The Exploration-Exploitation Trade-Off in Sequential Decision Making Problems

A thesis presented for the degree of

Doctor of Philosophy of Imperial College London

April, 2011

Adam M. Sykulski

Department of Mathematics
Imperial College London
180 Queen's Gate
London SW7 2BZ

School of Electronics and
Computer Science
University of Southampton
Southampton SO17 1BJ

I certify that this thesis, and the research to which it refers, are the product of my own work, and that any ideas or quotations from the work of other people, published or otherwise, are fully acknowledged in accordance with the standard referencing practices of the discipline.

Signed:

Copyright

Abstract

Sequential decision making problems require an agent to repeatedly choose between a series of actions. Common to such problems is the exploration-exploitation trade-off, where an agent must choose between the action expected to yield the best reward (exploitation) or trying an alternative action for potential future benefit (exploration). The main focus of this thesis is to understand in more detail the role this trade-off plays in various important sequential decision making problems, in terms of maximising finite-time reward.

The most common and best studied abstraction of the exploration-exploitation trade-off is the classic multi-armed bandit problem. In this thesis we study several important extensions that are more suitable than the classic problem to real-world applications. These extensions include scenarios where the rewards for actions change over time or the presence of other agents must be repeatedly considered. In these contexts, the exploration-exploitation trade-off has a more complicated role in terms of maximising finite-time performance. For example, the amount of exploration required will constantly change in a dynamic decision problem, in multi-agent problems agents can explore by communication, and in repeated games, the exploration-exploitation trade-off must be jointly considered with game theoretic reasoning.

Existing techniques for balancing exploration-exploitation are focused on achieving desirable asymptotic behaviour and are in general only applicable to basic decision problems. The most flexible state-of-the-art approaches, ϵ-greedy and ϵ-first, require exploration parameters to be set *a priori*, the optimal values of which are highly dependent on the problem faced. To overcome this, we construct a novel al-

gorithm, ϵ-ADAPT, which has no exploration parameters and can adapt exploration on-line for a wide range of problems. ϵ-ADAPT is built on newly proven theoretical properties of the ϵ-first policy and we demonstrate that ϵ-ADAPT can accurately learn not only *how much* to explore, but also *when* and *which actions* to explore.

Acknowledgements

First and foremost I would like to thank my supervisors Niall Adams and Nick Jennings. Without their expertise, innovative ideas and unwavering support the completion of this thesis would certainly have not been possible. I would also like to thank them both for their more than generous contribution of time and advice and the detailed corrections and suggestions made throughout the write-up of this thesis. Furthermore, I would like to thank them both for providing me with the opportunity to work with two excellent research groups at two different universities – this has certainly been a unique and invaluable experience.

I would also like to thank several people from both Imperial and Southampton that I have collaborated with or have provided me with invaluable advice and support. In particular, I would like to thank Nicos Pavlidis, Archie Chapman and Enrique Munoz de Cote for their collaboration and expertise throughout. I would also like to thank David Hand, Nicolo Cesa Bianchi, David Wolpert, David Nicholson, Robin Mason, Igor Golosnoy, Gerald Tesauro, Alex Rogers, Christoforos Anagnostopoulos, Patrick Rubin-Delanchy, Matthew Turnbull and Elena Ehrlich for their insights and expertise which were crucial in developing the research presented in this thesis. Additionally, I would like to thank my PhD examiners Alastair Young and David Leslie, for their insightful comments and important suggested corrections.

I gratefully acknowledge the financial support provided by BAE Systems and EPSRC through the ALADDIN project.

Finally, I would like to thank my parents Jan and Elżbieta, my partner Deepti and my sister Hanna for their loving support and encouragement throughout, and also to my late grandfather, Czesław Dorabialski, to whom this thesis is dedicated.

Table of contents

List of Figures

List of Tables

Chapter 1

Introduction

In many real-world situations, decision makers are required to repeatedly choose actions from a set of available options. Important examples include allocating drugs to patients in a clinical trial, choosing targets to track in a multi-target tracking problem and providing product recommendations for users visiting an e-commerce website. These are all examples of sequential decision making problems. In each case, the decision maker (henceforth called an agent) will learn which action it prefers based on past experiences and current circumstances. Agents accrue rewards from the actions they select, and seek to maximise the total reward gained over a period of time. At each time-step, however, the agent is faced with a trade-off between exploration and exploitation – where the agent must choose between what it believes is the best action (exploitation) and trying alternative actions for potential future benefit (exploration). For example, in a clinical trial, a patient can be allocated the tried and tested drug which is expected to perform best, or a new unknown drug which may perform better and subsequently benefit many patients – but may alternatively cause the patient adverse undesirable symptoms.

The exploration-exploitation trade-off is in fact central to any sequential decision making problem (beyond the examples given above) where agents are uncertain of future rewards and the rewards of unselected actions are not observed. Many sequential decision making problems have these features. The purpose of this thesis is therefore to study this trade-off in many important and applicable sequential decision making frameworks, in terms of maximising the total reward gained in finite-time problems.

The most common and best studied abstraction of the exploration-exploitation trade-off in sequential decision making problems is the multi-armed bandit problem (Robbins, 1952), which we comprehensively introduce and review in Chapter 2. The objective of the bandit problem is to select the optimal action from a set of available actions at each time-step, where the expected rewards for each action are unknown *a priori*. Throughout this thesis, we study the bandit problem in detail, to make new insights and develop new algorithms that maximise the reward gained by an agent. We study several important extensions of the bandit problem, however, that are more suitable to real-world applications than the classic bandit problem. These include problems where additional side information is observed that is relevant to the decision problem, which is then further extended to scenarios with multiple interacting agents that can communicate this side information with each other.

Most existing literature in the bandit problem has developed methods that maximise reward asymptotically (as discussed in Chapter 2), but in this thesis we are motivated by maximising reward in finite-time problems, as this objective is more useful in real-world problems. This is because real-world problems are always likely to be finite in length or change over time, such that asymptotic convergence is neither meaningful nor desirable.

1.1 Research Contributions

Throughout this thesis we consider two central themes that are fundamental to the various sequential decision making problems considered. First is the role that the exploration-exploitation trade-off plays and how it is fundamentally related to the problem of maximising reward in finite-time problems. This concept is well understood for simple single-agent sequential decision making problems (such as basic bandit problems), but it is still poorly understood in many other important sequential decision making problems such as:

- Bandit problems with side information, where additional relevant information must be considered prior to each action.
- Dynamic bandit problems, where the expected reward for an action is changing over time.
- Multi-agent decision making problems, where the presence of other decision makers must be considered.

These problems characterise several real-world decision making problems. Not only is it unclear *how much* an agent should explore in these environments, but it is also difficult to ascertain *when* this exploration should occur, *which actions* should be explored and – in the multi-agent case – in what way exploration should be dependent on the presence of other agents. This first theme therefore serves to provide a better understanding of the exploration-exploitation trade-off in sequential decision making problems. In particular, selecting the correct actions for exploration (and at the right time) is crucial in terms of maximising reward in finite-time problems, as demonstrated throughout this thesis.

The second central theme, which builds on the findings of the first, is the construction of practical and implementable algorithms for each type of sequential decision making problem that we study. The exact solution to optimally balancing the exploration-exploitation trade-off is almost always an intractable calculation (Sutton and Barto, 1998). Gittins (1979) provides a more tractable, but still computationally intensive solution to basic bandit problems, which assumes certain fixed reward distributions. Otherwise, in more complicated sequential decision making problems, the current state-of-the-art is to apply off-line stochastic policies such as ϵ-greedy or ϵ-first, which are introduced in more detail (together with other core policies and algorithms) in Chapter 2. These exploration policies can provide good results in finite-time problems, as compared with other exploration policies or no exploration at all, but their performance is inexorably linked with the setting of an exploration parameter *a priori* that governs the overall amount of exploration. The optimal value of this parameter is likely to be unknown to an agent *a priori* in real-world applications, and moreover performance can degrade rapidly with a poorly chosen parameter value (Sutton and Barto, 1998).

For these reasons, in this thesis we first attempt to find the optimal value of the exploration parameter for the standard ϵ-greedy and ϵ-first policies, for a basic bandit problem. This analysis allows an agent to select optimal parameters if the parameters of the problem are known beforehand, otherwise these can be learnt on-line. For more enhanced and realistic bandit problems, however, where the theoretical calculation of optimal parameters becomes intractable, we build an algorithm that can approximate the optimal exploration decision on-line, without the need for a prefixed exploration parameter. This algorithm, which we call ϵ-ADAPT, is much more applicable to practical domains due to the absence of an exploration parameter. In addition, ϵ-ADAPT can be extended to various multi-agent sequential decision making problems. The on-line approach of ϵ-ADAPT allows this algorithm to perform comparably with optimally tuned off-line policies, as ϵ-ADAPT can learn the circumstances of the decision making problem as it plays, and adapt to events such as the arrival of different side information, the dynamics in the environment or changes to the behaviour of other agents in the system. ϵ-ADAPT can therefore learn *how much*, *when* and *which action* to explore and how best to explore in the presence of other agents.

The main contributions of this thesis can hence be summarised as follows:

- Theoretical analysis and proofs of the behaviour of ϵ-first and ϵ-greedy policies in bandit problems, including derivations and proofs of the optimal exploration rates for a basic bandit problem. These findings are fundamental to the construction of the ϵ-ADAPT algorithm.

- An autonomous on-line algorithm, ϵ-ADAPT, which is the first algorithm that can adapt exploration on-line in sequential decision making problems, without the need for an *a priori* fixed exploration parameter.

- Extensive simulation results showing the favourable performance of ϵ-ADAPT for all decision making problems considered, as compared with optimally tuned off-line policies such as ϵ-first and ϵ-greedy.

- The construction and analysis of several off-line policies for multi-agent sequential decision making problems, specifically for the novel multi-agent bandit with communication problem studied in Chapter 6 and the repeated games with unknown rewards framework studied in Chapter 7.

1.2 Thesis Structure

This thesis is structured as follows:

Chapter 2 provides a background on the multi-armed bandit problem, which is central to the study of the exploration-exploitation trade-off in this thesis. We provide a review of well-studied frameworks and of existing policies and algorithms. We also provide an extensive analysis of the various policies and algorithms in terms of their flexibility and applicability to different problems. This motivates the need for the research contributions that follow.

In **Chapter 3** we theoretically examine a basic bandit with covariates problem. Specifically, we derive and prove optimal exploration rates for the ϵ-greedy and ϵ-first policies, which are frequently used and strong performing policies for bandit problems in general. These findings are then used to construct a novel on-line algorithm (free of exploration parameters) which is the building block of the ϵ-ADAPT algorithm.

Chapter 4 constructs ϵ-ADAPT, our on-line algorithm for multi-armed bandit problems, for problems with and without side information. The algorithm is built using newly derived theoretical properties of the ϵ-first policy. Several key additional features are introduced to the algorithm to improve finite-time performance, which we demonstrate through a detailed empirical study. We note that some of the findings in this chapter are also presented in Sykulski et al. (2010a).

In **Chapter 5** we study dynamic bandit problems, where the expected rewards received for actions change over time. This is an important and realistic extension of bandit problems to real-world applications. We make several key changes to the

ϵ-ADAPT algorithm of Chapter 4 and demonstrate the strong performance again through an extensive empirical study of several different dynamic reward processes.

Chapter 6 starts by providing a short background on relevant multi-agent sequential decision making problems. Motivated by this, we then study a multi-agent bandit problem with communication, to show the importance of agents *exploring* their communication decisions. We construct a novel off-line policy, which is an extension of ϵ-greedy and also extend ϵ-ADAPT to this multi-agent framework.

In **Chapter 7** we study repeated 2-agent, 2-action games (also known as $2{\times}2$ games) with unknown rewards, to study the impact of the exploration-exploitation trade-off in relation to game theoretic reasoning. We study several fundamental off-line policies before using these findings to extend ϵ-ADAPT to this framework. We show that optimal rates of exploration are dependent on the type of opponent, as well as the reward structure of the game.

Finally, in **Chapter 8** we conclude and present key directions for future work.

Chapter 2

Sequential Decision Making and Bandit Problems

This thesis first examines the exploration-exploitation trade-off in the single-agent domain by studying the multi-armed bandit problem (Chapters 3, 4 and 5) and then in the multi-agent domain by considering various extensions of the bandit problem (Chapters 6 and 7). Since the bandit problem is central, in this chapter we provide a thorough overview of existing literature on the multi-armed bandit problem, including the various types of problem considered, which we refer to as *bandit frameworks*, and the range of existing policies and algorithms that attempt to balance the exploration-exploitation trade-off. Finally, we evaluate the applicability of existing policies and algorithms to the different bandit frameworks and perform a critical analysis of their key advantages and disadvantages. This analysis identifies several shortcomings in the current state-of-the-art which we try to address in the proceeding chapters. Note that we review the relevant literature for multi-agent sequential decision making problems, which are (in general) not directly related to the bandit problem, at the beginning of Chapters 6 and 7.

The multi-armed (or k-armed) bandit problem is the most basic and best studied abstraction of the exploration-exploitation trade-off in sequential decision making problems. Originally documented by Robbins (1952), the problem is based on the analogy of a series of slot machines or one-armed bandits. A gambler selects one arm to pull at each time-step (arm a_i at time t where $i \in \{1, \ldots, k\}$) and then receives a corresponding reward $r_i(t)$ (which can be negative). The objective of the gambler is to maximise cumulative reward over a sequence of pulls. Note that we henceforth refer to the gambler as an agent, and the arms as actions, to keep the terminology consistent with the multi-agent system problems considered in Chapters 6 and 7.

The bandit problem has three key properties. First, the agent starts with little or no prior knowledge of the expected rewards of each action. Secondly, the agent has no knowledge of the rewards foregone from unselected actions – this makes the problem *opaque* (as opposed to *transparent* where the agent observes the rewards of all actions) and finally, the observed reward of a selected action is not always the same (it is either noisily observed or changes over time). Taking these properties together, the agent must consider exploring all actions in an effective policy to *learn* about their potential future rewards. The agent therefore faces an exploration-exploitation trade-off, where the agent must choose between selecting actions that are expected to perform best (exploitation) and selecting alternative actions for potential future benefit (exploration). The objective of the multi-armed bandit problem is to design a policy that can use past actions and rewards to select the next action whilst simultaneously balancing the exploration-exploitation trade-off.

The multi-armed bandit problem has been extensively studied in the fields of statistics (Berry and Fristedt, 1985; Lai and Robbins, 1985), machine learning (Sutton and Barto, 1998), economics (Rothschild, 1974) and multi-agent systems (Carmel and Markovitch, 1999) and has applications in areas as diverse as on-line auctions (Blum et al., 2003), clinical trials (Hardwick et al., 1998; Woodroofe, 1979), market pricing (Azoulay-Schwartz et al., 2004; Rothschild, 1974; Weitzman, 1979), organisational learning (March, 1991), web advertising (Kleinberg et al., 2008; Pandey et al., 2007) and multi-target tracking (Hero et al., 2006; Krishnamurthy and Evans, 2001). Notice that all of these applications match the key properties of the bandit problem mentioned above. In clinical trials, for example: there will usually be little prior knowledge about the effectiveness of an untested drug, the outcome of untried drugs on patients will be unknown, and the benefits/symptoms of tried drugs will vary from person to person and over time. In fact, several real-world applications of sequential decision making problems are likely to have these key properties and it is for these reasons that the multi-armed bandit problem has attracted such wide attention in the past 40 years.

This chapter is structured as follows. In Section 2.1 we introduce the various bandit frameworks that have been considered. Then, in Section 2.2, we summarise all the different algorithms and policies that have been constructed for the bandit problem. Then we evaluate the strengths and weaknesses of these algorithms and policies in Section 2.3 and also discuss how well they generalise to the different bandit frameworks. Summary remarks follow in Section 2.4.

2.1 Bandit Frameworks

All bandit frameworks considered in the literature have the three key properties mentioned earlier: little or no prior knowledge of expected rewards, opaqueness, and action rewards that are noisy or change over time. There are several variations of this general principle, however, which are covered in this section. These variations include: the number of actions available (Section 2.1.1), the availability of any side information (Section 2.1.2), the length of game (Section 2.1.3), the method by which rewards are generated (Section 2.1.4) and dynamics driving the reward process (Section 2.1.5). Finally, in Section 2.1.6 we introduce Markov Decision Processes (MDPs) and discuss their relationship with bandit problems.

2.1.1 Number of Actions

The simplest version is the one-armed bandit problem, introduced in Chernoff (1967). In this formulation, the agent must select between an unknown “risky” action and a known “safe” action. The application considered in this paper is sequential clinical trials, where patients in a trial can be allocated the best known and tested drug or a new untested drug (which is the subject of the trial). The objective of the trial is not only to learn about the performance of the untested drug, but also to maximise the benefits (or minimise the symptoms) to all the tested patients – which creates the exploration-exploitation trade-off. The one-armed bandit problem has since been extensively studied in Kumar and Seidman (1981), Rosenberg et al. (2007), Sarkar (1991) and Woodroofe (1979).

The reward structure of the alternative "known" action does not require any learning, and hence the agent does not need to explore this action. Exploration in the one-armed bandit problem therefore involves selecting the unknown action when it is expected to perform worst given the knowledge the agent has. This is why the problem is known as the one-armed bandit (as opposed to two-armed). The two-armed bandit problem, where the reward structures of both actions are unknown *a priori*, has been studied in Berry (1972), Li and Zhang (1992) and Rothschild (1974) amongst others. The extension to multiple-arms where the agent must select between a finite set of unknown actions is commonly referred to as the multi-armed bandit problem or k-armed bandit problem (Sutton and Barto, 1998; Vermorel and Mohri, 2005). This problem has also been extended in Whittle (1981) to *arm acquiring bandits* where new actions become available during the decision making problem. Finally, the continuous bandit problem or the infinitely-armed bandit problem studies scenarios where the agent must select from a continuous variable or an infinite set of actions (Flaxman et al., 2005; Kleinberg, 2005). In this thesis, we consider bandit problems for a set number of finite actions and continue to make the distinction between one-armed and multi-armed problems. Arm acquiring bandits and the infinite-action problem are both of interest and form part of planned future work (see Chapter 8 for more details).

2.1.2 Side Information

In many real-world applications, agents are likely to have additional side information that is received throughout a decision making process. This side information can be interpreted as additional information (other than observed rewards) that is related to, but does not fully reveal, the expected rewards of future actions. This concept was first introduced to bandit problems in Woodroofe (1979) for the one-armed bandit problem. This paper advanced the work in Chernoff (1967) for sequential clinical trials, by arguing that available side information such as age, severity of disease or general physical status, which is specific to the patient, should be incorporated into the decision making process. Woodroofe goes further to argue that side information is likely to be present in all applications. As examples, weather conditions, time of day and terrain topography can inform the agent in a multi-target tracking problem or daily volume of transactions, exchange rates and the availability of alternative products can assist a bidder in on-line auctions.

The agent observes the side information in the form of a covariate (also referred to as a concomitant variable) prior to the decision made at time t. This covariate is linked to the reward of both the known and unknown action. In Woodroofe (1979), the covariate takes a scalar value $x(t)$ (from a known distribution X) and the reward of the unknown action a_1 is simply:

$$r_1(t) = x(t) + z(t), \tag{2.1}$$

where $z(t)$ is drawn from a known distribution Z (independent of X) with unknown but fixed mean μ. In this instantiation of the problem, the agent must learn the parameter μ to establish which action is expected to yield a larger reward given a particular value of $x(t)$.

This problem has become known as the *bandit with covariates problem* and has since been extensively studied in Clayton (1989), Pavlidis et al. (2008a), Pavlidis et al. (2008b), Sarkar (1991), Woodroofe (1982) and Yang and Zhu (2002) for both the one-armed and multi-armed cases. In other literature, the problem has been referred to as the *contextual bandit problem* (Beygelzimer et al., 2011; Langford and Zhang, 2007; Lu et al., 2010), or simply as the *bandit problem with side observations* (Pandey et al., 2007; Wang et al., 2005a,b). In each case the covariate and reward structure has been enhanced from that seen in Woodroofe (1979), to include high-dimensional covariates and complicated reward structures. This is covered in more detail in Section 2.1.4 where we discuss the many different reward processes that have been investigated.

We consider the bandit with covariates problem to be important in the context of sequential decision making problems as they characterise many real-world problems where agents face the exploration-exploitation trade-off. The framework can also be viewed as a generalisation of the standard bandit problem with no covariates (where the covariate can simply be set to be degenerate (Wang et al., 2005b)). For these reasons, we focus our research on the covariates setting in the following chapters.

