar{\pi}_l}$ using value iteration. We then take $\pi_l \in GQ_l$ for $Q_l \sim P_Q^l$ as usual.

The performance of SU-ESARSA on the Deep Sea benchmark (fig. 8.1, bottom) is similar to that of GP-SARSA (fig. 7.3, top). Indeed, the two models themselves are similar. The Q-function covariance matrix for a given policy $\pi$ under SU-ESARSA is given by

$$\Sigma_Q = (I - \gamma\bar{P}^\pi)^{-1}\left[\alpha^{-1}I + \beta^{-1}\Psi\Psi^T\right]^{-1}(I-\gamma\bar{P}^\pi)^{-T}$$

Comparing this with the covariance matrix under GP-ESARSA, given in eq. (7.1) on page 139, the key difference is the location of the regularisation term $\alpha^{-1}I$. The position it takes within the SU-ESARSA covariance matrix allows us to use $\varphi_l^\pi = (I-\gamma\bar{P}^\pi)^{-1}$ in order to avoid having to explicitly construct the matrix $(I-\gamma\bar{P}^\pi)$; this leads to SU-ESARSA being easier to scale to the deep reinforcement learning setting.

When compared with SU-MAX, SU-ESARSA combines more naturally with GPI-based deep reinforcement learning. While SU-ESARSA still requires solving policy evaluation for each episode $l \in [L]$, since the difference between $P_w^l$ and $P_w^{l+1}$ is (in some sense) small, $\bar{\pi}_l$ and $\bar{\pi}_{l+1}$ are close. We can therefore warm-start the policy evaluation at episode $l+1$ using the solution from episode $l$, allowing for efficient (in practice) generalised policy iteration. In contrast, the dependence within SU-MAX on $P_w^l$ and $P_w^{l+1}$ is through the samples $w_l$ and $w_{l+1}$, which may be far apart.

## 8.3 Deep Successor Uncertainties

We now combine our SU-ESARSA model with neural network function approximation and benchmark our implementation on the Atari Learning Environment. The key change from the tabular setting is that we no longer assume $\psi$ to be given and instead learn a

[1]This can be estimated using simple Monte Carlo, see eq. (8.2) on page 158.

suitable embedding online. We now look at how to do so, and how to adapt the Deep Q-Networks network architecture and loss to our needs. The full algorithm is listed in fig. 8.2 on page 157.

### 8.3.1 Learning reward-predictive embeddings

The key relationship satisfied by the embedding $\psi$ is that

$$r(s,a) = \langle \mu_w, \psi(s,a) \rangle \quad (\forall (s,a) \in \mathcal{S} \times \mathcal{A})$$

for some $\mu_w \in \mathbb{R}^d$. This motivates learning a network $\hat{\psi}$ to use as the reward-predictive embedding by minimising a loss of the form

$$\mathcal{L}_r(s,a,r) = \Big(r - \langle \hat{\mu}_w, \hat{\psi}(s,a) \rangle\Big)^2$$

with respect to both $\hat{\mu}_w$ and the parameters of $\hat{\psi}$ in expectation over observations $(s,a,r)$.

Additionally, we suggest imposing the following two constraints on $\hat{\psi}$:

1. $\hat{\psi}(s,a) \geq 0$ for all $(s,a) \in \mathcal{S} \times \mathcal{A}$. This ensures that

$$\mathbf{E}\left[\sum_{\tau=0}^{\infty} \gamma^\tau \hat{\psi}(S_\tau, A_\tau) \mid S_0 = s, A_0 = a\right] > 0 \quad (\forall (s,a) \in \mathcal{S} \times \mathcal{A}),$$

   for any policy and transition model and so stops cancellation of embeddings in the sum, which could otherwise lead to zero posterior variance for the Q-function—a pathological situation.

2. $\|\hat{\psi}(s,a)\|_2 = 1$ for all $(s,a) \in \mathcal{S} \times \mathcal{A}$. This leads to all state-action pair having the same a priori mean reward variance and makes choosing $\alpha$ and $\beta$ easier.

With this in mind, we learn $\hat{\psi}$ as the output of a ReLU network, followed by explicit normalisation. The network takes as input $s \in \mathcal{S}$ and outputs $f(s) \in \mathbb{R}^{d'}$ at the penultimate layer. We then take

$$\hat{\psi}'(s,a) = \textsc{ReLU}(U_a^T f(s) + b_a) \quad \text{and} \quad \hat{\psi}(s,a) = \frac{\hat{\psi}'(s,a)}{\|\hat{\psi}'(s,a)\|_2 + \varepsilon}, \tag{8.1}$$

where $U_a \in \mathbb{R}^{d' \times d}$ is a weight matrix, $b_1, \ldots, b_{|A|} \in \mathbb{R}^d$ are bias terms for each action and we take $\varepsilon$ to be a small positive constant to avoid division by zero.

Learning the embedding function $\hat{\psi}$ online causes difficulty for the computation of $\Sigma_w^l$, which depends on $\hat{\psi}$ directly, and $\mu_w^l$, which in turn depends on $\Sigma_w^l$. Specifically, each time the function $\hat{\psi}$ changes, we need to recompute $\Sigma_w^l$ and $\mu_w^l$. This requires a pass through all previous observations, and thus also storing these. Instead:

- We use $\hat{\mu}_w$, the final weights of the neural network predicting the mean reward function, in place of the analytic mean $\mu_w^l$.
- At each step $t$, we compute $\hat{\psi}_t = \hat{\psi}(S_t, A_t)$, and use the approximate covariance

$$\hat{\Sigma}_w = \left[\alpha^{-1}\zeta^t I + \beta^{-1}\sum_{\tau<t}\zeta^{t-\tau}\hat{\psi}_\tau\hat{\psi}_\tau^T\right]^{-1}$$

  where $\zeta \in [0, 1)$ is a forgetting parameter that reduces the influence of outdated embeddings.

The mean approximation was previously used in Levine et al. (2017) and windowing schemes for covariance functions date back to at least Dearden et al. (1998).[2]

### 8.3.2 Neural network approximation of successor features

We use a neural network $\hat{\varphi}$ to estimate the successor features $\varphi_l$ for the ESARSA policy. From our constraints on $\hat{\psi}$, it is clear that $\hat{\varphi}$ needs to be non-negative. Again, we take this to be the output of a ReLU network with some penultimate layer $f(s) \in \mathbb{R}^{d'}$, $d' \in \mathbb{N}$, and

$$\hat{\varphi}(s, a) = \textsc{ReLU}((V_a + \bar{V})f(s) + c_a + \bar{c}),$$

where $V_1, \ldots, V_{|\mathcal{A}|}, \bar{V} \in \mathbb{R}^{d\times d'}$ are weight matrices and $c_1, \ldots, c_{|\mathcal{A}|}, \bar{c} \in \mathbb{R}^d$ offsets. We include $\bar{V}$ and $\bar{c}$ to act in the manner of duelling networks (Wang et al., 2016), potentially making learning easier where multiple actions lead to similar state-action distributions.

Next, we need a loss function for the network $\hat{\varphi}$. In line with the DQN work, we suggest a standard squared Bellman error loss applied elementwise on the entries of $\hat{\varphi}$. That is,

[2]A promising alternative to our heuristic approach would be to explicitly correct for the error introduced by the use of outdated embeddings in computing the covariance $\Sigma_w$. Consider an embedding function $\psi$ which induces a model $\mathcal{N}(\mu_w, \Sigma_w)$ and a modified function $\varphi'$ with model $\mathcal{N}(\mu'_w, \Sigma'_w)$. Assuming the reward is linearly realisable in both embeddings, then the likelihood of observing the rewards is the same under both models. This likelihood-matching condition yields a semidefinite program that computes $\Sigma'_w$ from $\Sigma_w$. The program can be solved iteratively using standard proximal point methods. This likelihood matching method has been investigated in the bandit setting by Zahavy et al. (2019) and Nabati et al. (2021).

