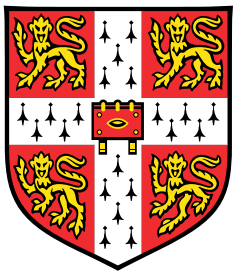
Exploration and Exploitation: From Bandits to Bayesian Optimisation



Eli Persky

Supervisor: Prof. C. E. Rasmussen

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Master of Philosophy in Machine Learning and Machine Intelligence

Wolfson College

August 2021

Declaration

I, Eli Persky of Wolfson College, being a candidate for the MPhil in Machine Learning and Machine Intelligence, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

The software written for this project used Python 3.7 and the NumPy and scikit-learn libraries. A squared-exponential Gaussian process algorithm for computing the Gittins index was previously written by Prof. Rasmussen in Matlab. This code was used only to verify the outputs of the original software written for the project. All software used for experiments is original and can be found on Github[1].

Word count: 14,916

Eli Persky
August 2021

[1] `github.com/Eli-Persky/EEBO`

Acknowledgements

I would like to express thanks to Prof. Carl Rasmussen for his indispensable role in this project. Your ideas and insights made this work possible, and your guidance made it an instructive and manageable task to undertake. I have learned a lot, not just about the subjects of this project, but about performing research itself from your supervision.

I would also like to acknowledge the fantastic work from Dr. Rich Turner and all of the staff running the MLMI course. Especially in this most challenging of years, their efforts have made it an entirely worthwhile and enjoyable experience.

Abstract

This project aims to offer a theoretically principled method for approaching the task of balancing exploration and exploitation in the setting of correlated multi-arm bandits, and to extend this application to encompass Bayesian optimisation. Our method is an attempt to generalise the Gittins index algorithm for multi-arm bandits to settings where bandit reward distributions can be correlated and, following that, settings with a continuum of bandits, leading to Bayesian optimisation. Our algorithm has no hyperparameters to set and is theoretically principled in the sense that it is motivated by consideration of the optimality of the Gittins index for independent bandits. We also derive an efficient algorithm for computing the Gittins index based on Gaussian process approximations to the value function using cubic spline covariance functions. Our Gittins based algorithm is demonstrated to outperform many standard approaches to multi-arm bandits when the bandits are correlated and is competitive with standard algorithms for Bayesian optimisation, especially outperforming those without free hyperparameters.

Table of contents

Chapter 1

Introduction

Two roads diverged in a wood, and I—
I took the one less traveled by,
And that has made all the difference.

— Robert Frost, *The Road Not Taken*

1.1 Motivation

The conflict between exploration and exploitation can be found within almost all decision making processes, human or machine. Wherever uncertainty exists, we must choose between actions about which we are the most confident of the consequences or to take the road less traveled by, taking on greater risk for the chance of greater reward. Exploration can uncover valuable information, and provide increased certainty about the options available to us which we may be able to exploit in the future.

This conflict is made explicit in the context of reinforcement learning. When we wish to design an agent which will make effective decisions in an uncertain environment, a fundamental consideration must be determining an appropriate level of exploration. This aspect is condensed in essential form in the class of problems called multi-arm bandits. Studying these problems can give us insight into the nature of the exploration and exploitation and how they can be managed.

In a meaningful sense, the classic multi-arm bandit problem is solved; the Gittins index algorithm provides an optimal method, under fairly general conditions. However, we wish to peel back some of these conditions to understand just how essential they are to the performance of this algorithm. We will allow bandits to have correlated rewards and adapt the algorithm in a

way which provides a principled approach to the problem, grounded in the theory established by Gittins and many others. This is in contrast to many existing algorithms which, while empirically effective, often lack an objective justification as to why they should be expected to be the best approach. We then go further and allow bandits to exist on a continuum, rather than in a discrete, finite set. This is a change which allows the problem to encompass Bayesian optimisation, taking the Gittins index even further away from its natural habitat.

A drawback to the Gittins index is that it is often ignored as a practical solution to the multi-arm bandit problem because of its perceived difficulty to calculate and implement. We hope to help in alleviating this perception by adding to the literature a method of computing the index which is accurate and computationally efficient. We will use this method to compute the index for ourselves which will enable us to implement our approaches to the aforementioned generalised multi-arm bandit problems.

In relaxing some of the assumptions which make the Gittins index an optimal algorithm we lose the theoretical guarantees which have been proved for it in its native form. Because of this, we cannot know for sure whether it will continue to be quite so effective when we exploit it in a context for which it was not specifically designed. However, by exploring these uncertain edges of the theory we hope to find that we can establish a principled and powerful method for these broader classes of problems.

1.2 Outline

Here we briefly outline the contents of each of the coming chapters and their place within the overall structure of the project.

This project brings together ideas from a number of different but related areas of machine learning and statistics. Chapter 2 aims to prime the reader with the necessary basics to be able to follow the subsequent chapters and to define a notation and terminology for the concepts discussed which will be carried through the rest of the work. We also offer references to related work for each of the subjects covered in the chapter, including works with demonstrate or prove useful results or apply the subjects to a diverse range of problems. The subjects covered in Chapter 2 are as follows:

- **Balancing exploration and exploitation in reinforcement learning.** We give a broad description of the aims of reinforcement learning and define some useful notation for Markov decision processes which lay the groundwork for discussing multi-arm

bandits. We discuss the idea of exploration and exploitation and the value of balancing these concepts and we refer to some related works on the subject.

- **Multi-arm bandits**. We restrict our discussion of reinforcement learning to this specific class of problems. We define some further terminology and discuss the general idea of selecting policies to balance exploration and exploitation in this context. Multi-arm bandits are the framework within which the problems this project aims to address can be defined.

- **The Gittins index**. We discuss this particular algorithm for selecting bandits in multi-arm bandits problems, which is provably optimal. We provide an alternative viewpoint on multi-arm bandits in terms of information states in order to reach a formulation of the Gittins index by intuitive arguments. We then offer a proof that the formulation we have arrived at is in fact the optimal algorithm. The Gittins index algorithm will form the basis for the algorithms we will go on to develop and we take its optimality in its own domain as justification for our approach.

- **Gaussian processes**. These form the backbone of the algorithms developed in this project. We heavily depend on Gaussian processes in both our method for computing the Gittins index and for our Gittins based algorithms for correlated bandits and Bayesian optimisation. We define some notation and terminology and describe the most central results needed to follow the algorithms to which they are applied.

- **Bayesian optimisation**. Since Bayesian optimisation is one of the domains we attempt to develop a Gittins based algorithm for, we use this opportunity to define this class of problems and to refer the reader to a number of existing methods, some of which will be used in our experiments.

Having covered the necessary background and pointed the reader towards related work, we proceed in Chapter 3 to develop the first algorithms of this project. This chapter details an algorithm for the computation of the Gittins index using Gaussian process approximations. We first detail a version of the algorithm using squared-exponential covariance functions and we then derive a version using cubic spline covariances. We finally illustrate how the computed Gittins index looks using various discount factors.

Chapter 4 details the algorithms we have designed, based off the Gittins index, to address the problem of multi-arm bandits with correlated bandits and Bayesian optimisation. Implimentation of these algorithms will require computation of the Gittins index, so this chapter builds on the work covered in Chapter 3. While these general problem classes were defined and discussed in Chapter 2, we go into greater detail on existing methods for these problems which will be used later in our experiments.

Chapter 5 covers our experiments on multi-arm bandits and Bayesian optimisation. We compare the performance of the Gittins based algorithms covered in Chapter 4 to the other algorithms which were also discussed in that chapter.

Finally, in Chapter 6 we provide our final discussion of the merits and weaknesses of the algorithms developed in this project. We discuss their performance in the experiments as well as their place in the broader landscape of algorithms for multi-arm bandits and Bayesian optimisation. We suggest a number of avenues for potential future work which could extend, improve and further investigate the work undertaken in this project.

Chapter 2

Background and Related Work

2.1 Balancing exploration and exploitation in reinforcement learning

A task in reinforcement learning (Sutton & Barto, 2018) can be summarised as an agent attempting to make good decisions with respect to some predefined goal, or equivalently developing a policy which produces good decisions, and it does this by interacting with an environment and observing the effects of its actions, possibly generating a model of the environment. These objects: The environment, the agent's goal, the policy and, if relevant, the model can be seen as the elements which provide a framework into which mostly any RL problem can be situated.

This framework contains multiple potential sources of uncertainty for the agent. The environment may be affected by the agent's actions in unpredictable ways and the agent must take into account this uncertainty when evaluating how it wishes to interact with the environment. On top of this, the state of the environment itself may be uncertain. Sometimes the agent simply does not have access to all the possible information which it must instead infer from its observations. Even when this information is provided we may model it has having some epistemic uncertainty associated with it; if a robot is equipped with a sensor we can not be sure that its measurements are perfectly accurate, and failing to account for this uncertainty can lead to failure in the task.

A further difficulty which must be overcome is that an agent's goal will typically be long-term in some sense. For example, we may be interested in maximising the accumulated reward received over a period of time. While the agent may observe the immediate rewards it

receives as a consequence of its actions, the effect of a given action on the long-term goal may be unpredictable. A policy which aims for maximal instantaneous reward may fail at the overall task. This thought motivates the study of the balance of exploration and exploitation which will be discussed after we establish a formal notation for the problems that have been described.

At any time t, the environment has a *state* $s_t \in \mathscr{S}$ where $\mathscr{S}$ is the state space containing all possible states of the environment. At each time, the agent must select an *action* $a_t \in \mathscr{A}$ where $\mathscr{A}$ is the action space of possible actions. The agent then receives a *reward* $r_t \in \mathbb{R}$. For each state of the environment there exists a reward function $r : a \to \mathbb{R}$ which determines r_t. Before they are observed these values are all uncertain so we will represent them as random variables by S_t, A_t and R_t for states, actions and rewards respectively which have s_t, a_t and r_t as their realisations.

We also define a distribution P_a for each action a which returns the transition probability between any two states given the action made in the first state, $P_a(s,s') = \mathbb{P}[S_{t+1} = s'|S_t = s, A_t = a]$. These elements are what we need to construct a Markov decision process (MDP) which will be the informational structure of the problem. As we have seen, the transition probabilities are Markov as they have no explicit dependence on states earlier than the current state S_t. Furthermore, we will restrict the agent's policies to be a mapping which takes the current state and returns a probability distribution over the action space, so the decision process has no explicit dependence on earlier states. For a policy π the agent selects actions according to the distribution it defines, $\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$. In general the policy is stochastic but in many situations we would use deterministic policies which are a pure mapping from a state to a single action.

In order to quantify the long-term value of a being in a state define the value function $V_\pi(s)$ for policy π in state s as

$$V_\pi(s) = \mathbb{E}_\pi[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots | S_t = s] \tag{2.1}$$

$$= \mathbb{E}_\pi[r_{t+1} + \gamma V_\pi(S_{t+1}) | S_t = s] \tag{2.2}$$

where $0 < \gamma < 1$ is the *discount factor*. A policy π is optimal if $V_\pi(s) \geq V_{\pi'}(s)$ for any state s and policy π'.

To see why γ is important, imagine setting $\gamma = 1$ in the above value function. If the length of time or *horizon* of the problem is infinite then, unless the problem is set up such that the expected rewards themselves decay sufficiently rapidly over time, this value will be infinite. As we will be studying problems with stationary reward distributions, this concern is very relevant. There is no way to distinguish between the desirability of two states if their values are both infinite. Intuitively we know that even if two states will both lead to an expected infinite total reward, we may prefer one to the other because it delivers its rewards sooner. This is in tidy analogy with the concept from finance that 'money today is worth more than money tomorrow', which is true because money received today can be safely invested and be worth more tomorrow. In finance, this interest rate informs the discount factor applied to a cash flow, to represent that future inflows and outflows experience a 'time cost' and are worth less in present value. This is the intuitive reason for using discounting when considering infinite horizons. Mathematically, they make value functions finite allowing the problem of selecting a policy to be well-defined.

A key way in which RL differs from supervised learning is that the reward received by an agent after taking an action gives a very uninformative numerical evaluation of the decision. In supervised learning, the loss function takes into account not only the strength of the model's prediction but also how it could have done better and the reward (the negative loss) is scaled according to how close the prediction was to optimal. In RL the agent does not receive this rich of information; if a policy results in a reward of, say, 10, that does very little to help the agent know if this was a good series of decisions or how the policy can be improved. It needs to learn for itself what rewards it could have received had it made different decisions. It is possible that there may be much better policies hidden somewhere in the very high dimensional policy space which the agent can only locate by trying many different things. If the agent only takes actions which it believes are the best out of what it has tried so far, it will never learn what other, possibly greater, options exist. This makes *exploration*, or trying actions which the agent knows little or nothing about the effects of, essential for finding good policies. Only when enough exploration has been performed that it is unlikely to result in much more long-term gain does *exploitation* of the learned information become the most profitable way forward. Exploiting actions which are known to be relatively rewarding may gain us some information about the reward distribution associated with that action and the local dynamics of the environment, but fundamentally it does little to help us learn about the breadth of options that exist in policy space. In other words, we cannot simultaneously explore and exploit- they must be traded off and balanced. To explore, in general, comes at a cost of immediate rewards and (considering discounting) defers ex-

ploitation to later times when its rewards are worth less. There must come a point where exploration is too expensive to justify its costs but finding that point is a complex and subtle challenge. Furthermore, it is not enough to simply instruct an agent to explore. We must decide where to explore; in what regions of policy space is exploration most likely to yield fruit?

MDPs were first studied by Wald (1949) and Bellman (1957). A review of this area can be found in Teugels (1976) and a more modern treatment is given by Puterman (2014). Blackwell (1965) proved important results about MDPs including the existence and uniqueness of a solution to the dynamic programming form of the problem and that the optimal policy is deterministic, stationary and Markov. The problem of balancing exploration and exploitation was discussed in the context of control theory by Feldbaum (1965) as the dual control problem and Witten (1976), referring to the balance between identification and control which are essentially the same concepts as exploration and exploitation. The concept is not usually studied in and of itself, but rather in the context of particular algorithms or particular RL problems such as multi-arm bandits, which are discussed in the following section.

2.2 Multi-arm bandits

The basic form of the multi-arm bandit (MAB) problem is as follows: There exists an action space $\mathscr{A}$ containing k actions which remain the same at all times. Each action is associated with a stationary probability distribution. At each time step, the agent must select one action and it receives a reward drawn from the distribution associated with the chosen action. The name derives from the concept of a one-armed bandit, a gambling machine in which the user pulls a single lever (an arm) and receives a random reward each time. We imagine our agent being faced with many machines or equivalently one machine with many arms, all of which generate different distributions of rewards. In some literature the choice of action is referred to as an *arm*, in others it is slightly counter intuitively called a *bandit*. We will opt to use the latter.

MABs were first introduced in the context of statistics by Thompson (1933, 1934) and by Robbins (1952) and were further developed by Bellman (1956). They have also been studied from an engineering context by Narendra and Thathachar (1989). Despite how old this field of study is, this class of problem continues to have a number of applications to this day, typically involving the efficient allocation of limited resources (Bouneffouf & Rish, 2019). Modern applications include the exploration of promising treatments in clinical trials (Durand et al., 2018), modeling psychological behaviour (Beneffouf et al., 2017), portfolio selection

in finance (Shen et al., 2015), recommender systems (Zhou et al., 2017) and information retrieval (Losada et al., 2017).

MABs might be considered the simplest setting of RL. The environment is stateless, or more precisely, the state space contains exactly one state which occurs at every time step. This fact means we do not need to consider the dynamics of the environment in terms of transition probabilities. This also entails the stationarity of the reward distributions, a convenient property which allows us to quite simply build a model of the distributions from the observed data. Furthermore, it is usually assumed in the literature that the distributions associated with the different bandits are independent, however this is often not explicitly stated but taken for granted. We explicitly mention this point here because independence of the bandits is an assumption which we will attempt to relax in this work.