2.1.3 Length of Game

The objective of the agent in bandit problems is to maximise reward over a sequence of pulls, or to minimise *regret*, where regret commonly refers to the difference between the reward of the action selected and the reward of the optimal action. In most early studies of bandit problems (Berry and Fristedt, 1985), the objective was to find a policy to maximise reward (or minimise regret) asymptotically, or in an infinite-length game. Regret measures are preferred for asymptotics, as the objective is to then design a zero-regret policy (whereas rewards will grow without bound). More recent studies however (Auer et al., 2002; Vermorel and Mohri, 2005), have attempted to design policies that are optimal or approximately optimal in finite-length games. The latter task is more difficult, as asymptotically optimal policies can yield poor results in finite time (Vermorel and Mohri, 2005). This is particularly relevant in real applications, where decision making problems are always likely to be finite in length. Furthermore, decision making environments will usually evolve over time and the decision problem will constantly change – such that asymptotic convergence is neither meaningful nor desirable. Given this, we focus our attention on finite-time problems in this thesis.

2.1.4 Reward Process

In the k-armed problem, the agent receives a reward $r_i(t)$ from action a_i at time t, where $i \in \{1, \dots, k\}$. The objective is to maximise the cumulative reward $R(T)$ in a game of length T where $t \in \{1, \dots, T\}$. Note that t takes discrete values and decisions occur at regular intervals – this is not a restrictive assumption in static bandit problems where reward processes do not change over time and the actual time between each decision is not important. The value of t can therefore be simply interpreted as the "iteration number".

The cumulative reward is sometimes discounted over time such that:

$$R(T) = \sum_{t=1}^{T} a^t r_i(t), \tag{2.2}$$

where $0 < a \leq 1$ is the discount factor ($a = 1$ corresponds to no discounting). Some action selection policies (as detailed in Section 2.2) require the discount factor to be strictly less than 1 for the policy to work. This is somewhat restrictive as several applications (such as clinical trials) consider undiscounted rewards (Berry and Fristedt, 1985). For this reason we impose no such condition on the policies and algorithms we build in this thesis and henceforth consider only undiscounted cumulative rewards.

The individual rewards for each action, $r_i(t)$, have been generated in a number of ways. In the simplest case rewards are generated using a Bernoulli distribution (Berry and Fristedt, 1985) or a normal distribution (Vermorel and Mohri, 2005). In these cases, the reward of each action a_i is an i.i.d. sample from a Bernoulli distribution with unknown success probability θ_i or a normal distribution with unknown mean and variance μ_i and σ_i^2 respectively. The agent has to learn these unknown parameters to identify the actions that yield the highest rewards. In fact, any probability distribution could be used, but analysis of i.i.d. rewards is typically restricted to Bernoulli and normal distributions (as these are commonplace in many applications and also easy to use in a Bayesian setting). This problem is typically referred to as the *stochastic* bandit problem (Vermorel and Mohri, 2005) or in some literature as a *stopping problem* (Woodroofe, 1979; Gittins, 1979), as the agent would prefer to perform all exploration first and must learn when to *stop* exploring and start exploiting.

An alternative mechanism for generating rewards is the *adversarial* or *non-stochastic* bandit problem (Auer et al., 1995, 2003) where rewards are set by an adversary *a priori* and can take any possible sequence of values, which can be highly unpredictable and constantly change the optimal action. Although this framework appears more flexible, analysis has been restricted to rewards that are bounded in the interval $[0, 1]$, which is not always practical. Moreover, the best policies in this framework attempt to maximise $\max_i \left(\sum_t r_i(t)\right)$ rather than $\sum_t \max_i \left(r_i(t)\right)$. Or in other words, the agent attempts to minimise regret against the action that performed best overall on average (rather than the regret against the best action at each time step t). In this work, we do not bound rewards and attempt to find the best action at each iteration (rather than the best action on average), we therefore do not consider the adversarial framework and generate our rewards stochastically.

Side information has also been incorporated into the reward process in a number of ways. A popular framework models the reward of each action as a linear function of a p-dimensional covariate $\boldsymbol{x}(t) = (x_1(t), \ldots, x_p(t))$ with added observation noise (Ginebra and Clayton, 1995; Pavlidis et al., 2008a,b; Yang and Zhu, 2002):

$$r_i(t) = \sum_{j=1}^{p} \alpha_{i,j} x_j(t) + \eta_i(t), \qquad \eta_i(t) \sim \mathcal{N}(0, \sigma_i^2), \tag{2.3}$$

where $x_1(t) = 1$, such that $\alpha_{i,1}$ becomes the intercept of the reward plane. The p-dimensional covariate $\boldsymbol{x}(t)$ is often assumed to be an i.i.d. draw from a multivariate normal distribution with unknown mean and covariance matrix. The agent has to learn the $k \times p$ matrix $\boldsymbol{\alpha}$ to partition the covariate space between regions where each action is optimal. For this reason, bandit problems with this characteristic are sometimes referred to as *allocation problems* (Woodroofe, 1979).

Another method is to model the relationship between the covariate and the rewards for each action using a set of finite hypotheses (Langford and Zhang, 2007; Wang et al., 2005b). In some cases, these hypotheses have been labelled as "experts" that provide advice on the best action (Auer et al., 2003; Beygelzimer et al., 2011). This setting allows for a richer set of reward functions to be considered, but restricts the agent to a finite set of parameter values. Moreover, existing analysis in this framework has been restricted to a 1-dimensional covariate and in some cases, rewards that are bounded in the interval $[0, 1]$ (Beygelzimer et al., 2011; Langford and Zhang, 2007). Variations of this setting include the *partial-label problem* (Kakade et al., 2008) and the *associative bandit problem* (Strehl et al., 2006), where the covariate can take high-dimensional values but the rewards for each action are further restricted to either 0 or 1.

High-dimensional covariates and unbounded rewards are likely to exist in several applications. For example in finance, side information (such as macro-economic measurements) is widely available and rewards are difficult to bound, particularly in times of economic instability. For these reasons, we use the linear reward and multivariate normal covariate setting as the test-bed in this thesis. At first glance, linear rewards and normal covariates may appear restrictive; however, non-linear reward functions can often be accurately approximated by a linear function (using an appropriate design of covariates) (Abe et al., 2003; Auer, 2003; Sutton and Barto, 1998) and multivariate normal distributions can accurately model several real-world data sources (Cox and Small, 1978). Although our algorithms can in fact also be applied to non-linear rewards and non-Gaussian covariates.

2.1.5 Dynamic Environments

In many real-world applications, the rewards received for an action will change over time. In the simplest case, many slot machines in casinos will be pre-programmed such that expected rewards incrementally grow as the agent plays (and loses) until the agent does eventually win a large reward. After this, the expected reward suddenly jumps down such that the agent is almost guaranteed to lose money if it continues to play (BettingCorp.com, 2009). In this way, the casino can guarantee a steady profit over long term play – in fact, this is where the “bandit” part of the term “one-armed bandit” originated (777OnlineSlots, 2011).

A good policy for selecting actions should therefore be time dependent and adapt to the type of dynamics observed. The optimal action may constantly change over time, or in the presence of side information, the regions of the covariate space for which each action is optimal will move or jump over time. The problem is no longer about convergence to an optimal action, but instead the agent needs to continuously adapt and respond to changes in the reward structure.

This type of dynamic reward process, which incorporates both drifts and jumps, has been implemented in the bandit with covariates framework by Pavlidis et al. (2008a). In this formulation, the coefficients of the linear reward function in Equation (2.3) each follow an ESTAR (Exponential Smooth Transition Autoregressive) process (Haggan and Ozaki, 1981):

$$\alpha_{i,j}(t) = \alpha_{i,j}^{\text{eq}} + \left(\alpha_{i,j}(t-1) - \alpha_{i,j}^{\text{eq}}\right) \exp\left(-\gamma(\alpha_{i,j}(t-1) - \alpha_{i,j}^{\text{eq}})^2\right) + \upsilon, \tag{2.4}$$

where $\alpha_{i,j}^{\text{eq}}$ is the equilibrium value of $\alpha_{i,j}$ and υ is zero-mean normally-distributed noise (with variance σ_υ^2). Coefficient values close to the equilibrium will change like a random walk, but values far from the equilibrium will become mean reverting and jump back to the equilibrium. The ESTAR model therefore allows for a "continuum" of regimes (Van Dijk et al., 2002). The parameter $\gamma \in (0, \infty)$ is a smoothness parameter which determines the balance between drift and mean-reversion (high values of γ allow no drift and as $\gamma \to \infty$ the process effectively becomes static).

This type of dynamics ensures that the decision making problem does not degenerate such that one action becomes globally optimal (as may naturally happen in a drifting system) but has the important effect of constantly changing the regions of the covariate space where each action is optimal. An effective policy would therefore constantly adapt to the problem with continuous exploration of all actions – rather than trying to converge to a specific partitioning of the covariate space. An alternative framework for considering dynamic rewards is the *restless bandit* problem (Whittle, 1988), where the state of all unselected actions change over time. This formulation, however, does not include side information and can therefore be seen as a degenerate version of the dynamic model in Pavlidis et al. (2008a).

We consider dynamic rewards to be an important generalisation of bandit problems, as they can capture many real world problems. For example, the price of financial assets will drift and jump over time, in response to various micro and macro-economic effects or targets will continuously adapt their evasive movements in a multi-target tracking problem. From henceforth, we refer to bandit problems with dynamic reward structures as *dynamic bandit problems* (as opposed to *static bandit problems*) and we study such problems in more detail in Chapter 5, where we consider new types of dynamic reward processes, previously not considered in a bandit setting.

2.1.6 Markov Decision Processes

Finally, we note that another well-studied class of sequential decision making problems are *Markov Decision Processes* (MDPs) (Bellman, 1957). In MDPs, the decision problem is in some state and transitions to new states are dependent on the past action and current state only, also known as the Markov Property. *Partially Observable Markov Decision Processes* (POMDPs) (Monahan, 1982) are generalised frameworks where the agent does not observe the current state, and instead has a probability distribution over all states.

MDPs and POMDPs have also been extended to situations where rewards are unknown, which then introduces the exploration-exploitation trade-off (Sutton and Barto, 1998). In this case, the classic multi-armed bandit problem is equivalent to a single-state MDP, whereas the inclusion of side-information can be viewed as an infinite-state MDP, where actions do not affect subsequent realisations of states. The dynamic bandit problem, however, can capture state transitions beyond those permitted by the Markov property and the agent is further not assumed to know these state transition probabilities, hence the MDP framework is not as general – it is for this key reason that we focus on bandit problems and not MDPs in this thesis. Nevertheless, we note that the policies and algorithms introduced in Section 2.2 for the bandit problem have also been widely applied to MDPs and POMDPs, in particular Q-learning which is introduced in more detail in Section 2.2.6.

2.2 Policies and Algorithms

In this section we detail the various policies and algorithms that have been constructed for bandit problems, many of which have been broadly applied to other sequential decision making and optimisation problems that require action exploration. We make a distinction between algorithms and policies such that algorithms are on-line methods for selecting actions, where an algorithm is executed at each iteration, and policies are off-line decision rules with all exploration parameters set in advance.

All policies and algorithms in the bandit problem require some form of initialisation. In most cases (Auer et al., 1995; Sutton and Barto, 1998) this requires selecting each action once to gain an unbiased estimate of the expected rewards. In other instances (Berry and Fristedt, 1985; Gittins, 1979) a Bayesian prior is attached to the reward of each action – but unless there are some particularly informative priors, this almost inevitably also leads to the agent selecting each action once at the beginning. The case of more actions than rounds was considered in Vermorel and Mohri (2005), and in this case the policies and algorithms were initialised by selecting two random actions and then estimating the reward of all other actions as being the average of the two selected actions. We note that the case of more actions than rounds ($k > T$) is interesting, but we restrict our attention to more rounds than actions ($T > k$) in this work – as this is more common in most bandit applications. Finally, in the presence of side information, a longer initialisation period is required to gain unbiased sample estimates. In the linear reward structure of Yang and Zhu (2002), each action must be first selected $p + 1$ times (where p is the dimension of the covariate), so that unbiased estimates of the reward coefficients $\alpha_{i,j}$ can be computed before any action selection policies are executed.

In this section we review the ϵ-greedy policy and its variants (Section 2.2.1), the SoftMax policy and probability matching variants (Section 2.2.2), interval estimation and UCB policies (Section 2.2.3), the Gittins Indices (Section 2.2.4), the POKER algorithm (Section 2.2.5) and reinforcement learning methods (Section 2.2.6). We discuss the applicability of these approaches throughout, but then compare their strengths and weaknesses in more detail in Section 2.3.

2.2.1 ϵ-greedy and Variants

The simplest policy in any bandit setting is a *greedy* policy (Sutton and Barto, 1998) which selects the action that has yielded the highest reward thus far. This is a pure exploitation policy – as exploration involves selecting an action that is not expected to yield the highest instantaneous reward, but might improve cumulative reward in the long-term. The lack of exploration and consideration of long-term rewards has lead this policy to be referred to as a *myopic* policy in some literature (Woodroofe, 1979) and also a *one-step look-ahead* policy in others (Gittins, 1979).

This policy has in fact been shown to perform optimally (Macready and Wolpert, 1998; Woodroofe, 1979), or near-optimally (Pavlidis et al., 2008b) for certain bandit frameworks. This can be attributed to a number of possible factors. First of all, the initialisation period can perform sufficient exploration for the agent to immediately learn the optimal action before the action selection policy is even implemented. This was discovered, for example, in the linear bandit with covariates framework in Pavlidis et al. (2008a), particularly for a high-dimensional covariate where the initialisation period is long. Secondly, the observation noise may be sufficiently low such that exploration is not required (Pavlidis et al., 2008b) and finally, the number of actions is small or reward structure is very simple (Macready and Wolpert, 1998; Woodroofe, 1979) and the problem is easy enough to learn from the rewards received through pure exploitation. The greedy policy has also been found to be optimal for many other sequential decision making and optimisation problems (Lew, 2006).

In most bandit settings however, the greedy policy will perform far from optimally as the agent is performing insufficient exploration and will often converge to the repeated selection of a suboptimal action (Sutton and Barto, 1998). In fact, the greedy policy is mainly used for its computational efficiency and simplicity (Lew, 2006). Unlike most other policies, it does not require any parameters to be set *a priori*.

Exploration can be incorporated into a policy in a number of ways. One of the simplest methods is by using an ϵ-*greedy* policy (Watkins, 1989), which is a straightforward extension of greedy selection. At each iteration, the agent adopts a greedy policy with probability $1 - \epsilon$ (where $0 \leq \epsilon \leq 1$) and selects a random action with probability ϵ. The value of ϵ is selected *a priori* and can be interpreted as an exploration parameter – higher values correspond to more exploration and vice-versa. Note that the greedy policy is recovered by setting $\epsilon = 0$. The explorative component is random and ensures every possible action is continuously explored – such that convergence to a suboptimal action is not possible. In a static reward setting this may seem undesirable as an agent should stop exploration once the optimal action has been learnt. Nevertheless, this policy has been found to perform well in finite time, in a number of empirical studies (Sutton and Barto, 1998; Vermorel and Mohri, 2005). Furthermore, this policy is suitable for dynamic problems where an agent should continuously explore to then adapt to any changes in the system (Pavlidis et al., 2010). In fact, ϵ-greedy methods for exploration are commonly used across many sequential decision making and optimisation problems (Neumann et al., 2007; Shani et al., 2005; Sutton and Barto, 1998) and we study this policy in more detail throughout this thesis.

A natural variant of the ϵ-greedy policy is an ϵ-*first* policy (Even-Dar et al., 2002), which performs all exploration at the beginning of the problem. Specifically, in a game of length T, the agent explores randomly for the first ϵT iterations and then selects greedily for the remaining $(1 - \epsilon)T$ iterations. This policy ensures that all exploration is performed at the beginning when the agent has the highest levels of uncertainty regarding the expected rewards of each action (in a static problem). In fact, for the same value of ϵ, an ϵ-first policy will on average perform better than an ϵ-greedy policy in a static problem, as shown empirically in Vermorel and Mohri (2005). This is because the benefits of exploration, through increased learning, are experienced over more iterations (as the exploration is performed earlier) whilst the short-term costs of exploration are the same in both cases. In other words, the "greedy" part is performed better using an ϵ-first policy, as the agent has a better idea of which action is optimal. In fact, this policy was found to perform best in the empirical evaluation by Vermorel and Mohri (2005). An ϵ-first policy however, requires knowledge of the game-length T (ϵ-greedy does not), which is not always available in certain applications. Furthermore, this policy is not usually suitable in a dynamic reward setting where exploration should not be performed all at once.

Another variant, which combines the above approaches, is an ϵ-*decreasing* policy (Auer et al., 2002), where the agent explores with probability $\min\left(1, \frac{\epsilon_0}{t}\right)$ at time t (where $\epsilon_0 \geq 0$) and otherwise selects greedily. Alternatively, in a variant called *GreedyMix* (Cesa-Bianchi and Fischer, 1998), the exploration is performed with probability $\min\left(1, \frac{\log(t)\epsilon_0}{t}\right)$. In both these cases, the rate of exploration decreases exponentially to zero, making these policies particularly suitable for problems that are static and where the game-length T is unknown (such that an ϵ-first policy is unsuitable). Moreover, for certain classes of problems, these policies can be shown to have optimal asymptotic properties and strong finite-time performance (Auer et al., 2002).

The ϵ-greedy, ϵ-first and ϵ-decreasing policies are sometimes collectively referred to as ϵ-greedy policies (Sutton and Barto, 1998), but to avoid ambiguity, in this thesis we refer to them separately with the above given names. In other literature, these policies are sometimes referred to as *semi-uniform* policies as the agent forms a binary distinction between greedy exploitation and random (uniform) exploration (Vermorel and Mohri, 2005). We consider these approaches to be the simplest but most fundamental exploration policies in sequential decision making, and pay particular attention to the ϵ-greedy and ϵ-first policies in this thesis, not least because they have been found to perform consistently well across a range of empirical studies (Auer et al., 2002; Pavlidis et al., 2008b,a; Sutton and Barto, 1998; Vermorel and Mohri, 2005). Note that ϵ-first performs best in static problems, but ϵ-greedy has particular application in a dynamic reward setting (see Chapter 5), or in certain multi-agent settings (Chapter 7). The main weakness of these policies, however, is that exploration is performed randomly which means that actions that are known to be suboptimal will continue to be selected for exploration. We compare and contrast ϵ-greedy and its variants against other policies in more detail in Section 2.3.

2.2.2 SoftMax and Probability Matching Variants

An alternative approach to randomly exploring actions is to weight the probability of selecting each action such that actions that are expected to perform better are selected with higher probability. This concept was first introduced in Luce (1959), where a *SoftMax* policy was proposed. Each action a_i is chosen with probability $p_i = e^{\bar{r}_i/\tau} / \sum_{j=1}^{k} e^{\bar{r}_j/\tau}$, where $\bar{r}_i$ is the mean reward of this action thus far. The parameter $\tau > 0$ determines the degree of exploration performed where large values correspond to more equal weighting between the actions and hence more exploration. Conversely, as $\tau \to 0$ the policy approaches a greedy policy. For a suitable value of τ, this policy ensures that actions that are unlikely to be optimal are rarely selected. The overall degree of exploration, however, does not change over time, which yields poor asymptotic properties. For these reasons, the temperature parameter is sometimes decreased over time at rate $1/t$ or $\log(t)/t$ (in a similar vein to ϵ-decreasing) – the latter case sometimes being referred to as a *SoftMix* policy (Cesa-Bianchi and Fischer, 1998), which was shown to be the theoretically appropriate choice for certain classes of problems in Singh et al. (2000). All of these weighted probability exploration policies are sometimes collectively referred to as *probability matching* policies (Vermorel and Mohri, 2005).

Another popular method, particularly in the non-stochastic adversarial framework, is the *Exp3* policy (Exponential weight algorithm for exploration and exploitation), designed in (Auer et al., 2003). In this instance, the probability of selecting each action at time t is weighted by:

$$p_i(t) = (1-\gamma)\frac{w_i(t)}{\sum_{j=1}^{k} w_j(t)} + \frac{\gamma}{k}, \tag{2.5}$$

where $w_i(t+1) = w_i(t)\exp\left(\gamma \frac{r_i(t)}{p_i(t)k}\right)$ (updated only if action a_i is selected). The parameter $\gamma \in [0,1]$ determines the degree of exploration, where $\gamma = 1$ yields pure random exploration. This policy ensures that actions with high rewards lead to higher probability weightings – where the increase in weighting is largest if an action with previously low probability weighting (and hence low prior rewards) then observes a high reward. Numerous variants are also proposed, including decreasing γ over time to achieve asymptotic optimality – but these policies are beyond the scope of this thesis, particularly as they are designed for the non-stochastic adversarial framework.

In this thesis, we choose not to focus on SoftMax policies and their variants as they do not perform well in finite time (Vermorel and Mohri, 2005), and are mainly designed for their guaranteed asymptotic properties (in some specific problems). For example, Exp3 policies are only asymptotically optimal for rewards bounded in the interval $[0, 1]$. Moreover, such methods have not always been extended to include side information or dynamics. This is discussed further in Section 2.3.