**Algorithm:** Deep SU-ESARSA with posterior sampling
**Parameters:** prior variance $\alpha > 0$; likelihood variance $\beta > 0$; covariance decay factor $\zeta \in [0, 1)$; discount factor $\gamma \in [0, 1)$.
**Initialisation:** $\Lambda \leftarrow \alpha^{-1} I$, $\hat{\Sigma}_w \leftarrow \Lambda^{-1}$

For each episode:

1. Sample a weight $w \sim \mathcal{N}(\hat{\mu}_w, \hat{\Sigma}_w)$.
2. Obtain initial state $S$.
3. For each step in episode:
   (a) Take action $A \in \arg\max_{b \in \mathcal{A}} \langle \hat{\varphi}(s, b), w \rangle$.
   (b) Observe reward $R$ and next state $S'$, and $D \in \{0, 1\}$ indicating end of episode. Add $(S, A, R, S', D)$ to the replay buffer.
   (c) Sample batch of observations $B$ uniformly from the replay buffer and a weight $u \sim \mathcal{N}(\hat{\mu}_w, \hat{\Sigma}_w)$. For each $(s, a, r, s', d) \in B$ compute next action $a' \in \arg\max_{b \in \mathcal{A}} \langle \varphi(s, b), u \rangle$ and targets

$$Y_Q = r + \gamma(1 - d)\langle \hat{\mu}_w, \hat{\varphi}(s', a') \rangle \quad \text{and} \quad Y_\varphi = \hat{\psi}(s, a) + \gamma(1 - d)\hat{\varphi}(s', a'),$$

   and then the corresponding losses

$$\mathcal{L}_Q = (\langle \hat{\mu}_w, \hat{\varphi}(s, a) \rangle - Y_Q)^2, \quad \mathcal{L}_\varphi = (\hat{\varphi}(s, a) - Y_\varphi)^2,$$

   and the reward loss

$$\mathcal{L}_r = (\langle \hat{\mu}_w, \hat{\psi}(s, a) \rangle - r)^2.$$

   Take gradient step minimising $\mathcal{L}_Q + \mathcal{L}_\varphi + \mathcal{L}_r$ with respect to $\hat{\psi}$, $\hat{\varphi}$ and $\hat{\mu}_w$.
   (d) Update the precision matrix

$$\Lambda \leftarrow \zeta \Lambda + \beta^{-1} \hat{\psi}(S, A) \hat{\psi}(S, A)^T.$$

4. Update $\hat{\Sigma}_w \leftarrow \Lambda^{-1}$.

Figure 8.2: Deep SU model with approximate ESARSA updates using a single sample Monte Carlo estimate for the target policy and with exploration via posterior sampling.

for a tuple $(s, a, r, s')$ sampled from the replay buffer and a policy $\pi$, we compute targets

$$Y_\varphi(s,a) = \hat{\psi}(s,a) + \gamma\langle \pi(s'), \hat{\varphi}(s', \cdot)\rangle,$$

and use a loss of the form $\mathcal{L}_\varphi = \sum_{i=1}^d [\hat{\varphi}(s,a) - Y_\varphi(s,a)]_i^2$. We take the policy $\pi$ to be given by a Monte Carlo estimate of that used in SU-ESARSA,

$$\pi(s)(a) = \frac{1}{k}\sum_{m=1}^{k} \mathbb{1}\{\langle w^m, \hat{\varphi}(s,a) - \hat{\varphi}(s,b)\rangle > 0 \text{ for all } b \neq a\} \quad (\forall (s,a) \in \mathcal{S}\times\mathcal{A}) \quad (8.2)$$

for $k \in \mathbb{N}$ a hyperparameter and $w^1, \ldots, w^k$ independent samples from $\mathcal{N}(\hat{\mu}_w, \hat{\Sigma}_w)$.

In practice, we found that in order to obtain strong performance on even very simple problems, we need to include an explicit mean Q-function loss in addition to the successor feature and reward losses. This takes the form

$$\mathcal{L}_Q(s,a,r) = \left(\langle \hat{\mu}_w, \hat{\varphi}(s,a)\rangle - Y_Q(s,a,r)\right)^2 \quad \text{with} \quad Y_Q(s,a,r) = r + \gamma\langle \hat{\mu}_w, \hat{\varphi}(s', a')\rangle.$$

We take gradients with respect to both $\hat{\varphi}$ and the weights $\hat{\mu}_w$. It is perhaps unsurprising that for a DQN-based method to work well, it needs to include a DQN-like loss.

### 8.3.3 Results on the Atari Learning Environment

We benchmarked the deep implementation of the SU algorithm, presented in fig. 8.2, on the standard set of 49 games from the Arcade Learning Environment. We use 200M training frames under the 'no-ops start 30 minute emulator time' test protocol described in Hessel et al. (2018) and averaged the results over three seeds. We use a standard network architecture, as in Mnih et al. (2015) and Van Hasselt et al. (2016), with the head replaced by ReLU networks predicting the successor features and reward-predictive embeddings. See appendix 8.A for more in-depth implementation details.[3]

Deep SU obtains a median human normalised score of 2.09. As shown in table 8.1, this is a significant improvement over competing methods. We do not compare against the results in Azizzadenesheli et al. (2018); there the authors only report scores for a small subset of the games and use a non-standard testing procedure. Osband, Aslanides et al. (2018), where Bootstrap+Prior is introduces, does not report Atari results; we thus compare with results for the original Bootstrapped DQN method instead (scores reported

[3]Code for the Atari experiments: djanz.org/successor_uncertainties/atari_code

Table 8.1: Human normalised Atari scores. Superhuman performance denotes the percentage of games on which human performance is surpassed (as in Mnih et al., 2015).

| Algorithm | Human normalised score percentiles 25% | 50% | 75% | Superhuman performance |
|---|---|---|---|---|
| Successor Uncertainties | **1.06** | **2.09** | **5.95** | **77.55%** |
| Bootstrapped DQN | 0.76 | 1.60 | 5.16 | 67.35% |
| UBE | 0.38 | 1.07 | 4.14 | 51.02% |
| DQN + $\varepsilon$-greedy | 0.50 | 1.00 | 3.41 | 48.98% |