The precise goal of the agent depends on the specific setup of the problem. If we consider a finite horizon, i.e. the agent has a number $T \in \mathbb{N}$ of steps to perform its actions then it may be reasonable to aim to optimise the total reward received. If, however, the horizon is infinite then, as discussed above, we must apply discounting to the rewards. We will be focusing on this case so at any time t we attempt to maximise the objective G_t known as the *payoff* where

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}. \tag{2.3}$$

Since the agent begins with no knowledge of the reward distributions, we must construct a model based on the observations it has seen so far at any moment. For this work we will restrict our problems to a narrow class of reward of distributions. We assume that for any action a and time t,

$$R_t \sim \mathcal{N}(q_*(a), 1) \tag{2.4}$$

where $q_*(a) = \mathbb{E}[R_t]$, an unknown value, and $\mathcal{N}(\mu, \sigma^2)$ is the Gaussian distribution with mean μ and variance σ^2. There are two simple ways of tracking our current estimate $Q_t(a)$ for $q_*(a)$. The first is to simply track the total reward received so far $\Sigma_t(a)$ and divide it by $n_t(a)$, the number of times the bandit has produced a reward. The other approach which is also useful for non-stationary distributions is to make the update

$$Q_t(a) \leftarrow Q_{t-1}(a) + \frac{r_t - Q_{t-1}(a)}{n_t(a)}. \tag{2.5}$$

Since a was the bandit chosen at time t we have $n_{t-1}(a) = n_t(a) - 1$, so we can rearrange this expression to see that it is equivalent to

$$Q_t(a) \leftarrow \frac{Q_{t-1}(a)n_{t-1}(a) + r_t}{n_t(a)}. \tag{2.6}$$

With $Q_0(a) = 0$ we can see by induction that this update must produce the running mean of the rewards for $t \geq 1$.

As with any RL problem, the MAB problem requires a balance of exploration and exploitation. At any time it is possible for the agent to select the *greedy* option which is to select

$$a_t = \arg\max_a Q_t(a) \tag{2.7}$$

however, using this approach too much and too early will lead to poor results for the reasons discussed in the earlier section. Many strategies have been designed to tackle this problem, a number of which will be explained in chapter 4. These include upper confidence bound (Lai & Robbins, 1985), (Kaelbing, 1993), (Agrawal, 1995), (Auer et al., 2002) and Thompson sampling (Thompson, 1933), (Wyatt, 1998). Most of these algorithms attempt to associate the level of exploration with the degree of uncertainty about the bandits. This seems to be the correct approach, however the precise relationship between uncertainty and exploration is quite subtle and isn't entirely captured by those algorithms, leading to suboptimal performance. However, one algorithm, which will be the basis for this work, strikes this balance in the precisely optimal way and that is the Gittins index algorithm.

2.3 The Gittins index

The Gittins index (GI) was originally introduced as the *dynamic allocation index* (Gittins, 1974) however it was not widely known or recognised as the solution to the multi-arm bandit problem for a number of years, after which point it gained its current name. In the meantime a number of proofs of the GI's optimality have emerged, one of which we will outline in this chapter. First we must fully define the problem which the algorithm claims to solve. The following explanation paraphrases Gittins et al. (2011) but with simplified notation and with some of the more generalised aspects which are not relevant to this particular work elided.

A discrete time *bandit process* is a particular Markov decision process. *Controls* must be applied to the process at a succession of *decision times*, the first of which being $t = 0$. The

interval between decision times is a random variable (however, we will soon restrict our attention to bandit processes with constant intervals). The control which must be applied at each decision time must come from the set $\{0,1\}$ which are referred to as *freezing* and *continuation* respectively. If ever the freezing control is applied then every subsequent time step is considered a decision time until the continuation control is applied and the state of the system does not change until then. If, on the other hand, the continuation control is applied at time t then a reward of $\gamma^t r(S_t)$ is generated and the state changes. The next decision time is at a random point in the future. Note that we have reintroduced the notion of a state S_t despite earlier claiming that MAB problems do not have states. This represents a shift in point of view; while the state of the bandit is in some 'objective' sense constant, our knowledge of the bandit changes whenever it produces a reward. Hence, from the Bayesian point of view we can consider the posterior distribution of the rewards to have experienced an update and to have changed state. We will call these states *information states* to reflect the fact that in reality the bandit does not change, only our state of knowledge about it. We will parameterise the information state $S_t = (\Sigma_t, n_t)$ by Σ_t, the total reward received up to time t and n_t, the number of times the bandit has been activated up to time t. Under the assumptions we will impose on the reward distributions, these statistics are sufficient to make the Bayesian updates. The dynamics are very simple: when the bandit is continued and produces reward r_t then with probability 1, we have $S_{t+1} = (\Sigma_t + r_t, n_t + 1)$. The GI is not limited to processes in which the state is purely informational, in fact there is really no distinction for our purposes between a stationary system described by Bayesian information states and a system whose reward distributions genuinely change in perfect lock-step with those information states.

In a *Markov bandit process* the interval between decision times when the continuation control is applied is a constant, which we will set to 1 without loss of generality. A policy in this setting is a (possibly stochastic) mapping from states to the control set $\{0,1\}$. With this notion in mind we define a *simple family of alternative bandit processes* (SFABP) which is a collection of k Markov bandits processes, all of which are available to be continued at each time step, with the additional restriction that at any given time only one process can take the continuation control and all the rest must be frozen. Given this, we can redefine the control set for the SFABP as $\{1,\ldots,k\}$ where applying control i corresponds to applying continuation to bandit i and freezing all the rest. We will refer to the control applied at time t as i_t (this is also the identifier of the bandit with the continuation control) and the state of bandit i_t at this time as $S_t^{i_t} \in \mathscr{S}$ and the state of the entire SFABP as $S_t \in \mathscr{S}^k$. Furthermore we define the reward function as a function of the state so we will receive rewards from $r_{i_t}(S_t^{i_t})$

We are interested in determining a past-measurable policy π which optimises the payoff of this MDP i.e. we wish to find a policy which attains the value function

$$V(s) = \sup_{\pi} \mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t r_{i_t}(S_t^{i_t}) \middle| S_0 = s \right]. \tag{2.8}$$

By past-measurable, we simply mean the intuitive property that the decision made by the policy at time t is determined by the events up to time t. This value function can be expressed as a dynamic programming (DP) equation

$$V(s) = \max_{i \in 1,\ldots,k} \left\{ r_i(s^i) + \gamma \sum_{y \in \mathscr{S}} P_i(y|s^i) V(s^1, \ldots, s^{i-1}, y, s^{i+1}, \ldots, s^k) \right\} \tag{2.9}$$

where $s = (s^1, \ldots, s^k)$ and $P_i(y|s^i)$ is the transition probability from state s^i to state y for bandit i. If the states are continuous then this sum becomes an integral. The theory of MDPs tells us that there is at least one deterministic, stationary, Markov policy which is optimal and any optimal policy must optimise (2.9) for each state s.

However, the DP form of the problem is usually not useful for solving for the policy as the size of the problem grows exponentially with k, the number of bandits. This is without even considering the implications of the fact that the state space is continuous, owing to the continuous distribution we have assumed for the rewards in (2.4). If we were to use a discrete reward distribution e.g. Bernoulli, or if we were to discretise and truncate the rewards from a Gaussian distribution then the full state space would be of size $|\mathscr{S}^k| = |\mathscr{S}|^k = N > 2^k$ and the number of stationary, deterministic policies is k^N. In a linear programming formulation, we would need a number of variables of order $O(2^n)$ so the algorithm would be computationally intractable (Gittins et al., 2011).

The remarkable insight of Gittins is that the problem does not need to be solved by considering the DP equation for the whole SFABP, but rather the optimal policy can be found as an *index policy*. This means that for each bandit process B_i in the SFABP there exists an index $\nu(B_i, S^i) \in \mathbb{R}$ which can be computed independently for each bandit such that continuing the bandit with the greatest index yields an optimal strategy. The space of index policies is considerably smaller than the entire space of policies, as we are placing a strong restriction on the form the policy can take. For an index policy we can take each bandit individually, ignore all information from all other bandits and calculate the index, only mixing the information from the different bandits at the end when we compare the size of the indexes. Any policy

which cannot be computed in this way does not fall into the class of index policies. Hence, this restricting of the space of potentially optimal policies down to just the space of index policies makes hypothesising and proving an optimal policy a manageable task, as we will see.

Assuming this claim to be true, we can derive what that index must be. To do this, we imagine a SFABP containing only two bandits. One of these bandits is constructed as a *standard bandit process* meaning it always pays a reward λ when the continuation control is applied and its state never changes. We will call the other bandit B_i. An important observation is that, since continuing the standard bandit does not alter the state of the SFABP, if this action is optimal at any time it must also be optimal at all future times. The payoff for using the standard bandit in this way is

$$\lambda(1+\gamma+\gamma^2+\dots) = \frac{\lambda}{1-\gamma}. \tag{2.10}$$

The maximal payoff must be

$$\sup_{\tau>0} \mathbb{E}\left[\sum_{t=0}^{\tau-1} \gamma^t r_i(S_t^i) + \gamma^\tau \frac{\lambda}{1-\gamma} \middle| S_0^i = s^i\right] \tag{2.11}$$

where τ is a past-measurable stopping time whose value must be some decision time. This simply says that we receive rewards from the bandits B_i up to time τ, after which we start using the standard bandit for the rest of time, and the rewards for that are discounted by the factor γ^τ.

The difference between the payoff in (2.11) and the quantity generated by the standard bandit in (2.10) is how much better it is to continue the non-standard bandit than the standard bandit. Subtracting these and setting equal to 0 gives an equation satisfied by the value of λ such that we would be indifferent between continuing either of the processes.

$$0 = \sup_{\tau>0} \mathbb{E}\left[\sum_{t=0}^{\tau-1} \gamma^t r_i(S_t^i) - (1-\gamma^\tau)\frac{\lambda}{1-\gamma} \middle| S_0^i = s^i\right]. \tag{2.12}$$

This expression is convex and decreasing in λ so it has a unique root $\bar{\lambda}$ such that

$$\bar{\lambda} = \sup\left\{\lambda : \frac{\lambda}{1-\gamma} \leq \sup_{\tau>0} \mathbb{E}\left[\sum_{t=0}^{\tau-1} \gamma^t r_i(S_t^i) + \gamma^\tau \frac{\lambda}{1-\gamma} \middle| S_0^i = s^i\right]\right\} \tag{2.13}$$

$$= \sup\left\{\lambda : \sup_{\tau>0} \mathbb{E}\left[\sum_{t=0}^{\tau-1} \gamma^t (r_i(S_t^i) - \lambda) \middle| S_0^i = s^i\right] \geq 0\right\}. \tag{2.14}$$

From this expression we claim that this crucial value of λ is in fact exactly the index $\nu(B_i, s^i)$ for bandit B_i in state s^i as it can be interpreted as the maximum 'rent' one would be willing to pay per time step for ownership of the rewards the bandit will produce.

Rearranging the expression we see that

$$\nu(B_i, s^i) = \sup_{\tau>0} \frac{\mathbb{E}\left[\sum_{t=0}^{\tau-1} \gamma^t r_i(S_t^i) \middle| S_0^i = s^i\right]}{\mathbb{E}\left[\sum_{t=0}^{\tau-1} \gamma^t \middle| S_0^i = s^i\right]} \tag{2.15}$$

which can be seem as maximising the expected total discounted reward over τ steps divided by the total discounted time over the same number of steps. This formulation hopefully gives an intuitive idea of why this particular index yields the optimal strategy. Another way to express the index, which we will use when applying it later in the project is

$$\nu(\Sigma, n, \gamma) = \frac{\Sigma}{n} + \nu(0, n, \gamma). \tag{2.16}$$

Here we have reparametrised the arguments so that the state s^i is represented by Σ and n and the reference to the bandit B_i itself is replaced with just an input for the discount factor γ. We introduce this notation here because it is the way we will formulate the index going forward. This decomposition can be viewed as an expected immediate reward Σ/n and a term which is independent of Σ which we might call the *exploration bonus*. This number depends on n so, in effect, it is a function of the uncertainty in the estimate of q_*.

The index theorem for SFABPs (using the original notation) can simply be stated as:

Theorem 1. *A policy for a simple family of alternative bandits is optimal if it is an index policy with respect to* $\nu(B_1, \cdot), \nu(B_2, \cdot), \ldots, \nu(B_k, \cdot)$.

2.3.1 Proof of the Gittins index theorem

Many proofs of this theorem have been presented over the years including the original proof by interchanging bandit portions (Gittins, 1974) and others (El Karoui & Karatzas, 1993, 1994), (Ishikida & Varaiya, 1994), (Tsitsiklis, 1994). The one that perhaps provides the most intuition about the index, the proof by prevailing charges argument, is due to Weber (1992) and is outlined here.

We imagine a setting with a single bandit process B_i which can be continued or frozen at each time step and we return to the analogy mentioned above of paying a 'rent' each time the bandit is used for the rewards produced by the bandit. Taking into account the current state of the bandit s^i, we set up a sort of game which produces a stochastic process. This process will not ever actually be realised but by considering its expected behaviour we can prove the required result about the optimal decision at the start. The game works as follows:

We refer to the rent as the *prevailing charge*, $\lambda_i^*(S_t^i)$. If the prevailing charge is too large then we will freeze the bandit but if the charge is small enough then we will continue the bandit and make an expected profit. The *fair charge* $\lambda_i(S_t^i)$ is the level of rent at which we would be indifferent between freezing the bandit and continuing it for at least one more step (with the knowledge that we can optionally stop the bandit at some time later). The value of $\lambda_i(S_t^i)$ is the same as $\bar{\lambda}$ in (2.14) by definition.

The prevailing charge is initially set equal to the fair charge and if at any point the fair charge falls below the prevailing charge, making it optimal to stop playing the game, the prevailing charge is dropped to equal the current fair charge. This means that throughout the game $\lambda_i^*(S_t^i) = \min_{t' \leq t} \lambda_i(S_{t'}^i)$. The prevailing charge is non-increasing in time. Under the rules of this game, the expected profit at the outset is 0. Due to this, the total expected discounted reward must be equal to the total expected discounted prevailing charge.

We can now consider a SFABP of k bandits whose behaviours all follow the above description. By always choosing to continue the bandit with the greatest prevailing charge we can receive an expected profit of 0. This is because any policy corresponds to an interleaving of the k sequences of prevailing charges into a single sequence. Selecting the greatest prevailing charge at each moment interleaves them such that the discounted total is maximised. Since prevailing charges are never set below fair charges, the expected profit for playing each bandit is never positive, so there is no policy which receives a positive profit in expectation. Therefore this policy attains the maximum so it is optimal. At any moment when selecting

from multiple bandits we can imagine setting up this game using the current state of the bandits. The above argument tells us that we should pick the bandit with the greatest prevailing charge which is exactly equal, by construction, to the fair charge: the Gittins index.

2.4 Gaussian processes

Having hopefully established what the GI is and argued why it must produce an optimal policy, we need to consider how to calculate it. We will present an approach in chapter 3 to approximately calculate the GI which is based around Gaussian processes (GP). As well as calculating the GI, GPs will be central in our approach to utilising it in the context of correlated bandits and Bayesian optimisation in chapter 4. So we take some time here to define the GP and lay out some of the central facts which will be useful in the rest of this work. A key reference for all of the information relating to GPs is Rasmussen & Williams (2006).

A Gaussian process is a generalisation of the multivariate Gaussian distribution. While a Gaussian distribution describes the probabilistic behaviour of a random vector (a vector of random variables), a Gaussian process describes the behaviour of a function. One intuitive way to understand this difference is to consider a random vector $\boldsymbol{v} \in \mathbb{R}^d$ which follows a d-dimensional multivariate Gaussian distribution $\boldsymbol{v} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$ where $\boldsymbol{\mu} \in \mathbb{R}^d$ and $\Sigma \in \mathbb{R}^{d \times d}$ is symmetric and positive semi-definite. To select a single component from this vector we use an index i from the set $I = \{1, \ldots, d\}$ and take the component $(\boldsymbol{v})_i = v_i$. A Gaussian process can be thought of in much the same way except for the fact that the set containing the possible indices becomes uncountably infinite, in particular $I = \mathbb{R}$. This replaces the vector $\boldsymbol{v}$ with a function $f(x)$ indexed by $x \in \mathbb{R}$.