2.2.3 Interval Estimation and UCB

Another approach to exploration is to be "optimistic in the face of uncertainty" (a term introduced in Kaelbling (1993)) and select actions using a combination of expected reward values and the uncertainty of the reward estimates – such that actions with high uncertainty are selected more often. This approach is sometimes referred to as using "exploration bonuses" (Dearden, 2000; Meuleau and Bourgine, 1999). This idea was first introduced in Kaelbling (1993) where the upper bound of the $100(1-\beta)\%$ confidence interval is calculated for the expected reward of each action, and then the action with the highest value (sometimes referred to as an "optimistic reward estimate") is selected. This action selection policy subsequently became known as an *interval estimation* policy. Higher β values correspond to less exploration and as $\beta \to 1$, action selection approaches that of a greedy policy. The rewards are typically assumed to follow a normal distribution such that the confidence bounds can be easily estimated on-line (Sutton and Barto, 1998). This policy has also been extended to the covariates setting in Pavlidis et al. (2008b) for a multivariate normal covariate and to dynamic problems in Li et al. (2010) for a web-advertising application (where user preferences change over time).

A further enhanced approach, which does not require distributional assumptions regarding the reward processes, was developed in Auer et al. (2002). Specifically, several *UCB* (Upper Confidence Bound) policies were designed for rewards bounded in the interval $[0, 1]$ (and an additional policy for normally distributed rewards). These policies were designed to achieve bounded regret in finite time (as well as having optimal asymptotic properties). In more detail, the first proposed policy *UCB1*, selects the action a_i that maximises:

$$\bar{r}_i + \sqrt{\frac{2\ln t}{n_i}}, \tag{2.6}$$

where n_i is the number of times that action a_i has been selected thus far. This parameter free method has optimal asymptotic properties for rewards in the interval $[0, 1]$ but was found to perform poorly in finite-time problems. To improve on finite-time performance however, Auer *et al.* modify this policy and instead select the action that maximises:

$$\bar{r}_i + \sqrt{\frac{\ln t}{n_i}\min\left(\frac{1}{4}, V_i(n_i)\right)}, \tag{2.7}$$

where,

$$V_i(n_i) = \left(\frac{1}{n_i}\sum_{\tau=1}^{n_i} r_i^2\left(t_i(\tau)\right)\right) - \bar{r}_i^2 + \sqrt{\frac{2\ln t}{n_i}}, \tag{2.8}$$

where t_i is the sequence of time-steps for which action a_i has been selected. This policy, referred to as *UCB1-Tuned*, performs well with rewards bounded in the interval $[0, 1]$, but the authors are not able to provide any finite time or asymptotic performance bounds. Auer *et al.* then propose a policy *UCB2* which performs well in finite-time analysis and can be bounded. This policy chooses an action and selects this for an “epoch” (an interval in time). The action selected is the one that maximises:

$$\bar{r}_i + \sqrt{\frac{(1+\kappa)\ln\left(et/\lceil(1+\kappa)^{e_i}\rceil\right)}{2\lceil(1+\kappa)^{e_i}\rceil}}, \tag{2.9}$$

and is selected exactly $\lceil(1+\kappa)^{e_i+1}\rceil + \lceil(1+\kappa)^{e_i}\rceil$ times, where e_i is the number of epochs selected with action i thus far (and e is Euler's number). We note that this policy introduces a parameter $0 < \kappa < 1$, which must be set *a priori* and affects the rate of exploration. Finally, Auer *et al.* construct a policy for normally distributed rewards, *UCB1-Normal*, which selects the action that maximises:

$$\bar{r}_i + \sqrt{\frac{16\left(\sum_{\tau=1}^{n_i} r_i^2\left(t_i(\tau)\right) - n_i\bar{r}_i^2\right)\ln(t-1)}{n_i(n_i-1)}}. \tag{2.10}$$

All of these policies were found to generally perform worse than a well-tuned ϵ-decreasing policy in an empirical evaluation. Although, it is noted that the performance of ϵ-decreasing degrades rapidly with a poorly tuned ϵ – the performance of UCB2 for example, is much less sensitive to the κ parameter. Nevertheless, all UCB policies are designed to work with specific reward structures and cannot be easily extended to include covariates or dynamics. We therefore do not use these policies within our algorithms in this thesis but nevertheless refer to them frequently and compare performance in relevant empirical studies in Chapter 4.

Finally, we note that the UCB approach of Auer et al. (2002) has also been extended to other sequential decision making problems not considered in this thesis such as Markov Decision Processes (MDPs) and game-tree searches (used to solve games such as chess, Go and backgammon) in Kocsis and Szepesvári (2006) where a *UCT* (Upper Confidence bounds for Trees) algorithm was constructed, based on Monte Carlo planning.

2.2.4 Gittins Indices

An optimal policy has been found for a discounted stochastic bandit problem in the classic paper by Gittins (1979). The framework assumes Bernoulli distributed rewards for each action a_i, with unknown success probability θ_i. At time $t = 0$, the agent has a prior probability density $\theta_i \sim$ beta$(\alpha_i(0), \beta_i(0))$. The posterior distribution for θ_i at time t is therefore also a beta distribution with parameters $(\alpha_i(t), \beta_i(t)) = (\alpha_i(0) + s_i, \beta_i(0) + n_i - s_i)$, where s_i is the number of successes (rewards of 1) received from action a_i up to time t. At each time-step t the agent calculates an index for each action a_i which is given by:

$$\nu^T(\alpha_i(t), \beta_i(t), a) = \sup_{\tau \leq T} \frac{R_\tau(\alpha_i(t), \beta_i(t), a)}{\sum_{t=0}^{\tau-1} a^t}, \tag{2.11}$$

for a discount factor of $0 < a < 1$, where

$$R_\tau(\alpha_i(t), \beta_i(t), a) = \mathrm{E} \sum_{t=0}^{\tau-1} a^t r_i(t), \tag{2.12}$$

is the expected total discounted reward from τ trials on action a_i, which can be calculated iteratively for $\tau = 1, 2, \ldots,$ by calculating conditional expectations of the beta posterior distributions (see Gittins (1979) for the full algorithm). The optimal value of τ is referred to as the “optimal stopping time”.

The action with the highest $\nu^T(\alpha_i(t), \beta_i(t), a)$ index value is selected at each iteration. These indices were referred to as *dynamic allocation indices* (DAIs) in Gittins’ papers but are now commonly known as the *Gittins indices*. The expected probability of success for each action a_i at time t is the mean of the beta$(\alpha_i(t), \beta_i(t))$ posterior distributions, namely $\alpha_i(t)/(\alpha_i(t) + \beta_i(t))$, which serve as minimum bounds for the Gittins indices (as they are equal to $\nu^{T=1}(\alpha_i(t), \beta_i(t), a)$). For small values of a (i.e. a large discount factor), the Gittins indices will be close to these values. They will however be considerably larger for high values of a where the optimal stopping time is large.

Computation of the Gittins indices is difficult, particularly as the optimal stopping time τ becomes large. For these reasons Gittins and Jones (1979) provides pre-computed index values for a grid of α and β integer values ranging from 1 to 10 – this however is with a 0.75 discount factor, as higher discount values require longer stopping times. In fact, as $a \to 1$ the indices all tend towards 1 and precisely determining the optimal action can require extremely heavy computation. In addition, large values of τ require α and β values far greater than 10. The use of a beta prior for Bernoulli rewards is also significant, as this prior is conjugate and hence allows an analytic representation of the posterior and avoids numerical approximation techniques. The Gittins indices were also extended to normally distributed rewards with unknown mean μ_i but with known variance σ_i^2. The prior distribution for μ_i is also normal and hence conjugate, but the indices are even more complicated to calculate (see Section 9 in Gittins (1979) for more details).

The optimality of the Gittins indices for the discounted bandit problem has been proved several times, see for example (Ishikida and Varaiya, 1994; Tsitsiklis, 1994; Weber, 1992; Whittle, 1980). Gittins indices have also been shown to be optimal for several variations of the multi-armed bandit problem, for example with arm acquiring bandits (Whittle, 1981) and restless bandits (Whittle, 1988); but have also been shown to not exist or be suboptimal for other variations, such as bandits with switching costs (Banks and Sundaram, 1994; Van Oyen et al., 1992) and bandits with multiple plays (Ishikida, 1992; Pandelis and Teneketzis, 1999). Analytical representations of Gittins indices in bandit problems are typically unattainable, however such solutions were found in a continuous time bandit where rewards evolve according to a Brownian motion (Karatzas, 1984), with unknown drift but known volatility.

The Gittins indices do not converge in the undiscounted setting and have only been computed for certain reward distributions and parameter priors. Given this, and the fact that no extensions to covariates or dynamic rewards exist, we do not consider the Gittins indices in this thesis. Nevertheless, our novel algorithm for the bandit problem has similarities with the Gittins approach, in that we attempt to iteratively calculate the long-term value of selecting each action individually. We discuss this analogy in more detail in Chapter 4.

In addition to Gittins indices, there are other Bayesian approaches which are much less computationally demanding. Thompson sampling (Thompson, 1933) samples a value from the posterior distribution for the expected reward of each action and then selects the action with the highest sampled value. Although this method was originally constructed for Bernoulli bandits, and is focused towards achieving desirable asymptotic properties, Thompson sampling has been implemented by Graepel et al. (2010) for web advertising with Microsoft's Bing search engine. Bayesian Reinforcement Learning (Strens, 2000) uses a similar approach of estimating the posterior distribution of the unknown parameters, but then uses these distributions to sample a hypothesis of the decision problem faced. The optimal action can then be approximated by optimising over this hypothesis using techniques such as Q-learning (see Section 2.2.6). These Bayesian sampling methods are useful but not considered further in this thesis as they are not designed for the finite-time or dynamic problems that we consider in this thesis – but more for guaranteeing convergence to the optimal action in an infinite-time static decision making problem.

2.2.5 POKER

The “Price Of Knowledge and Estimated Reward” (POKER) algorithm (Vermorel and Mohri, 2005) uses similar principles to interval estimation and UCB methods in that optimistic estimates are given to under-explored actions. At each time-step t, an index p_i is calculated for each action:

$$p_i = \bar{r}_i + \delta_\mu \mathrm{Pr}[\mu_i \geq \bar{r}^* + \delta_\mu](T - t), \tag{2.13}$$

where μ_i is the unknown true mean reward of action a_i, $\bar{r}^*$ is the highest observed mean reward of all actions and δ_μ is the expected reward mean improvement. The action with the highest index value p_i is selected. δ_μ is multiplied by the probability of a reward improvement and the horizon $T - t$. This can be collectively interpreted as an estimate of the “knowledge acquired” if action a_i is selected. δ_μ is not known to the agent and is estimated by $\delta_\mu = (\bar{r}_{i_1} - \bar{r}_{i_{\sqrt{q}}})/\sqrt{q}$, where $\bar{r}_{i_1} \geq \ldots \geq \bar{r}_{i_q}$ are the ordered reward means of all the q actions selected thus far. Note that q can be less than the total number of actions k as this paper studies scenarios where $T < k$ and not every action is selected during initialisation (as discussed on page 30 earlier).

The probability of a reward improvement $\mathrm{Pr}[\mu_i \geq \bar{r}^* + \delta_\mu]$ is also approximated by:

$$\Phi_{\bar{r}_i, \frac{\bar{\sigma}_i}{\sqrt{n_i}}}(\bar{r}_i - \bar{r}^* - \delta_\mu) = \int_{\bar{r}^* + \delta_\mu}^{\infty} \mathcal{N}\left(x, \bar{r}_i, \frac{\bar{\sigma}_i}{\sqrt{n_i}}\right) dx. \tag{2.14}$$

This approximation assumes that the unknown parameter μ_i is normally distributed and centred around the mean estimate $\bar{r}_i$ with standard deviation proportional to the inverse square root of the number of times action a_i has been selected. This approximation is exact if the rewards themselves follow a normal distribution and otherwise converges to the true distribution (as n_i increases) under the central limit theorem. There are no finite-time theoretical guarantees on the POKER algorithm (in fact we show in Chapter 4 that the algorithm can also perform poorly in simulation studies). The authors do however provide a proof showing that the algorithm is zero-regret asymptotically, under certain assumptions regarding the reward generating mechanism.

The POKER algorithm is in fact intrinsically an on-line algorithm, with no fixed exploration parameter, and is therefore an alternative approach to the work presented in Chapter 4, where we develop methods for controlling exploration on-line. This algorithm, however, has no natural extension to incorporate side information (or dynamics), and hence cannot be generalised to the bandit with covariates problem. Furthermore, the choice of δ_μ is not a principled approach, and changing this value will scale the amount of exploration performed accordingly (in the same way that β controls the degree of optimism in the interval estimation policy). The choice is motivated by the authors from the fact that it performs well empirically and is suitable for problems with a large number of actions. Nevertheless, despite these restrictions, we include the POKER algorithm in various simulations as a benchmark for our algorithms, in the standard bandit problem with no side information.

2.2.6 Reinforcement Learning

Several other policies and algorithms exist for balancing the exploration-exploitation trade-off in sequential decision making and bandit problems. Many of these are only for specially designed frameworks that are not relevant to this thesis, but some are general enough to encompass a wide range of sequential decision making problems. In this section we give an overview of these policies/algorithms, which are all reinforcement learning type approaches.

A popular reinforcement learning approach is to use *Q-learning* (Watkins, 1989) to learn the expected rewards from each action over time, such that:

$$Q_i(t) = Q_i(t-1) + \lambda(r_i(t) - Q_i(t-1)), \tag{2.15}$$

where $Q_i(t)$ are referred to as the *Q-values* for each action and $0 < \lambda \leq 1$ is the learning rate. Note that this is a simplified version of the full Q-learning algorithm which can be used in settings with multiple states (such as those considered in MDPs, see Section 2.1.6), but this extension is beyond the scope of this thesis. Returning to the single-state formulation given in Equation (2.15), if we set $\lambda = 1/(n_i)$ then we recover $Q_i(t) = \bar{r}_i(t)$, which is the average reward from action a_i thus far. Setting λ to be a constant value allows the Q-values to be an adaptive estimate of the mean reward of each action, which is useful for dynamic systems. Selecting the action with the highest Q-value, however, can yield similar results to that of a greedy policy, as there is no explicit action exploration. This is unless Q-learning is combined with using *optimistic initial values* (Sutton and Barto, 1998), i.e. setting high values for $Q_i(0)$. This causes all actions to be explored at the beginning until their Q-values converge to their true values – this will however only encourage exploration at the beginning of a game and is therefore only suitable for exploration in a static system. Furthermore, there are two obvious issues with setting these optimistic initial values. First, the agent may not know what rewards to expect and hence does not know what range of initial values are optimistic and secondly, even with this knowledge, the degree of optimism influences the total amount of exploration performed before converging to a particular action. This algorithm is consequently more useful for guaranteeing asymptotic convergence, rather than maximising finite-time performance. We note that the multi-agent extension of this algorithm, known as R-Max, was constructed in Brafman and Tennenholtz (2003). Essentially, R-Max initialises optimistically by allocating each state and joint action the maximum possible reward (which is assumed to be bounded and known). This is covered in more detail in Section 6.2 where we provide a more detailed background on multi-agent sequential decision making.

Alternative reinforcement learning approaches include *reinforcement compar-*

ison methods (Sutton, 1984) and *pursuit* methods (Thathachar and Sastry, 1985). The former adjusts the probability of selecting an action dependent on whether an observed reward is less than or greater than some *reference reward*, such that high rewards increase the probability (and vice-versa). A natural choice for the reference reward is the mean from past observations (although optimistic Q-values could also be used). As an example, the probability p_i of selecting each action can be determined using a SoftMax policy $p_i(t) = e^{\tau_i(t)} / \sum_{j=1}^{k} e^{\tau_j(t)}$ where $\tau_i(t+1) = \tau_i(t) + \beta(r_i(t) - Q_i(t))$. This algorithm was shown to work well in finite time (Sutton and Barto, 1998) with optimistic Q-values for suitably tuned parameters. Although we note that the inclusion of the step-size parameter β (together with the Q-learning step-size λ and setting the optimistic initial values) over-parameterises the problem, and complicates the implementation of this algorithm in many practical domains.

Pursuit methods are similar, but rather than changing the probability of selecting each action dependent on their relative performance, the algorithm actively "pursues" the greedy action. Specifically, the probability of selecting the greedy action $p_{i^*}(t)$ is adjusted by $p_{i^*}(t+1) = p_{i^*}(t) + \beta(1 - p_{i^*}(t))$ and all other actions are adjusted by $p_i(t+1) = p_i(t) - \beta p_i(t)$, which ensures the probabilities always sum to 1. The initial probabilities $p_i(0)$ are usually set to be $1/k$. This algorithm is more suitable than reinforcement comparison for the problems encountered in this thesis, as there is only one parameter that controls exploration, but we do not consider this algorithm any further as it does not extend to the more enhanced frameworks considered in this thesis.

2.3 Evaluation and Comparison

In Section 2.1 we outlined the key variations between the different bandit frameworks studied in the existing literature and in Section 2.2 we highlighted the key policies and algorithms constructed for these various frameworks. In this section we bring this work together and analyse in detail the robustness and flexibility of each policy and algorithm in the context of how well they generalise to different types of bandit problems. To keep this evaluation succinct and easy to follow, we detail this analysis in Table 2.1, where a grid of different policies/algorithms and bandit frameworks are cross-compared. The table shows that the ϵ-greedy policy[1] is the most flexible: the policy is computationally efficient, can incorporate covariates, does not require knowledge of T, does not require bounded rewards or any distributional assumptions, and can be applied to dynamic reward environments.

The other policies and algorithms have significant shortfalls – most cannot be extended to dynamics and covariates and several require bounded rewards. Despite these increased requirements, the simpler and more general ϵ-greedy approach (and its variants) have been consistently found to perform best across a range of empirical studies (Auer et al., 2002; Pavlidis et al., 2008b,a; Vermorel and Mohri, 2005), which is a significant finding that warrants further investigation. Furthermore, there are two obvious shortfalls of the ϵ-greedy policy: first, the requirement of an exploration parameter (in contrast to POKER and UCB1) and secondly, the fact that actions that are sure to be suboptimal are repeatedly selected through random exploration. The first of these issues is particularly important. In many applications, setting an exploration parameter optimally *a priori* is not feasible and, as noted in Auer et al. (2002), performance can degrade rapidly with a poorly tuned ϵ.

As mentioned earlier, the ϵ-first policy will outperform an ϵ-greedy policy in static problems. Despite the requirement of knowing the length of the game T, we also investigate this policy (together with ϵ-greedy) in much more detail in the chapters that follow. In addition, we include the POKER and UCB algorithms as benchmarks for the bandit with no covariates problem.

[1]and of course the greedy policy, which is a special case of ϵ-greedy.

Table 2.1: The applicability of policies and algorithms for the exploration-exploitation trade-off in various bandit problems

Policy/Algorithm	No. of Param	Comp. complex	Covariates	T not required	Unbounded rewards	No distbn. assumptions	Dynamics
Greedy	0	$\mathcal{O}(T)$	✓	✓	✓	✓	✓
ϵ-greedy	1	$\mathcal{O}(T)$	✓	✓	✓	✓	✓
ϵ-first	1	$\mathcal{O}(T)$	✓	✗	✓	✓	✗
ϵ-decreasing	1	$\mathcal{O}(T)$	✓	✓	✓	✓	✗
SoftMax	1	$\mathcal{O}(kT)$	✓	✓	✓	✓	✓[a]
Exp3	1	$\mathcal{O}(kT)$	✗	✓	✗	✓	✓
Interval Estimation	1	$\mathcal{O}(T)$	✓	✓	✓	✗	✓
UCB1/UCB1-Tuned	0	$\mathcal{O}(kT)$	✗	✓	✗	✓	✗
UCB2	1	$\mathcal{O}(kT)$	✗	✓	✗	✓	✗
UCB1-Normal	0	$\mathcal{O}(kT)$	✗	✓	✓	✗	✗
Gittins Indices	0	$\mathcal{O}(k^3T^3)$[b]	✗	✗	✓	✗	✗
POKER	0	$\mathcal{O}(kT)$	✗	✗	✓	✗	✗
Q-learning	1[c]	$\mathcal{O}(T)$	✗	✓	✓	✓	✓
Reinforcement Comparison	2[d]	$\mathcal{O}(kT)$	✗	✓	✓	✓	✓
Pursuit Methods	1	$\mathcal{O}(kT)$	✗	✓	✓	✓	✓[a]

[a]If used with Q-values rather than mean rewards

[b]as proven in Nino-Mora (2010)

[c]$k+1$ with optimistic initial estimates

[d]$k+2$ with Q-learning using optimistic initial estimates

2.4 Summary

In this chapter we have reviewed the multi-armed bandit problem and the various policies and algorithms that have been constructed for balancing the exploration-exploitation trade-off. These are the state-of-the-art methods and have been applied to many sequential decision making problems outside of the bandit framework. Nevertheless, this is the first comprehensive review that encompasses the full range of bandit problem literature and the first attempt to analyse the applicability of various policies and algorithms across the various types of bandit problems.