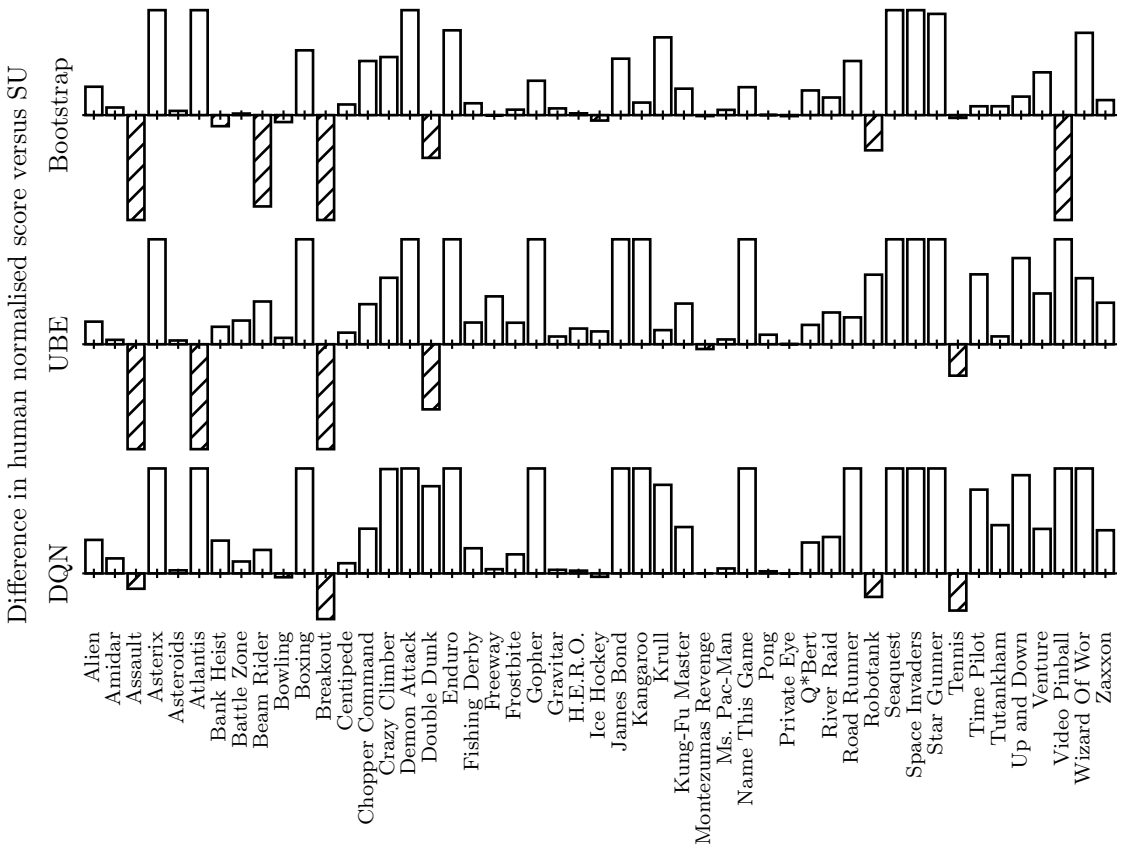



Figure 8.3: Difference in the human normalised score between SU and each of Bootstrapped DQN, UBE and DQN for each ALE game, clipped to the range $[-2.5, 2.5]$. Positive values indicate SU outperformed the baseline algorithm. SU outperforms the baselines on 36/49, 43/49 and 42/49 of the games respectively. Hatching is used to highlight negative values.

Table 8.2: Raw ALE scores for SU alongside DQN, UBE and Bootstrapped DQN. Baseline scores taken from Hessel et al. (2018).

| Game | DQN | UBE | Bootstrapped DQN | SU |
|---|---|---|---|---|
| Alien* | 1,620.0 | 3,345.3 | 2,436.6 | **6,924.4** |
| Amidar* | 978.0 | 1,400.1 | 1,272.5 | **1,574.4** |
| Assault | 4,280.4 | **11,521.5** | 8,047.1 | 3,813.8 |
| Asterix | 4,359.0 | 7,038.5 | 19,713.2 | **42,762.2** |
| Asteroids | 1,364.5 | 1,159.4 | 1,032.0 | **2,270.4** |
| Atlantis | 279,987.0 | **4,648,770.8** | 994,500.0 | 2,026,261.1 |
| Bank Heist* | 455.0 | 718.0 | **1,208.0** | 1,017.4 |
| Battle Zone | 29,900.0 | 19,948.9 | 38,666.7 | **39,944.4** |
| Beam Rider | 8,627.5 | 6,142.4 | **23,429.8** | 11,652.3 |
| Bowling | 50.4 | 18.3 | **60.2** | 38.3 |
| Boxing | 88.0 | 34.2 | 93.2 | **99.7** |
| Breakout | 385.5 | 617.3 | **855.0** | 352.7 |
| Centipede | 4,657.7 | 4,324.1 | 4,553.5 | **7,049.3** |
| Chopper Command | 6,126.0 | 7,130.8 | 4,100.0 | **15,787.8** |
| Crazy Climber | 110,763.0 | 132,997.5 | 137,925.9 | **171,991.1** |
| Demon Attack | 12,149.4 | 25,021.1 | 82,610.0 | **183,243.2** |
| Double Dunk | -6.6 | **4.7** | 3.0 | -0.2 |
| Enduro | 729.0 | 30.8 | 1,591.0 | **2,216.3** |
| Fishing Derby | -4.9 | 3.1 | 26.0 | **53.3** |
| Freeway** | 30.8 | 0.0 | **33.9** | 33.8 |
| Frostbite* | 797.4 | 546.0 | 2,181.4 | **2,733.3** |
| Gopher | 8,777.4 | 13,808.0 | 17,438.4 | **19,126.2** |
| Gravitar** | 473.0 | 224.5 | 286.1 | **684.4** |
| H.E.R.O.* | 20,437.8 | 12,808.8 | 21,021.3 | **22,050.8** |
| Ice Hockey | -1.9 | -6.6 | **-1.3** | -2.9 |
| James Bond | 768.5 | 778.4 | 1,663.5 | **2,171.1** |
| Kangaroo | 7,259.0 | 6,101.2 | 14,862.5 | **15,751.1** |
| Krull | 8,422.3 | 9,835.9 | 8,627.9 | **10,103.9** |
| Kung-Fu Master | 26,059.0 | 29,097.1 | 36,733.3 | **50,878.9** |
| Montezumas Revenge** | 0.0 | **499.1** | 100.0 | 0.0 |
| Ms. Pac-Man* | 3,085.6 | 3,141.3 | 2,983.3 | **4,894.8** |
| Name This Game | 8,207.8 | 4,604.4 | 11,501.1 | **12,686.7** |
| Pong | 19.5 | 14.2 | 20.9 | **21.0** |
| Private Eye** | 146.7 | -281.1 | **1,812.5** | 133.3 |
| Q*Bert* | 13,117.3 | 16,772.5 | 15,092.7 | **22,895.8** |
| River Raid | 7,377.6 | 8,732.3 | 12,845.0 | **17,940.6** |
| Road Runner | 39,544.0 | 56,581.1 | 51,500.0 | **61,594.4** |
| Robotank | 63.9 | 42.4 | **66.6** | 58.5 |
| Seaquest | 5,860.6 | 1,880.6 | 9,083.1 | **68,739.9** |
| Space Invaders | 1,692.3 | 2,032.4 | 2,893.0 | **13,754.3** |
| Star Gunner | 54,282.0 | 44,458.6 | 55,725.0 | **78,837.8** |
| Tennis | **12.2** | 10.2 | 0.0 | -1.0 |
| Time Pilot | 4,870.0 | 5,650.6 | 9,079.4 | **9,574.4** |
| Tutankham | 68.1 | 218.6 | 214.8 | **247.7** |
| Up and Down | 9,989.9 | 12,445.9 | 26,231.0 | **29,993.4** |
| Venture** | 163.0 | -14.7 | 212.5 | **1,422.2** |
| Video Pinball | 196,760.4 | 51,178.2 | **811,610.0** | 515,601.9 |
| Wizard Of Wor* | 2,704.0 | 8,425.5 | 6,804.7 | **15,023.3** |
| Zaxxon* | 5,363.0 | 5,717.9 | 11,491.7 | **14,757.8** |

in Osband, Blundell et al., 2016). We display the relative performance of SU against that of Bootstrapped DQN, UBE and DQN visually in fig. 8.3. SU outperforms these on 36/49, 43/49 and 42/49 of the games respectively. We report the raw scores attained by Deep SU on individual games in table 8.2. We highlight 'hard exploration games with dense rewards' by * and 'hard exploration games with sparse rewards' by ** (as classified by Ostrovski et al., 2017). SU outperforms DQN, UBE and Bootstrapped DQN on 8/9 of the former and 2/5 of the latter.

## 8.4 Discussion

We are now at the end of part II. Our main contribution here has been in the identification of a successor feature structure necessary for scalable feature-linear Gaussian Q-function models. Based on this, we developed two tabular algorithms: SU-MAX, which matches the performance of RLSVI on the Deep Sea MDPs, and which we showed is stochastically optimistic for the RLSVI Q-function estimator; and SU-ESARSA, which matches the performance of GP-SARSA but is better suited for problems with a large state-action space cardinality. We extended SU-ESARSA to the deep reinforcement learning setting and showed that it attains strong scores in the Atari Learning Environment. We now turn to discussing the shortcomings of our work and potential directions for future work.

From a theoretical perspective, our finite MDP setting is very limited. An immediate extension of our work would be to integrate it with the linear MDP assumption of Jin et al. (2020). The methodology therein, based on the OFUL concentration inequality, might allow for a regret bound for SU-MAX with linear function approximation. Thereafter, the natural research direction would be in looking at regret under neural network function approximation. Dong et al. (2021) have some interesting results for this non-linear setting, but these suggest that optimism (and, equivalently, posterior sampling) may be insufficient.

From a practical perspective, our results across the Atari Learning Environment are strong, particularly so on 'hard exploration with dense rewards' games. However, SU scored poorly on the hardest of the exploration challenges: it scored zero on Montezuma's Revenge on all three seeds. We have two hypotheses for why:

1. We extended the weaker SU-ESARSA model due to its better integration with generalised policy iteration. It is possible that a suitably extended SU-MAX algorithm would attain stronger scores on hard exploration tasks.