Making this change requires us to alter the objects which serve to describe the mean and covariance of the components. Instead of a mean vector $\boldsymbol{\mu}$ we require a *mean function* $\mu(\cdot)$. Instead of a covariance matrix Σ, we require a *covariance function*, or *kernel*, $k(\cdot, \cdot)$. A crucial fact about GPs is that the marginal distribution of f evaluated at any finite set of input points is always multivariate Gaussian. To be precise

$$\boldsymbol{f}(\boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{x}), K) \tag{2.17}$$

where $(\boldsymbol{\mu}(\boldsymbol{x}))_i = \mu(x_i)$ and $K_{ij} = k(x_i, x_j) = \mathrm{cov}(f(x_i), f(x_j))$ for any vector of inputs $\boldsymbol{x}$. This allows us to work with the GP, despite the uncountability which is built into it, by only

ever asking questions about the distribution on finite sets of points.

The choice of μ and k are very important and can be seen as describing the prior distribution over functions. The choice of k informs how the distributions of nearby points depend on each other. Finiteness of k ensures that the functions are almost surely continuous but the particular way in which they can vary over any given length scale depends on the form of the covariance function and the setting of its parameters.

An important covariance function is the *squared-exponential* (SE) covariance function (also known as radial basis function (RBF) or Gaussian kernel). This can be expressed as

$$k(x_i, x_j) = v^2 \exp\left(\frac{-|x_i - x_j|^2}{2l^2}\right) \tag{2.18}$$

where are v and l are (hyper)parameters which need to be set. We can see that the covariance approaches the constant v^2 as $|x_i - x_j|$ approaches 0 so this number can be seen of the marginal variance at this point. The magnitude of the covariance decays exponentially as the distance between the two points increases. The rate of this decay depends on the length scale l. The larger l is then the greater the distance can be between two points to still have a given level of positive covariance. As well as the squared-exponential, a less commonly seen covariance function, the cubic spline, is applied in our method for calculating the Gittins index and is described in chapter 3.

2.4.1 GP prediction

A useful property of GPs is that we can calculate the posterior distribution in closed form given a dataset of input points and function evaluations. We will assume that measurements are made with some level of noise so we would like the posterior GP to produce function which pass through, or close to, the observed points. We would also expect the uncertainty in the distribution to be low near those points and larger at points far away from the datapoints. We observe this exact behaviour when we make the Bayesian update and compute the posterior GP. If we observe a dataset $X, \boldsymbol{y}$ and wish to find the predictive distribution at a set

of test points X_* then we have

$$\begin{aligned} \boldsymbol{f}_*|X,\boldsymbol{y},X_* &\sim \mathcal{N}(\bar{\boldsymbol{f}}_*, \text{cov}(\boldsymbol{f}_*)) \\ \bar{\boldsymbol{f}}_* &= K(X_*,X)(K(X,X)+\sigma_n^2 I)^{-1}\boldsymbol{y} \qquad (2.19) \\ \text{cov}(\boldsymbol{f}_*) &= K(X_*,X_*) - K(X_*,X)(K(X,X)+\sigma_n^2 I)^{-1}K(X,X_*) \qquad (2.20) \end{aligned}$$

a result which we will use extensively. Here σ_n^2 is the noise level added to the diagonal of the covariance matrix of the training data to represent the measurement noise. It can be important to ensure the numerical stability of the matrix inversion by keeping the eigenvalues of the matrix away from 0. This computation can become expensive due to the need to invert a matrix of dimension equal to the number of datapoints. Scalability can be improved with a technique such as sparse Gaussian processes (Titsias, 2009).

This allows us to perform regression in a Bayesian way as we can fit a GP to a set of datapoints and make predictions about the function values at points in between the datapoints. We will especially make use of this in this project to interpolate functions for which we only have computed values at a discrete grid of points.

2.4.2 Hyperparemeter adaptation

Given a choice of covariance function, we will in general have some number of hyperparameters to determine. For example, the length scale of the squared-exponential covariance controls how rapidly nearby function values are able to fluctuate. If the length scale is short then the functions can vary rapidly and points beyond a short distance apart are effectively uncorrelated, whereas with a long length scale the functions can only fluctuate much more slowly with high probability. This entails a trade-off between the simplicity of the functions prioritised by the distribution and the ability of those functions to explain the observed data. The way of evaluating the strength of a choice of hyperparameters is the *marginal likelihood*, the integral of the prior multiplied by the likelihood

$$p(\boldsymbol{y}|X) = \int p(\boldsymbol{y}|\boldsymbol{f},X)p(\boldsymbol{f}|X)d\boldsymbol{f} \qquad (2.21)$$

which will be a function of the hyperparameters.

Performing the calculations it can be shown that for a GP

$$\log p(\boldsymbol{y}|X) = -\frac{1}{2}\boldsymbol{y}^T(K+\sigma_n^2 I)^{-1}\boldsymbol{y} - \frac{1}{2}\log|K+\sigma_n^2 I| - \frac{n}{2}\log 2\pi. \tag{2.22}$$

where n is the number of datapoints. This is the function which should be optimised to find the best hyperparameters. Any optimisation method can be used for this task depending on the specific circumstances, e.g. whether the gradient with respect to the hyperparameters θ, $\nabla_\theta \log p(\boldsymbol{y}|X)$ is available.

2.5 Bayesian optimisation

One of the goals of this project is to attempt to apply a Gittins index based algorithm to the problem of Bayesian optimisation (BO). Here we will briefly define BO and the general framework of its methods.

Bayesian optimisation is the problem of globally optimising a continuous black-box objective function which is expensive to evaluate (Kushner, 1964), (Jones at al., 1998), (Frazier, 2018). By this we mean that we do not have access to information about the derivatives of the function nor does the function have easily exploitable properties such as linearity or convexity. The expensiveness of evaluating the function places a premium on the number of function evaluations involved in the optimisation algorithm, perhaps in terms of time or cloud computing fees. To achieve this aim we must intelligently select where to sample the objective function based on the observations we have made so far.

A central application of BO is the tuning of hyperparameters in machine learning models, especially neural networks which can involve very compute intensive training (Snoek at al., 2012). The objective function in this case could be the test loss of a trained network as a function of the hyperparemeters e.g. learning rate or layer width. This function then includes the entire process of training a neural network- a highly expensive process which does not have readily available derivatives.

BO is also useful when evaluating the objective function involves expensive real-world actions such as performing an experiment. For this reason it has seen use in drug discovery (Negoescu et al., 2011), chemical design (Griffiths & Hernández-Lobato, 2020), materials design (Frazier & Wang, 2016) and designing engineering systems (Forrester et al., 2008).

Typically BO involves two components, a statistical model of the objective function f and an acquisition function which informs the decision about where to sample the function next. The model of the objective function is usually a GP and this is what we will be using. GPs are useful because they can capture the continuous behaviour we assume about the objective function and can incorporate observation noise which is relevant to this particular project. There are a number of choices for acquisition functions which take the posterior distribution of the objective function model $f(x)|f(x_{1:t})$, computed in the way described in the earlier section, and convert it to a score which tells us where to sample the function.

A key question when performing BO is the choice of acquisition function. The application to BO of this project is the development of a new acquisition function based on principles from MABs. Some of the most popular choices are expected improvement (discussed in chapter 4) (Močkus, 1975), (Jones, 1998), knowledge gradients (Frazier et al., 2009), entropy search (Hennig & Schuler, 2012) and predictive entropy search (Hernández-Lobato, 2014). In chapter 4 we also discuss UCB and Thompson sampling which can be seen as generalisations of the equivalent techniques from MABs. What unifies the different approaches is that they give preference to sampling regions of input space where the distribution is highly uncertain. This can be seen as an example of balancing exploration and exploitation- it is beneficial to explore these uncertain areas because there is a reasonable chance that they will produce information which will be useful for the long term goal.

Chapter 3

Approximate Computation of the Gittins Index

Methods for computing the Gittins index typically revolve around *calibration* (Edwards, 2019), (Gittins, 2011), (Chakravorty, 2014) which broadly speaking means solving equation (2.14) for the value of a standard bandit for which we are indifferent between it and another bandit. This is done by binary search; repeatedly evaluating the value function on a shrinking range of values of λ to locate its root. These methods can find the GI up to a fixed accuracy, the level of which effects the optimality of the GI algorithm when applied to MABs. Increasing the accuracy of binary search can rapidly increase the computational cost.

An approximately accurate GI leads to an approximately optimal MAB policy (Katehakisa & Veinott Jr, 1987) and Glazebroke (1982) provides a bound on the loss of reward due to an imprecise GI. This means that a level of error is acceptable, which justifies the approximations we will make in computing the GI, but we would like a method which retains the ability to control the accuracy of the approximation (as a trade-off with computational expense).

We present here an alternative method which recursively approximates the value function using Gaussian processes. This particular choice of approximation leads to closed form calculations of subsequent value functions. This means that accuracy can be improved, rather than by increasing the depth of a binary search, but by increasing the density of points used to fit each GP, thus improving the accuracy of the GP approximation. We will first demonstrate the way that this approach arrives at the GI, then formulate the specific calculations for two particular choices of GP covariance function, squared-exponential and cubic spline.

3.1 GI from recursive formulation of the value function

Following an optimal policy, the value function for a bandit in the scenario where we imagine a second bandit with fixed returns λ is defined recursively as

$$\tilde{V}(\Sigma, n, \gamma, \lambda) = \max\left\{\frac{\Sigma}{n} + \gamma \mathbb{E}_Y[\tilde{V}(\Sigma + Y, n+1, \gamma, \lambda)], \frac{\lambda}{1-\gamma}\right\}. \tag{3.1}$$

We have the choice between using the variable bandit which returns an expected reward of $\frac{\Sigma}{n}$ or using the fixed bandit. If we choose the former then the expected future rewards are contained within the same value function but with arguments reflecting the new information about the bandit, and after discounting the function by γ. If the fixed arm is the optimal choice then no information is gained so the optimal choice will remain the same forever, resulting in a reward of $\frac{\lambda}{1-\gamma}$. Taking the maximum of the rewards from these two options yields the optimal reward.

A more convenient formulation of the value function is $V = \tilde{V} - \frac{\lambda}{1-\gamma}$, the excess reward of the variable arm compared to the fixed arm under the optimal policy. This can be expressed as

$$V(\Sigma, n, \gamma, \lambda) = \max\left\{\frac{\Sigma}{n} - \lambda + \gamma \mathbb{E}_Y[V(\Sigma + Y, n+1, \gamma, \lambda)], 0\right\} \tag{3.2}$$

and then the Gittins index is

$$\nu(\Sigma, n, \gamma) = \min\{\lambda : V(\Sigma, n, \gamma, \lambda) = 0\}. \tag{3.3}$$

The function V is awkward in that is is a non-constant function over part of its domain and a constant over the rest. We can transform it further to get a single smooth function which will be even more convenient to work with. In effect we will reverse the order of the max operation and the 'step' operation (moving from n to $n-1$) of the dynamic programming to get a function U such that $V = \max\{U, 0\}$. That is to say,

$$U(\Sigma, n) = \frac{\Sigma}{n} - \lambda + \gamma \mathbb{E}_Y\left[\max\left\{U(\Sigma + Y, n+1), 0\right\}\right] \tag{3.4}$$

where γ and λ are treated as constants. From here on, this function U will be what is referred to as the value function. Figure 3.1 shows the function $U(\Sigma, 1)$ for a particular horizon, the zero-intercept on the Σ axis β is marked as this will be an important feature of our calculations.

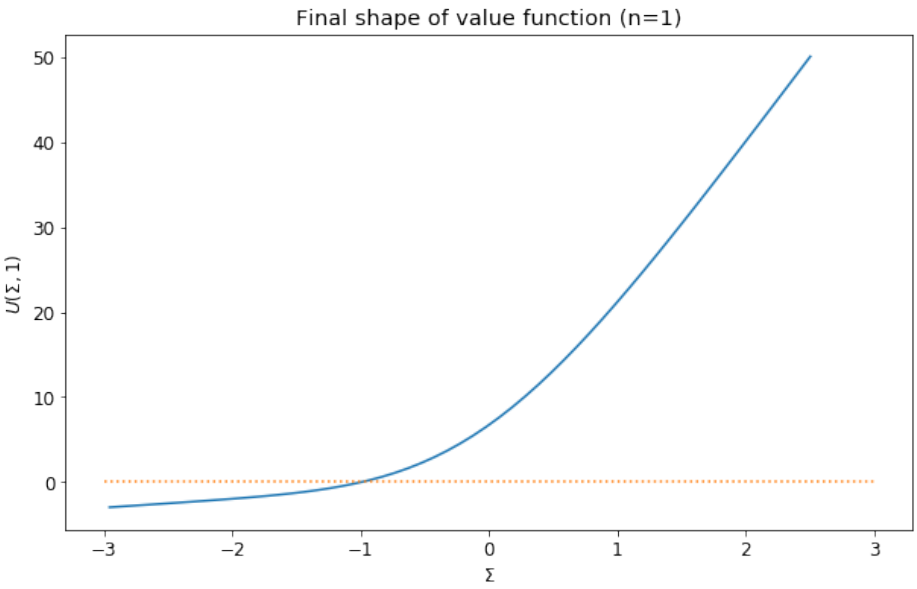


Fig. 3.1 Value function $U(\Sigma,1)$ for $h=20$, $N=40$ and $\gamma=0.95$. The zero-intercept can be observed as $\beta\approx-1$.

The expectation in (3.4) is taken over the posterior distribution of Y,

$$Y|\Sigma,n\sim\mathscr{N}\left(\frac{\Sigma}{n},\frac{n+1}{n}\right). \tag{3.5}$$

While the problem is formulated as having an infinite horizon, in practice, the dynamic programming problem needs a starting point for it to be well defined. In the limit of large n, information gained on successive uses of a bandit has a vanishing effect on the future strategy and rewards. The uncertainty is so small that the bandit can be treated as if it produces a constant reward of Σ/n. It has excess value $\max\left\{\frac{\Sigma/n-\lambda}{(1-\gamma)},0\right\}$ so we set our DP starting point to

$$U(\Sigma,N)=\frac{\Sigma/N-\lambda}{(1-\gamma)} \tag{3.6}$$

for a large number N. The effect of this approximation on steps early in the DP computation might be large but it will begin to vanish for later steps (meaning earlier time-steps since the DP calculation works backwards from N). This means we should select N so that it is much larger than the effective horizon so that the approximation is good for steps before the effective horizon.

Notice that $U(\Sigma,n)$ is non-decreasing in Σ and is unbounded. This means that it has a unique root β. In fact, if β is defined in Σ-space then it is equal to $-n$ times the exploration bonus component of the Gittins index. i.e.

$$\nu(\Sigma,n,\gamma) = \frac{\Sigma}{n} - \frac{\beta}{n}. \tag{3.7}$$

Noting the distribution $\Sigma + y|\Sigma, n-1 \sim \mathcal{N}\left(\frac{n\Sigma}{n-1}, \frac{n}{n-1}\right)$, we can now reformulate the recursion as

$$\begin{aligned} U(\Sigma,n-1) &= \frac{\Sigma}{n-1} - \lambda + \gamma \int_{-\infty}^{+\infty} \max\{U(z,n),0\} \mathcal{N}\left(\frac{n\Sigma}{n-1}, \frac{n}{n-1}\right) dz \\ &= \frac{\Sigma}{n-1} - \lambda + \gamma \int_{\beta}^{\infty} U(z,n) \mathcal{N}\left(\frac{n\Sigma}{n-1}, \frac{n}{n-1}\right) dz. \end{aligned} \tag{3.8}$$

Once the value function is in this formulation we can simplify it for our purposes by setting $\lambda = 0$. Performing this integral is where the problem opens itself to innovation and a variety of ideas present themselves. At each step we will only have access to $U(\Sigma,n)$ from the previous step at a set of sample points, rather than the full continuous function so some approximation must be made which faithfully recreates the true value function but retains the tractability of the integral.

This approximation can be viewed as an interpolation or regression problem. A powerful tool for these tasks is the Gaussian process; it can transform the grid of points into a smooth function while being easy to compute and the hyper-parameters can be optimised by maximising marginal likelihood. In other words $U(\Sigma,n)$ will be replaced by the posterior mean of a GP conditioned on the values computed in the previous step.

In order to use a GP we must select a covariance function. We will implement and compare GP approximations using squared exponential and cubic spline covariance functions which are described by Rasmussen & Williams (2006).