Our critical analysis has shown that the ϵ-greedy and ϵ-first policies are the best-performing and most flexible approaches to exploration-exploitation, particularly for the important generalisations of the bandit framework that include side information and dynamic rewards. These policies have significant shortfalls though, such as the requirement for an exploration parameter that needs to be set *a priori*. Given this, in this thesis we construct a novel on-line algorithm, ϵ-ADAPT, which removes the need for an exploration parameter and learns which action to explore over time – we extend this algorithm to both covariates (Chapter 4) and dynamics (Chapter 5), such that ϵ-ADAPT is more generalisable than the UCB and POKER algorithms. In addition, in the next chapter we conduct a theoretical analysis of a basic one-armed problem to demonstrate how we can find the optimal ϵ values off-line for both ϵ-greedy and ϵ-first.

Chapter 3

Optimal Exploration Rates in Bandit Problems

In the previous chapter we performed a critical analysis of the existing policies and algorithms for various multi-armed bandit problems. We found that the ϵ-greedy policy is the simplest and most robust approach across the range of bandit frameworks that have been considered. Furthermore, this policy (together with ϵ-first) has been found to perform best in a number of empirical studies, however this is only for a well chosen value of ϵ. Indeed, performance can degrade rapidly if this parameter is chosen incorrectly.

For these reasons, in this chapter we theoretically examine the performance of the ϵ-greedy and ϵ-first policies for a simple bandit framework, namely the one-armed bandit with covariates problem (with a one-dimensional covariate). We investigate this framework as it is the most basic formulation of the exploration-exploitation trade-off, where there is only one action to explore and one covariate value to consider. The contributions of this chapter therefore serve as an important first step towards theoretically finding optimal parameters off-line and also understanding how exploration-exploitation needs to be balanced in finite-time problems. Furthermore, we use these theoretical findings to construct algorithms for adapting exploration on-line which removes the need for an *a priori* fixed exploration parameter.

In more detail, we present a novel approach for reasoning about the expected reward of action selection policies, by modelling the distribution of estimated parameters in the reward function. This helps us find the *probability of error*; that is, the probability that the agent selects the action with lower expected reward when trying to select the best action, given a specific policy. This measure is important because it helps us find the expected reward when the agent is selecting greedily between actions. The probability of error is therefore crucial to finding the expected reward (in finite time) of any policy or algorithm that exploits the covariate values to select between actions, in this case the ϵ-greedy and ϵ-first policies.

In a one-armed bandit problem, there is only one “unknown action”. Explorative policies will select this action more often as this helps the agent to learn about the expected rewards more quickly, thus reducing the probability of error – this is the *benefit of exploration*. Conversely, such policies have an attributed *cost of exploration*, as the agent might be selecting the action with lower expected instantaneous reward. In the one-dimensional covariate setting considered in this chapter, we can explicitly calculate this benefit and cost of exploration and hence capture the exploration-exploitation trade-off in the same currency. We can then find the expected cumulative reward of ϵ-greedy and ϵ-first for finite-time problems, and hence reason about their optimal tuning.

In this chapter we prove that in the one-dimensional setting, the expected reward of the ϵ-greedy policy is maximised by $\epsilon = 0$ irrespective of the length of the game (T) and all other parameters. This means that, on average, a greedy policy will outperform any ϵ-greedy policy in finite time. This result is in line with the infinite-time statements proved in Woodroofe (1979) and Sarkar (1991), where it was proved that the greedy policy is asymptotically optimal (for a one-dimensional covariate using slightly more generalised reward functions). Moreover, contrary to the findings of finite-time analyses of multi-armed bandits (Auer et al., 2002; Vermorel and Mohri, 2005; Pavlidis et al., 2008b), we have proved the interesting result that the ϵ-greedy policy (with $\epsilon > 0$) is a suboptimal policy for the one-armed bandit problem considered here. We discuss the implications of these findings more throughout this thesis.

For the ϵ-first policy, however, we show that the optimal value of ϵ will be non-zero for certain game-lengths T. In particular, we find the optimal value of ϵ numerically and present results to show its dependence on T. The significance of this result, is that a well-tuned ϵ-first policy will, on average, outperform the greedy policy. This motivates the construction of an on-line algorithm that can learn the optimal rate of exploration over time – we construct such an algorithm for the one-armed problem considered here using the theoretical properties of ϵ-first. We then use the ideas from this on-line approach to construct an algorithm for adapting exploration on-line for more general multi-armed bandit problems which can be used with high-dimensional covariates (Chapter 4) and also with dynamic rewards (Chapter 5).

This chapter is structured as follows. In Section 3.1 we introduce the one-armed bandit with covariates framework. In Section 3.2 we model the distribution of estimated parameters in the reward function and use this to find the probability of error over time. In Section 3.3 we derive the expected reward of the ϵ-greedy policy and prove that this is maximised with $\epsilon = 0$. In Section 3.4 we derive the expected reward of the ϵ-first policy and show numerically that non-zero values of ϵ can be optimal. Finally, in Section 3.5, we use these theoretical findings to construct an on-line algorithm that removes the need for an *a priori* fixed exploration parameter. Summary remarks follow in Section 3.6.

3.1 The One-Armed Bandit with Covariates Framework

An agent is faced with a one-armed bandit problem and must choose at time $t = 1, \ldots, T$ between action a_1 with unknown expected reward and action a_2 with known expected reward. The agent only receives a reward from the action that is selected, which is a function of an observed covariate at time t denoted $x(t)$. We consider the linear reward structure used in Ginebra and Clayton (1995), Yang and Zhu (2002) and Pavlidis et al. (2008a,b) (see Equation (2.3) in Section 2.1.4), simplified to a one-dimensional covariate, such that:

$$\begin{aligned} r_1(t) &= \alpha x(t) + \eta(t), \\ r_2(t) &= \beta x(t) + \omega(t), \end{aligned} \tag{3.1}$$

where $\eta(t)$ and $\omega(t)$ are i.i.d. noise terms drawn from $\mathcal{N}(0, \sigma_\eta^2)$ and $\mathcal{N}(0, \sigma_\omega^2)$, respectively. The coefficient β is known to the agent ***a priori***, but α is unknown and must be estimated from observations. The covariate $x(t)$ is an i.i.d. draw from $\mathcal{N}(0, \sigma_x^2)$ and the agent must then either select action a_1 and receive reward $r(t) = r_1(t)$ or select action a_2 and receive reward $r(t) = r_2(t)$. The objective is for the agent to maximise the cumulative reward $R(T) = \sum_{t=1}^{T} r(t)$. The reward generating mechanism is simple, using a normal covariate centred at zero, but allows for a clear analysis of the exploration-exploitation trade-off – as exploration is only needed for the agent to learn one parameter. We note that the initialisation in this framework is for the agent to select the unknown action a_1 once, in order to receive an estimate of its expected average reward.

3.2 The Probability of Error

The agent must learn the value of α in Equation (3.1) over time. Suppose the agent has selected action a_1 n times prior to time t (where $n \in \{1, \ldots, t-1\}$) and these selections occurred at time-steps $t_1, \ldots, t_n$. α is estimated using $\widehat{\alpha}_n$, the solution of the linear least squares equation:

$$\widehat{\alpha}_n = \frac{\sum_{j=1}^{n} x(t_j) r_1(t_j)}{\sum_{j=1}^{n} x(t_j)^2}. \tag{3.2}$$

The parameter estimate $\widehat{\alpha}_n$ has a distribution that is centred at α and dependent on the number of pulls n and the distribution of $x(t)$ and $\eta(t)$. As $\eta(t)$ is i.i.d. and normally distributed it follows that (see Daly et al. (1995, p.407)):

$$\widehat{\alpha}_n \sim \mathcal{N}\left(\alpha, \frac{\sigma_\eta^2}{\sum_{j=1}^n x(t_j)^2}\right) \Rightarrow (\widehat{\alpha}_n - \alpha)\sqrt{\frac{n\sigma_x^2}{\sigma_\eta^2}} \sim \frac{\mathcal{N}(0,1)}{\sqrt{\sum_{j=1}^n \left(x(t_j)^2/\sqrt{\sigma_x^2}\right)/n}}.$$

If the agent uses an ϵ-greedy or ϵ-first policy, then the distribution of covariates $x(t_j)$ used for estimating α does not follow a $\mathcal{N}(0, \sigma_x^2)$ distribution (unless $\epsilon = 1$ and action a_1 is always selected) as the agent only selects a_1 for certain regions of the covariate space. Nevertheless, action a_1 is selected based only on whether $x(t)$ is positive or negative and hence the distribution of $x(t_j)^2$ is the same as $x(t)^2$, i.e.: $\sum_{j=1}^n x(t_j)^2/\sqrt{\sigma_x^2} \sim \chi_n^2$ – a chi-square distribution with n degrees of freedom. In addition, this chi-squared distribution is independent of the normal distribution in the above equation, as the magnitude of the observed covariates and the observation errors are independent of each other (which will not always be the case in other settings). It then follows from the definition of the t-distribution that:

$$(\widehat{\alpha}_n - \alpha)\sqrt{\frac{n\sigma_x^2}{\sigma_\eta^2}} \sim t_n, \tag{3.3}$$

where t_n is the t-distribution with n degrees of freedom. From Equation (3.3) we can find the probability of error after n pulls, that is the probability that the agent selects the wrong action when being greedy. Specifically this occurs if $\widehat{\alpha}_n > \beta$ when $\alpha < \beta$ and vice-versa. We therefore define this probability as:

$$F(n) = \begin{cases} \Pr(\widehat{\alpha}_n < \beta) & \text{when } \alpha > \beta; \\ \Pr(\widehat{\alpha}_n > \beta) & \text{when } \alpha < \beta \end{cases} \tag{3.4}$$

First consider $\Pr(\widehat{\alpha}_n < \beta)$ when $\alpha > \beta$. It follows from Equation (3.3) that:

$$\begin{aligned}\Pr(\widehat{\alpha}_n < \beta) &= \Pr\left(\sqrt{\frac{n\sigma_x^2}{\sigma_\eta^2}}\,(\widehat{\alpha}_n - \alpha) < \sqrt{\frac{n\sigma_x^2}{\sigma_\eta^2}}\,(\beta - \alpha)\right) \\ &= T\left((\beta - \alpha)\sqrt{\frac{n\sigma_x^2}{\sigma_\eta^2}}, n\right), \end{aligned} \tag{3.5}$$

where $T(x, n)$ is the t-distribution cumulative density function at ordinate x, with n degrees of freedom. By considering $\Pr(\widehat{\alpha}_n > \beta)$ when $\alpha < \beta$ in the same way, it follows from Equation (3.4) and Equation (3.5) that:

$$F(n) = T\left(-|\alpha - \beta|\sqrt{\frac{n\sigma_x^2}{\sigma_\eta^2}}, n\right), \tag{3.6}$$

for all values of α and β. The probability of error, $F(n)$, has the following four properties:

1. $F(n)$ is (strictly) bounded above by 0.5.
2. $F(n)$ decreases as n increases, as both the ordinate becomes more negative and the degrees of freedom increase (reducing the weight in the tails). $F(n)$ is also a convex sequence in n (proved later).
3. Increasing the difference between α and β reduces the value of $F(n)$.
4. The ratio σ_x^2/σ_η^2 can be interpreted as a 'signal to noise ratio' – larger values of this ratio reduce $F(n)$.

Property 1 ensures that the agent can do no worse than guessing between the actions. Notice however, that as $\sigma_\eta^2 \to \infty$, $F(n) \to 0.5$. Figure 3.1 shows the probability of error over n for several values of σ_x^2/σ_η^2 where properties 1, 2 and 4 are demonstrated. Note that we use the reward coefficients $\alpha = 0.5$ and $\beta = 0.3$ for all figures and simulations in this chapter and that in all figures the displayed curves are not continuous and can only be identified at integer values of n or t.

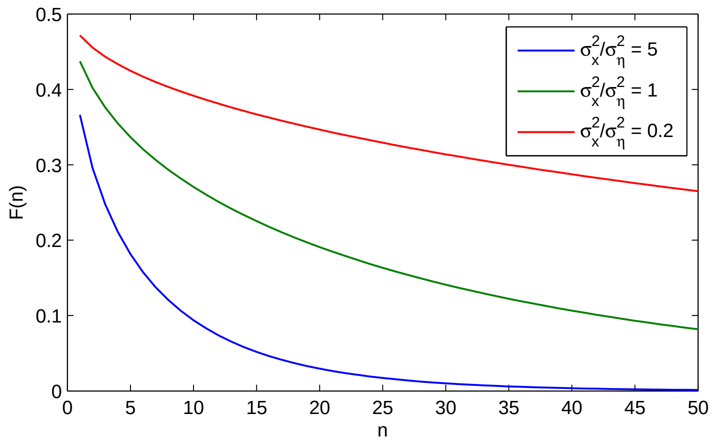


Figure 3.1: The sequence $F(n)$ from Equation (3.6) for various values of σ_x^2/σ_η^2.

3.3 The ϵ-greedy Policy

The ϵ-greedy policy dictates that the agent selects action a_1 with probability ϵ but selects the action with highest expected reward with probability $1-\epsilon$. In the previous section, we found the probability of error given n pulls of action a_1 by time t. In fact, we can find the distribution of n given t by the symmetry of the problem, as $x(t)$ is centrally distributed and therefore action a_1 is pulled 50% of the time when the agent is greedy. One pull of action a_1 is guaranteed at time $t=1$ (for initialisation), so the probability of having pulled the action n times by time t, $B(n,t,\epsilon)$, follows a binomial distribution:

$$B(n,t,\epsilon) = \binom{t-2}{n-1}\left(\frac{1}{2}(1+\epsilon)\right)^{n-1}\left(\frac{1}{2}(1-\epsilon)\right)^{t-n-1}, \tag{3.7}$$

where $t \geq 2$ and $n \in \{1,\ldots,t-1\}$. This probability is the probability that the action is selected $n-1$ times from the $t-2$ opportunities whilst using ϵ-greedy (recall that initialisation occurs at the first time-step and the decision at time t has not been made yet). The probability of selection at each round is $(1+\epsilon)/2$ as 1/2 of the pulls are guaranteed using ϵ-greedy and a further ϵ/2 through ϵ-greedy exploration.

The distribution of $\widehat{\alpha}_n$ after n pulls of action a_1 using the ϵ-greedy policy is as given in Equation (3.3). The distribution of n can then be used to find the probability of error at time t of the ϵ-greedy policy:

$$F_{\epsilon g}(t,\epsilon) = \sum_{n=1}^{t-1} B(n,t,\epsilon)F(n). \tag{3.8}$$

As $F(n) < 0.5$ and $\sum_{n=1}^{t-1} B(n,t,\epsilon) = 1$ it follows that $F_{\epsilon g}(t,\epsilon) < 0.5$ (for $t \geq 2$, note that $F_{\epsilon g}(1,\epsilon) = 0.5 \forall \epsilon$). It then immediately follows that all other properties of the probability of error mentioned in Section 3.2 still hold, and $F_{\epsilon g}(t,\epsilon)$ has the additional property that it decreases as ϵ increases, for a fixed t. Figure 3.2 shows the sequence $F_{\epsilon g}(t,\epsilon)$ for a selection of ϵ values from $t = 1,\ldots,50$, where this property is demonstrated.

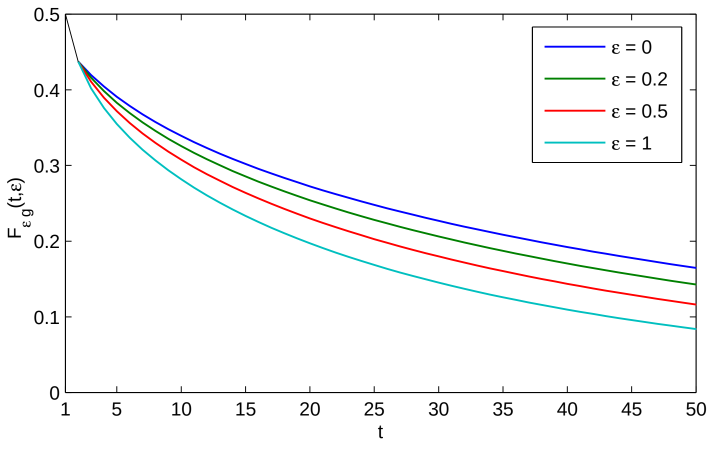


Figure 3.2: The sequence $F_{\epsilon g}(t,\epsilon)$ from Equation (3.8) for various ϵ ($\sigma_x^2/\sigma_\eta^2 = 1$).

The expected instantaneous reward of the ϵ-greedy policy at time t, $r_{\epsilon g}(t,\epsilon)$, can now be found by considering the cases when the agent explores and exploits separately:

$$r_{\epsilon g}(t,\epsilon) = \epsilon \mathrm{E}(r_1(t)) + (1-\epsilon)r_g(t,\epsilon), \tag{3.9}$$

where $r_g(t,\epsilon)$ is the expected instantaneous reward when the agent is greedy. It follows that as $x(t) \sim \mathcal{N}(0,\sigma_x^2)$:

$$\mathrm{E}(r_1(t)) = \int_{-\infty}^{\infty} \alpha x(t) \frac{1}{\sqrt{2\pi\sigma_x^2}} \exp\left(-\frac{x(t)^2}{2\sigma_x^2}\right) \mathrm{d}x(t) = 0. \tag{3.10}$$

From the probability of error $F_{\epsilon g}(t, \epsilon)$ we can find the expected instantaneous reward when the agent is greedy, by separately considering the expected instantaneous reward when the correct/incorrect action is selected, dependent on the probability of error (see Appendix A.1). It follows that:

$$r_g(t, \epsilon) = |\alpha - \beta| \sqrt{\frac{\sigma_x^2}{2\pi}} \left(1 - 2F_{\epsilon g}(t, \epsilon)\right), \tag{3.11}$$

which yields the expected instantaneous reward of the ϵ-greedy policy:

$$r_{\epsilon g}(t, \epsilon) = (1 - \epsilon) |\alpha - \beta| \sqrt{\frac{\sigma_x^2}{2\pi}} \left(1 - 2F_{\epsilon g}(t, \epsilon)\right). \tag{3.12}$$

This expected reward is greater than zero (for $t \geq 2$) as $F_{\epsilon g}(t, \epsilon) < 0.5$, so the policy performs better than guessing between the actions for all values of ϵ (except $\epsilon = 1$). Larger values of ϵ reduce the probability of error $F_{\epsilon g}(t, \epsilon)$, which increases the expected reward – this is the *benefit of exploration*. Conversely, larger values of ϵ reduce the $(1 - \epsilon)$ term in the expected reward and this is the *cost of exploration*. Despite this exploration-exploitation trade-off, the expected instantaneous reward in Equation (3.12) is maximised by $\epsilon = 0$ for all values of $t > 0$, α, β and σ_x^2/σ_η^2, which we prove in Theorem 3.1 below.

Theorem 3.1 *$r_{\epsilon g}(t, 0) > r_{\epsilon g}(t, \epsilon)$ for all $0 < \epsilon \leq 1$ and for all $t \in \mathbb{Z}^+ \backslash \{1\}$, $\alpha, \beta \in \mathbb{R}$ and $\sigma_x^2, \sigma_\eta^2 \in \mathbb{R}^+$.*

Proof We prove Theorem 3.1 by contradiction. First consider the case $t \geq 3$. Suppose there exists $0 < \epsilon \leq 1$ such that:

$$r_{\epsilon g}(t, \epsilon) \geq r_{\epsilon g}(t, 0) \quad \text{for some } t \in \mathbb{Z}^+ \backslash \{1\}, \alpha, \beta \in \mathbb{R}, \sigma_x^2, \sigma_\eta^2 \in \mathbb{R}^+. \tag{3.13}$$

Substituting from Equation (3.12) and Equation (3.8), the inequality in Equation (3.13) becomes:

$$\sum_{n=1}^{t-1} F(n) G(n) \geq \frac{\epsilon}{2}, \tag{3.14}$$

where $G(n) = B(n,t,0) - (1-\epsilon)B(n,t,\epsilon)$. Notice that $F(n) < 1/2$ and $\sum_{n=1}^{t-1} G(n) = \epsilon$, however it does not follow from this alone that $\sum_{n=1}^{t-1} F(n)G(n) < \epsilon/2$ (by Cauchy-Schwarz, for example), as $G(n)$ can be negative for certain values of n. To proceed, the following three lemmas allow for a useful upper bound to be constructed on the left-hand side of Equation (3.14).