2. The problem might not be with exploration but with learning. Montezuma's revenge and other sparse reward tasks require learning and credit assignment over long sequences of time-steps. As shown in O'Donoghue et al., 2018, multi-step temporal difference learning methods (Sutton, 1988) can significantly improve performance on such sparse tasks. Combining Deep SU-ESARSA with multi-step learning and other tricks for dealing with long horizons (as in, for example, Pohlen et al., 2018) might improve scores on sparse reward tasks.

Both of these hypothesis provide directions for future research.

Testing hypothesis 1 requires some thought. We suspect a combination of SU-MAX with multi-task learning methods may yield a strong extension deep reinforcement learning algorithm. For example:

- Barreto et al. (2017) consider a multi-task reinforcement learning setting, tackling a set of MDPs with shared transition dynamics but different reward functions, and use successor features and an interesting approximation result to quickly estimate near-optimal value functions for new MDPs, given those previously solved. They apply their methodology to the deep reinforcement learning setting. Interpreting SU-MAX as solving such a sequence of tasks, we may be able to use their method to reduce the computational complexity of SU-MAX and thus combine it with deep reinforcement learning. Their approximation result might even yield a regret bound (under some suitable regression oracle assumptions).
- That same multi-task reinforcement learning problem could be tackled with more general meta-learning methods, for example those of Finn et al. (2017). This approach may give an alternative extension of SU-MAX to deep reinforcement learning, but is unlikely to provide for new theoretical insights.

Implementing either approach, we would be very interested in combining it with a more modern framework for deep reinforcement learning, such as that given by the AlphaStar/AlphaZero/MuZero line of work (Silver, Huang et al., 2016; Team, 2019; Schrittwieser et al., 2020).

Examining hypothesis 2 is conceptually easier, but requires a great deal of engineering effort and, importantly, computational budget. It may be made easier by implementing SU within the framework of parallel data collection using multiple agents (Nair et al., 2015) and using seed sampling to coordinate their exploration (Dimakopoulou et al., 2018). Such approaches can reduce the time required to run the Atari Learning Environment

benchmark by an order of magnitude or more. Also, since our focus is on exploration, and we could restrict our attention to the subset of games classified as posing a hard exploration challenge with sparse rewards. Note that hypothesis 2 addresses not just our algorithm, but exploration more broadly.

Looking beyond our contributions, deep reinforcement learning literature is still missing exploration methods that robustly tackle hard exploration problems. While heuristic methods have performed well on particular environments, as in the case of the Go-Explore method on Montezuma's Revenge (Ecoffet et al., 2019), it is unclear whether these are broadly applicable. But perhaps the bandit-based view taken in this thesis is too rigid, and the future of practical reinforcement is in a development of effective heuristics and the combination of these with domain specific knowledge to solve new problems.

# Appendix 8.A ALE experimental details

**Training procedure** We train for 200M frames (50M action selections with each action repeated for 4 frames), using the ADAM optimiser (Kingma et al., 2014) with a learning rate of $5 \times 10^{-5}$ and a batch size of 32. A target network is utilised, as in Mnih et al. (2015), and is updated every 10,000 steps, as in Van Hasselt et al. (2016).

**Network architecture** We use a single neural network for both $\hat{\psi}$ and $\hat{\varphi}$.

1. Features: the neural network converts $4 \times 84 \times 84$ pixel states (obtained through standard frame max-pooling and stacking) into a 3136-dimensional feature vector, using a convolution network with the same architecture as in Mnih et al., 2015.
2. Hidden layer: the feature vector is then mapped to a hidden representation of size 1024 by a fully connected layer followed by a ReLU activation.
3. $\psi$ estimation: the hidden representation is mapped by a fully connected layer with ReLU activation to a 64 vector for each action $a$ in $\mathcal{A}$, which is then normalised as in eq. (8.1) on page 155 with $\varepsilon = 10^{-6}$. This gives $\hat{\psi}(s, a)$.
4. $\varphi$ estimation: the hidden representation is mapped to $1 + |\mathcal{A}|$ vectors of size 64. The first vector gives the average successor features for that state, denoted $\bar{\varphi}(s)$, whilst each of the $|\mathcal{A}|$ vectors predicts an advantage $\hat{\varphi}'(s, a)$. The overall successor feature prediction is given by $\hat{\varphi}(s, a) = \bar{\varphi}(s) + \hat{\varphi}'(s, a)$.
5. Q-function and reward prediction: a final linear layer with weights $\hat{\mu}_w$ is shared for both $\hat{\varphi}$ and $\hat{\psi}$, mapping $\hat{\varphi}$ to Q value predictions and $\hat{\psi}$ to reward predictions.

**Hyperparameter selection** We used six games for hyperparameter selection: ASTERIX, ENDURO, FREEWAY, HERO, QBERT, SEAQUEST, a subset of the games commonly used for this purpose (Munos et al., 2016). 12 combinations of parameters in the 'search set' column of table 8.3 were tested (that is, not an exhaustive gridsearch), for a total of $12 \times 6 = 72$ full game runs, or approximately a third of the entire computational cost.

Table 8.3: Hyperparameters used for Deep SU in ALE experiments.

| Hyperparameter | Search set | Value used |
|---|---|---|
| Action repeat | — | 4 |
| Train interval | — | 4 |
| Learning rate | $\{2.5 \times 10^{-4}, 5 \times 10^{-5}\}$ | $5 \times 10^{-5}$ |
| Batch size | — | 32 |
| Gradient clip norm cut-off | — | 10 |
| Target update interval | $\{10^3, 10^4\}$ | $10^4$ |
| Successor feature size | $\{32, 64\}$ | 64 |
| Hidden layer size | — | 1024 |
| Prior variance $\alpha$ | — | 1 |
| Likelihood variance $\beta$ | $\{10^{-3}, 10^{-2}\}$ | $10^{-3}$ |
| $\hat{\Sigma}_w$ decay factor $\zeta$ | $\{1 - 10^{-5}, 1 - 10^{-4}\}$ | $1 - 10^{-5}$ |

# Conclusion and future work

In this thesis, we considered the problem of sequential decision making within the bandit and the reinforcement learning settings, with a view of combining theory with practical insights to produce effective, fast and robust algorithms. Our main contributions in the area of bandit/global optimisation and kernel-based algorithms were:

- *Results on Matérn kernel spectrum.* We developed a novel bound that relates the information gain associated with a Matérn kernel regression problem as a function of the volume of the domain and showed an immediate application of this bound to the GP-UCB algorithm. These results may find wider application.
- *Partitioned GP-UCB algorithm.* We built on the information gain bound to design a hierarchical Gaussian process optimisation algorithm for bandit and global optimisation with strong guarantees on performance and computational complexity for a large class of problems. We tested the algorithm on a set of synthetic benchmarks.

And in the area of applied reinforcement learning:

- *Flaws in uncertainty modelling.* We identified a group of related flaws in the way some common probabilistic model-free reinforcement learning methods model uncertainty about the state-action value function, particularly in the context of neural function approximation. We provided some basic tools for the analysis of these methods and showed how careful empirical methodology can be used to identify such problems.
- *Successor Uncertainties.* We introduced the Successor Uncertainties Q-function model and demonstrated how combined with posterior sampling it yields a reinforcement learning algorithm capable of solving hard tabular problems as well as tackling the Atari Learning Environment under neural network function approximation.

In both areas, our contributions significantly advance understanding and methodology.