3.2 Squared exponential covariance

The squared exponential covariance function is

$$k(x,x') = v^2 \exp\left(\frac{-(x-x')^2}{2l^2}\right) \tag{3.9}$$

where v is a scale the magnitude of the covariance and l is the length scale. We define the matrix K_y such that $(K_y)_{ij} = k(x_i, x_j) + \delta_{i,j} v^2 \tau^2$ where τ^2 represents the signal-to-noise ratio. Adding this noise to the diagonal ensures positive-definiteness of the matrix and helps with the numerical stability of inverting it.

The data provided to the GP is $\{x_i, y_i\}_{i=1}^d$ so we can expect reasonable behaviour for x values within the range of the data. We can ensure good behaviour for larger x by using a prior mean function $m_{gp}(x)$. We will assume that if x, which represents Σ the total reward received from the bandit, is very large then there is a negligible posterior probability that the true reward is negative. Approximating this probability by 0 allows us to say that the optimal policy is to activate this bandit forever (instead of the fixed bandit which for our purposes gives reward $\lambda = 0$). Hence we should find that the value function $U(\Sigma, n) \approx \frac{\Sigma}{n(1-\gamma)}$. Therefore we set the prior mean

$$\boldsymbol{m}_{gp}(\boldsymbol{x}) = \frac{\boldsymbol{x}}{n(1-\gamma)}. \tag{3.10}$$

We can now find the posterior mean of the GP evaluated at a test point x_*

$$\begin{aligned} U(x_*, n) &= m_{gp}(x_*) + \boldsymbol{k}(x_*, \boldsymbol{x}) K_y^{-1} (\boldsymbol{y} - \boldsymbol{m}_{gp}(\boldsymbol{x})) \\ &= m_{gp}(x_*) + \sum_{i=1}^{d} \alpha_i v^2 \exp\left(-\frac{(x_* - x_i)^2}{2l^2} \right) \end{aligned} \tag{3.11}$$

where $\alpha_i = \sum_{j=1}^d (K_y^{-1})_{ij} (y_j - m_{gp}(x_j))$.

Inserting this expression into (3.8) we can reformulate $U(\Sigma, n-1)$ in terms of the probability density function $\mathcal{N}$ and cumulative density function Φ of the standard Gaussian distribution.

Let $m = \frac{n\Sigma}{n-1}$, $s^2 = \frac{n}{n-1}$. Then,

$$\begin{aligned}
U(\Sigma, n-1) &= \frac{\Sigma}{n-1} + \gamma \int_\beta^\infty U(z,n)\mathcal{N}(z|m,s^2)dz \\
&= \frac{\Sigma}{n-1} + \gamma \int_\beta^\infty m_{gp}(z)\mathcal{N}(z|m,s^2)dz + \gamma \sum_{i=1}^{d} \int_\beta^\infty \alpha_i v^2 \exp\left(-\frac{(z-x_i)^2}{2l^2}\right)\mathcal{N}(z|m,s^2)dz \\
&= \frac{\Sigma}{n-1} + \frac{\gamma}{n(1-\gamma)} \int_\beta^\infty z\mathcal{N}(z|m,s^2)dz + \gamma v^2 l\sqrt{2\pi} \sum_{i=1}^{d} \alpha_i \int_\beta^\infty \mathcal{N}(z|x_i,l^2)\mathcal{N}(z|m,s^2)dz \\
&= \frac{\Sigma}{n-1} + \frac{\gamma}{n(1-\gamma)} \left(m\Phi\left(\frac{m-\beta}{s}\right) + s\mathcal{N}\left(\frac{m-\beta}{s}\right)\right) + \gamma v^2 l \sum_{i=1}^{d} \alpha_i Z_i \Phi\left(\frac{\widetilde{m}_i - \beta}{\tilde{s}}\right)
\end{aligned} \tag{3.12}$$

where $Z_i = \frac{1}{\sqrt{l^2+s^2}} \exp\left(-\frac{(m-x_i)^2}{2(l^2+s^2)}\right)$, $\tilde{s} = \left(\frac{1}{l^2} + \frac{1}{s^2}\right)^{-\frac{1}{2}}$ and $\widetilde{m}_i = \tilde{s}^2\left(\frac{x_i}{l^2} + \Sigma\right)$.

Here we have used the following useful identity which will be proved as part of Proposition 1

$$\int_\beta^\infty z\mathcal{N}(z|\mu,\sigma^2)dz = \mu\Phi\left(\frac{\mu-\beta}{\sigma}\right) + \sigma\mathcal{N}\left(\frac{\mu-\beta}{\sigma}\right). \tag{3.13}$$

3.2.1 Hyperparameter optimisation

At each step of the DP we fit a new GP to a set of points. While the behaviour of the GPs should not vary much from step to step, it is still important to ensure that a good distribution has been learned which reasonably balances parsimony with goodness of fit. The standard method to do this is maximising the marginal likelihood $p(\boldsymbol{y}|\boldsymbol{x},\theta)$ over the hyperparameters θ. For a squared-exponential covariance function, the hyperparameters are $\theta = (v,l)$ for the covariance scale and length scale.

The marginal likelihood of any GP is

$$p(\boldsymbol{y}|\boldsymbol{x},\theta) = -\frac{1}{2}\boldsymbol{y}^T(K+\sigma^2 I)^{-1}\boldsymbol{y} - \frac{1}{2}\log|K+\sigma^2 I| - \frac{d}{2}\log(2\pi). \tag{3.14}$$

We have reparameterised the noise parameter σ^2 to be $v^2\tau^2$ in our formulation so that it scales with the covariances in K. The log of the determinant in this expression can be computed conveniently as the sum of the diagonal of the Cholesky decomposition of the matrix $K_y = K + v^2\tau^2 I$.

In order to maximise the marginal likelihood we can use gradient ascent or any variation thereof. To do this requires the partial derivatives of the marginal likelihood with respect to the two hyperparameters. These are

$$\frac{\partial}{\partial v}p(\boldsymbol{y}|\boldsymbol{x},\theta) = (\boldsymbol{y}-\boldsymbol{m}_{gp}(\boldsymbol{x}))^T(K_y)^{-1}(\boldsymbol{y}-\boldsymbol{m}_{gp}(\boldsymbol{x})) - d \tag{3.15}$$

$$\frac{\partial}{\partial l}p(\boldsymbol{y}|\boldsymbol{x},\theta) = \frac{1}{l^2}x^T\left(W\boldsymbol{x} - \left(x_j\sum_{i=1}^{d}W_{ji}\right)_{j=1}^{d}\right) \tag{3.16}$$

where $W = K\circ((K_y)^{-1}-\boldsymbol{\alpha}\boldsymbol{\alpha}^T)$ with $\circ$ denoting element-wise multiplication and $\boldsymbol{\alpha} = (\alpha_1,\ldots,\alpha_d)$ as defined above.[1]

Being a non-convex optimisation problem, finding the global maximum of the marginal likelihood is not trivial. However, as long as a fine enough resolution of data points is calculated at each step of the DP, functions drawn from the GP do not have much room to oscillate in between the data points. This means that a posterior distribution which fits the data well will be forced to almost entirely contain sensible functions. Therefore, perfectly optimising the hyperparameters should not be necessary, and points in the vicinity of local maxima are likely to be adequate.

3.3 Cubic spline covariance

An alternative covariance function which may be more suitable for computation of the Gittins index is the cubic spline covariance function:

$$k(x,x') = \frac{|x-x'|}{2}\min(x,x')^2 + \frac{1}{3}\min(x,x')^3 \tag{3.17}$$

where σ^2 is a scale hyperparameter. Given the data points $\{(x_i,y_i)\}_{i=1}^d$, let $(K_y)_{ij} = \sigma^2(k(x_i,x_j)+r^2\delta_{ij})$ where r^2 is a parameter which controls the signal-to-noise ratio.[2] We define the basis $\boldsymbol{h}(x) = [1,x]^T$ and $H = [\boldsymbol{h}(x_i),\ldots,\boldsymbol{h}(x_N)]$. Finally define $\boldsymbol{B} = [B_1,B_2]^T = (HK_y^{-1}H^T)^{-1}HK_y^{-1}\boldsymbol{y}$. The cubic spline GP has linear asymptotes so there is no need to

[1] The derivative with respect to l involves some slightly confusing notation which represents the vector of row sums of W element-wise multiplied by $\boldsymbol{x}$. This is difficult to notate clearly so this footnote is included to hopefully add some clarity to the expression.

[2] The cubic spline GP is non-stationary because the variance $k(x,x)$ grows with x^3. This means it is not possible to define a single signal-to-noise ratio when using a a stationary noise process. Hence for a given value of r^2 the signal-to-noise ratio will be different at every point in the process so r^2 should not be interpreted as literally being the signal-to-noise ratio.

enforce a linear prior as we did with the squared-exponential covariance. The posterior GP mean is

$$\begin{aligned} U(x_*,n) &= \boldsymbol{h}(x_*)^T\boldsymbol{B} + \sigma^2\boldsymbol{k}(x_*,\boldsymbol{x})^T K_y^{-1}(\boldsymbol{y}-H^T\boldsymbol{B}) \\ &= B_1 + x_* B_2 + \sum_{i-1}^{N} \alpha_i \left(\frac{|x_*-x_i|}{2}\min(x_*,x_i)^2 + \frac{1}{3}\min(x_*,x_i)^3 \right) \end{aligned} \tag{3.18}$$

where $\alpha_i = \sum_{j=1}^{d}(K_y^{-1})_{ij}(\boldsymbol{y}-H^T\boldsymbol{B})_j$. A derivation of this can be found in Rasmussen and Williams (2006). Note that this expression is independent of the covariance scale σ^2 so it can be arbitrarily set to 1 and there is no need to optimise it.

This mean function is piecewise cubic in x so to solve the integral in (3.8) with this GP we require a few facts about integrals of Gaussian distributions in z multiplied by z (which leads to (3.13)), z^2 and z^3. These will be proved by repeated application of integration by parts.

Proposition 1. *For $a,b,\mu \in \mathbb{R}, \sigma > 0$ the following all hold*

$$i) \int_a^b z\mathcal{N}(z|\mu,\sigma^2) = \sigma\left(\mathcal{N}\left(\frac{a-\mu}{\sigma}\right) - \mathcal{N}\left(\frac{b-\mu}{\sigma}\right)\right) + \mu\left(\Phi\left(\frac{b-\mu}{\sigma}\right) - \Phi\left(\frac{a-\mu}{\sigma}\right)\right)$$

$$\begin{aligned} ii) \int_a^b z^2\mathcal{N}(z|\mu,\sigma^2) =& \sigma(a+\mu)\mathcal{N}\left(\frac{a-\mu}{\sigma}\right) - \sigma(b+\mu)\mathcal{N}\left(\frac{b-\mu}{\sigma}\right) \\ &+ \left(\sigma^2+\mu^2\right)\left(\Phi\left(\frac{b-\mu}{\sigma}\right) - \Phi\left(\frac{a-\mu}{\sigma}\right)\right) \end{aligned}$$

$$\begin{aligned} iii) \int_a^b z^3\mathcal{N}(z|\mu,\sigma^2) =& \sigma(a^2+\mu^2+\mu a+2\sigma^2)\mathcal{N}\left(\frac{a-\mu}{\sigma}\right) \\ &- \sigma(b^2+\mu^2+\mu b+2\sigma^2)\mathcal{N}\left(\frac{b-\mu}{\sigma}\right) \\ &+ \mu(3\sigma^2+\mu^2)\left(\Phi\left(\frac{b-\mu}{\sigma}\right) - \Phi\left(\frac{a-\mu}{\sigma}\right)\right) \end{aligned}$$

Proof. We first let $\tilde{a} = \frac{a-\mu}{\sigma}$ and $\tilde{b} = \frac{b-\mu}{\sigma}$ and note the following 4 facts derived by integration by parts:

$$\int_{\tilde{a}}^{\tilde{b}} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) dx = \Phi(\tilde{b}) - \Phi(\tilde{a}) \tag{3.19}$$

$$\int_{\tilde{a}}^{\tilde{b}} \frac{1}{\sqrt{2\pi}} x \exp\left(-\frac{x^2}{2}\right) dx = -\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) \Big|_{\tilde{a}}^{\tilde{b}} = \mathcal{N}(\tilde{a}) - \mathcal{N}(\tilde{b}) \tag{3.20}$$

$$\begin{aligned} \int_{\tilde{a}}^{\tilde{b}} \frac{1}{\sqrt{2\pi}} x^2 \exp\left(-\frac{x^2}{2}\right) dx &= -\frac{1}{\sqrt{2\pi}} x \exp\left(-\frac{x^2}{2}\right) \Big|_{\tilde{a}}^{\tilde{b}} + \int_{\tilde{a}}^{\tilde{b}} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) dx \\ &= \tilde{a}\mathcal{N}(\tilde{a}) - \tilde{b}\mathcal{N}(\tilde{b}) + \Phi(\tilde{b}) - \Phi(\tilde{a}) \end{aligned} \tag{3.21}$$

$$\begin{aligned} \int_{\tilde{a}}^{\tilde{b}} \frac{1}{\sqrt{2\pi}} x^3 \exp\left(-\frac{x^2}{2}\right) dx &= -\frac{1}{\sqrt{2\pi}} x^2 \exp\left(-\frac{x^2}{2}\right) \Big|_{\tilde{a}}^{\tilde{b}} + 2\int_{\tilde{a}}^{\tilde{b}} \frac{1}{\sqrt{2\pi}} x \exp\left(-\frac{x^2}{2}\right) dx \\ &= \tilde{a}^2\mathcal{N}(\tilde{a}) - \tilde{b}^2\mathcal{N}(\tilde{b}) + 2\left(\mathcal{N}(\tilde{a}) - \mathcal{N}(\tilde{b})\right) \end{aligned} \tag{3.22}$$

Deriving (3.21) uses (3.19) and deriving (3.22) uses (3.20).

We can now consider the integrals stated in the proposition and use a change of variables to find integrals of the forms of (3.19) - (3.22) making their solution a simple matter of substituting in already derived results.

$$\begin{aligned} \int_a^b z\mathcal{N}(z|\mu,\sigma^2)dz &= \int_a^b \frac{1}{\sqrt{2\pi}\sigma} z \exp\left(\frac{-1}{2\sigma^2}(z-\mu)^2\right) dz \\ &= \int_{\tilde{a}}^{\tilde{b}} \frac{1}{\sqrt{2\pi}} (\sigma x + \mu) \exp\left(-\frac{x^2}{2}\right) dx \\ &= \sigma \int_{\tilde{a}}^{\tilde{b}} \frac{1}{\sqrt{2\pi}} x \exp\left(-\frac{x^2}{2}\right) dx + \mu \int_{\tilde{a}}^{\tilde{b}} \exp\left(-\frac{x^2}{2}\right) dx \end{aligned}$$

and result (i) follows by application of (3.19) and (3.20).

We also have

$$\begin{aligned}\int_a^b z^2 \mathcal{N}(z|\mu,\sigma^2)dz &= \int_a^b \frac{1}{\sqrt{2\pi}\sigma} z^2 \exp\left(\frac{-1}{2\sigma^2}(z-\mu)^2\right) dz \\ &= \int_{\tilde{a}}^{\tilde{b}} \frac{1}{\sqrt{2\pi}} (\sigma x+\mu)^2 \exp\left(-\frac{x^2}{2}\right) dx \\ &= \sigma^2 \int_{\tilde{a}}^{\tilde{b}} \frac{1}{\sqrt{2\pi}} x^2 \exp\left(-\frac{x^2}{2}\right) dx + 2\sigma\mu \int_{\tilde{a}}^{\tilde{b}} \frac{1}{\sqrt{2\pi}} x \exp\left(-\frac{x^2}{2}\right) dx \\ &\quad + \mu^2 \int_{\tilde{a}}^{\tilde{b}} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) dx\end{aligned}$$

and result (ii) follows by application of (3.19), (3.20) and (3.21).