Lemma 3.1 *$F(n)$ is a convex sequence in n.*

Proof From Equation (3.6) and Abramowitz and Stegun (1965),

$$F(n) = \mathrm{T}\left(-c\sqrt{n}, n\right) = \frac{1}{2} I_x\left(\frac{n}{2}, \frac{1}{2}\right), \text{ where } x = \frac{n}{n + (-c\sqrt{n})^2} = \frac{1}{1+c^2},$$

x is a constant where $0 < x < 1$ as $c \in \mathbb{R}^+$. $I_x(a,b)$ is the regularized incomplete beta function ($a, b > 0$ and $0 \leq I_x \leq 1$) defined by:

$$I_x(a,b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \int_0^x t^{a-1}(1-t)^{b-1} dt.$$

It therefore suffices to prove that for all $a > 0$,

$$\frac{I_x(a,1/2) + I_x(a+1,1/2)}{2} \geq I_x(a+1/2, 1/2).$$

To prove this we use the following 4 relations found in (Abramowitz and Stegun, 1965):

<u>Property 1</u> $B_x(a,b) = \frac{1}{a} x^a \, {}_2F_1(a, 1-b, a+1; x)$

<u>Property 2</u> ${}_2F_1(a,b,c,x) = \frac{\Gamma(c)}{\Gamma(b)\Gamma(c-b)} \int_0^1 t^{b-1}(1-t)^{c-b-1}(1-xt)^{-a} dt$

<u>Property 3</u> ${}_2F_1(a,b,c,x) = \frac{1}{(1-x)^b} \, {}_2F_1(b, c-a, c, \frac{x}{x-1})$

<u>Property 4</u> ${}_2F_1(a,b,c,x) = {}_2F_1(b,a,c,x)$

where ${}_2F_1(a,b,c,x)$ is the Gauss hypergeometric series and $B_x(a,b)$ is the nonregularized incomplete beta function where:

$$I_x(a,b) = \frac{B_x(a,b)}{B(a,b)} = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} B_x(a,b), \tag{3.15}$$

and $B(a,b)$ is the beta function. From Properties 1 and 3:

$$\begin{aligned} B_x(a+1/2,1/2) &= \frac{x^{a+1/2}}{a+1/2}\, {}_2F_1(a+1/2,1/2,a+3/2;x) \\ &= \frac{x^{a+1/2}}{(a+1/2)\sqrt{1-x}}\, {}_2F_1\left(1/2,1,a+3/2;\frac{x}{x-1}\right), \end{aligned}$$

similarly,

$$\begin{aligned} B_x(a,1/2) &= \frac{x^{a}}{a\sqrt{1-x}}\, {}_2F_1\left(1/2,1,a+1;\frac{x}{x-1}\right), \\ B_x(a+1,1/2) &= \frac{x^{a+1}}{(a+1)\sqrt{1-x}}\, {}_2F_1\left(1/2,1,a+2;\frac{x}{x-1}\right). \end{aligned} \tag{3.16}$$

Therefore from Equation (3.15), Equation (3.16) and Properties 2 and 4:

$$\begin{aligned} I_x(a,1/2) &= \frac{\Gamma(a+1/2)}{\Gamma(a)\Gamma(1/2)} B_x(a,1/2) \\ &= \frac{\Gamma(a+1/2)}{\Gamma(a)\Gamma(1/2)} \frac{x^a}{a\sqrt{1-x}}\, {}_2F_1\left(1/2,1,a+1;\frac{x}{x-1}\right) \\ &= \frac{\Gamma(a+1/2)}{\Gamma(a)\Gamma(1/2)} \frac{x^a}{a\sqrt{1-x}}\, {}_2F_1\left(1,1/2,a+1;\frac{x}{x-1}\right) \\ &= \frac{\Gamma(a+1/2)}{\Gamma(a)\Gamma(1/2)} \frac{x^a}{a\sqrt{1-x}} \frac{\Gamma(a+1)}{\Gamma(1/2)\Gamma(a+1/2)} \int_0^1 t^{-1/2}(1-t)^{a-1/2}(1-zt)^{-1}dt \\ &= \frac{x^a}{\pi\sqrt{1-x}} \int_0^1 t^{-1/2}(1-t)^{a-1/2}(1-zt)^{-1}dt, \end{aligned}$$

$z = \frac{x}{x-1}$ and $\Gamma(1/2) = \sqrt{\pi}$. Notice that $z < 0$ as $0 < x < 1$. It follows that:

$$\begin{aligned} I_x(a+1/2,1/2) &= \frac{x^{a+1/2}}{\pi\sqrt{1-x}} \int_0^1 t^{-1/2}(1-t)^{a}(1-zt)^{-1}dt, \\ I_x(a+1,1/2) &= \frac{x^{a+1}}{\pi\sqrt{1-x}} \int_0^1 t^{-1/2}(1-t)^{a+1/2}(1-zt)^{-1}dt. \end{aligned}$$

Therefore,

$$\begin{aligned}
&I_x(a,1/2) + I_x(a+1,1/2) \\
&= \frac{x^a}{\pi\sqrt{1-x}} \int_0^1 t^{-1/2}(1-t)^{a-1/2}(1-zt)^{-1}[1+x(1-t)]dt \\
&\geq \frac{x^a}{\pi\sqrt{1-x}} \int_0^1 t^{-1/2}(1-t)^{a-1/2}(1-zt)^{-1}[2\sqrt{x(1-t)}]dt = 2I_x(a+1/2,1/2).
\end{aligned}$$

This holds as $1 + x(1-t) \geq 2\sqrt{x(1-t)}$, for $0 < x < 1$ and $0 \leq t \leq 1$. To verify, set $u = x(1-t)$ and square both sides:

$$(1+u)^2 = 1 + 2u + u^2 = (1-u)^2 + 4u > 4u.$$

The relation holds as $0 < u < 1$ and the proof of convexity is complete. ∎

Lemma 3.2 *There exists an integer q where $2 \leq q \leq t-1$ such that:*

$$\begin{aligned}
&G(n) \geq 0 \quad \text{for } n = 1, \ldots, q \\
&G(n) < 0 \quad \text{otherwise.}
\end{aligned}$$

Proof

$$\begin{aligned}
G(n) &= \text{bin}\left(n-1, t-2, \frac{1}{2}\right) - (1-\epsilon)\text{bin}\left(n-1, t-2, \frac{1}{2}(1+\epsilon)\right) \\
&= \left(\frac{1}{2}\right)^{t-2} \binom{t-2}{n-1} \left(1 - (1+\epsilon)^{n-1}(1-\epsilon)^{t-n}\right).
\end{aligned}$$

Notice that:

$$G(1) = \left(\frac{1}{2}\right)^{t-2} \left(1 - (1-\epsilon)^{t-1}\right) > 0, \tag{3.17}$$

$$G(2) = \left(\frac{1}{2}\right)^{t-2} (t-2)\left(1 - (1-\epsilon^2)(1-\epsilon)^{t-3}\right) > 0 \quad (\text{for } t \geq 3),$$

$$\left(\frac{1}{2}\right)^{t-2} \binom{t-2}{n-1} > 0, \tag{3.18}$$

for $0 < \epsilon \leq 1$ and $n = 1, \ldots, t-1$. From Equation (3.17) and Equation (3.18) it suffices to show that the sequence $H(n) = (1-(1+\epsilon)^{n-1}(1-\epsilon)^{t-n})$ is decreasing in n for $n = 1, \ldots, t-1$.

$$\begin{aligned} H(n+1) - H(n) &= \left(1-(1+\epsilon)^{n}(1-\epsilon)^{t-n-1}\right) - \left(1-(1+\epsilon)^{n-1}(1-\epsilon)^{t-n}\right) \\ &= \left((1+\epsilon)^{n-1}(1-\epsilon)^{t-n-1}\right)(-2\epsilon) < 0, \end{aligned}$$

for $n = 1, ..., t-1$ and $0 < \epsilon \leq 1$. Therefore, the sequence $G(n)$ has all negative terms preceded by non-negative terms. The integer q in the lemma is set to be the last non-negative term in the sequence $G(n)$, where $2 \leq q \leq t-1$. ∎

Lemma 3.3

$$\sum_{n=1}^{t-1} F(n)G(n) \leq \sum_{n=1}^{t-1} F'(n)G(n), \quad \text{where} \quad F'(n) = \frac{q-n}{q-1}F(1) + \frac{n-1}{q-1}F(q).$$

Proof It follows from Lemmas 3.1 and 3.2 that $F(n) \leq F'(n)$ and $G(n) \geq 0$ for $n = 1, \ldots, q$, therefore:

$$\sum_{n=1}^{q} F(n)G(n) \leq \sum_{n=1}^{q} F'(n)G(n).$$

It also follows from Lemmas 3.1 and 3.2 that $F(n) > F'(n)$ and $G(n) < 0$ for $n = q+1, \ldots, t-1$, therefore:

$$\sum_{n=q+1}^{t-1} F(n)G(n) \leq \sum_{n=q+1}^{t-1} F'(n)G(n). \quad ∎$$

After expanding the binomial coefficients and rearranging (see Appendix A.2):

$$\sum_{n=1}^{t-1} F'(n)G(n) = \frac{1}{2}(2\epsilon - \epsilon^2)F(1) + \frac{1}{2}\epsilon^2 F(q) + \frac{1}{2}\frac{t-q-1}{q-1}\epsilon^2(F(q) - F(1)). \quad (3.19)$$

As $\frac{1}{2}\frac{t-q-1}{q-1}\epsilon^2 > 0$, $F(q) < F(1)$ and $F(n) < 0.5$, then the third term is negative and hence from Lemma 3.3 and Equation (3.19):

$$\sum_{n=1}^{t-1} F(n)G(n) \leq \frac{1}{2}(2\epsilon - \epsilon^2)F(1) + \frac{1}{2}\epsilon^2 F(q) < \frac{1}{2}(2\epsilon - \epsilon^2)\frac{1}{2} + \frac{1}{2}\epsilon^2\frac{1}{2} = \frac{\epsilon}{2}.$$

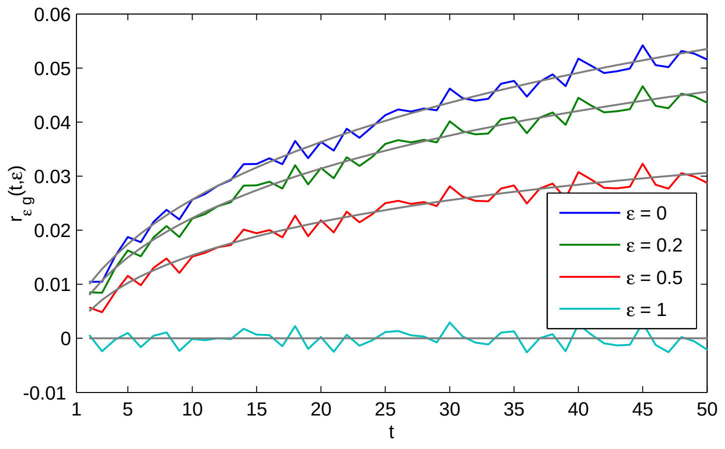
A contradiction has been made and $\sum_{n=1}^{t-1} F(n)G(n) < \epsilon/2$, therefore Theorem 3.1 has been proved for $t \geq 3$. It remains to show that the theorem holds for $t = 2$. Using Equation (3.12) and Equation (3.8):

$$\begin{aligned} r_{\epsilon g}(2,0) &= |\alpha - \beta| \sqrt{\frac{\sigma_x^2}{2\pi}} \left(1 - 2F(1)\right) \\ &> (1-\epsilon)\,|\alpha - \beta| \sqrt{\frac{\sigma_x^2}{2\pi}} \left(1 - 2F(1)\right) = r_{\epsilon g}(2,\epsilon), \end{aligned}$$

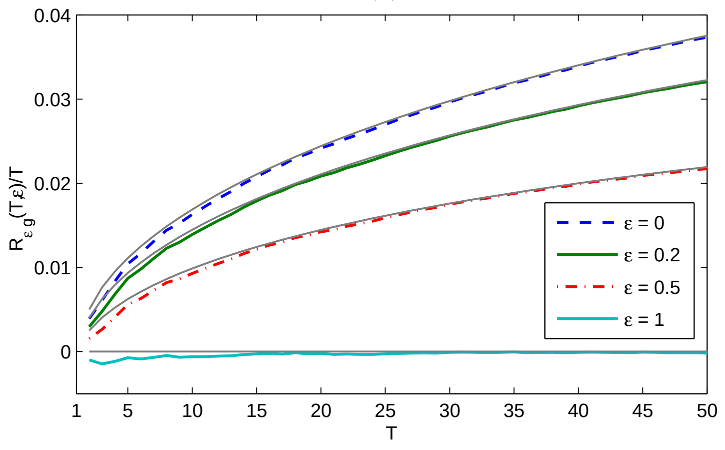
as $0 < \epsilon \leq 1$ and $F(1) < 0.5$. This completes the proof of Theorem 3.1. ∎

Theorem 3.1 states that the expected instantaneous reward at time t is maximised by $\epsilon = 0$. It is then immediate that the cumulative reward $R_{\epsilon_g}(T,\epsilon) = \sum_{t=1}^{T} r_{\epsilon_g}(t,\epsilon)$, which is what we wish to maximise, is also maximised by $\epsilon = 0$. This implies that the greedy policy, on average, outperforms any ϵ-greedy policy for this one-armed bandit problem. Given these findings, Figure 3.3 shows the averaged instantaneous and cumulative reward at time t from 20,000 repeated simulations of the same problem, with the theoretical expectations overlayed. The empirical evidence verifies the theoretical findings that the instantaneous reward (and hence cumulative reward also) is maximised by $\epsilon = 0$. In other words, the ϵ-greedy policy is a suboptimal policy for this one-armed bandit problem with covariates.

In contrast, finite-time analyses of multi-armed bandit problems (Auer et al., 2002; Vermorel and Mohri, 2005; Pavlidis et al., 2008b) have concluded, through empirical evidence, that the optimally tuned ϵ-greedy policy can have $\epsilon > 0$. The difference between this evidence and our findings, is due to the exploration requirements of the two problems. In our one-armed bandit problem only one action requires any exploration, and since this action is already selected 50% of the time with a greedy policy, no further exploration is required with an ϵ-greedy policy. Conversely, in a multi-armed bandit problem, more actions require exploration.



(a)



(b)

Figure 3.3: Average rewards for a range of ϵ-greedy policies ((a) instantaneous reward and (b) average cumulative reward) where $\sigma_x^2/\sigma_\eta^2 = 1$. Theoretical expectations are overlayed (in grey).

Moreover there are no such guarantees on exploring each action sufficiently with a greedy policy and consequently optimal actions are often overlooked. As a result, an ϵ-greedy policy with $\epsilon > 0$, can outperform the greedy policy.

3.4 The ϵ-first Policy

The ϵ-first policy dictates that all the agent's exploration is at the beginning (for the first ϵT iterations) followed by greedy selection for the remaining iterations. When the agent explores, action a_1 is always pulled and it follows from Equation (3.10) that the expected reward of ϵ-first $r_{\epsilon f}(t) = E(r_1(t)) = 0$ for $t \leq \epsilon T + 1$. To find the expected reward for $t > \epsilon T + 1$, consider the probability of error $F_{\epsilon f}(t, \epsilon)$ with this policy. Using the same reasoning as before, we find:

$$F_{\epsilon f}(t, \epsilon) = \sum_{n=1}^{t-\epsilon T-1} B(n, t - \epsilon T, 0) F(n + \epsilon T), \tag{3.20}$$

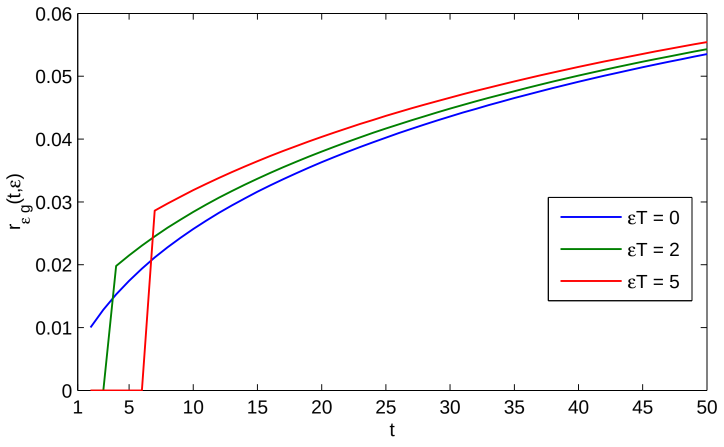
with $F(n)$ as given in Equation (3.6) and $B(n, t, \epsilon)$ as given in Equation (3.7). Notice that $F_{\epsilon f}(t, \epsilon) < 0.5$ (for $t \geq 2$) as with the ϵ-greedy policy. This probability of error follows as there is a minimum of $\epsilon T + 1$ guaranteed pulls from exploration and initialisation, and any further pulls occur from greedy selection over the remaining $t - \epsilon T - 2$ opportunities, each with probability 50%. In the same way that Equation (3.11) was derived (see Appendix A.1), it follows that:

$$r_{\epsilon f}(t, \epsilon) = |\alpha - \beta| \sqrt{\frac{\sigma_x^2}{2\pi}} \left(1 - 2F_{\epsilon f}(t, \epsilon)\right) \quad \text{for } t > \epsilon T + 1. \tag{3.21}$$

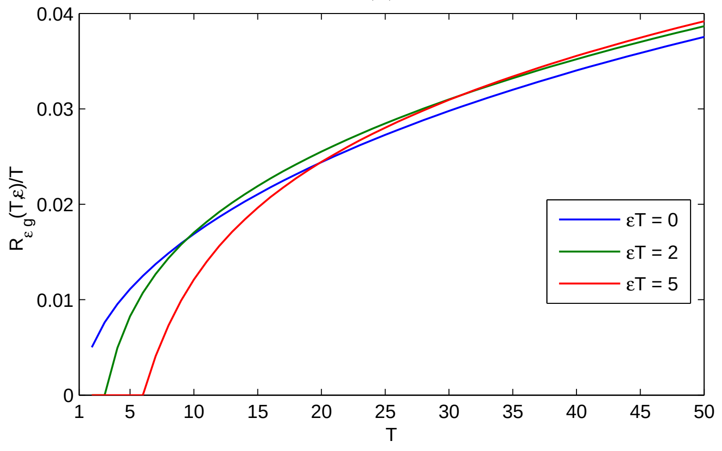
Again this expected reward is positive as $F_{\epsilon f}(t, \epsilon) < 0.5$. Larger values of ϵ reduce the probability of error for $t > \epsilon T + 1$ and thus have a higher expected reward in this region – this is the *benefit of exploration*. Conversely, larger values of ϵ correspond to a longer period of exploration where the expected reward is zero – this is the *cost of exploration*. The expected cumulative reward is $R_{\epsilon_f}(T, \epsilon) = \sum_{t=1}^{T} r_{\epsilon_f}(t, \epsilon)$ and we can maximise this numerically using Equation (3.21) to find the optimal ϵ.

This optimal value will not necessarily be zero as the following numerical results show. In particular, Figure 3.4(a) displays the expected reward at time t, for the game of length $T = 50$ shown in Figure 3.3, for various values of ϵT, where the benefit and cost of exploration can be clearly seen. Summing the rewards from Figure 3.4(a) generates Figure 3.4(b) which is the expected cumulative reward at time $T = 1, \ldots, 50$ for the fixed values of ϵT. Fixing ϵT in this way has shown that the greedy policy can be beaten and there are regions of T where $\epsilon T = 0, 1, 2, \ldots$ are optimal in terms of maximising the expected cumulative reward.

Figure 3.5(a) displays how the optimal value of ϵT grows with T for various values of σ_x^2/σ_η^2. The range of values of T where a specific value of ϵT is optimal



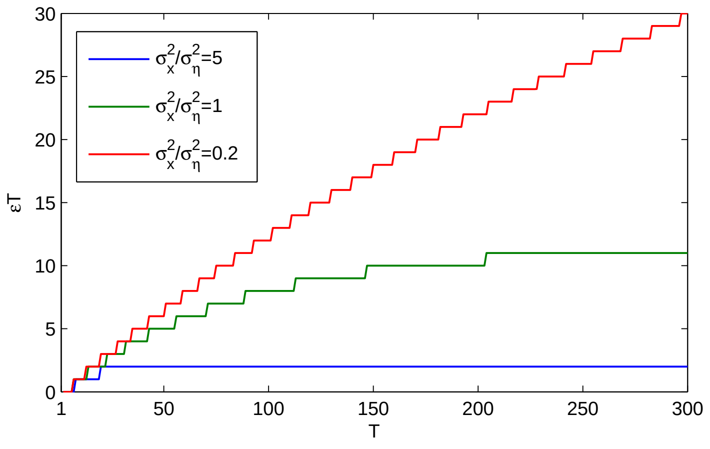
(a)



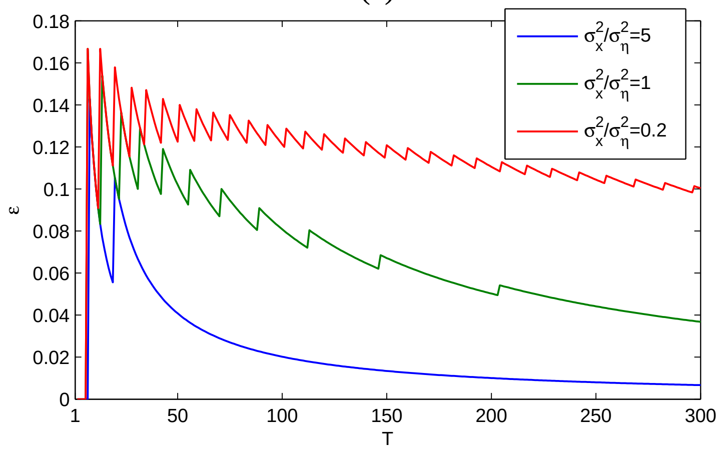
(b)

Figure 3.4: Expected rewards for a range of ϵ-first policies ((a) instantaneous reward (with $T = 50$) and (b) average cumulative reward) where $\sigma_x^2/\sigma_\eta^2 = 1$.

become larger as ϵT increases, indicating that ϵT grows more slowly than T. This idea can also be seen in Figure 3.5(b), which shows a plot of optimal ϵ for values of T. The non-smooth shape of this plot is due to the fact that ϵT is represented as a step-function taking integer values only. The key observation, however, is that the optimal ϵ decreases and approaches zero as $T \to \infty$ (although for $\sigma_x^2/\sigma_\eta^2 = 0.2$ this happens very slowly), which concurs with the studies of Woodroofe (1979) and Sarkar (1991) which proved the greedy policy was asymptotically optimal. Nevertheless, in finite time, a well chosen ϵ in the ϵ-first policy will outperform the greedy policy, signifying the benefits of correctly balancing exploration and exploitation in an action selection policy.