Each of the two parts of this thesis ends with an outline of short-to-medium-term future work directions. In the longer term, we see the following as the most important research directions related to the topic of this thesis:

- *Theory for neural network function approximation.* Much of the recent impetus in the research and application of machine learning, including reinforcement learning, has come from the advent of deep learning. However, deep learning is largely an empirical field; our understanding of the sample complexity and generalisation properties of neural networks is poor, especially so as it pertains to the very complex neural network models often used in practice. A better theoretical understanding of deep learning would likely advance the entirely of the machine learning field. In the context of this thesis, such development would aid in the design of mechanisms for provably effective sequential decision making in domains where neural network function approximation is virtually necessary: computer games, biological problems, autonomous driving, robotics and others.
- *Fast practical sequential optimisation with guarantees.* The area of bandit algorithms and the theory of sequential optimisation has gathered huge interest in the recent years. Applied global optimisation/Bayesian optimisation has been a very popular research topic since at least Osborne et al. (2009). What is missing is work at the intersection. While we attempt to address this area in part I of this thesis, our work is superficial. For example, we fail to address the important topic of hyperparameter optimisation within our method, either in theory or practice.
- *Robust exploration within deep reinforcement learning systems.* Despite the advances in deep reinforcement learning, and indeed notwithstanding the contributions of this thesis, exploration within deep reinforcement learning remains an unsolved problem. While development of theory for neural network function approximation and its combination with sequential optimisation may provide for many developments in this area, there is also a large scope for empirical work: for example, a core question in our opinion is in identifying the difficulty posed by benchmarks like Montezuma's Revenge and Pitfall. Is the problem really in exploration, or is it more related to the long-term credit allocation required therein? The answer to this determines the important directions within this subfield.

# Bibliography


Abbasi-Yadkori, Yasin (2009). 'Forced-Exploration Based Algorithms for Playing in Bandits with Large Action Sets'. PhD thesis. University of Alberta.

Abbasi-Yadkori, Yasin, Dávid Pál and Csaba Szepesvári (2011). 'Improved algorithms for linear stochastic bandits'. In: *Advances in Neural Information Processing Systems*.

Adams, Robert A and John JF Fournier (2003). *Sobolev spaces*. Elsevier.

Alaoui, Ahmed and Michael W Mahoney (2015). 'Fast Randomized Kernel Ridge Regression with Statistical Guarantees'. In: *Advances in Neural Information Processing Systems*.

Aronszajn, Nachman (1950). 'Theory of reproducing kernels'. In: *Transactions of the American mathematical society* 68.3, pp. 337–404.

Auer, Peter (2002). 'Using confidence bounds for exploitation-exploration trade-offs'. In: *Journal of Machine Learning Research* 3.Nov, pp. 397–422.

Auer, Peter, Nicolo Cesa-Bianchi and Paul Fischer (2002). 'Finite-time analysis of the multiarmed bandit problem'. In: *Machine learning* 47.2-3, pp. 235–256.

Auer, Peter, Ronald Ortner and Csaba Szepesvári (2007). 'Improved rates for the stochastic continuum-armed bandit problem'. In: *International Conference on Computational Learning Theory*.

Azar, Mohammad Gheshlaghi, Ian Osband and Rémi Munos (2017). 'Minimax regret bounds for reinforcement learning'. In: *International Conference on Machine Learning*.

Azizzadenesheli, Kamyar, Emma Brunskill and Animashree Anandkumar (2018). 'Efficient Exploration through Bayesian Deep Q-Networks'. In: *arXiv preprint arXiv:1802.04412*.

Barreto, André, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt and David Silver (2017). 'Successor features for transfer in reinforcement learning'. In: *Advances in Neural Information Processing Systems*.

Bastani, Hamsa and Mohsen Bayati (2020). 'Online decision making with high-dimensional covariates'. In: *Operations Research* 68.1, pp. 276–294.

Bauer, Matthias, Mark van der Wilk and Carl Edward Rasmussen (2016). 'Understanding probabilistic sparse Gaussian process approximations'. In: *Advances in Neural Information Processing Systems*.

Bellemare, M. G., Y. Naddaf, J. Veness and M. Bowling (June 2013). 'The Arcade Learning Environment: An Evaluation Platform for General Agents'. In: *Journal of Artificial Intelligence Research* 47, pp. 253–279.

Bellemare, Marc, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton and Remi Munos (2016). 'Unifying count-based exploration and intrinsic motivation'. In: *Advances in Neural Information Processing Systems*.

Blundell, Charles, Julien Cornebise, Koray Kavukcuoglu and Daan Wierstra (2015). 'Weight uncertainty in neural network'. In: *International Conference on Machine Learning*.

Boucheron, Stéphane, Gábor Lugosi and Pascal Massart (2013). *Concentration inequalities: A nonasymptotic theory of independence*. Oxford university press.

Bouneffouf, Djallel (2014). 'Freshness-aware thompson sampling'. In: *International Conference on Neural Information Processing*.

Bouneffouf, Djallel, Amel Bouzeghoub and Alda Lopes Gançarski (2013). 'Contextual bandits for context-based information retrieval'. In: *International Conference on Neural Information Processing*.

Brafman, Ronen I and Moshe Tennenholtz (2002). 'R-Max — A general polynomial time algorithm for near-optimal reinforcement learning'. In: *Journal of Machine Learning Research* 3.Oct, pp. 213–231.

Bubeck, Sébastien, Rémi Munos, Gilles Stoltz and Csaba Szepesvári (2011). 'X-armed bandits'. In: *Journal of Machine Learning Research* 12.May, pp. 1655–1695.

Burda, Yuri, Harrison Edwards, Amos Storkey and Oleg Klimov (2018). 'Exploration by random network distillation'. In: *International Conference on Learning Representations*.

Calandriello, Daniele, Luigi Carratino, Alessandro Lazaric, Michal Valko and Lorenzo Rosasco (2019). 'Gaussian Process Optimization with Adaptive Sketching: Scalable and No Regret'. In: *Conference on Learning Theory*.

Calandriello, Daniele, Luigi Carratino, Alessandro Lazaric, Michal Valko and Lorenzo Rosasco (2020). 'Near-linear time Gaussian process optimization with adaptive batching and resparsification'. In: *International Conference on Machine Learning*.

Calandriello, Daniele, Alessandro Lazaric and Michal Valko (2017a). 'Distributed adaptive sampling for kernel matrix approximation'. In: *Artificial Intelligence and Statistics*.

Calandriello, Daniele, Alessandro Lazaric and Michal Valko (2017b). 'Second-order kernel online convex optimization with adaptive sketching'. In: *International Conference on Machine Learning*.

Chowdhury, Sayak Ray and Aditya Gopalan (2017). 'On kernelized multi-armed bandits'. In: *International Conference on Machine Learning*.

Christmann, Andreas and Ingo Steinwart (2008). *Support vector machines*. Springer.

Cover, Thomas M (1999). *Elements of information theory*. John Wiley & Sons.

Dani, Varsha, Thomas P Hayes and Sham M Kakade (2008). 'Stochastic linear optimization under bandit feedback'. In: *Conference on Learning Theory*.

Dann, Christoph, Tor Lattimore and Emma Brunskill (2017). 'Unifying PAC and regret: uniform PAC bounds for episodic reinforcement learning'. In: *International Conference on Neural Information Processing Systems*.

Dayan, Peter (1993). 'Improving generalization for temporal difference learning: the successor representation'. In: *Neural Computation* 5.4, pp. 613–624.

Dearden, Richard, Nir Friedman and Stuart J. Russell (1998). 'Bayesian Q-Learning'. In: *AAAI/IAAI*. AAAI Press / The MIT Press, pp. 761–768.

Desautels, Thomas, Andreas Krause and Joel W Burdick (2014). 'Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization'. In: *Journal of Machine Learning Research* 15, pp. 3873–3923.

Dimakopoulou, Maria and Benjamin Van Roy (2018). 'Coordinated exploration in concurrent reinforcement learning'. In: *International Conference on Machine Learning*.

Dong, Kefan, Jiaqi Yang and Tengyu Ma (2021). 'Provable model-based nonlinear bandit and reinforcement learning: Shelve optimism, embrace virtual curvature'. In: *Advances in Neural Information Processing Systems*.

Drineas, Petros, Michael W Mahoney and Nello Cristianini (2005). 'On the Nyström Method for Approximating a Gram Matrix for Improved Kernel-Based Learning.' In: *Journal of Machine Learning Research* 6.12.

Durand, Audrey, Charis Achilleos, Demetris Iacovides, Katerina Strati, Georgios D Mitsis and Joelle Pineau (2018). 'Contextual bandits for adapting treatment in a mouse model of de novo carcinogenesis'. In: *Machine Learning for Healthcare Conference*.