Finally, we have

$$\begin{aligned}\int_a^b z^3 \mathcal{N}(z|\mu,\sigma^2)dz &= \int_a^b \frac{1}{\sqrt{2\pi}\sigma} z^3 \exp\left(\frac{-1}{2\sigma^2}(z-\mu)^2\right) dz \\ &= \int_{\tilde{a}}^{\tilde{b}} \frac{1}{\sqrt{2\pi}} (\sigma x+\mu)^3 \exp\left(-\frac{x^2}{2}\right) dx \\ &= \sigma^3 \int_{\tilde{a}}^{\tilde{b}} \frac{1}{\sqrt{2\pi}} x^3 \exp\left(-\frac{x^2}{2}\right) dx + 3\sigma^2\mu \int_{\tilde{a}}^{\tilde{b}} \frac{1}{\sqrt{2\pi}} x^2 \exp\left(-\frac{x^2}{2}\right) dx \\ &\quad + 3\sigma\mu^2 \int_{\tilde{a}}^{\tilde{b}} \frac{1}{\sqrt{2\pi}} x \exp\left(-\frac{x^2}{2}\right) dx + \mu^3 \int_{\tilde{a}}^{\tilde{b}} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) dx\end{aligned}$$

and result (iii) follows by application of (3.19), (3.20), (3.21) and (3.22). □

The integral given in (3.13) follows by letting $b \to \infty$ in part (i) of the proposition. The other two parts of the proposition can also give us integrals of this form but they are not useful for our calculations here.

We can now derive an expression for the integral in (3.8) for the cubic-spline GP. Recall that we are setting $\lambda = 0$ so the DP equation can be stated as

$$U(\Sigma, n-1) = \frac{\Sigma}{n-1} + \gamma \int_\beta^\infty U(z,n) \mathcal{N}\left(\frac{n\Sigma}{n-1}, \frac{n}{n-1}\right) dz. \tag{3.23}$$

First, let $m = \frac{n\Sigma}{n-1}$ and $s^2 = \frac{n}{n-1}$ and replace $U(z,n)$ in the integral with the GP posterior mean (3.18). This gives

$$\begin{aligned} U(\Sigma, n-1) =& \frac{\Sigma}{n-1} + \gamma \int_\beta^\infty \boldsymbol{h}(z)^T \boldsymbol{B} \mathcal{N}(z|m,s^2) dz \\ &+ \gamma \sum_{i=1}^{d} \alpha_i \int_\beta^\infty \left(\frac{|z-x_i|}{2} \min(z,x_i)^2 + \frac{\min(z,x_i)^3}{3} \right) \mathcal{N}(z|m,s^2) dz \end{aligned} \tag{3.24}$$

Note that $\boldsymbol{h}(z)^T \boldsymbol{B} = B_1 + zB_2$ so the fist integral in (3.24) can be solved as

$$\int_\beta^\infty \boldsymbol{h}(z)^T \boldsymbol{B} \mathcal{N}(z|m,s^2) dz = B_1 \Phi\left(\frac{m-\beta}{s}\right) + B_2 \left(m\Phi\left(\frac{m-\beta}{s}\right) + s\mathcal{N}\left(\frac{m-\beta}{s}\right)\right) \tag{3.25}$$

once again making use of (3.13).

To deal with the sum of d integrals in (3.24) we need to separate the integrals depending on whether $x_i \leq \beta$ or $x_i > \beta$ as each collection of integrals will have different behaviour.

If $x_i \leq \beta$ then the integral is over a range such that $z \geq x_i$ always so the integral is simply

$$\int_\beta^\infty \left(\frac{|z-x_i|}{2} \min(z,x_i)^2 + \frac{\min(z,x_i)^3}{3} \right) \mathcal{N}(z|m,s^2) dz = \int_\beta^\infty \left(\frac{x_i^2}{2} z - \frac{x_i^3}{6} \right) \mathcal{N}(z|m,s^2) dz. \tag{3.26}$$

This can once again be solved using (3.13).

If $x_i > \beta$ then the behaviour of the integral switches when $z = x_i$ so it must be split into two integrals

$$\begin{aligned} &\int_\beta^\infty \left(\frac{|z-x_i|}{2} \min(z,x_i)^2 + \frac{\min(z,x_i)^3}{3} \right) \mathcal{N}(z|m,s^2) dz = \\ &\int_\beta^{x_i} \left(\frac{x_i}{2} z^2 - \frac{1}{6} z^3 \right) \mathcal{N}(z|m,s^2) dz + \int_{x_i}^\infty \left(\frac{x_i^2}{2} z - \frac{x_i^3}{6} \right) \mathcal{N}(z|m,s^2) dz. \end{aligned} \tag{3.27}$$

we now see the importance of Proposition 1 as we have integrals involving z^2 and z^3 multiplied by a Gaussian density.

Careful application of Proposition 1 to (3.26) and (3.27), noting the various limits of the different integrals and adding the term given in (3.25) finally leads us to a cumbersome but

useful expression for $U(\Sigma, n-1)$. It is given entirely in terms of n, Σ and the previously computed values B_1, B_2, β, x_i and α_i for $i = 1, \ldots, d$ using the Gaussian probability density function $\mathscr{N}$ and cumulative density function Φ.

$$
\begin{aligned}
&U(\Sigma, n-1) = \\
&\quad \frac{\Sigma}{n-1} + \gamma B_1 \Phi\left(\frac{m-\beta}{s}\right) + \gamma B_2 \left(m\Phi\left(\frac{m-\beta}{s}\right) + s\mathscr{N}\left(\frac{m-\beta}{s}\right)\right) \\
&\quad + \gamma \sum_{i|x_i \le \beta} \alpha_i \left\{ \frac{x_i^2}{2}\left[m\Phi\left(\frac{m-\beta}{s}\right) + s\mathscr{N}\left(\frac{m-\beta}{s}\right)\right] - \frac{x_i^3}{6}\Phi\left(\frac{m-\beta}{s}\right)\right\} \\
&\quad + \gamma \sum_{i|x_i > \beta} \alpha_i \left\{ \frac{x_i}{2}\left[(s^2+m^2)\left(\Phi\left(\frac{m-\beta}{s}\right) - \Phi\left(\frac{m-x_i}{s}\right)\right)\right.\right. \\
&\qquad\qquad + s(\beta+m)\mathscr{N}\left(\frac{m-\beta}{s}\right) - s(x_i+m)\mathscr{N}\left(\frac{m-x_i}{s}\right)\Big] \\
&\qquad\qquad - \frac{1}{6}\left[s(\beta^2+m^2+m\beta+2s^2)\mathscr{N}\left(\frac{m-\beta}{s}\right) - s(x_i^2+m^2+mx_i+2s^2)\mathscr{N}\left(\frac{m-x_i}{s}\right)\right. \\
&\qquad\qquad \left. + m(3s^2+m^2)\left(\Phi\left(\frac{m-\beta}{s}\right) - \Phi\left(\frac{m-x_i}{s}\right)\right)\right] \\
&\qquad\qquad \left. + \frac{x_i^2}{2}\left[m\Phi\left(\frac{m-x_i}{s}\right) + s\mathscr{N}\left(\frac{m-x_i}{s}\right)\right] - \frac{x_i^3}{6}\Phi\left(\frac{m-x_i}{s}\right)\right\}. \qquad (3.28)
\end{aligned}
$$

A subtlety that has so far been ignored in these calculations is the fact that ordinarily the cubic spline covariance function is only defined for $x > 0$ but here we have allowed x to exist on the entire real line. In fact, allowing $x \le 0$ leads to invalid covariance matrices K which may have negative values on the diagonal. Fortunately, this abuse we have committed against the covariance function does not cause problems for the posterior mean function which continues to sensibly follow the data despite being generated from an invalid covariance matrix. On the other hand, if we were to calculate the posterior covariance of the GP then problems would arise and it would not be possible to make valid inferences. However, we never calculate or use the posterior covariance so we escape unscathed. Possible resolutions would be to shift the x values upwards before fitting the GP or to calculate two separate GPs, one for $x > 0$ and one for $-x$ when $x < 0$. While going to these efforts may make the method feel more theoretically sound, we do not see this as a necessary step since our method works without it.

The above discussion may make the cubic spline covariance seem to be a more complicated choice than squared-exponential, involving unpleasant expressions like (3.28). However, this choice brings a meaningful advantage in that the covariance depends on only one hyperparameter σ^2 and the posterior mean is independent of it. This saves all the computational cost

of optimising hyperparameters which had to be done at every step of the algorithm with the squared-exponential covariance.

3.4 Algorithm details

The starting point N for dynamic programming should be much larger than the effective horizon h so we set $N = 2h$. The choice of N is a handle by which the precision of the algorithm can be adjusted at the expense of compute time. The appropriate discounting factor is set at $\gamma = 1 - 1/h$.

At each step of the program an appropriate range of values for Σ must be chosen for which we wish to compute $U(\Sigma, n-1)$. The lower bound Σ_0 needs to be low enough that the range contains the zero-intercept of the function. Setting this bound to $\Sigma_0 = -\sqrt{n}$ always achieves this. While choosing this bound works, it is very loose, especially for large n. We can improve it by empirically checking how low the zero intercepts can become. Figure 3.2 shows the linear relationship between the minimum zero-intercept and $\sqrt{N}$. The slope is approximately -0.62, so setting $\Sigma_0 = -0.62\sqrt{n}$ should always keep the zero-intercept within the range of Σ without allowing for an increasingly large amount of unnecessary calculation below the intercept.

The choice of upper bound for the range of Σs impacts the accuracy of the approximation. A single reward has a variance of 1 so for the sum of n rewards the variance grows with n and the standard deviation with $\sqrt{n}$. To capture almost all of the mass of a Gaussian distribution we should consider at least 3 standard deviations from the mean. Hence we will set the upper bound on Σ to $3\sqrt{n}$. This can be increased to make a small gain in accuracy or reduced to accelerate the algorithm.

The range of Σs also needs an appropriate resolution δ. It should be fine enough that the zero-intercept of the value function can be accurately approximated. However, the dimension of the covariance matrix scales linearly with the resolution so this significantly effects the time needed for matrix inversion. We used a resolution of $\delta = 0.1$ in our experiments.

$U(\Sigma, N)$ is set to the linear function (3.6). This means that for the first step of dynamic programming, no Gaussian process approximation is needed. The calculation is more simply

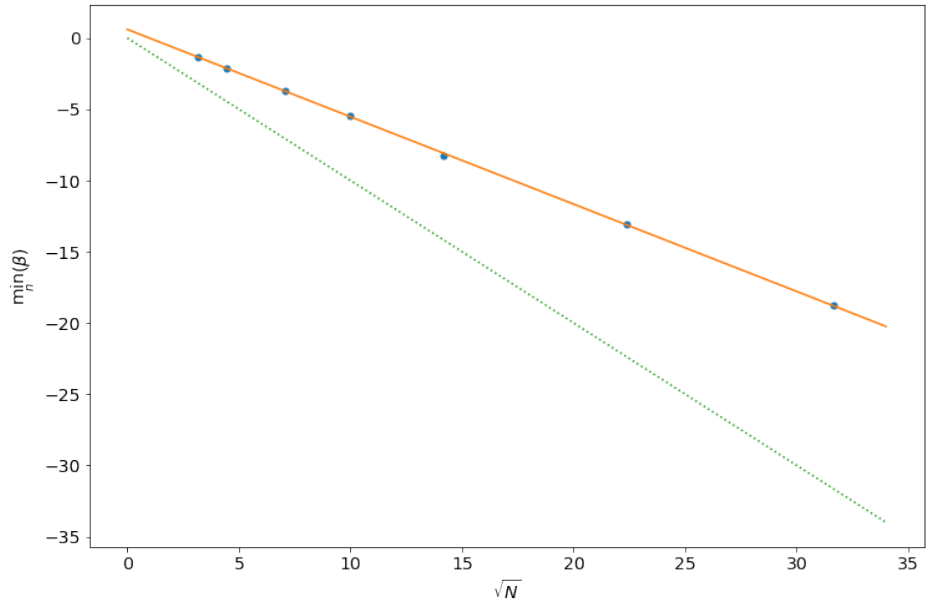


Fig. 3.2 Minimum value of β against $\sqrt{N}$. The line $y = -\sqrt{N}$ is marked.

expressed as

$$U(\Sigma, N-1) = \frac{\Sigma}{N-1} + \frac{\gamma}{1-\gamma}\left(\frac{\Sigma}{N-1}\Phi\left(\frac{\sqrt{N}\Sigma}{\sqrt{N-1}}\right) + \frac{1}{\sqrt{N(N-1)}}\mathcal{N}\left(\frac{\sqrt{N}\Sigma}{\sqrt{N-1}}\right)\right). \tag{3.29}$$

It can be assumed that this function has a single zero-intercept β so this can be approximated by linear interpolation between the two computed points between which the change of sign occurs.

To compute $U(\Sigma, n-1)$ for $n = N-1, \ldots, 1$ we set the test points $\Sigma = \{\Sigma_i\}$ where Σ_0 is the lower bound discussed above and $\Sigma_i = \Sigma_{i-1} + \delta$. There are enough of these points that it spans the space between the earlier discussed lower and upper bounds. We then compute the value function approximation using (3.28) or (3.12) and the value of β computed from $U(\Sigma, n)$. The new zero-intercept is then computed and the next step of dynamic programming can begin. Calculation of all of the points $U(\Sigma, n-1)$ for a given n can be computed in vectorised form using NumPy or Matlab, which improves efficiency significantly over the unvectorised form of the calculation.

The value functions themselves are not useful but the zero-intercepts on the Σ axis are important. At each step we can find the exploration bonus component of the Gittins index as $\nu(0, n, \gamma) = -\beta(n)/n$.

3.5 Results

Figure 3.3 shows how the value function $U(\Sigma, n)$ evolves as n changes going from $n = 20$ to the final shape at $n = 1$. Figure 3.1 shows the shape at $n = 1$. Note that the function becomes linear far away from the zero-intercept as little information can be learned about a bandit which is in this area so its expected reward becomes a constant.

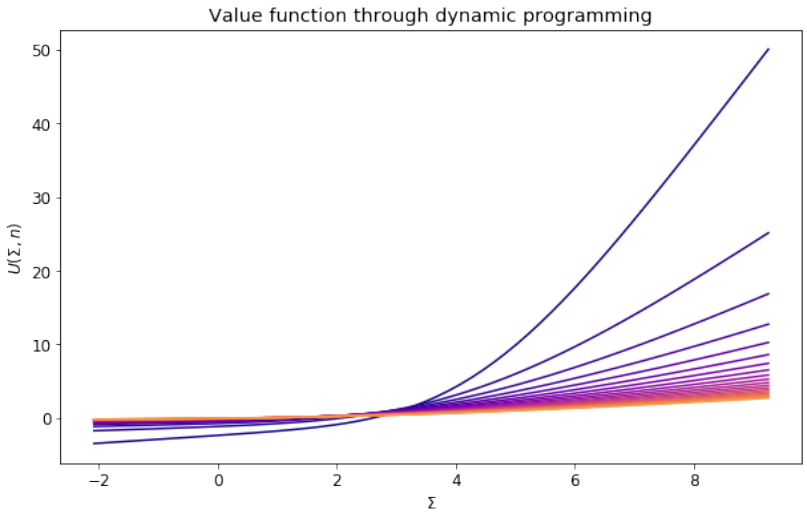


Fig. 3.3 $U(\Sigma, n)$ at each step of DP for $\gamma = 0.95$. $n = 20$ is in yellow and $n = 1$ is in blue.

Figure 3.4 shows the evolution of β as n changes for different horizons/discount factors. Recall that we always set $\gamma = 1 - 1/h$. The stength of the approximation improves for smaller n, as we move away from the start of the DP. For this reason we should only really count as valid the values of β for $n < h$. The more useful value that can be derived from this is the exploration bonus $\nu(0, n, \gamma) = -\beta(n)/n$. These are shown for various horizons/discount factors in Figure 3.5. We see the expected behaviour; the exploration bonus decreases as the number of observations of a bandit increases. In settings with a longer horizon (larger discount factor) there is more incentive for exploration so the bonus is greater.

As expected, using the cubic spline covariance runs much faster than squared-exponential due to the lack of a need to optimise hyperparameters. In order to use the calculated values in practice, they only need to be computed once for a given γ however, for long horizons the speed of the algorithm may become a relevant factor in how the algorithm should be executed. The speed can be augmented by reducing the starting point for dynamic programming N or the upper end of the range of values of Σ used in each step of computation. These choices involve of a trade off with precision of the computed Gittins indices.