(a)



(b)

Figure 3.5: Optimal values of (a) ϵT and (b) ϵ for the ϵ-first policy, with various values of σ_x^2/σ_η^2.

3.5 An On-line Algorithm

In this section, we construct an algorithm that can effectively attempt to learn the optimal ϵ-first policy on-line. This algorithm is simple, and makes use of the theoretically derived expected reward of the ϵ-first policy, given in Equation (3.21). This expected reward is dependent on the reward function parameters: α, σ_x^2 and σ_η^2. The agent does not know the true values of these parameters, but can recursively estimate them on-line. α is estimated using least squares Equation (3.2), σ_x^2 is estimated using the sample variance of observed covariate values and σ_η^2 is estimated from the residuals from the regression:

$$\hat{\sigma}_\eta^2 = \frac{1}{n-1}\hat{\xi}'\hat{\xi}, \tag{3.22}$$

where the jth component of the vector $\hat{\xi}$ is,

$$\hat{\xi}_j = r_1(t_j) - \hat{\alpha}x(t_j). \tag{3.23}$$

With these estimates the agent can then compute estimates of the expected cumulative reward (from the current time t to the final time-step T) of exploring for one more iteration $(R_{xplr}(t))$ or selecting greedily $(R_{xplt}(t))$ for the rest of the game:

$$R_{xplt}(t) = \sum_{v=t}^{T} \left(r_{\epsilon f}(v, (t-2)/T)|\hat{\alpha}, \hat{\sigma}_\eta^2, \hat{\sigma}_x^2\right), \tag{3.24}$$

$$R_{xplr}(t) = \sum_{v=t}^{T} \left(r_{\epsilon f}(v, (t-1)/T)|\hat{\alpha}, \hat{\sigma}_\eta^2, \hat{\sigma}_x^2\right), \tag{3.25}$$

which are computed using Equation (3.21) where the sample estimates $\hat{\alpha}, \hat{\sigma}_\eta^2, \hat{\sigma}_x^2$ are used instead of their unknown true values. Notice that the number of exploration steps for $R_{xplr}(t)$ and $R_{xplt}(t)$ is $t-1$ and $t-2$, respectively, as the first time-step is initialisation and the decision at time-step t is the one being determined and is hence ϵ-first exploration with $R_{xplr}(t)$ or greedy selection with $R_{xplt}(t)$. The agent simply follows the policy ($R_{xplr}(t)$ or $R_{xplt}(t)$) with the highest estimated reward for the next iteration, and then recomputes for the next iteration with the new sample estimates.

If the greedy policy dictates selecting action a_1 anyway then the agent does not need to execute the algorithm at each iteration (although parameter estimates are still updated). If the agent determines that a greedy policy is better than exploration then, unlike with ϵ-first, exploration can still be revisited at future time-steps if the new sample estimates dictate this. In this way, the algorithm is self-correcting and performs better than simply estimating the optimal ϵ on-line during the initial time-steps, where sample errors are still large. Note that an initialisation period of length

2 is required, where action a_1 is pulled twice, such that sample estimates of the covariance and noise variance ($\hat{\sigma}_x^2$ and $\hat{\sigma}_\eta^2$) can be obtained.

Table 3.1 shows the average performance of this on-line algorithm (averaged over 20,000 repeats[1]) for various values of σ_x^2/σ_η^2, as compared with various ϵ-first policies, where the game-length[2] is $T = 100$. The rewards are normalised between 0 and 1, where 1 is the expected reward to an oracle that knows all parameters and 0 is the expected reward of a random policy. We report all rewards in this thesis in this manner as this allows results between experiments to be comparable.

Table 3.1: Average rewards of the on-line algorithm and various ϵ-first policies

σ_x^2/σ_η^2	$\epsilon = 0$	0.02	0.05	0.1	0.15	0.2	On-line
5	0.882	**0.887**	0.883	0.858	0.822	0.780	0.894
2	0.755	0.764	**0.769**	0.764	0.744	0.717	0.776
1	0.618	0.626	0.638	**0.639**	0.629	0.614	0.640
0.5	0.478	0.482	0.493	**0.501**	0.499	0.492	0.501
0.2	0.318	0.322	0.327	0.333	**0.340**	0.333	0.331
Avg.	0.610	0.616	**0.622**	0.619	0.606	0.587	0.628

As expected, performance degrades for each policy as σ_x^2/σ_η^2 decreases and the relative amount of noise increases. For a given value of σ_x^2/σ_η^2, the on-line approach yields a reward close to or better than the best performing ϵ-first policy (denoted in bold), and when rewards are averaged across experiments then the on-line approach performs best overall. In fact, Table 3.2 compares the on-line algorithm against the optimal ϵ-first policy, and we see that the on-line approach yields rewards that are comparable with the optimal off-line reward. Note that the relative performance degrades for higher noise problems, as the sample estimates have larger variance. Moreover, notice that the average number of exploration steps has increased accordingly with increases in the noise variance, albeit slowly. This shows that the amount of exploration is being driven, to some extent, by the degree of uncertainty the agent has regarding the rewards of the unknown action.

[1]This number of repeats ensures the standard errors of all average rewards are less than 1×10^{-3}.

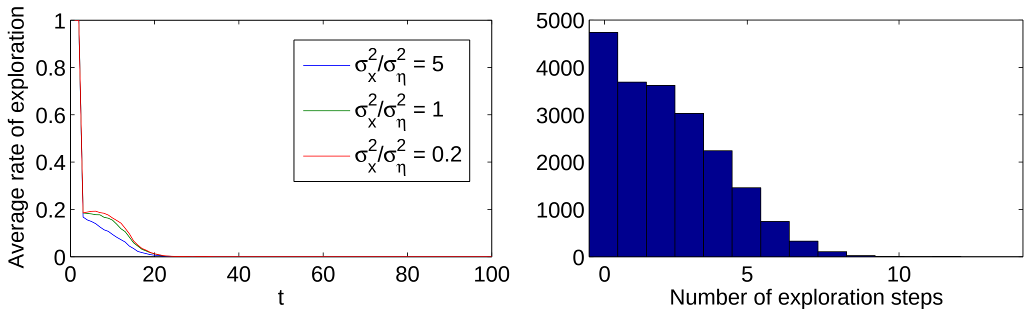
[2]This value of T is selected as it is used in Chapter 4, justified by the fact that real-world static problems are likely to be short-length, so we want algorithms that can yield high rewards quickly.

Table 3.2: Comparison of on-line and optimal off-line policies

σ_x^2/σ_η^2	Off-line (ϵ-first)		On-line	
	Opt. ϵ	Reward	% Optimal	Avg. exp. steps
5	0.03	0.888	100.8%	1.43
2	0.06	0.769	100.8%	1.84
1	0.08	0.640	100.0%	2.04
0.5	0.10	0.501	100.1%	2.14
0.2	0.13	0.340	97.5%	2.21

The on-line algorithm often marginally outperforms the optimal ϵ-first policy and this is because the on-line algorithm can respond to the circumstances of each game – for example, a favourable start to the game, where rewards are observed with little noise and yield faster than expected convergence to the true regression parameter α, will mean no more exploration is required. Conversely, noisy observations that lead to a high probability of error, will lead to more exploration. In this way, the on-line approach is again showing that it is driven by the amount of uncertainty as it plays.

Figure 3.6(a) shows the average rate of exploration over time for various values of σ_x^2/σ_η^2. The rate decays to zero over time and does this more slowly for high noise problems, which is desirable behaviour. Figure 3.6(b) however, displays a histogram of the number of explorative steps over the 20,000 repeats for a high-noise problem. Notice that the algorithm often does not perform any additional exploration after the 2 initialisation steps. As previously mentioned, this is sometimes due to a fortuitous initialisation period where observations are not noisy and the correct value of α has been immediately learnt. On other occasions however, the observations could be noisy but still ‘aligned’ on the regression plane, yielding an under-estimate of the noise variance (as the residuals are small) and an incorrect estimate of α which causes suboptimal actions to be selected. In such cases, the algorithm will prematurely cease exploration, explaining the slight degradation in performance with high-noise problems. This issue is addressed in the next chapter, where we build an algorithm for a more general problem that is more responsive to the uncertainty of parameter estimates when choosing the next action.



(a) (b)

Figure 3.6: Explorative behaviour of on-line approximation algorithm: (a) the average rate of exploration over time and (b) a histogram of the total number of explorative steps (for $\sigma_x^2/\sigma_\eta^2 = 0.2$).

3.6 Summary

In this chapter we have theoretically derived the expected rewards of the ϵ-greedy and ϵ-first policies for a one-dimensional one-armed bandit problem. We have proved that ϵ-greedy is optimised by setting ϵ=0 but ϵ-first is optimised with $\epsilon > 0$ (beyond some time-length T). In addition, we have constructed an algorithm for approximating ϵ-first on-line, without the need for an *a priori* fixed exploration parameter. We do this by estimating unknown parameters on-line and then approximating the optimal action. A key component of this is measuring the amount of uncertainty (by estimating the variance of the noise and covariate) and using this to drive exploration – more uncertainty leads to greater levels of exploration being required.

Extending these theoretical findings to higher-dimensional covariates or multi-armed problems is not trivial. The distributions of parameter estimates can be found, but their interactions in terms of how they influence the probability of error means finding analytic representations of errors and rewards is usually not possible. In addition, the introduction of multivariate distributions almost inevitably leads to numerical approximations of any representations that can be found. As a result, the expected rewards of various policies can simply be found through Monte Carlo simulations (rather than numerically approximating reward representations that are intractable and contain high dimensional distributions). For these reasons, we do not find any more specific reward representations of bandit frameworks. Instead we focus on developing algorithms for adapting exploration on-line in the proceeding chapters, using the concept of uncertainty driven exploration introduced in this chapter.

Chapter 4

On-line Adaptation of Exploration in Bandit Problems

In Chapter 2 we performed a critical analysis of the various policies and algorithms constructed for the bandit problem. We demonstrated that there is a shortage of parameter-free methods that can generalise across the many bandit frameworks, in particular for settings that include side information and dynamics (see Table 2.1). In fact, the ϵ-greedy policy is the only such method (and ϵ-first for static rewards), but these policies are not parameter-free and require the exploration parameter ϵ to be fixed *a priori*. As previously noted, the performance of these algorithms can deteriorate rapidly with a poorly selected ϵ value and setting this value correctly is unfeasible in many applications, as this requires prior knowledge of the problem faced, which is precisely what the agent is trying to learn.

For these reasons, in this chapter and the next, we construct an algorithm, ϵ-ADAPT, that can adapt exploration on-line in a computationally efficient manner, without the need for any exploration parameters. In this chapter, we focus on the static reward case (we focus exclusively on dynamic rewards in Chapter 5), and construct an algorithm for settings that are with or without side information. The latter setting is considered as most policies and algorithms cannot be used with side information – and we can therefore evaluate the performance of our algorithm against more alternatives. The generalisation to include side information, however, is particularly significant as our algorithm is the first on-line (and parameter-free) algorithm that can be used in this setting.

In Chapter 3, we performed a theoretical analysis of the ϵ-greedy and ϵ-first policies for a one-armed bandit problem. We derived the expected rewards and the optimal exploration parameters. In particular, we proved that the optimal ϵ in ϵ-first will be non-zero (beyond some game-length) and then constructed an algorithm that attempted to find this optimal rate of exploration on-line. The empirical results were favourable, with average rewards that were close to or better than the optimal off-line policy. In this chapter, we extend this approach first to higher-dimensional covariates and then to multi-armed problems. In addition, we make several improvements to the basic approach introduced in the last chapter, such as learning which regions of the covariate space should be explored and which actions require most exploration. In other words, ϵ-ADAPT not only learns *how much* to explore, but also *when* to explore and *which actions* to explore.

In general, planning out an optimal exploration policy for the duration of a finite-length game is an intractable computation (Sutton and Barto, 1998), scaling exponentially in the length of the game and the number of actions available. To reduce the computation to quadratic-time, we make use of the properties of the ϵ-first policy, such that ϵ-ADAPT need only decide whether to explore or exploit for the next time-step. In more detail, the likelihood of exploring at each iteration is driven by the amount of *uncertainty* the agent currently has – captured by calculating statistics from past interactions with the environment. Our algorithm therefore *adapts as it plays* to effectively tune the exploration parameter without the need for any other free parameters. We note that many policies and algorithms presented in Table 2.1 scale linearly in time (rather than quadratically), but this increased level of computation with ϵ-ADAPT is required to remove the need for an exploration parameter to be set *a priori*.

The remainder of this chapter is structured as follows. In Section 4.1 we construct ϵ-ADAPT for the one-armed bandit with covariates problem, and then perform a preliminary empirical evaluation. We extend ϵ-ADAPT to the multi-armed problem in Section 4.2 and then perform a detailed empirical evaluation in Section

4.3, for both simulated and real data. Summary remarks follow in Section 4.4.

4.1 The One-Armed Bandit with Covariates Problem

In the one-armed bandit with covariates problem, the agent must select between actions $\{a_1, a_2\}$, where the rewards for each action are given by:

$$r_1(t) = f(\boldsymbol{x}(t), \boldsymbol{\alpha}) + \eta(t), \tag{4.1}$$

$$r_2(t) = g(\boldsymbol{x}(t), \boldsymbol{\beta}) + \omega(t), \tag{4.2}$$

where $\boldsymbol{x}(t)$ is the p-dimensional covariate observed at time t. $\eta(t)$ and $\omega(t)$ are i.i.d. noise processes assumed to be normally distributed and centred at zero with variance σ_η^2 and σ_ω^2 respectively (both unknown to the agent). It is assumed the agent knows the functions f and g and the parameters $\boldsymbol{\beta}$ of the known action, but not the parameter values of the unknown action $\boldsymbol{\alpha}$ – this is precisely what the agent must learn. As in Pavlidis et al. (2008a), the covariate is assumed to be drawn from a known distribution, with unknown parameters (for example a multivariate Gaussian with unknown mean vector $\boldsymbol{\mu}_x$ and unknown covariance matrix Σ_x). The objective of the agent is to correctly partition the covariate space between areas where each action is optimal – the agent hence has to learn this decision boundary accurately and quickly to achieve a high reward.

As previously mentioned, exploration in the one-armed bandit with covariates problem involves selecting unknown action a_1, when action a_2 is expected to yield a larger reward, given the observed covariate value $\boldsymbol{x}(t)$. To illustrate the need to control the amount of exploration performed in this problem, in Figure 4.1 we show expected rewards using the ϵ-first policy with the 10-dimensional covariate problem studied in Section 4.1.2, for various values of the noise variance σ_η^2. As in Section 3.1.5, we normalise the rewards between 0 and 1, where 1 is the expected reward to an oracle that knows all parameters and 0 is the expected reward of a random policy. The optimal value of ϵ (denoted by a star) is highly dependent on the level of noise variance, and furthermore the performance of ϵ-first can degrade quickly

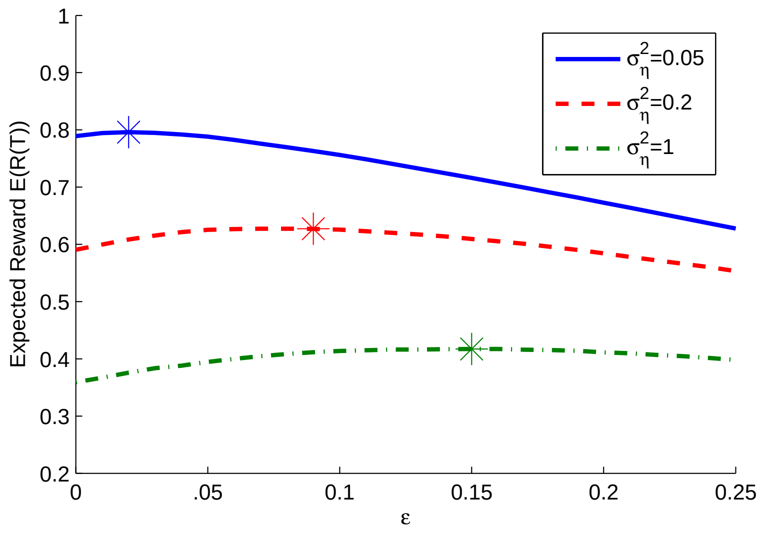


Figure 4.1: Expected reward of the ϵ-first policy (averaged over 10,000 repeats) for $0 \leq \epsilon \leq 0.25$ in the 10-dimensional setup used in Section 4.1.2.

for badly tuned ϵ, particularly when the amount of noise is low – this motivates the construction of an algorithm to control exploration on-line, which we describe in the next section.

4.1.1 The ϵ-ADAPT Algorithm

In this section, we construct an on-line algorithm, ϵ-ADAPT, that learns to effectively control the amount of exploration in the one-armed bandit with covariates problem. At each iteration, the agent updates its predictions of the unknown parameters of the reward function and the covariate. For example, with a multivariate Gaussian covariate, the agent maintains estimates $\hat{\boldsymbol{\mu}}_x$ and $\hat{\Sigma}_x$, along with $\hat{\boldsymbol{\alpha}}$ and $\hat{\sigma}_\eta^2$, using the sample estimates from the past history of interactions. These statistics indicate the likelihood of each action being optimal for future covariate values and the amount of uncertainty the agent has regarding this. It is this uncertainty that dictates the likelihood with which the agent will explore or exploit at each iteration.

In an on-line setting, the agent only needs to select an action for the next iteration, and does not have to submit a policy for the remainder of the game. Nevertheless, the agent cannot ignore the possible action choices that follow the current one, which presents the agent with an intractable calculation. To combat this issue, we use the ϵ-first policy as the building block of our algorithm (as motivated in the previous chapters). In particular, we make use of Theorem 4.1 – a new theorem regarding the optimality of ϵ-first policies in finite time.

Theorem 4.1 $R_{\epsilon f}(T, c/T) > R_{\epsilon f}(T, 0)$ *and* $c \geq 1 \Rightarrow R_{\epsilon f}(T, 1/T) > R_{\epsilon f}(T, 0)$ *(for all* $T, c \in \mathbb{Z}^+$*).*

$R_{\epsilon f}(T, c/T)$ is the expected cumulative reward of the ϵ-first policy for a game of length T, where $\epsilon = c/T$. This theorem states that if an ϵ-first policy that explores for one or more iterations outperforms a greedy policy then an ϵ-first policy that explores for exactly one iteration also outperforms a greedy policy. This property can be clearly seen in Figure 4.1 where the expected reward of the ϵ-first policy is always monotonically increasing between $\epsilon = 0$ and the optimal ϵ value.

Proof *(sketch)*: Suppose that $R_{\epsilon f}(T, 1/T) < R_{\epsilon f}(T, 0)$. As the two policies are identical after one iteration (i.e. they are both greedy), it follows that $R_{\epsilon f}(T, 0)$ selects the known action first and $R_{\epsilon f}(T, 1/T)$ selects the unknown action first. The proof now has two key steps:

1. If the game were to be one iteration shorter, then $R_{\epsilon f}(T-1, 1/(T-1)) < R_{\epsilon f}(T-1, 0)$. To see this consider removing the last action at time T, which is a greedy action. This subtracts more from the reward of $R_{\epsilon f}(T, 1/T)$ than $R_{\epsilon f}(T, 0)$), as $R_{\epsilon f}(T, 1/T)$ would have a larger number of samples with which to estimate α and hence a reduced probability of error (and hence higher expected reward) in this final iteration.