Ecoffet, Adrien, Joost Huizinga, Joel Lehman, Kenneth O. Stanley and Jeff Clune (2019). 'Go-Explore: a New Approach for Hard-Exploration Problems'. In: *arXiv preprint arXiv:1901.10995*.

Engel, Yaakov, Shie Mannor and Ron Meir (2005). 'Reinforcement learning with Gaussian processes'. In: *International Conference on Machine Learning*.

Evans, Richard and Jim Gao (2016). 'Deepmind AI reduces Google data centre cooling bill by 40%'. In: *DeepMind blog* 20, p. 158.

Finn, Chelsea, Pieter Abbeel and Sergey Levine (2017). 'Model-agnostic meta-learning for fast adaptation of deep networks'. In: *International Conference on Machine Learning*.

Gal, Yarin (2016). 'Uncertainty in deep learning'. PhD thesis. University of Cambridge.

Gal, Yarin and Zoubin Ghahramani (2016). 'Dropout as a bayesian approximation: Representing model uncertainty in deep learning'. In: *International Conference on Machine Learning*.

Gómez-Bombarelli, Rafael, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams and Alán Aspuru-Guzik (2018). 'Automatic chemical design using a data-driven continuous representation of molecules'. In: *ACS central science* 4.2, pp. 268–276.

Guimaraes, Gabriel Lima, Benjamin Sanchez-Lengeling, Carlos Outeiral, Pedro Luis Cunha Farias and Alán Aspuru-Guzik (2017). 'Objective-reinforced generative adversarial networks (ORGAN) for sequence generation models'. In: *arXiv preprint arXiv:1705.10843*.

Hazan, Elad, Amit Agarwal and Satyen Kale (2007). 'Logarithmic regret algorithms for online convex optimization'. In: *Machine Learning* 69.2-3, pp. 169–192.

Henderson, Peter, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup and David Meger (2018). 'Deep reinforcement learning that matters'. In: *AAAI Conference on Artificial Intelligence*.

Hessel, Matteo, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Gheshlaghi Azar and David Silver (2018). 'Rainbow: Combining Improvements in Deep Reinforcement Learning'. In: *AAAI Conference on Artificial Intelligence*.

Houthooft, Rein, Xi Chen, Yan Duan, John Schulman, Filip De Turck and Pieter Abbeel (2016). 'VIME: Variational information maximizing exploration'. In: *Advances in Neural Information Processing Systems*.

Hutter, Frank, Holger H Hoos and Kevin Leyton-Brown (2011). 'Sequential model-based optimization for general algorithm configuration'. In: *International conference on learning and intelligent optimization*. Springer, pp. 507–523.

Janz, David, David Burt and Javier González (2020). 'Bandit optimisation of functions in the Matérn kernel RKHS'. In: *International Conference on Artificial Intelligence and Statistics*.

Janz, David, Jiri Hron, Przemysław Mazur, Katja Hofmann, José Miguel Hernández-Lobato and Sebastian Tschiatschek (2019). 'Successor Uncertainties: Exploration and Uncertainty in Temporal Difference Learning'. In: *Advances in Neural Information Processing Systems*.

Janz, David, Jos van der Westhuizen, Brooks Paige, Matt J Kusner and José Miguel Hernández-Lobato (2018). 'Learning a generative model for validity in complex discrete structures'. In: *International Conference on Learning Representations*.

Jin, Chi, Zhuoran Yang, Zhaoran Wang and Michael I Jordan (2020). 'Provably efficient reinforcement learning with linear function approximation'. In: *Conference on Learning Theory*.

Jones, Donald R, Cary D Perttunen and Bruce E Stuckman (1993). 'Lipschitzian optimization without the Lipschitz constant'. In: *Journal of Optimization Theory and Applications* 79.1, pp. 157–181.

Jumper, John, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko et al. (2021). 'Highly accurate protein structure prediction with AlphaFold'. In: *Nature* 596.7873, pp. 583–589.

Kanagawa, Motonobu, Philipp Hennig, Dino Sejdinovic and Bharath K Sriperumbudur (2018). 'Gaussian Processes and Kernel Methods: A Review on Connections and Equivalences'. In: *arXiv preprint arXiv:1807.02582*.

Kearns, Michael and Satinder Singh (2002). 'Near-optimal reinforcement learning in polynomial time'. In: *Machine Learning* 49.2, pp. 209–232.

Kendall, Alex, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley and Amar Shah (2019). 'Learning to drive in a day'. In: *International Conference on Robotics and Automation*.

Kingma, Diederik P and Jimmy Ba (2014). 'Adam: A method for stochastic optimization'. In: *arXiv preprint arXiv:1412.6980*.

Kirschner, Johannes, Manuel Nonnenmacher, Mojmír Mutný, Andreas Krause, Nicole Hiller, Rasmus Ischebeck and Andreas Adelmann (2019). 'Bayesian optimisation for fast and safe parameter tuning of SwissFEL'. In: *FEL2019, Proceedings of the 39th International Free-Electron Laser Conference*. JACoW Publishing, pp. 707–710.

Kleinberg, Robert, Aleksandrs Slivkins and Eli Upfal (2008). 'Multi-armed bandits in metric spaces'. In: *The ACM Symposium on Theory of computing*.

Lai, Tze Leung and Herbert Robbins (1985). 'Asymptotically efficient adaptive allocation rules'. In: *Advances in applied mathematics* 6.1, pp. 4–22.

Lattimore, Tor and Csaba Szepesvári (2020). *Bandit Algorithms*. Cambridge University Press.

Le Maître, Olivier and Omar M Knio (2010). *Spectral methods for uncertainty quantification: with applications to computational fluid dynamics*. Springer Science & Business Media.

Levine, Nir, Tom Zahavy, Daniel J Mankowitz, Aviv Tamar and Shie Mannor (2017). 'Shallow Updates for Deep Reinforcement Learning'. In: *Advances in Neural Information Processing Systems*.

Lilwall, Scott (2021). *ISL Adapt uses ML to make water treatment cleaner & greener*. URL: https://www.amii.ca/latest-from-amii/isl-adapt-uses-ml-make-water-treatment-cleaner-greener/ (visited on 09/02/2022).

Lin, Long-Ji (1992). *Reinforcement learning for robots using neural networks*. Carnegie Mellon University.

Lipton, Zachary C., Xiujun Li, Jianfeng Gao, Lihong Li, Faisal Ahmed and Li Deng (2018). 'BBQ-Networks: Efficient Exploration in Deep Reinforcement Learning for Task-Oriented Dialogue Systems'. In: *AAAI Conference on Artificial Intelligence*.

Locatelli, Andrea and Alexandra Carpentier (2018). 'Adaptivity to smoothness in x-armed bandits'. In: *Conference on Learning Theory*.

Louizos, Christos and Max Welling (2017). 'Multiplicative normalizing flows for variational bayesian neural networks'. In: *International Conference on Machine Learning*.

Matérn, Bertil (1960). 'Spatial variation: Stachastic models and their application to some problems in forest surveys and other sampling investigations'. PhD thesis. Stockholm University.

Minh, Ha Quang, Partha Niyogi and Yuan Yao (2006). 'Mercer's theorem, feature maps, and smoothing'. In: *International Conference on Computational Learning Theory*.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski et al. (2015). 'Human-level control through deep reinforcement learning'. In: *Nature* 518.7540, p. 529.

Moerland, Thomas M, Joost Broekens and Catholijn M Jonker (2017). 'Efficient exploration with Double Uncertain Value Networks'. In: *arXiv preprint arXiv:1711.10789*.

Munos, Rémi (2011). 'Optimistic optimization of deterministic functions without the knowledge of its smoothness'. In: *Advances in Neural Information Processing Systems*.

Munos, Rémi, Tom Stepleton, Anna Harutyunyan and Marc Bellemare (2016). 'Safe and efficient off-policy reinforcement learning'. In: *Advances in Neural Information Processing Systems*.

Mutný, Mojmír and Andreas Krause (2019). 'Efficient high dimensional bayesian optimization with additivity and quadrature fourier features'. In: *Advances in Neural Information Processing Systems*.