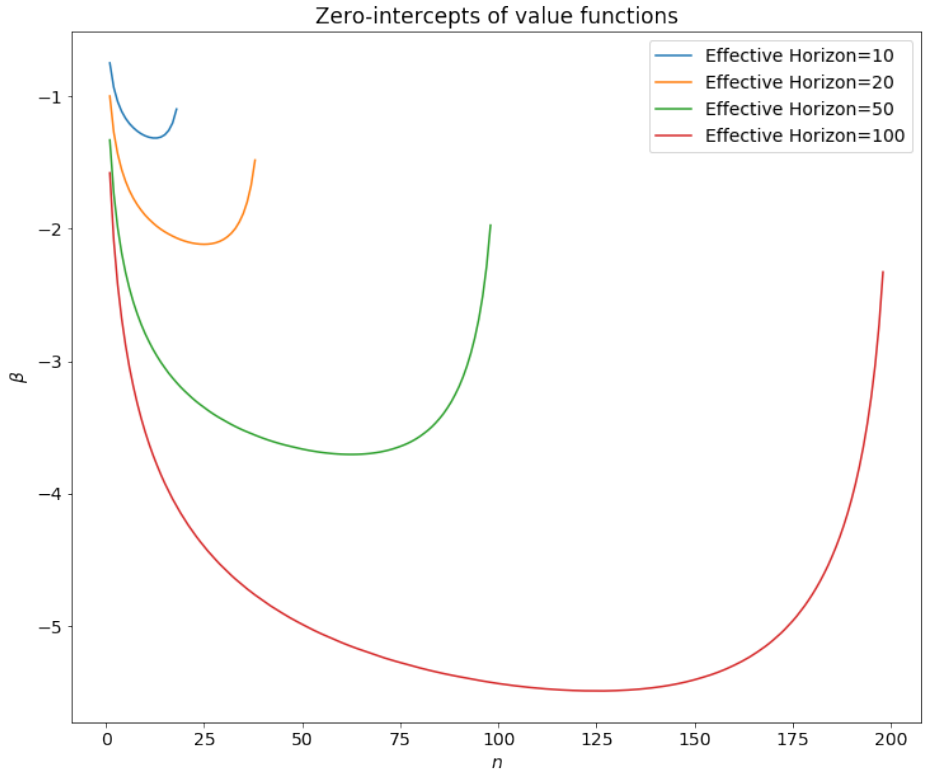


Fig. 3.4 Zero-intercepts β of value functions. These are accurate for $n < h = 1/(1-\gamma)$.

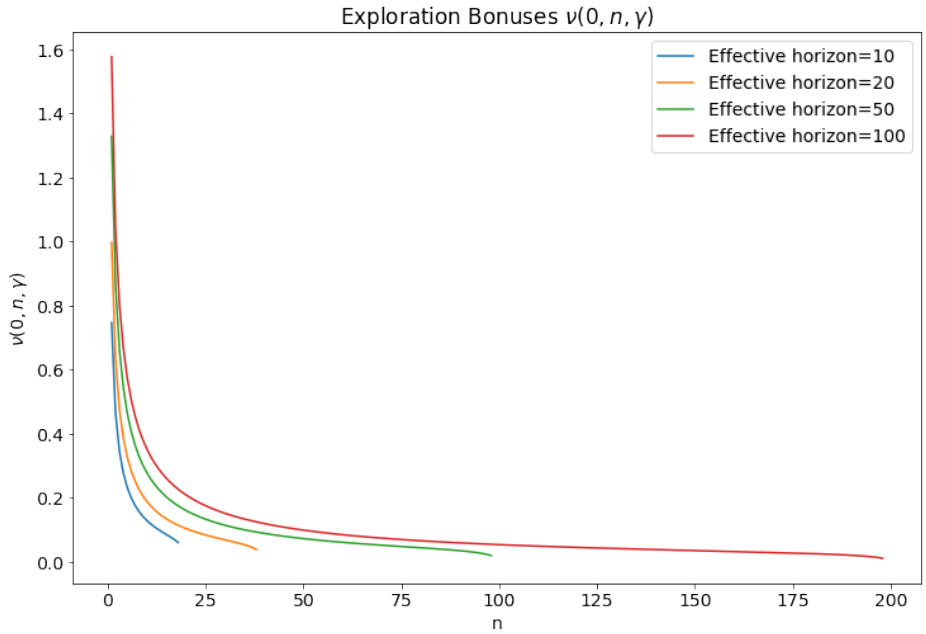


Fig. 3.5 Exploration bonus derived from value functions as a function of n. These are calculated up to double the effective horizon.

Chapter 4

Correlated Bandits and Bayesian Optimisation

Having computed the Gittins index, we now aim to extend its use beyond that which it is designed for. We only have theoretical guarantees of optimality for the algorithm in the case of uncorrelated bandits but it stands to reason that with appropriate modification it could form a useful basis for an algorithm with correlated bandits. As we are entering territory without theoretical guarantees, we will design an algorithm and evaluate its strength empirically rather than theoretically.

Once an approach is established for correlated discrete bandits, the theoretical step to move to Bayesian optimisation is relatively straight forward. In essence, the discrete bandit setting is a special case of the continuous bandit setting in which we only consider a discrete set of points in the learned reward function. While the two settings are theoretically similar, we will find in chapter 5 that the empirical results are quite different.

4.1 Correlated bandits

The key insight that motivates our 'correlated Gittins' algorithm is that the uncertainty associated with each bandit is purely a function of n, the number of times the bandit has been observed. This means that if we can produce an alternative measure of the uncertainty then we can convert it into $\tilde{n}$ which we might call the *effective n*.

The way we calculate the uncertainty is to fit a GP to the observed rewards and extract the marginal variances from the posterior GP at points $1,\ldots,k$ if there are k bandits. These

marginal variances σ_i^2 can be transformed into the effective n as $\tilde{n}_i = 1/\sigma_i^2$ for $i = 1\ldots,k$. This value is used to extract a Gittins value which has already been computed for the appropriate γ.

However, the Gittins index is only designed for $n \in \mathbb{N}$. We need to extend the definition of $\nu(\Sigma, \tilde{n}, \gamma)$ to positive real values $\tilde{n} > 0$. For the time being, it can suffice to use to linear interpolation of the pre-computed values, but we will use a more sophisticated approach when we graduate to Bayesian optimisation. In other words, we compute the interpolated Gittins index as

$$\tilde{\nu}(\tilde{n}\mu, \tilde{n}, \gamma) = \mu + \nu(0, \lfloor\tilde{n}\rfloor, \gamma) + (\tilde{n} - \lfloor\tilde{n}\rfloor)\,(\nu(0, \lfloor\tilde{n}\rfloor + 1, \gamma) - \nu(0, \lfloor\tilde{n}\rfloor, \gamma)) \tag{4.1}$$

where we replace the usual total reward Σ by $\tilde{n}\mu$ where μ is the posterior mean of the GP.

Once this index has been calculated, it is a simple task to select a bandit i to maximise $\tilde{\nu}(\tilde{n}_i\mu_i, \tilde{n}_i, \gamma)$. This bandit is then observed, the reward received and the GP is updated with this new information. The algorithm for k bandits with noisy reward function r is summarised in algorithm (1).

Algorithm 1: Correlated Gittins for multi-arm bandits

Require: k, maxIter, $\nu(0, n, \gamma)$ for $n = 1, \ldots, N$
$\boldsymbol{x} \leftarrow [\quad]$
$\boldsymbol{R} \leftarrow [\quad]$
firstAction $\leftarrow$ random$(1, \ldots, k)$
append($\boldsymbol{x}$, firstAction)
append($\boldsymbol{R}$, r(firstAction))
for step $= 1, \ldots,$ maxIter **do**
  $\boldsymbol{t} \leftarrow [1, \ldots, k]$
  $\boldsymbol{\mu} \leftarrow K(\boldsymbol{t}, \boldsymbol{x})(K(\boldsymbol{x}, \boldsymbol{x}) + I)^{-1} R$
  $\text{cov}(\boldsymbol{\mu}) \leftarrow K(\boldsymbol{t}, \boldsymbol{t}) - K(\boldsymbol{t}, \boldsymbol{x})(K(\boldsymbol{x}, \boldsymbol{x}) + I)^{-1} K(\boldsymbol{x}, \boldsymbol{t})$
  $\boldsymbol{\sigma^2} \leftarrow \text{diag}(\text{cov}(\boldsymbol{\mu}))$
  $\tilde{\boldsymbol{n}} \leftarrow 1/\boldsymbol{\sigma^2}$
  for $i = 1, \ldots, k$ **do**
    $\tilde{\nu}_i \leftarrow \mu_i + \nu(0, \lfloor\tilde{n}_i\rfloor, \gamma) + (\tilde{n}_i - \lfloor\tilde{n}_i\rfloor)\,(\nu(0, \lfloor\tilde{n}_i\rfloor + 1, \gamma) - \nu(0, \lfloor\tilde{n}_i\rfloor, \gamma))$
  end for
  action $\leftarrow \arg\max_i \tilde{\nu}_i$
  reward $\leftarrow r$(action)
  append($\boldsymbol{x}$, action)
  append($\boldsymbol{R}$, reward)
end for

Note that the first action is chosen entirely at random due to the lack of information and the need for some data $\boldsymbol{x}$ and $\boldsymbol{R}$ to make sense of the GP equations.

Before exploring further, we will describe some alternative algorithms which are commonly used in multi-arm bandit problems. These algorithms, as well as the regular Gittins index algorithm, will be used as comparisons for the performance of our own algorithms. We will also consider how some of the approaches have similarities to our own which will help us to gain an intuition as to what constitutes a 'fair' comparison.

4.1.1 ε-greedy

Considering the aim of exploiting known information while exercising a non-zero but limited amount of exploration, the most immediate and straight forward algorithm one might construct is ε-greedy. The idea is that most of the time the algorithm will strictly choose the bandit which has delivered the greatest average reward in the past, but will test bandits uniformly at random a proportion ε of the time. The degree of exploration can be easily and explicitly controlled by the single parameter ε; greater ε leads to more exploration. ε-greedy for k bandits with reward function r is detailed in algorithm (2)

Algorithm 2: ε-greedy

Require: ε, maxIter
$\boldsymbol{Q} \leftarrow [0 \dots 0] \in \mathbb{R}^k$
$\boldsymbol{n} \leftarrow [0 \dots 0] \in \mathbb{N}^k$
for $t = 1, \dots,$ maxIter **do**
    Sample $p \sim U(0,1)$
    if $p < \varepsilon$ **then**
        $a \leftarrow \text{random}(1, \dots, k)$
    else
        $a \leftarrow \arg\max_i Q_i$
    $R \leftarrow r(a)$
    $\boldsymbol{Q}[a] \leftarrow \boldsymbol{Q}[a] + (R - \boldsymbol{Q}[a])/(\boldsymbol{n}[a] + 1)$
    $\boldsymbol{n}[a] \leftarrow \boldsymbol{n}[a] + 1$
end for

Owing to its simplicity, this algorithm is severely flawed. If ε is too small, then the algorithm will under-explore, and in general will be trapped in local maxima for long periods of time. If the unknown means of the bandits happen to be set such that only a small number of them have values close to the maximum and most bandits have much smaller means then this

problem will be particularly bad. It may take many random observations before the algorithm discovers one of the bandits with a large mean and in all the time up to that it will be greedily exploiting low reward yielding bandits. In the discounted infinite horizon setting, delaying rewards like this causes a great reduction in the total discounted reward.

On the other hand, if ε is set too large, then the algorithm will always be limited in how well it can perform in the long run. Although it may identify the best bandits relatively rapidly, it will continue observing random, sub-optimal bandits a large proportion of the time. The fact that the algorithm is never able to converge is a severe handicap.

This problem of correctly tuning the hyperparameters of the algorithm is a major negative side to algorithms which have this property, as we will see in the upper confidence bound algorithm as well.

4.1.2 Upper confidence bound

The upper confidence bound (UCB) algorithm bears a resemblance to the Gittins algorithm. It can be viewed as constructing an index for each bandit which consists of the current estimate of the mean of the reward function plus an exploration bonus. To be precise, the choice of bandit follows the formula given in Sutton and Barto (2018),

$$a_t = \arg\max_a \left(Q(a) + c\sqrt{\frac{\log t}{n(a)}} \right) \tag{4.2}$$

where t is the current step in the algorithm. To make this algorithm valid we must always have $n(a) > 0$ so we observe each bandit once before starting the algorithm itself. This can be a non-negligible downside if the number of bandits is not very small relative to the horizon. In the infinite horizon setting this is still a problem if we are considering discounted rewards, as the more cleverly selected actions are deferred until a number of steps after the start. The full algorithm is detailed in algorithm (3).

This index penalises bandits which have been observed a large number of times relative to the total number of observations t. In fact, noting that the numerator is the natural logarithm of t we see that the exploration bonus decays increasingly quickly the greater the number of observations that have occurred. The idea of this index is that it represents the upper bound of a confidence interval which contains the expected reward $q_*(a)$ with overwhelmingly large

Algorithm 3: UCB

Require: c, maxIter
$\boldsymbol{Q} \leftarrow [0 \dots 0] \in \mathbb{R}^k$
$\boldsymbol{n} \leftarrow [0 \dots 0] \in \mathbb{N}^k$
for $i = 1, \dots, k$ **do**
    $R \leftarrow r(i)$
    $\boldsymbol{Q}[i] \leftarrow R$
    $\boldsymbol{n}[i] \leftarrow 1$
end for
for $t = 1, \dots,$ maxIter **do**
    $a \leftarrow 0$
    score $\leftarrow -\infty$
    for $i = 1, \dots, k$ **do**
        $s \leftarrow \boldsymbol{Q}[i] + c\sqrt{\log t / \boldsymbol{n}[i]}$
        if $s >$ score **then**
            $a \leftarrow i$
            score $\leftarrow s$
        end if
    end for
    $R \leftarrow r(a)$
    $\boldsymbol{Q}[a] \leftarrow \boldsymbol{Q}[a] + (R - \boldsymbol{Q}[a])/(\boldsymbol{n}[a] + 1)$
    $\boldsymbol{n}[a] \leftarrow \boldsymbol{n}[a] + 1$
end for

probability (Auer et al., 2002).

This algorithm is generally superior to ε-greedy because the exploration is performed in a controlled and reasonable way, rather than randomly selecting bandits which may even have been repeatedly observed to be poor. In UCB, a bandit which has received very few observations will be preferred to one with more observations, possibly even more than a bandit which has produced greater rewards in the past. This can allow the algorithm to avoid ignoring a bandit after it has given one or two low rewards which may have simply been a result of the noise in the reward function. The algorithm follows the intuitive notion that the size of the exploration bonus should be tied to the degree of uncertainty about a bandits rewards.

However, we once again face the problem that the algorithm requires a free hyperparameter c. A poor choice of c can cause under- or over-exploration and it is not clear how to select this value without repeatedly running the algorithm and performing validation. The appropriate value will depend on the specifics of the problem. When we use the algorithm in experiments

we will run UCB with several values of c and we will report the few choices which produced the strongest results to give the strongest benchmark against which to compare our algorithm. However, it should be kept in mind that in practice, a single run of UCB may have a less well chosen parameter and so cannot be expected to perform as well.

4.1.3 Thompson sampling

In Thompson sampling we iteratively update a Bayesian posterior for the reward distributions, and generate a stochastic policy by sampling from the posterior distribution at each time step. This approach is most convenient when the set up involves conjugate priors. Since the sampling distribution we are interested in is Gaussian with known variance, we can use a Gaussian prior and this becomes a very simple task.

If we apply a $\mathcal{N}(0,1)$ prior to each reward function mean $q_*(a)$, and recalling that the rewards have distribution $r(a) \sim \mathcal{N}(q_*(a),1)$, the posterior distribution $p(q_*(a)|\Sigma,n) = \mathcal{N}(\Sigma/n, 1/(n+1))$. At each step we take a sample from the posterior distribution of the parameters $q(a)$. In general, we would then select the bandit for which the sampled parameter maximises the expected reward. In the special case that we are considering with Gaussian reward functions, this is simply a matter of selecting the largest sample. The Thompson sampling algorithm is detailed in algorithm (4).