2. It then follows that $R_{\epsilon f}(T, 2/T) < R_{\epsilon f}(T, 1/T)$. This is because adding an extra exploration step at the beginning will add more reward to $R_{\epsilon f}(T-1, 0)$ than $R_{\epsilon f}(T-1, 1/(T-1))$, as there are $T-1$ proceeding exploitation steps (rather than $T-2$), so the increased knowledge of α, and hence reduced probability of error, benefits more future iterations.

We can then consider a game two iterations shorter and continue this inductive process ($c-1$ times) to show $R_{\epsilon f}(T, c/T) < R_{\epsilon f}(T, 0)$, which completes the proof.
$\blacksquare$

The significance of this result is that all the agent now needs to compute at time t is which policy is expected to yield a larger reward: $R_{\epsilon f}(T^*, 0)$ or $R_{\epsilon f}(T^*, 1/T^*)$, where $T^* = T - t + 1$ (the length of the game remaining). This follows because if any more exploration needs to be performed in a static problem, it should be performed immediately. Therefore, from Theorem 4.1, the agent needs to explore if and only if an ϵ-first policy with one initial exploration step outperforms a greedy policy (or ϵ-first with 0 initial exploration steps) for a game starting at time t. If the agent could calculate this exactly, the optimal on-line policy can be computed using Algorithm 4.1 – an algorithmic representation of the optimal ϵ-first policy.

Algorithm 4.1 Optimal on-line policy

```
1:  for $t = 1$ to $T$ do
2:      Observe $\boldsymbol{x}(t)$ {Covariate}
3:      Update unknown parameters of covariate distribution
4:      if $t \leq D$ or $R_{\epsilon f}(T^*, 1/T^*) > R_{\epsilon f}(T^*, 0)$ or $E[r_1(t)|\boldsymbol{x}(t), \hat{\boldsymbol{\alpha}}] >$
        $E[r_2(t)|\boldsymbol{x}(t), \boldsymbol{\beta}]$ then
5:          Select action $a_1$ and receive reward $r_1(t)$
6:          Update $\hat{\boldsymbol{\alpha}}$ and $\hat{\sigma}_\eta^2$ {Only updated when $a_1$ is selected}
7:      else
8:          Select action $a_2$ and receive reward $r_2(t)$
9:      end if
10: end for
11: $D$ is the required length of initialisation for sample estimates to exist.
```

This policy receives exactly the same reward as the optimally tuned off-line ϵ-first policy, but requires knowledge of all the unknown parameters. The challenge therefore lies in approximating the unknown expected rewards, $R_{\epsilon f}(T^*, 1/T^*)$ and $R_{\epsilon f}(T^*, 0)$. We use sample estimates so that the agent, at time t, can perform a Monte Carlo (MC) simulation of the rest of the game, to see which policy yields the highest expected cumulative reward. This involves generating future covariate values ($\boldsymbol{x'}(s)$) and rewards ($r'_i(s)|\boldsymbol{x'}(s), \boldsymbol{\beta}, \hat{\boldsymbol{\alpha}}, \hat{\sigma}_\eta^2$) for $i = 1, 2$ and $s = 1, \ldots, T^*$, and then simulating the game with each policy to see which performs better. An alternative approach would be to to try and find these values analytically, but this in an intractable calculation – as the number of future action sequences rises exponentially with the length of the game. The analytical expressions can be approximated, but these can only be done case-by-case dependent on the covariate distribution and type of reward function. We therefore resort to MC approximation due to its simplicity and generality.

The shortcoming of the Monte Carlo approach, however, is that if we simulate the rest of the game from time t using the sample estimates, then the greedy approach will always outperform ϵ-first. This is because the MC estimate of $\hat{\boldsymbol{\alpha}}$ (denoted $\bar{\boldsymbol{\alpha}}$) will already have converged to $\hat{\boldsymbol{\alpha}}$ and is then used to drive future rewards, so exploration is deemed not to be required. To avoid this, we need to introduce uncertainty in the estimate of $\hat{\boldsymbol{\alpha}}$, which reflects the uncertainty in the true game. So in addition to simulating the rest of the game, we also regenerate covariate values and rewards that were used to estimate $\hat{\boldsymbol{\alpha}}$ prior to time t, such that the MC estimate $\bar{\boldsymbol{\alpha}}$ is perturbed from the true sample estimate. This creates uncertainty in the simulated game that mimics the uncertainty in the true game, and provides a reason to explore. The on-line approximation of $R_{\epsilon f}(T^*, 0)$ (and $R_{\epsilon f}(T^*, 1/T^*)$) follows Algorithm 4.2. We note that an alternative method is to directly sample the MC estimate of $\bar{\boldsymbol{\alpha}}$ at time t (rather than resampling covariates prior to time t), but this can only be done when an analytical representation of the distribution of $\hat{\boldsymbol{\alpha}}$ can be found – which is only easily attained in the 1-dimensional problem considered in Chapter 3.

Our algorithm for controlling exploration on-line, ϵ-ADAPT, follows Algorithm 4.1, with $R_{\epsilon f}(T^*, 0)$ and $R_{\epsilon f}(T^*, 1/T^*)$ approximated using Algorithm 4.2. The MC computation can in fact be repeated several times to smooth the estimates of the two competing policies, but this is not necessary for our algorithm to work. In fact, in the simulations performed in Section 4.1.2, the MC estimate was only repeated twice at each iteration, as more repeats had no particular extra benefit – an interesting result which we discuss more in Section 4.2 for the full multi-armed bandit problem.

The covariate value at time t is *not* replaced by a new sample in Algorithm 4.2, but kept at the true observed value – a key component of ϵ-ADAPT. This allows ϵ-ADAPT to decide which regions of the covariate space are worth exploring and which are not. For example, if the expected reward of the unknown action is

Algorithm 4.2 On-line MC approximation of $R_{\epsilon f}(T^*, 0)$ (and $R_{\epsilon f}(T^*, 1/T^*)$)

```
1: Inputs: $t$ (current time-step), $T$ (length of game) $n$ (no. of times $a_1$ selected
   prior to time $t$), sample estimates ($\hat{\boldsymbol{\alpha}}, \hat{\sigma}^2_\eta, \ldots$)
2: for $s = 1$ to $T^* + n$ do
3:   Generate $\boldsymbol{x'}(s)$ {New Covariate}
4:   if $s = n + 1$ then
5:     $\boldsymbol{x'}(s) = \boldsymbol{x}(t)$ {True covariate value kept at time $t$ only}
6:   end if
7:   if $s \leq n$ or $E[r'_1(s)|\boldsymbol{x'}(s), \bar{\boldsymbol{\alpha}}] > E[r'_2(s)|\boldsymbol{x'}(s), \boldsymbol{\beta}]$ then
8:     Select action $a_1$ and receive reward $r'(s) = r'_1(s)$. Update $\bar{\boldsymbol{\alpha}}$
9:   else
10:    Select action $a_2$ and receive reward $r'(s) = r'_2(s)$
11:  end if
12: end for
13: $R_{\epsilon f}(T^*, 0) = \sum_{s=n+1}^{T^*+n} r'(s)$ {MC approximation}
14: Replace $n$ with $n + 1$ in Line 7 to calculate approximation of $R_{\epsilon f}(T^*, 1/T^*)$.
```

only marginally smaller than the known action (given the covariate value), then the benefits of exploration (through increased learning of parameters $\boldsymbol{\alpha}$ and σ^2_η) can outweigh the costs (the myopic loss of selecting a sub-optimal action). Whereas with other covariate values the short-term costs might exceed the long-term benefits.

This includes a notion of cost-inclusive exploration to ϵ-ADAPT, where at earlier steps the algorithm is willing to forego a lot of reward for 1 exploration step whereas later exploration is only worthwhile if the cost to the reward is negligible. In this sense, ϵ-ADAPT attempts to detect *when* best to explore and not just how much. This is something that ϵ-greedy and ϵ-first policies are not able to do, as they are off-line policies, and further motivates the use of an on-line exploration algorithm.

ϵ-ADAPT is based on the ϵ-first policy, but does not require any parameters to be set *a priori* that govern the amount of total exploration. ϵ-ADAPT only requires parameters that are used within the estimation module or for the distributions specified in the MC approximation, but these can be estimated during play, and do not need to be set *a priori*. In addition, the need for MC distributions can be removed (by using nonparametric bootstrapping techniques (Efron and Tibshirani, 1993) for example) and we nevertheless provide evidence in Section 4.1.2 that ϵ-ADAPT can work well even with misspecified modelling assumptions.

Occasionally ϵ-ADAPT might undershoot (due to poor sample estimates and/or error from the MC approximation) and start exploiting too early. For these reasons, we continue to decide whether to explore or exploit at *every* iteration until the end of the game. In this way, ϵ-ADAPT is naturally self-correcting and will explore at a later stage (if necessary) to compensate for any lack of earlier exploration.

ϵ-ADAPT is computationally efficient, scaling quadratically in T (as the MC approximation in Algorithm 4.2 is of maximum length T and is repeated T times). The relationship with the dimensionality of the covariate p depends on the inference procedure used to estimate the parameter $\boldsymbol{\alpha}$. For linear reward models, least squares can be used (order p^3) or recursive least squares (Haykin, 2002) (order p^2) if a further saving is required (at a marginal increase to the error of the estimates for low sample-length data), see Appendix B for the full algorithm. ϵ-ADAPT can handle non-linear reward functions, using techniques such as non-linear regression. In addition, the algorithm can deal with non-Gaussian covariates and error terms by either explicitly coding them in (where tractable) or by using nonparametric bootstrapping techniques. In the case of bootstrapping, covariate values and rewards could be resampled (with replacement) from the original dataset of observed side information and past rewards. This is particularly applicable in our setting where covariate information is generally assumed to be i.i.d. over time.

4.1.2 Numerical Results

In this section, we test ϵ-ADAPT for the one-armed bandit with covariates problem. We first compare ϵ-ADAPT against the on-line algorithm developed in Section 3.5 for the one-dimensional covariate problem, to examine the benefits of using the MC approximation technique. We then consider linear reward functions with both a 5-dimensional and 10-dimensional covariate, to test the ability of ϵ-ADAPT to learn from a higher-dimensional covariate. Finally, we also test the robustness of ϵ-ADAPT to a misspecified noise model, where the noise is assumed to be normally distributed, but is in fact either t-distributed (heavy-tailed) or gamma distributed (skewed). This tests the dependence of ϵ-ADAPT on correct modelling assumptions in the MC approximation. As this chapter considers static problems, we compare ϵ-ADAPT against the ϵ-first policy in all numerical experiments in this chapter. All rewards are again normalised between 0 and 1, where 0 is the average reward of a policy that always randomly selects between actions and 1 is the average reward of an oracle that knows all reward coefficients beforehand.

- *1-dimensional Covariate*

We first repeat the numerical experiments of Section 3.5 with the ϵ-ADAPT algorithm for a game of length $T = 100$. We consider this game-length throughout this chapter for two reasons. First, static problems are likely to be short-length in real-world applications, because the environment often changes and the decision problem evolves to a new one. Secondly, we want to test the performance of algorithms in finite time and expose policies and algorithms that only perform well after several time-steps (many of which will have optimal asymptotic behaviour).

Table 4.1 displays the average rewards and average rates of exploration of ϵ-ADAPT (over 20,000 repeats), along with the results for the optimal ϵ-first policy and the alternative on-line algorithm. ϵ-ADAPT is the best performing algorithm in each case. Notice also, that the average number of exploration steps has increased much more than with the alternative approach, as the noise increases, which suggests the algorithm is being driven to a greater extent by the uncertainty surrounding the sample estimates.

Table 4.1: Comparison of on-line and optimal off-line policies (with ϵ-ADAPT)

σ_x^2/σ_η^2	Off-line (ϵ-first)		On-line Algorithm of Section 3.5			ϵ-ADAPT		
	Opt. ϵ	Reward	Reward	% Optimal	Avg. exp.	Reward	% Optimal	Avg. exp.
5	0.03	0.888	0.894	100.8%	1.43	0.897	101.0%	1.33
2	0.06	0.769	0.776	100.8%	1.84	0.783	101.7%	2.78
1	0.08	0.640	0.640	100.0%	2.04	0.659	102.9%	5.12
0.5	0.10	0.501	0.501	100.1%	2.14	0.514	102.7%	6.28
0.2	0.13	0.340	0.331	97.5%	2.21	0.370	108.8%	7.35

To gain further insight, in Figure 4.2(a) we show the rate of exploration over time (averaged over all simulations) for various values of σ_x^2/σ_η^2. In contrast to

Figure 3.5(a), notice that the rate of exploration decays more slowly and does not reach zero until the final time-step. This is due to the fact that the probability of error will be non-zero for the duration of the game and exploration can be worthwhile, even at a late stage, with certain covariate values and coefficent estimates – it is only until the final time-step that the algorithm stops exploring and fully exploits. There is also a much bigger difference between the amount of exploration performed throughout for different values of σ_x^2/σ_η^2, again showing that exploration is driven by uncertainty. Figure 4.2(b) displays a histogram of the number of explorative time-steps (over the 20,000 repeats) for the high noise problem. In contrast to Figure 3.5(b), ϵ-ADAPT does not get caught under-exploring as often, which explains the much stronger performance in terms of cumulative reward gained.

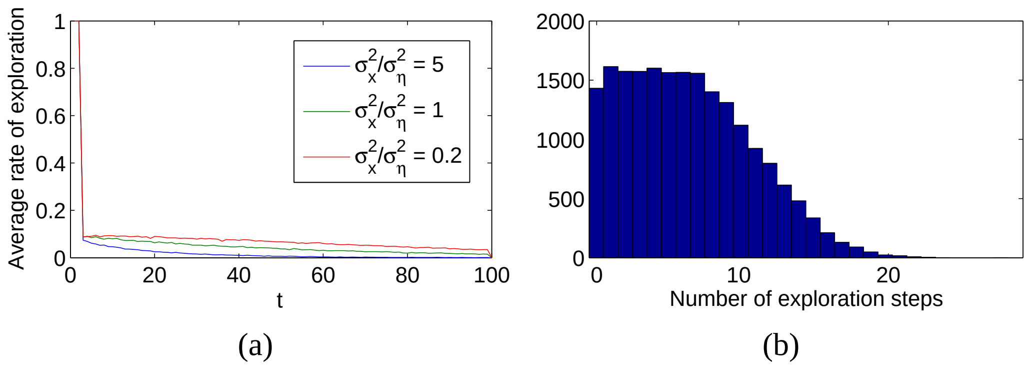


Figure 4.2: Explorative behaviour of ϵ-ADAPT with a 1-D covariate: (a) the average rate of exploration over time and (b) a histogram of the total number of explorative steps (for $\sigma_x^2/\sigma_\eta^2 = 0.2$).

To further explain the strong performance of ϵ-ADAPT in this problem we investigate to what extent the algorithm has determined *when* exploration should be performed, as well as how much overall. Figure 4.3(a) displays a histogram of all covariate values observed in the entire simulation and partitions them according to whether ϵ-ADAPT decides to explore or exploit (where $\sigma_x^2/\sigma_\eta^2 = 1$). First of all, ϵ-ADAPT is usually partitioning the covariate space correctly, by selecting a_1 for $x(t) > 0$ and a_2 otherwise. Interestingly, ϵ-ADAPT appears to explore for the full range of covariate values, but proportionately less so when $x(t)$ is close to zero. This is due to the fact that reward observations in this region are not particularly informative about α, so exploration has little value. This is a feature of this type of reward function, where rewards are fixed through the origin, and does not extend to reward functions where the intercept of the reward function is unknown (as we see in later sections).

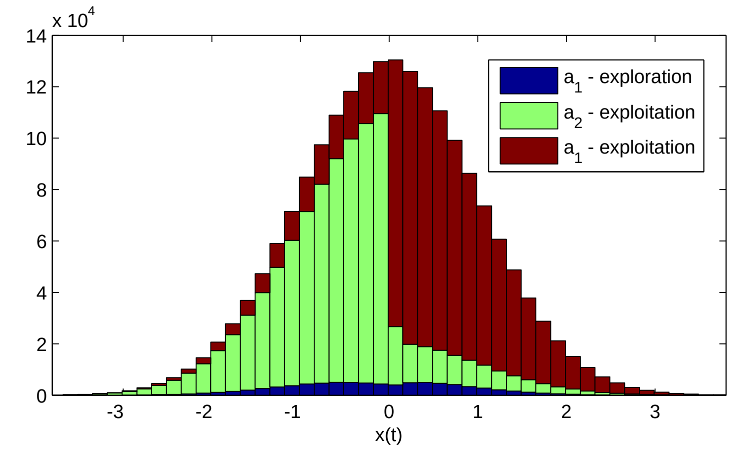
Figure 4.3(b) displays a frequency plot showing the average proportion of times ϵ-ADAPT explores when a_2 is predicted to be optimal (for the covariate bins generated in Figure 4.3(a)). This plot reveals that ϵ-ADAPT is explicitly considering the covariate value when making exploration decisions. Aside from less exploration near $x(t) = 0$, notice that the algorithm explores more for positive covariate values – this is where the greedy action is in fact suboptimal, and the algorithm is responding to this by exploring more. This is a key feature of ϵ-ADAPT and explains its strong performance. When the covariate is positive, but the greedy action determines a_2 is optimal, then ϵ-ADAPT is likely to have high noise variance estimates and the difference between $\hat{\alpha}$ and β is likely to be small. These factors make exploration more likely and mean ϵ-ADAPT can correct erroneous greedy decisions more effectively then an ϵ-first or ϵ-greedy policy. In addition, exploration is more likely for covariate values that are closer to (but not near) $x(t) = 0$ rather than values from the tails of the distribution, as this is deemed less costly to the immediate reward. ϵ-ADAPT therefore also incorporates the cost of exploration, a feature of the algorithm we investigate more in future sections.

- *5-dimensional Covariate*

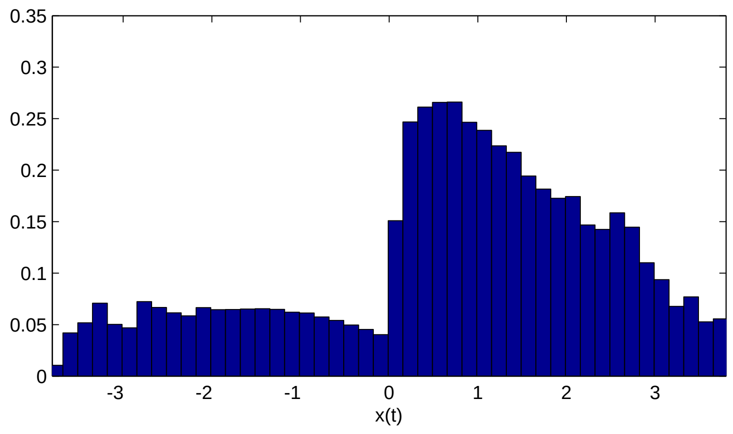
We now test ϵ-ADAPT for a 5-dimensional covariate using linear reward functions given by:

$$r_1(t) = \sum_{i=1}^{p} \alpha_i x_i(t) + \eta_t, \quad r_2(t) = \sum_{i=1}^{p} \beta_i x_i(t) + \nu_t,$$

where $\boldsymbol{\beta}$ is known and $\boldsymbol{\alpha}$ is unknown. p is the dimension of the covariate where $x_1(t) = 1$ (so that α_1 becomes the intercept of the reward plane) and $x_{2,\dots,p}(t) \sim \mathcal{N}(\boldsymbol{\mu}_x, \Sigma_x)$. A multivariate normal covariate is used as this distribution can accurately model several real-world data sources (Cox and Small, 1978), but this is not



(a)



(b)

Figure 4.3: Histograms of (a) the covariate distribution partitioned into regions of exploration/exploitation and (b) the frequency of explorative actions when a_2 is greedy optimal, using ϵ-ADAPT on a 1-D covariate problem.

a requirement for our algorithm to work. We tested ϵ-ADAPT over 20,000 repeated simulations for a game of length 100 with the following parameter values:[1]

[1]In this section we set $\boldsymbol{\beta}$ to be a vector of zeros, without loss of generality. Any problem can be transformed to this by subtracting $\sum_{i=1}^{p} \beta_i x_i(t)$, which is known, from each observed reward.

$$\boldsymbol{\alpha} = \begin{bmatrix} 0 \\ 0.2 \\ -0.2 \\ 0.1 \\ -0.1 \end{bmatrix} \boldsymbol{\mu}_x = \begin{bmatrix} -0.1 \\ 0.2 \\ 0.3 \\ -0.3 \end{bmatrix} \Sigma_x = \begin{bmatrix} 1 & 0.1 & -0.2 & 0.5 \\ 0.1 & 0.2 & -0.2 & 0 \\ -0.2 & -0.2 & 0.4 & 0.1 \\ 0.5 & 0 & 0.1 & 0.8 \end{bmatrix}$$

The parameter values are selected such that each action is optimal in approximately 50% of the covariate space, which maximises the importance of learning the correct decision boundary quickly. We simulated the problem for various values of the noise variance σ_η^2 and quantify this using the *covariance to noise ratio* (Pavlidis et al., 2008b), CNR= $\frac{||\Sigma_x||_1}{\sigma_\eta^2}$, where $||\Sigma_x||_1$ is the 1-norm of Σ_x. The larger the CNR, the more informative observations are about $\boldsymbol{\alpha}$, making the learning problem easier. The only other parameters we could change are the reward coefficients – this has a similar effect to changing the CNR though, in that separating the distance between actions dissipates the effect of noise (and vice-versa). We choose to report CNR values, however, as this allows our results to be commensurate across dimensions. Note that the CNR values for the experiments performed in the last section with a 1-dimensional covariate (and in Section 3.5) are simply equal to σ_x^2/σ_η^2.