Nabati, Ofir, Tom Zahavy and Shie Mannor (2021). 'Online Limited Memory Neural-Linear Bandits with Likelihood Matching'. In: *arXiv preprint arXiv:2102.03799*.

Nair, Arun, Praveen Srinivasan, Sam Blackwell, Cagdas Alcicek, Rory Fearon, Alessandro De Maria, Vedavyas Panneershelvam, Mustafa Suleyman, Charles Beattie, Stig Petersen et al. (2015). 'Massively parallel methods for deep reinforcement learning'. In: *arXiv preprint arXiv:1507.04296*.

Nikolov, Nikolay, Johannes Kirschner, Felix Berkenkamp and Andreas Krause (2018). 'Information-directed exploration for deep reinforcement learning'. In: *arXiv preprint arXiv:1812.07544*.

O'Donoghue, Brendan, Ian Osband, Remi Munos and Volodymyr Mnih (2018). 'The Uncertainty Bellman Equation and Exploration'. In: *International Conference on Machine Learning*.

Olivecrona, Marcus, Thomas Blaschke, Ola Engkvist and Hongming Chen (2017). 'Molecular de-novo design through deep reinforcement learning'. In: *Journal of Cheminformatics* 9.1, pp. 1–14.

Osband, Ian, John Aslanides and Albin Cassirer (2018). 'Randomized prior functions for deep reinforcement learning'. In: *Advances in Neural Information Processing Systems*.

Osband, Ian, Charles Blundell, Alexander Pritzel and Benjamin Van Roy (2016). 'Deep exploration via bootstrapped DQN'. In: *Advances in Neural Information Processing Systems*.

Osband, Ian, Yotam Doron, Matteo Hessel, John Aslanides, Eren Sezener, Andre Saraiva, Katrina McKinney, Tor Lattimore, Csaba Szepesvari, Satinder Singh et al. (2019). 'Behaviour Suite for Reinforcement Learning'. In: *International Conference on Learning Representations*.

Osband, Ian, Daniel Russo and Benjamin Van Roy (2013). '(More) efficient reinforcement learning via posterior sampling'. In: *Advances in Neural Information Processing Systems*.

Osband, Ian and Benjamin Van Roy (2016). 'On lower bounds for regret in reinforcement learning'. In: *arXiv preprint arXiv:1608.02732*.

Osband, Ian, Benjamin Van Roy, Daniel J Russo, Zheng Wen et al. (2019). 'Deep Exploration via Randomized Value Functions'. In: *Journal of Machine Learning Research* 20.124, pp. 1–62.

Osband, Ian, Benjamin Van Roy and Zheng Wen (2016). 'Generalization and exploration via randomized value functions'. In: *International Conference on Machine Learning*.

Osborne, Michael A, Roman Garnett and Stephen J Roberts (2009). 'Gaussian processes for global optimization'. In: *International Conference on Learning and Intelligent Optimization*, pp. 1–15.

Ostrovski, Georg, Marc G Bellemare, Aäron Oord and Rémi Munos (2017). 'Count-based exploration with neural density models'. In: *International Conference on Machine Learning*.

Pathak, Deepak, Pulkit Agrawal, Alexei A Efros and Trevor Darrell (2017). 'Curiosity-driven exploration by self-supervised prediction'. In: *International Conference on Machine Learning*.

Pena, Victor H de la, Michael J Klass and Tze Leung Lai (2004). 'Self-normalized processes: exponential inequalities, moment bounds and iterated logarithm laws'. In: *Annals of probability*, pp. 1902–1933.

Petersen, Kaare Brandt, Michael Syskind Pedersen et al. (2008). 'The matrix cookbook'. In: *Technical University of Denmark*.

Plappert, Matthias, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y Chen, Xi Chen, Tamim Asfour, Pieter Abbeel and Marcin Andrychowicz (2018). 'Parameter space noise for exploration'. In: *International Conference on Learning Representations*.

Pohlen, Tobias, Bilal Piot, Todd Hester, Mohammad Gheshlaghi Azar, Dan Horgan, David Budden, Hado van Barth-Maron Gabriel Hasselt, John Quan, Mel Večerík, Matteo Hessel, Rémi Munos and Olivier Pietquin (2018). 'Observe and look further: Achieving consistent performance on atari'. In: *arXiv preprint arXiv:1805.11593*.

Pohlmeyer, Eric A, Babak Mahmoudi, Shijia Geng, Noeline W Prins and Justin C Sanchez (2014). 'Using reinforcement learning to provide stable brain-machine interface control despite neural input reorganization'. In: *PloS one* 9.1, e87253.

Pollack, Jordan B and Alan D Blair (1997). 'Why did TD-gammon work?' In: *Advances in Neural Information Processing Systems* 9.9, pp. 10–16.

Quinonero-Candela, Joaquin, Carl Edward Rasmussen and Christopher KI Williams (2007). 'Approximation methods for Gaussian process regression'. In: *Large-scale kernel machines*. MIT Press, pp. 203–223.

Rasmussen, Carl Edward and Christopher K. I. Williams (2006). *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press. ISBN: 026218253X.

Ritter, Klaus, Grzegorz W Wasilkowski, Henryk Wozniakowski et al. (1995). 'Multivariate integration and approximation for random fields satisfying Sacks-Ylvisaker conditions'. In: *Annals of Applied Probability* 5.2, pp. 518–540.

Riutort-Mayol, Gabriel, Paul-Christian Bürkner, Michael R Andersen, Arno Solin and Aki Vehtari (2020). 'Practical Hilbert space approximate Bayesian Gaussian processes for probabilistic programming'. In: *arXiv preprint arXiv:2004.11408*.

Rummery, Gavin A and Mahesan Niranjan (1994). *On-line Q-learning using connectionist systems*. Vol. 37.

Russo, Daniel and Benjamin Van Roy (2014). 'Learning to optimize via information-directed sampling'. In: *Advances in Neural Information Processing Systems*.

Salgia, Sudeep, Sattar Vakili and Qing Zhao (2020). 'A Computationally Efficient Approach to Black-box Optimization using Gaussian Process Models'. In: *arXiv preprint arXiv:2010.13997*.

Scarlett, Jonathan (2018). 'Tight Regret Bounds for Bayesian Optimization in One Dimension'. In: *International Conference on Machine Learning*.

Scarlett, Jonathan, Ilija Bogunovic and Volkan Cevher (2017a). 'Lower Bounds on Regret for Noisy Gaussian Process Bandit Optimization'. In: *Conference on Learning Theory*.

Scarlett, Jonathan, Ilija Bogunovic and Volkan Cevher (2017b). 'Lower bounds on regret for noisy gaussian process bandit optimization'. In: *Conference on Learning Theory*.

Schmidhuber, Jürgen (1991). 'A possibility for implementing curiosity and boredom in model-building neural controllers'. In: *Proc. of the International Conference on Simulation of Adaptive Behavior: From Animals to Animats*.

Schrittwieser, Julian, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel et al. (2020). 'Mastering atari, go, chess and shogi by planning with a learned model'. In: *Nature* 588.7839, pp. 604–609.

Seeger, Matthias, Christopher Williams and Neil Lawrence (2003). *Fast forward selection to speed up sparse Gaussian process regression*. Tech. rep.

Seeger, Matthias W, Sham M Kakade and Dean P Foster (2008). 'Information consistency of nonparametric Gaussian process methods'. In: *IEEE Transactions on Information Theory* 54.5, pp. 2376–2382.

Shahriari, Bobak, Kevin Swersky, Ziyu Wang, Ryan P Adams and Nando De Freitas (2015). 'Taking the human out of the loop: A review of Bayesian optimization'. In: *Proceedings of the IEEE* 104.1, pp. 148–175.

Shang, Xuedong, Emilie Kaufmann and Michal Valko (2019). 'General parallel optimization a without metric'. In: *Algorithmic Learning Theory*.

Shawe-Taylor, John and Steffen Grünewalder (2010). *Advanced Topics in Machine Learning: Part 1*. URL: http://www.cs.ucl.ac.uk/staff/J.Shawe-Taylor/courses/ATML-2.pdf (visited on 03/07/2021).