Algorithm 4: Thompson sampling for $r(a) \sim \mathcal{N}(q(a),1)$ and $q(a) \sim \mathcal{N}(0,1)$

Require: maxIter
$\boldsymbol{Q} \leftarrow [0\dots0] \in \mathbb{R}^k$
$\boldsymbol{n} \leftarrow [0\dots0] \in \mathbb{N}^k$
for $t = 1,\dots,$ maxIter **do**
  for $i = 1,\dots,k$ **do**
    Sample $\xi_i \sim \mathcal{N}(\boldsymbol{Q}[i], 1/(1+\boldsymbol{n}[i]))$
  end for
  $a \leftarrow \arg\max_a \xi_a$
  $R \leftarrow r(a)$
  $\boldsymbol{Q}[a] \leftarrow \boldsymbol{Q}[a] + (R - \boldsymbol{Q}[a])/(\boldsymbol{n}[a]+1)$
  $\boldsymbol{n}[a] \leftarrow \boldsymbol{n}[a] + 1$
end for

4.2 Bayesian optimisation

Since we are modelling the rewards in the MAB problem by a GP, it is not a large conceptual step to extend the action space to a continuous subset of the real line. Rather than just considering the GP at a discrete set of points, corresponding to different bandits, we will consider the entirety of the GP over a predetermined domain. We can continue to use the uncertainty estimates produced by the GP to select the appropriate level of exploration, from the precomputed GI. For this algorithm it will be more convenient for us to have a continuous representation of the GI rather than the discrete one we have used previously. Since we have computed the GI on a grid of integer points, this is an interpolation problem. Once again GPs provide a convenient solution.

To make things as convenient as possible, we first reparameterise the GI in terms of the standard deviation in the posterior distribution of the reward means $q_*(x)$, i.e. in terms of $1/\sqrt{n}$ instead of n. In this form, we can evaluate the GP at any level of posterior uncertainty and directly read off the appropriate exploration bonus. Let us call this version of the Gittins index $\tilde{\nu}(\sigma(x))$ where $\sigma(x)$ is the marginal standard deviation of the reward GP at input x. An example of the GI in this form is shown in 4.1. An interesting point is that under this parametrisation, the index is approximately linear. This will be useful in comparing the correlated Gittins method to UCB, as will be discussed later in this chapter.

The reward GP, giving us $\mu(x)$ and $\sigma(x)$ is computed in exactly the same way as it was in the MAB algorithm, except now we are interested in its values at real valued points x rather than just integer points.

These calculations lead to an acquisition function, which is in analogy with the regular Gittins index for bandits, which can be maximised to select the next sample point:

$$A(x) = \mu(x) + \tilde{\nu}(\sigma(x)) \tag{4.3}$$

with the next sample point x_* chosen such that

$$x_* = \arg\max_x A(x). \tag{4.4}$$

As is the normal goal of acquisition functions in BO, we have replaced an optimisation problem on an expensive, black-box function with an optimisation problem on a relatively cheap function for which we can compute gradients. Any standard optimisation algorithm

Fig. 4.1 GP mean $\tilde{\nu}(\sigma(x))$ of Gittins index parametrised as a function of standard deviation $\sigma(x)$ of reward distribution. $\gamma = 0.98$.

can be used to solve (4.4) such as L-BFGS. The correlated Gittins algorithm is the same as algorithm 1 except that we first fit the GP for $\tilde{\nu}(\sigma(x))$, values for $\tilde{\nu}$ are calculated by evaluating this GP at $\sigma(x)$ rather than by linear interpolation and we do not test the reward GP on the vector $\boldsymbol{t} = [1,\dots,k]$ we instead evaluate where ever is required by our choice of optimisation algorithm.

Before continuing on to the experiments we must describe the BO algorithms which will be used as comparisons against our algorithm, these being expected improvement, Thompson sampling and UCB.

4.2.1 Expected improvement

In BO the objective is the largest value sampled from the target function throughout the running of the algorithm. An intuitive criterion then for selecting a point to sample is how much we expect it to improve our value for this objective. If the greatest value so far sampled at time t is f_t^* and the function value at x is $f(x)$ then sampling at x will lead to an improvement of $\max(f(x) - f_t^*, 0)$. This is because if $f(x)$ is greater than f_t^* then we have improved our greatest reward by $f(x) - f_t^*$, otherwise the value of our objective is unchanged- the improvement is 0. Since we are using a statistical model as a surrogate

for f (and because our function evaluations are noisy), we are interested in the expected improvement acquisition function (Močkus, 1975), (Jones, 1998)

$$EI_t(x) = \mathbb{E}[\max(f(x) - f_t^*, 0)] \tag{4.5}$$

and the next sample point is selected to maximise this function.

Using a GP with posterior mean $\mu_t(x)$ and marginal variance $\sigma_t^2(x)$ at time t we can compute this acquisition function as

$$EI_t(x) = \mu(x) - f_t^* + \sigma_t(x)\mathcal{N}\left(\frac{\mu_t(x) - f_t^*}{\sigma_t(x)}\right) - |\mu_t(x) - f_t^*|\Phi\left(\frac{\mu_t(x) - f_t^*}{\sigma_t(x)}\right). \tag{4.6}$$

This choice of acquisition function is widely used due to its simplicity and intuitive appeal. It is not specifically designed to be applied to noisy objective functions and the exact interpretation as described does not strictly apply in this case. It is not uncommon to use this acquisition in this form even when noise is present, however this has been shown to negatively effect the algorithm's performance compared to other algorithms (Wu et al., 2017).

4.2.2 Thompson sampling for Bayesian optimisation

Thompson sampling for Bayesian optimisation (Kirthevasan et al., 2018) is a simple extension of the intuition behind Thompson sampling for MABs. We will simply sample a function from the GP which represents the current posterior distribution of the rewards and then maximise this function to select the next input value. It is clear how this approach is almost identical to the MAB method we took a sample form the predictive distribution of each bandit and selected the bandit with the largest sample.

This encourages exploration due to the fact that points at which the GP has a large mean are likely to produce large function values but also exploration as regions of input space with high uncertainty may allow the sampled function to take especially large values. This method again has an intuitive appeal despite having no guarantee of optimal performance.

4.2.3 GP-UCB

UCB for Bayesian optimisation, known as GP-UCB (Srinivas, 2009) as another intuitive generalisation from MABs. This acquisition function puts a probabilistic bound on the reward such that the sampled reward will be less than this bound with overwhelming probability.

In this case we simple need to extract the standard deviation of the reward GP and add a constant multiple c of it to the estimated mean.

$$UCB(x) = \mu(x) + c\sigma(x). \tag{4.7}$$

A difficulty of this acquisition function is that fact that the constant c must be selected. This is the same problem found in the MAB version of the algorithm. The level of exploration is left to be determined by the user. As was noted earlier, the correlated Gittins index method takes as the exploration bonus the Gittins index which is approximately linear as a function of the standard deviation. This makes the algorithm very similar to UCB which also has this behaviour. The correlated GI method does not have this free hyperparameter c to select, as it is effectively set automatically by the slope of $\tilde{\nu}(\sigma(x))$. Considering this, it may be fair to compare the correlated GI to a UCB algorithm where c has not been carefully selected. However, this seems like an excessively large handicap to place on the UCB algorithm and so in chapter 5 we will roughly tune UCB by running it with a few different values for c and selecting a small number with the best performance. It should be kept in mind when comparing these two algorithms that this tuning has gone on behind the scenes, arguably giving UCB an unfair advantage.

Chapter 5

Experiments

5.1 Multi-arm bandits

We first conduct experiments on the discrete multi-arm bandit problem using the correlated Gittins algorithm. We do this by simulating a large number (on the order of 1000) of runs in which a set of bandits with randomly generated mean rewards is created and we execute the algorithms that have been discussed on that set of bandits. The cumulative discounted reward at each time step is averaged across the runs. The criterion that our algorithm is designed to optimise is cumulative discounted rewards over an infinite horizon so in order to test this we first select a discount factor γ (or equivalently an effective horizon $h = 1 - 1/\gamma$) and run the algorithm for a multiple (e.g. 2-3) times the effective horizon. This gives a approximate idea of the long term behaviour.

To generate bandits with correlated means, a good approach is to use a Gaussian process. The hyperparamters of a squared-exponential covariance function can be used to modulate the level of correlation between them. To generate the mean rewards we simply sample the GP at locations $1,\ldots,k$ where k is the number of bandits. The length scale l is key to controlling the amount of correlation; since the bandits are at a separation of 1 unit, the length scale indicates how many bandits on either side of a given bandit are correlated with it in a non-negligible way. For example, if $l = 2$ then bandit i should be strongly correlated with bandits $i-1$ and $i+1$, weakly correlated with bandits $i-2$ and $i+2$ and very little with any other bandits. If $l \ll 1$ then the bandits will hardly be correlated at all, since functions from the GP can oscillate freely in between two bandits, information about one bandit tells us nothing about any others. This setting provides a useful first experiment- we can test whether the correlated Gittins algorithm reduces (approximately) to the uncorrelated version when

the bandits are uncorrelated.

Figure 5.1 shows that the discounted rewards from the correlated and uncorrelated Gittins algorithms are very similar and they outperform the other algorithms tested.

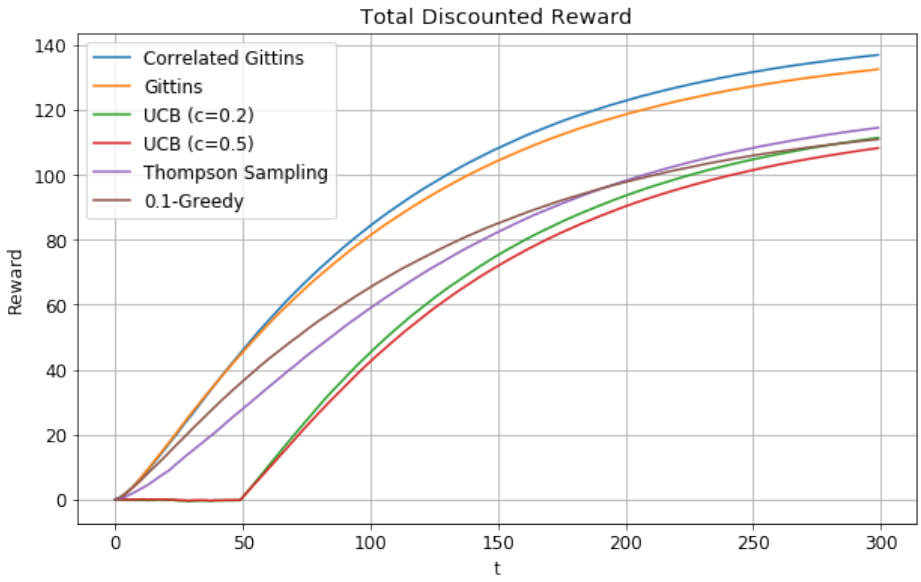


Fig. 5.1 Uncorrelated bandits. $\gamma = 0.99$, $l = 0.1$, $k = 50$. The Gittins index algorithms outperform all others.

When the length scale is increased, the bandits become correlated. A sample of the generated sets of bandits is shown in Figure 5.2. Now there is a large amount of information that can be gained by observing nearby bandits. Figure 5.3 shows that in this scenario, with $l = 4$ the correlated Gittins algorithm far outperforms all the other algorithms tested. In fact, all algorithms other than correlated Gittins take a significant penalty to performance in the correlated setting. We see that the regular Gittins algorithm is no longer superior to UCB, Thompson sampling or ε-greedy.

A further interesting metric is the largest reward found during a run. While this metric is not explicitly optimised for by any of these algorithms it can be an instructive statistic. It is influenced by both the accuracy with which an algorithm locates the optimum of the reward function and the amount of time spent at or near the optimum. An algorithm which rapidly converges on the optimum will have the best opportunity to receive large individual rewards due to the inherent randomness of the rewards. We measure this metric by identifying the algorithm which receives the largest individual reward on each run and calculating the proportion of 'wins' for each algorithm. Results for average total discounted rewards and

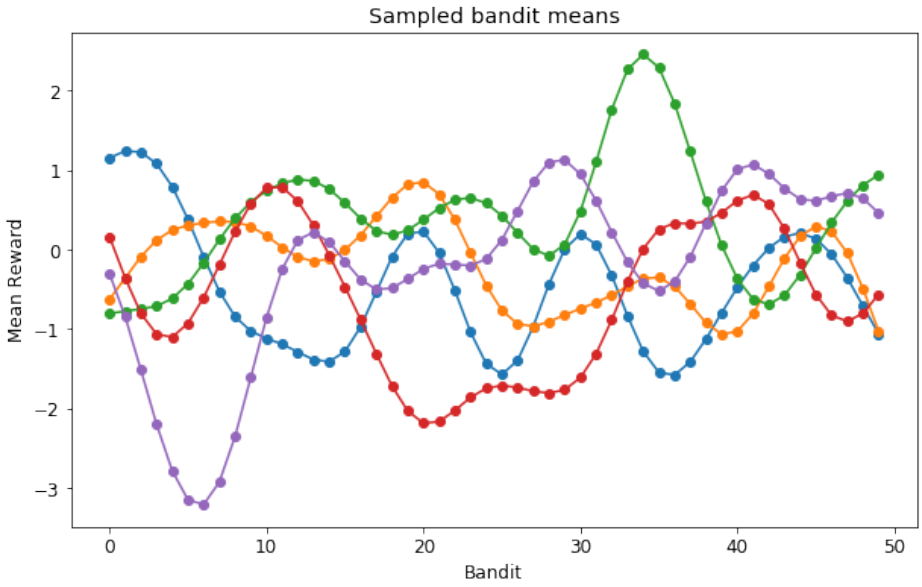


Fig. 5.2 Samples of bandits drawn from a GP with length scale $l = 4$ and $k = 50$. There is a high degree correlation between neighboring bandits.

probability of receiving the greatest reward are given in Tables 5.1 and 5.2 for $k = 10$ and $k = 50$ respectively.

Table 5.1 Average total discounted reward and probability of receiving greatest reward with $k = 10$.

	Total Discounted Reward					% Largest Reward				
Length scale	0.1	1	2	3	4	0.1	1	2	3	4
Corr. Gittins	**138.1**	**96.66**	**77.57**	**63.91**	**50.06**	**0.231**	0.190	0.202	**0.202**	**0.214**
Gittins	132.5	87.59	65.78	50.36	37.56	0.204	0.142	0.155	0.166	0.185
UCB (c=1.0)	104.2	87.94	65.82	52.77	38.22	0.193	**0.203**	0.141	0.179	0.185
UCB (c=1.5)	94.91	78.90	58.97	45.16	31.84	0.104	0.180	0.175	0.145	0.162
Thompson	103.1	87.23	66.17	51.89	37.75	0.175	0.149	**0.215**	0.161	0.151
0.1-Greedy	98.16	74.82	60.23	48.63	37.35	0.093	0.135	0.112	0.141	0.106

These results show that the correlated Gittins algorithm consistently outperforms the other algorithms when $l \geq 1$ and this advantage is greater for higher degrees of correlation between the bandits. The effect is also increased by having a greater number of individual bandits. We see that the correlated Gittins algorithm typically has the greatest chance of receiving the largest reward, however the relationship between this metric and the level of correlation and number of bandits is not so obvious. This data also indicates that the two metrics are not as closely related as one might imagine; it is possible for an algorithm to receive far greater

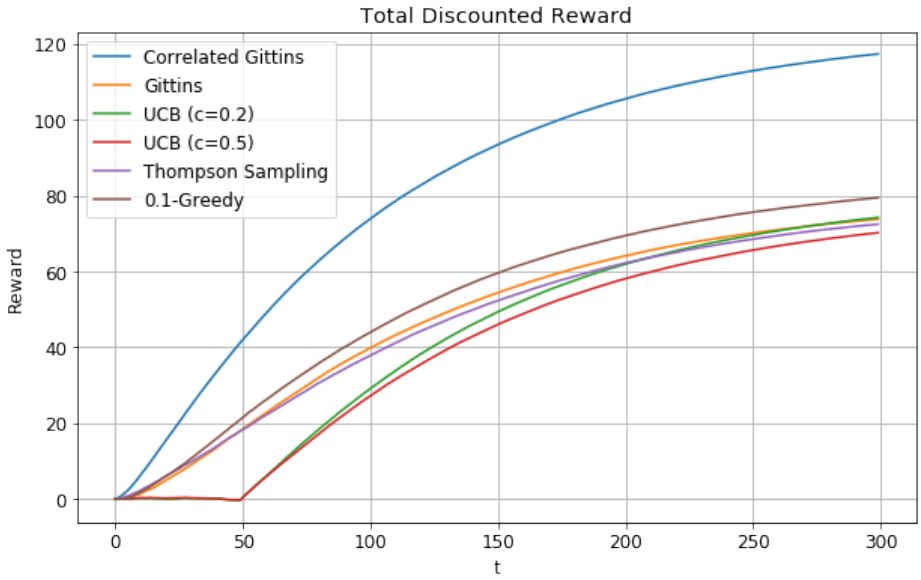


Fig. 5.3 Correlated bandits. $\gamma = 0.99$, $l = 4.0$, $k = 50$.