Silver, David, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot et al. (2016). 'Mastering the game of Go with deep neural networks and tree search'. In: *nature* 529.7587, pp. 484–489.

Silver, David, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel et al. (2017). 'Mastering chess and shogi by self-play with a general reinforcement learning algorithm'. In: *arXiv preprint arXiv:1712.01815*.

Simm, Gregor, Robert Pinsler and José Miguel Hernández-Lobato (2020). 'Reinforcement learning for molecular design guided by quantum mechanics'. In: *International Conference on Machine Learning*.

Snoek, Jasper, Hugo Larochelle and Ryan P Adams (2012). 'Practical bayesian optimization of machine learning algorithms'. In: *Advances in Neural Information Processing Systems*.

Solin, Arno and Simo Särkkä (2020). 'Hilbert space methods for reduced-rank Gaussian process regression'. In: *Statistics and Computing* 30.2, pp. 419–446.

Srinivas, Niranjan, Andreas Krause, Sham Kakade and Matthias Seeger (2010). 'Gaussian process optimization in the bandit setting: no regret and experimental design'. In: *International Conference on Machine Learning*.

Srinivas, Niranjan, Andreas Krause, Sham M Kakade and Matthias Seeger (2009). 'Gaussian process optimization in the bandit setting: No regret and experimental design'. In: *International Conference on Machine Learning*.

Stadie, Bradly C, Sergey Levine and Pieter Abbeel (2015). 'Incentivizing exploration in reinforcement learning with deep predictive models'. In: *arXiv preprint arXiv:1507.00814*.

Stein, Elias M (1970). *Singular Integrals and Differentiability Properties of Functions*. Princeton University Press.

Stein, Michael L (1999). *Interpolation of Spatial Data: Some Theory for Kriging*. Springer Science & Business Media.

Strehl, Alexander L, Lihong Li, Eric Wiewiora, John Langford and Michael L Littman (2006). 'PAC model-free reinforcement learning'. In: *International Conference on Machine Learning*.

Strens, Malcolm (2000). 'A Bayesian framework for reinforcement learning'. In: *International Conference on Machine Learning*.

Surjanovic, S. and D. Bingham (2013). *Virtual Library of Simulation Experiments: Test Functions and Datasets*. URL: http://www.sfu.ca/~ssurjano (visited on 22/02/2020).

Sutton, Richard S (1988). 'Learning to predict by the methods of temporal differences'. In: *Machine learning* 3.1, pp. 9–44.

Sutton, Richard S, Andrew G Barto et al. (1998). *Reinforcement learning: An introduction*. MIT press.

Szepesvári, Csaba (2010). 'Algorithms for reinforcement learning'. In: *Synthesis lectures on artificial intelligence and machine learning* 4.1, pp. 1–103.

Szepesvári, Csaba (2021). *RL Theory*. URL: https://rltheory.github.io (visited on 14/08/2021).

Talebi, Mohammad Sadegh, Zhenhua Zou, Richard Combes, Alexandre Proutiere and Mikael Johansson (2017). 'Stochastic online shortest path routing: The value of feedback'. In: *IEEE Transactions on Automatic Control* 63.4, pp. 915–930.

Tang, Haoran, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip DeTurck and Pieter Abbeel (2017). '#Exploration: A study of count-based exploration for deep reinforcement learning'. In: *Advances in Neural Information Processing Systems*.

Tang, Yunhao and Alp Kucukelbir (2017). 'Variational deep q network'. In: *arXiv preprint arXiv:1711.11225*.

Team, AlphaStar (2019). 'Alphastar: Mastering the real-time strategy game starcraft ii'. In: *DeepMind blog* 24.

Tesauro, Gerald et al. (1995). 'Temporal difference learning and TD-Gammon'. In: *Communications of the ACM* 38.3, pp. 58–68.

Thompson, William R (1933). 'On the likelihood that one unknown probability exceeds another in view of the evidence of two samples'. In: *Biometrika* 25.3/4, pp. 285–294.

Touati, Ahmed, Harsh Satija, Joshua Romoff, Joelle Pineau and Pascal Vincent (2019). 'Randomized Value Functions via Multiplicative Normalizing Flows'. In: *Uncertainty in Artificial Intelligence*.

Turner, Ryan, David Eriksson, Michael McCourt, Juha Kiili, Eero Laaksonen, Zhen Xu and Isabelle Guyon (2021). 'Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020'. In: *NeurIPS 2020 Competition and Demonstration Track*. PMLR, pp. 3–26.

Ubelacker, Sheryl (2019). *From bionic arms to predicting patient surges in ER, AI is reshaping patient care*. URL: https://www.cbc.ca/news/canada/edmonton/bionic-arms-artificial-intelligence-patient-care-alberta-1.5090172 (visited on 09/02/2022).

Vakili, Sattar, Kia Khezeli and Victor Picheny (2021). 'On information gain and regret bounds in Gaussian process bandits'. In: *International Conference on Artificial Intelligence and Statistics*.

Valko, Michal, Alexandra Carpentier and Rémi Munos (2013). 'Stochastic simultaneous optimistic optimization'. In: *International Conference on Machine Learning*.

Valko, Michal, Nathan Korda, Rémi Munos, Ilias Flaounas and Nello Cristianini (2013). 'Finite-Time Analysis of Kernelised Contextual Bandits'. In: *Uncertainty in Artificial Intelligence*.

Van Hasselt, Hado, Arthur Guez and David Silver (2016). 'Deep Reinforcement Learning with Double Q-Learning'. In: *AAAI Conference on Artificial Intelligence*.

Van Seijen, Harm, Hado Van Hasselt, Shimon Whiteson and Marco Wiering (2009). 'A theoretical and empirical analysis of Expected Sarsa'. In: *Adaptive Dynamic Programming and Reinforcement Learning*.

Vinyals, Oriol, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev et al. (2019). 'Grandmaster level in StarCraft II using multi-agent reinforcement learning'. In: *Nature* 575.7782, pp. 350–354.

Wang, Ziyu, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot and Nando De Freitas (2016). 'Dueling Network Architectures for Deep Reinforcement Learning'. In: *International Conference on Machine Learning*.

Watkins, Christopher JCH (1989). 'Learning from delayed rewards'. PhD thesis. King's College, Cambridge.

Watkins, Christopher JCH and Peter Dayan (1992). 'Q-learning'. In: *Machine learning* 8.3-4, pp. 279–292.

Wendland, Holger (2004). *Scattered data approximation*. Vol. 17. Cambridge University Press.

Widom, Harold (1963). 'Asymptotic behavior of the eigenvalues of certain integral equations'. In: *Transactions of the American Mathematical Society* 109.2, pp. 278–295.

Yang, Yun, Mert Pilanci, Martin J Wainwright et al. (2017). 'Randomized sketches for kernels: Fast and optimal nonparametric regression'. In: *The Annals of Statistics* 45.3, pp. 991–1023.

Youla, D (1957). 'The solution of a homogeneous Wiener-Hopf integral equation occurring in the expansion of second-order stationary random functions'. In: *IRE Transactions on Information Theory* 3.3, pp. 187–193.

Zahavy, Tom and Shie Mannor (2019). 'Deep neural linear bandits: Overcoming catastrophic forgetting through likelihood matching'. In: *arXiv preprint arXiv:1901.08612*.

Zanette, Andrea, David Brandfonbrener, Emma Brunskill, Matteo Pirotta and Alessandro Lazaric (2020). 'Frequentist regret bounds for randomized least-squares value iteration'. In: *International Conference on Artificial Intelligence and Statistics*.

Zhou, Ding-Xuan (2002). 'The covering number in learning theory'. In: *Journal of Complexity* 18.3, pp. 739–767.

Zhou, Qian, XiaoFang Zhang, Jin Xu and Bin Liang (2017). 'Large-scale bandit approaches for recommender systems'. In: *International Conference on Neural Information Processing*.

Zhou, Zhenpeng, Steven Kearnes, Li Li, Richard N Zare and Patrick Riley (2019). 'Optimization of molecules via deep reinforcement learning'. In: *Scientific Reports* 9.1, pp. 1–10.