Table 5.2 Average total discounted reward and probability of receiving greatest reward with $k = 50$.

	Total Discounted Reward					Probability of Greatest Reward				
Length scale	0.1	1	2	3	4	0.1	1	2	3	4
Corr. Gittins	**134.6**	**142.1**	**140.6**	**128.5**	**116.9**	0.083	0.140	**0.230**	**0.223**	**0.220**
Gittins	130.7	116.4	98.60	83.51	70.44	0.052	0.123	0.097	0.073	0.099
UCB (c=0.2)	113.6	108.5	97.19	84.09	72.09	0.213	**0.278**	0.217	0.210	0.207
UCB (c=0.5)	110.5	105.1	92.73	79.14	68.44	0.245	0.193	0.210	0.181	0.173
Thompson	116.8	111.7	98.03	82.58	69.74	**0.271**	0.150	0.168	0.167	0.162
0.1-Greedy	117.3	104.3	94.20	88.84	76.79	0.136	0.101	0.078	0.147	0.139

discounted rewards than another algorithm but not necessarily receive the largest reward on a consistent basis.

5.2 Bayesian optimisation

To test the correlated Gittins Bayesian optimisation algorithm we generate the reward mean function from a GP with a squared-exponential covariance function with length scale l. The algorithm and the comparison algorithms are all run on the same function for a fixed number of steps and the total discounted rewards is calculated. This is averaged over a large number of runs with different functions generated from the same GP. UCB was once again roughly tuned and the best performing values recorded. We can link the BO problem to MABs by

noticing that if the length scale is l and the domain of the inputs has width w then we can roughly divide the domain into w/l approximately independent regions. We take this number to be an analogue for the number bandits in a MAB problem, $\widetilde{k}$.

Table 5.3 Average total discounted reward and probability of receiving greatest reward in Bayesian optimisation.The Gittins index uses $\gamma = 0.98$. In these experiments $w = 2$, so $\widetilde{k} = 2/l$.

	Total Discounted Reward				Prob. Greatest Reward			
Length scale	0.2	0.4	0.8	1.0	0.2	0.4	0.8	1.0
$\widetilde{k}$	10	5.0	2.5	0.5	10	5.0	2.5	0.5
Corr. Gittins	23.64	21.24	**15.77**	**16.02**	0.22	0.22	0.24	**0.26**
Expected Improvement	19.73	17.77	13.69	13.56	0.10	0.18	0.18	0.16
UCB (c=2.1)	**23.94**	**21.71**	14.07	15.56	0.20	**0.24**	**0.28**	0.18
UCB (c=2.5)	23.92	21.32	13.50	14.75	0.22	**0.24**	0.14	0.14
Thompson	17.10	12.36	7.63	7.97	**0.26**	0.12	0.16	**0.26**

These results suggest that the correlated Gittins algorithm is less successful in this setting than with discrete bandits. While it achieved relatively stronger performance for longer length scales, its performance is much more inconsistent at shorter scales and is outperformed by UCB. As noted in chapter 4, the correlated Gittins algorithm can approximately be viewed as a version of UCB with the exploration parameter already determined by the Gittins index. However, these experiments show that this approach does not find an optimal level for this exploration as it can be improved upon by tuning UCB. This suboptimality may be because our algorithm only considers the marginal variance at each point when determining the exploration bonus to apply. The lack of reference to the posterior GP covariance between points may be causing it to under-explore. It is not immediately obvious how to solve this problem, or why it seems to be more impactful in BO than in MABs.

Despite this, we found that the correlated Gittins algorithm had comparable or better performance in terms of total discounted reward than the other methods tested on longer length scales. This is especially useful as our algorithm does not require any hyperparameter tuning, unlike the next best performing algorithm, UCB. Therefore, especially in settings where it is too expensive to test different exploration parameters, our algorithm may be considered better than UCB.

The results are less encouraging in terms of the probability of correlated Gittins receiving a larger single greatest reward compared to the other algorithms. This is maybe not entirely surprising. The intuition which suggests using the GI as a basis for an algorithm of this kind

comes from the optimality of GI for total discounted rewards. We have no such guarantees of optimality for the single largest reward attained. Therefore this metric it testing a property which this approach is not strictly designed for, however this limitation should be kept in mind when comparatively evaluating the algorithm. Ultimately, the usefulness of the algorithm very much depends on the specifics of the problem and the criteria which one wants to optimise.

Chapter 6

Conclusion and Future Directions

The aims of this project can be divided into three parts; an algorithm for the computation of the Gittins index, a Gittins based algorithm for correlated bandits and a Gittins based algorithm for Bayesian optimisation. We will give conclusions to these sections individually and following that we suggest a number of possible directions that these ideas could be taken in.

One of the main aims of the work on calculating the Gittins index was to build upon the squared-exponential GP approach and to create a version which does not require gradient based hyper parameter tuning. Since the cubic spline GP has a posterior mean which is independent of the hyperparemeter of the covariance function, this was successful. This can lead to a much faster run-time of the algorithm for a given horizon length and level of precision. It is not entirely meaningful to quantify the difference in run-time, since this is heavily determined by the user's choice of how many iterations to optimise the squared-exponential hyperparameters for. However, it is clear that avoiding this step entirely, which involves many expensive matrix inversion operations, will necessarily increase the speed. This allows for computing the index for longer horizons in a given length of time. The closed form of the DP step of the calculation is dependent on the choice of covariance function, so this work required deriving this expression by solving the integral of the cubic-spline GP mean multiplied by a Gaussian distribution. All together, this aspect of the project achieved its aims, and the ability to efficiently compute the Gittins index enabled the work that followed.

In designing an algorithm for correlated multi-arm bandits we had two criteria for a successful job. One criterion was that the design of the algorithm comes from a principled position where our approach to balancing exploration and exploitation was based on well established theoretical concepts. This is why we believe our Gittins based approach is sensible, we

are basing our method on an algorithm which is provably optimal in a special case of the problem we are addressing. This is in contrast to some other existing approaches which, while they have good justifications behind them for why they should give good performance, do not not necessarily have a good reason to say they should be optimal. This is especially the case for algorithms which require the selecting of hyperparameters like UCB, an issue which our algorithm avoids. The second criterion for evaluating our algorithm is, of course, its performance. In the special case of uncorrelated bandits, we saw the expected result that our algorithm outperforms the others, since we are approximately just performing the Gittins index algorithm. Importantly, when the bandits become correlated we see the correlated Gittins algorithm greatly outperform the other methods.

The algorithm for Bayesian optimisation had similar aims. The algorithm is almost identical to the bandits algorithm, so we are happy that the idea comes from a theoretically principled position. However, it seems that something is lacking that is needed to make this algorithm thoroughly outperform its competition. It has comparable performance to the other algorithms that were tested however it could be outperformed in terms of total discounted rewards by a tuned UCB algorithm. Also, Bayesian optimisation places emphasis on a different evaluation criterion from bandits. In BO we are usually more interested in the largest function evaluation found rather than a total reward. Our algorithm was not very consistent on this front compared to alternatives, which is not necessarily surprising as this criterion is quite distinct from that which the Gittins index is designed for. Perhaps this change in criterion is a step too far away from the assumptions which informed our approach.

One further direction to take this work in is to attempt to understand the cause of the shortfall in performance in the BO algorithm. Experiments performed to this end didn't expose a clear reason or bring much value to the discussion (leading to their omission from this paper), but with more time it would be valuable to learn if the issue can be fixed by a small adjustment or if the problem runs deeper in the concept itself. It would be interesting to understand why the performance is so different (relative to other algorithms) between the MAB algorithm and the BO algorithm despite the two being conceptually very similar.

There are also directions which could be investigated with regards to computation of the Gittins index. One possibility is to create a version of the algorithm for finite horizons. In this work we always assumed an infinite horizon with discounted rewards which led to a notion of an effective horizon. It may be more accurate to real world experiments to work with actually finite horizons. We can see this because in the infinite horizon case the exploration bonus is

always positive, however if the experiment must end at some time N there should be no value in exploring at this final step. Computing these values for a finite horizon would require computing a value of the Gittins index for every pair of n, the usual statistic for the number of observations of the bandit and $N-t$, the number of remaining steps to the horizon N. This would be a grid of $O(N^2)$ numbers rather than the $O(N)$ from infinite horizons. These values could be used in our algorithms to see how much removing this source of approximation affects the performance in experiments with finite horizons.

Another direction may be to consider other variations on the MAB problem. One change would be to consider bandits with Bernoulli rewards rather than Gaussian, or rewards from other exponential family distributions or otherwise. One could also consider other variants of the MAB such as restless bandits (Whittle, 1988), contextual bandits or combinatorial bandits (Chen et al. 2013). While the specifics of the approach would vary under the differing assumptions of these problems, they all depend on the fundamental question of balancing exploration and exploitation which means the ideas discussed in this project should be useful to their study.

References

Agrawal, R. (1995). Sample mean based index policies by o(log n) regret for the multi-armed bandit problem. *Advances in Applied Probability*, 27(4):1054–1078.

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2):235–256.

Bellman, R. (1956). A problem in the sequential design of experiments. *Sankhyā: The Indian Journal of Statistics (1933-1960)*, 16(3/4):221–229.

Berry, D. A. and Fristedt, B. (1985). Bandit problems: sequential allocation of experiments (monographs on statistics and applied probability). *London: Chapman and Hall*, 5(71-87):7–7.

Blackwell, D. (1965). Discounted dynamic programming. *The Annals of Mathematical Statistics*, 36(1):226–235.

Bouneffouf, D. and Rish, I. (2019). A survey on practical applications of multi-armed and contextual bandits. *arXiv preprint arXiv:1904.10040*.

Bouneffouf, D., Rish, I., and Cecchi, G. A. (2017). Bandit models of human behavior: Reward processing in mental disorders. *CoRR*, abs/1706.02897.

Chakravorty, J. and Mahajan, A. (2014). Multi-armed bandits, gittins index, and its calculation. *Methods and applications of statistics in clinical trials: Planning, analysis, and inferential methods*, 2(416-435):455.

Chen, W., Wang, Y., and Yuan, Y. (2013). Combinatorial multi-armed bandit: General framework and applications. In *International Conference on Machine Learning*, pages 151–159. PMLR.

Durand, A., Achilleos, C., Iacovides, D., Strati, K., Mitsis, G. D., and Pineau, J. (2018). Contextual bandits for adapting treatment in a mouse model of de novo carcinogenesis. In *Machine learning for healthcare conference*, pages 67–82. PMLR.

Edwards, J. (2019). Practical calculation of gittins indices for multi-armed bandits. *arXiv preprint arXiv:1909.05075*.

El Karoui, N. and Karatzas, I. (1993). General gittins index processes in discrete time. *Proceedings of the National Academy of Sciences*, 90(4):1232–1236.

El Karoui, N. and Karatzas, I. (1994). Dynamic allocation problems in continuous time. *The Annals of Applied Probability*, pages 255–286.

Forrester, A., Sobester, A., and Keane, A. (2008). *Engineering design via surrogate modelling: a practical guide*. John Wiley & Sons.

Frazier, P., Powell, W., and Dayanik, S. (2009). The knowledge-gradient policy for correlated normal beliefs. *INFORMS journal on Computing*, 21(4):599–613.

Frazier, P. I. (2018). A tutorial on bayesian optimization.

Frazier, P. I. and Wang, J. (2016). Bayesian optimization for materials design. In *Information science for materials discovery and design*, pages 45–75. Springer.

Gittins, J. (1974). A dynamic allocation index for the sequential design of experiments. *Progress in statistics*, pages 241–266.

Gittins, J., Glazebrook, K., and Weber, R. (2011). *Multi-armed bandit allocation indices*. John Wiley & Sons.

Gittins, J. C. (1979). Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society: Series B (Methodological)*, 41(2):148–164.

Griffiths, R.-R. and Hernández-Lobato, J. M. (2020). Constrained bayesian optimization for automatic chemical design using variational autoencoders. *Chemical science*, 11(2):577–586.

Hennig, P. and Schuler, C. J. (2012). Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13(6).

Hernández-Lobato, J. M., Hoffman, M. W., and Ghahramani, Z. (2014). Predictive entropy search for efficient global optimization of black-box functions. *arXiv preprint arXiv:1406.2541*.

Ishikida, T. and Varaiya, P. (1994). Multi-armed bandit problem revisited. *Journal of Optimization Theory and Applications*, 83(1):113–154.

Jones, D. R., Schonlau, M., and Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492.

Kaelbling, L. P. (1993). *Learning in embedded systems*. MIT press.

Kandasamy, K., Krishnamurthy, A., Schneider, J., and Poczos, B. (2018). Parallelised bayesian optimisation via thompson sampling. In Storkey, A. and Perez-Cruz, F., editors, *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 133–142. PMLR.

Katehakis, M. N. and Veinott Jr, A. F. (1987). The multi-armed bandit problem: decomposition and computation. *Mathematics of Operations Research*, 12(2):262–268.

Kushner, H. J. (1964). A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*.

Lai, T. L. and Robbins, H. (1985). Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22.

Losada, D. E., Parapar, J., and Barreiro, A. (2017). Multi-armed bandits for adjudicating documents in pooling-based evaluation of information retrieval systems. *Information Processing & Management*, 53(5):1005–1025.

Močkus, J. (1975). On bayesian methods for seeking the extremum. In *Optimization techniques IFIP technical conference*, pages 400–404. Springer.

Narendra, K. S. and Thathachar, M. A. (1989). *Learning automata: an introduction*. Courier corporation.

Negoescu, D. M., Frazier, P. I., and Powell, W. B. (2011). The knowledge-gradient algorithm for sequencing experiments in drug discovery. *INFORMS Journal on Computing*, 23(3):346–363.

Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Rasmussen, C. E. and Williams, C. K. (2006). *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA.

Robbins, H. (1952). Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–535.

Shen, W., Wang, J., Jiang, Y.-G., and Zha, H. (2015). Portfolio choices with orthogonal bandit learning. In *Twenty-fourth international joint conference on artificial intelligence*.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25.

Srinivas, N., Krause, A., Kakade, S. M., and Seeger, M. (2009). Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Teugels, J. L. (1976). A bibliography on semi-markov processes. *Journal of Computational and Applied Mathematics*, 2(2):125–144.

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294.

Thompson, W. R. (1935). On the theory of apportionment. *American Journal of Mathematics*, 57(2):450–456.

Titsias, M. (2009). Variational learning of inducing variables in sparse gaussian processes. In *Artificial intelligence and statistics*, pages 567–574. PMLR.

Tsitsiklis, J. N. (1994). A short proof of the gittins index theorem. *The Annals of Applied Probability*, pages 194–199.

Wald, A. (1949). Statistical decision functions. *The Annals of Mathematical Statistics*, 20(2):165–205.

Weber, R. et al. (1992). On the gittins index for multiarmed bandits. *The Annals of Applied Probability*, 2(4):1024–1033.

Whittle, P. (1988). Restless bandits: Activity allocation in a changing world. *Journal of applied probability*, 25(A):287–298.

Witten, I. H. (1976). The apparent conflict between estimation and control—a survey of the two-armed bandit problem. *Journal of the Franklin Institute*, 301(1-2):161–189.

Wu, J., Poloczek, M., Wilson, A. G., and Frazier, P. I. (2017). Bayesian optimization with gradients. *arXiv preprint arXiv:1703.04389*.

Wyatt, J. (1998). Exploration and inference in learning from reinforcement.

Zhou, Q., Zhang, X., Xu, J., and Liang, B. (2017). Large-scale bandit approaches for recommender systems. In *International Conference on Neural Information Processing*, pages 811–821. Springer.