Table 4.2 displays results for ϵ-ADAPT and several ϵ-first policies for various CNR values (where CNR values are set with consideration of the range of ϵ-values considered). Lower CNR values correspond to lower rewards for each policy, as the learning problem is more difficult. For each CNR value, however, ϵ-ADAPT performs close to the best performing ϵ-first policy.[2] Moreover, ϵ-ADAPT is the best overall when the rewards are averaged, even though these problems naturally favour exploration rates of 5-10% (which will not always be the case).

Table 4.3 compares ϵ-ADAPT with the optimally tuned off-line ϵ-first policy from the same set of experiments. The reward is always within 95% of the off-line optimal, although the performance degrades as the noise increases – this is because

[2]Note that the optimal ϵ in all future experiments in this thesis is now found empirically rather than theoretically.

Table 4.2: Average rewards with a 5-D covariate

CNR	$\epsilon = 0$	0.02	0.05	0.1	0.15	0.2	ϵ-ADAPT
100	0.845	**0.857**	0.850	0.816	0.775	0.730	0.856
50	0.772	0.796	**0.802**	0.783	0.750	0.710	0.800
20	0.630	0.661	0.687	**0.696**	0.680	0.654	0.685
10	0.501	0.535	0.567	0.590	**0.591**	0.576	0.574
5	0.383	0.411	0.442	0.470	**0.480**	0.475	0.458
Avg.	0.626	0.652	0.670	0.671	0.655	0.629	**0.675**

the errors in the sample estimates are larger, yielding on-line approximations that are not as accurate. Nevertheless, lower values of the CNR require more exploration from the agent, and ϵ-ADAPT has responded to this by performing more exploration steps on average (last column). The performance of ϵ-ADAPT no longer exceeds that of the optimal ϵ-first policy, as was the case in the 1-dimensional problem. This is due to the increased number of reward coefficient estimates and covariate distribution parameters used in the MC approximation of Algorithm 4.2, which make optimal exploration decisions harder to learn on short time-scales. Nevertheless, ϵ-ADAPT still performs best on average and we further test the robustness of this algorithm with a 10-dimensional covariate in the next section.

Table 4.3: Comparison of ϵ-ADAPT and optimal ϵ-first with a 5-D covariate

CNR	Off-line (ϵ-first)		On-line (ϵ-ADAPT)	
	Opt. ϵ	Reward	% Optimal	Avg. exp. steps
100	0.02	0.857	99.9%	3.04
50	0.04	0.803	99.7%	4.17
20	0.09	0.697	98.3%	5.76
10	0.13	0.592	96.9%	6.78
5	0.15	0.480	95.5%	7.55

Figure 4.4 displays the average amount of exploration performed at each time-step t (after initialisation) for various CNR values. As in the 1-dimensional case, the amount of exploration performed is naturally higher for low CNR values and generally decreases as the game is played. For low CNR values, there is initially a small increase in exploration over time. This is due to the small number of sample estimates when the ϵ-ADAPT algorithm commences, leading to potential under-

estimates in the amount of noise and subsequent under-exploration. This is soon corrected however, as the algorithm learns that more exploration is needed to correctly partition the covariate space. This may seem to be a weakness of the algorithm, but after relatively few sample estimates, it is difficult to distinguish between the different types of problems. Forcing the algorithm to explore more initially would reduce the rewards for problems with higher CNR values where high rates of exploration are not needed. ϵ-ADAPT is therefore adapting to the game as it plays.

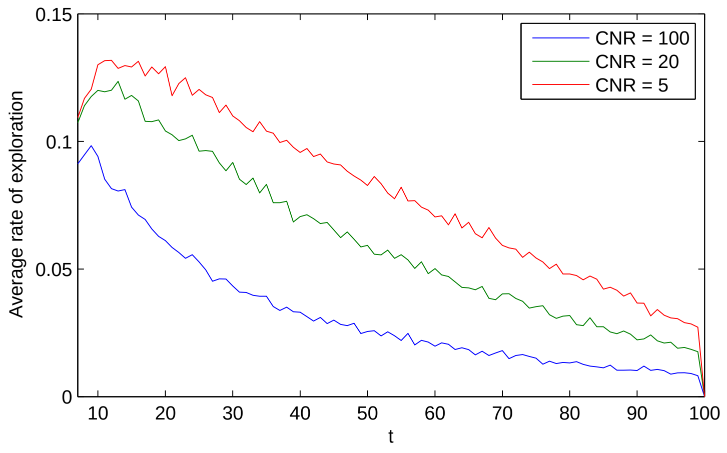


Figure 4.4: Average rate of exploration performed at time t with a 5-D covariate using ϵ-ADAPT, for various CNR values. Rates for $t < 7$ are not reported as this is during the initialisation period.

Figure 4.5 displays histograms of the number of exploration steps performed within a game for low and high CNR values – the spread is due to both the noise in the sample estimates (and subsequent MC approximation) and the circumstances of each game (a favourable start to the game means less exploration needs to be performed later and vice-versa). Overall, ϵ-ADAPT rarely gets caught under-exploring, as the algorithm naturally self-corrects if it learns the incorrect partitioning of the covariate space.

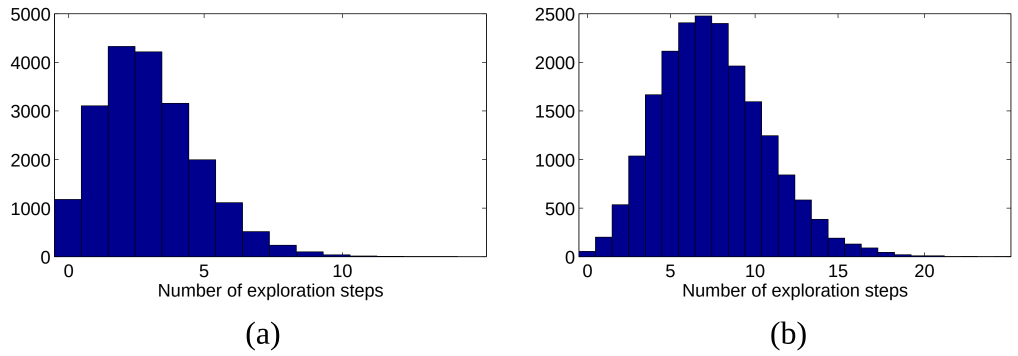


Figure 4.5: Histograms showing the total number of exploration steps performed with a 5-D covariate by ϵ-ADAPT for CNR values of (a) 100 (low noise) and (b) 5 (high noise).

- *10-dimensional Covariate*

To test the robustness of ϵ-ADAPT to a high-dimensional covariate, we repeat the same experiments using a 10-dimensional covariate with parameters:

$$\begin{aligned}
\boldsymbol{\alpha} &= \begin{bmatrix} 0.1 & 0.4 & -0.4 & 0 & 0.4 & -0.2 & 0.4 & -0.3 & 0.1 & -0.1 \end{bmatrix}, \\
\boldsymbol{\mu}_x &= \begin{bmatrix} -0.5 & -0.2 & 0.1 & 0.4 & 0.2 & -0.4 & 0 & 0.3 & -0.3 \end{bmatrix}, \\
\Sigma_x &= \mathrm{diag}\left(\begin{bmatrix} 0.5 & 0.3 & 0.7 & 0.1 & 0.9 & 0.8 & 0.1 & 1 & 0.1 \end{bmatrix} \right).
\end{aligned}$$

We choose a diagonal matrix for Σ_x to maximise the effect of the increased dimensionality on the learning problem, and the remaining values are set such that each covariate has a different impact on the reward function. Tables 4.4 and 4.5 display the expected rewards for the same on-line and off-line policies, where the magnitude of CNR values has been deliberately reduced by increasing the noise. This is because higher dimension problems require less exploration (see Pavlidis et al. (2008b) for a detailed explanation). ϵ-ADAPT has again yielded a reward that is within 95% of the optimal and furthermore, has performed best on average against the range of ϵ-first policies considered. The performance of ϵ-ADAPT has not been affected by the increased number of parameters it is required to learn as the algorithm is robust to any p-dimensional covariate.

Table 4.4: Average rewards with a 10-D covariate

CNR	$\epsilon = 0$	0.02	0.05	0.1	0.15	0.2	ϵ-ADAPT
20	0.832	**0.833**	0.817	0.777	0.732	0.685	0.831
10	0.788	**0.795**	0.787	0.755	0.715	0.672	0.791
5	0.718	0.731	**0.734**	0.715	0.684	0.646	0.731
2	0.586	0.605	0.620	**0.622**	0.606	0.581	0.618
1	0.471	0.490	0.509	**0.522**	0.518	0.504	0.511
Avg.	0.679	0.691	0.693	0.678	0.651	0.618	**0.696**

Table 4.5: Comparison of ϵ-ADAPT and optimal ϵ-first with a 10-D covariate

CNR	Off-line (ϵ-first)		On-line (ϵ-ADAPT)	
	Opt. ϵ	Reward	% Optimal	Avg. exp. steps
20	0.01	0.835	99.6%	2.36
10	0.02	0.795	99.5%	3.31
5	0.04	0.735	99.5%	4.42
2	0.08	0.624	99.0%	5.87
1	0.11	0.522	97.9%	6.73

- *Misspecified Noise Models*

Finally, we explore the behaviour of ϵ-ADAPT when assumptions fail (as they may in real-world applications). In particular, we look at two common departures from a Gaussian noise model – asymmetric and heavy-tailed noise distributions. Specifically, we generate the noise process using $t(3)$ (heavy-tailed) and gamma$(2, \theta)$ (skewed) distributions (where the gamma distribution is recentred at mean zero and the first parameter is the shape parameter and the second is the scale which we vary in simulations). This allows us to check whether ϵ-ADAPT is robust to misspecified noise models. We ran simulations for the 10-dimensional covariate with the alternative noise models, and scaled the noise so that we used the same range of CNR values. The results are presented in Table 4.6. As can be seen, the performance of ϵ-ADAPT has not been affected despite false assumptions regarding the noise model – which is a particularly desirable type of robustness if these methods are to be applied to real-world problems.

Table 4.6: Performance of ϵ-ADAPT with misspecified noise models

	t(3) noise			gamma(2,θ) noise		
CNR	ϵ-first		ϵ-ADAPT	ϵ-first		ϵ-ADAPT
	Opt. ϵ	Reward	% Opt.	Opt. ϵ	Reward	% Opt.
20	0.01	0.841	99.7%	0.01	0.836	99.5%
10	0.01	0.805	99.7%	0.02	0.799	99.3%
5	0.03	0.751	99.8%	0.03	0.743	99.4%
2	0.06	0.649	99.6%	0.07	0.634	99.0%
1	0.09	0.554	98.9%	0.10	0.532	98.7%

4.2 The Multi-Armed Bandit with Covariates Problem

In this section we extend ϵ-ADAPT so that it can be applied to multi-armed bandit problems, where there is more than one unknown action available. This is achieved by constructing an index for each action on-line, and then selecting at each iteration the action with the highest index value, in a similar vein to the Gittins Indices (as described in Section 2.2.4). Using this method, ϵ-ADAPT is then able to reason about *which action* to explore in addition to learning *when* to explore and *how much* overall.

In the multi-armed bandit with covariates problem, the agent selects between actions $\{a_i, i = 1, \ldots, k\}$, with rewards given by:

$$r_i(t) = f_i(\boldsymbol{x}(t), \boldsymbol{\alpha}_i) + \eta_i(t), \tag{4.3}$$

where, as in the one-armed case, $\boldsymbol{x}(t)$ is a p-dimensional covariate observed at time t and $\eta_i(t)$ are i.i.d. noise processes which are normal and centred at zero with variance σ_i^2. The agent is assumed to know the form of each function f_i, but not the reward coefficients $\boldsymbol{\alpha}_i$ – which is what the agent must learn to correctly partition the covariate space between the actions.

We seek to extend the ϵ-ADAPT approach of Section 4.1 to multiple action problems. Theorem 4.1 also holds for multi-armed problems, so we can construct ϵ-ADAPT based on the ϵ-first policy as before. In the simplest case, this could be performed by following Algorithms 4.1 and 4.2 in exactly the same way, except that $R_{\epsilon f}(T^*, 1/T^*)$ is instead approximated by randomly selecting one of the non-greedy actions at time t in the MC approximation of Algorithm 4.2. Then if $R_{\epsilon f}(T^*, 1/T^*) > R_{\epsilon f}(T^*, 0)$, a random non-greedy action is also selected in the actual game. This method retains the characteristics of ϵ-ADAPT for one-armed problems, in that the agent controls *how much* and *when* exploration is performed, but has the same disadvantages as the ϵ-first and ϵ-greedy policies in that exploration is performed randomly. So to further improve on the ϵ-first methodology we also choose *which action* should be explored at each time-step, such that globally suboptimal actions (actions that are suboptimal for all covariate values) are never selected again after sufficient exploration, allowing other actions to be explored more. We do this by constructing an index for each action $R_{\epsilon f}(T, t, i)$, which is the expected reward of selecting action i next, and then selecting greedily for the remainder of the game (length $T - t + 1$).

We therefore define ϵ-ADAPT as an algorithm that sequentially approximates a value ($R_{\epsilon f}(T, t, i)$) for each action, and then selects the action with the highest value at each iteration. We provide the pseudo-code for ϵ-ADAPT in Algorithm 4.3 and the MC approximation of $R_{\epsilon f}(T, t, i)$ in Algorithm 4.4. Note that D is the required number of times each action must be selected during initialisation. With linear rewards (see Equation (2.3)) for example, $D = p + 1$, such that ϵ-ADAPT has estimates of the noise variance σ_i^2 before the indices are calculated.

We generate past and future rewards and covariates in exactly the same way as before, by sampling from the covariate distribution using the sample estimates updated in Line 5 of Algorithm 4.3. Again, we retain the same covariate value at time t, so that the agent can decide *when* to explore (Line 10, Algorithm 4.3). ϵ-ADAPT can then calculate the indices $R_{\epsilon f}(T, t, i)$ for each action, as detailed in Algorithm 4.4. The MC approximation simulates the game for the full duration T, but uses only rewards gained from time t onwards to find $R_{\epsilon f}(T, t, i)$. In the time-steps before t, each action must be selected the number of times it has been selected in the actual game, such that the uncertainty of each individual action in the approximation mimics the true uncertainty. It is in this way that ϵ-ADAPT can then determine which action to explore.

Algorithm 4.3 ϵ-ADAPT for the Multi-Armed Bandit

```
1:  $\boldsymbol{C} = \mathbf{0}$ {Initialise Action count}
2:  for $t = 1$ to $T$ do
3:    $T^* = T - t + 1$
4:    Observe covariate $\boldsymbol{x}(t)$
5:    Update unknown parameters of covariate distribution
6:    if $t \leq kD$ then
7:      Select action $i$ where $i = 1 + t \bmod k$ {Initialisation}
8:    else
9:      Generate new Covariates $\boldsymbol{x'}(s)\ \forall 1 \leq s \leq T$
10:     $\boldsymbol{x'}(t) = \boldsymbol{x}(t)$ {Keep same covariate at time $t$}
11:     Generate new rewards $r'_i(t)\ \forall 1 \leq i \leq k, 1 \leq t \leq T$ using estimated
        reward coefficients $\hat{\boldsymbol{\alpha}}_i$ and estimated noise variance $\hat{\sigma}^2_i$
12:     for $i = 1$ to $k$ do
13:       Approximate $R_{\epsilon f}(T, t, i)$ {Algorithm 4.4}
14:     end for
15:     Select action $i$ ($1 \leq i \leq k$) that maximises $R_{\epsilon f}(T, t, i)$
16:   end if
17:   Receive reward $r(t) = r_i(t)$
18:   Update $\hat{\boldsymbol{\alpha}}_i$ and $\hat{\sigma}^2_i$
19:   $C(i) = C(i) + 1$ {Update action count}
20: end for
```

The ϵ-ADAPT approach is somewhat similar to the Gittins Indices (see Section 2.2.4), in that an index is attributed to the value of selecting each action. Our method, however, offers three important advantages. First, we do not require that rewards are discounted over time. Secondly, Gittins Indices do not generalise to include covariates, and are restricted to Bernoulli or normally distributed rewards for the non-covariates setting – where their extension to other reward distributions are non-trivial. Finally, ϵ-ADAPT is significantly less expensive to compute. Specifically, ϵ-ADAPT scales linearly in the number of arms k and quadratically in the length of the game T – this is computationally more expensive than ϵ-first (which is linear in T), but is a necessary cost to remove the need for an exploration parameter. Note also that the estimation of reward coefficients $\alpha_{i,j}$ for a linear reward model is quadratic in the dimension p of the covariate $x(t)$, if we use recursive least squares (see Appendix B).

Algorithm 4.4 MC Approximation of $R_{\epsilon f}(T, t, i)$ index

```
1:  for $s = 1$ to $T$ do
2:      if $s < t$ then
3:          Select an action $i$ such that each action is selected $C(i)$ times when $s = t$
            and receive reward $r'(s) = r'_i(s)$
4:          Update $\bar{\boldsymbol{\alpha}}_i$
5:      else if $s = t$ then
6:          Select action $i$ and receive reward $r'(s) = r'_i(s)$
7:          Update $\bar{\boldsymbol{\alpha}}_i$
8:      else
9:          Select action $j$ that maximises $\mathrm{E}\big(r'_j(s)|\bar{\boldsymbol{\alpha}}_j, \boldsymbol{x'}(s)\big)$ ($1 \leq j \leq k$) and receive
            reward $r'(s) = r'_j(s)$
10:         Update $\bar{\boldsymbol{\alpha}}_j$
11:     end if
12: end for
13: $R_{\epsilon f}(T, t, i) = \sum_{s=t}^{T} r'(s)$
```

Finally, the MC approximation of Algorithm 4.4 can be repeated several times, and the indices can be averaged, to smooth the overall estimates. In practice however, when performing numerical tests we again found that 2 repeats were usually sufficient to find the best action to explore, and more repeats had no particular extra benefit. This is because if the optimal action is not clear from only a few MC repeats, then it is likely that all of the competing best actions are ‘good choices’ and the cost of picking a marginally suboptimal action is low. Furthermore, if several actions have high uncertainty then those not selected immediately are extremely likely to be selected in subsequent iterations. The ordering of these selections is therefore not of prime importance – identifying which actions and how much to explore is more significant.

4.3 Numerical Results

In this section we test the performance of ϵ-ADAPT for a range of multi-armed bandit problems. We first study the bandit with covariates problem in Section 4.3.1, with varying numbers of actions and covariate dimensions. Then in Section 4.3.2, we evaluate ϵ-ADAPT for the bandit problem with no covariates, which allows a comparison with existing parameter-free methods (which have not been extended for problems with side information) such as the POKER algorithm and the UCB policy (reviewed in Sections 2.2.5 and 2.2.3 respectively). Finally, in Section 4.3.3, we test ϵ-ADAPT with a real data set from the *Content Distribution Network* problem (CDN) where an agent must minimise the sum of delays whilst retrieving data through a network with several sources available. This data set is used as the data is spiky and non-normal. We can therefore test the robustness of ϵ-ADAPT to misspecified modelling assumptions and to real-world non-smooth data sets.

4.3.1 Bandit with Covariates Problem

In this section we test ϵ-ADAPT for the bandit with covariates problem for a 2-armed problem, a 3-armed problem (with a globally suboptimal action) and a 10-armed problem (with a 10-dimensional covariate).

- *2-armed problem*

We start with a basic 2-armed problem with reward functions and covariate distribution as given in Figure 4.6. These are selected such that each action is optimal in 50% of the covariate space – in order to maximise the difficulty of the learning problem faced.

Table 4.7 displays the average rewards for ϵ-ADAPT and various ϵ-first policies and yet again ϵ-ADAPT has performed best overall. Table 4.8 compares ϵ-ADAPT with the optimal ϵ-first policy where ϵ-ADAPT performs particularly well with the lower range of CNR values where the MC approximation is most accurate.

Table 4.7: Average rewards with a 2-armed problem

CNR	$\epsilon = 0$	0.02	0.05	0.1	0.15	0.2	ϵ-ADAPT
200	0.876	**0.894**	0.888	0.849	0.803	0.755	0.906
100	0.843	0.862	**0.868**	0.840	0.796	0.750	0.878
50	0.780	0.808	**0.826**	0.813	0.778	0.737	0.828
20	0.662	0.686	0.719	**0.730**	0.717	0.689	0.720
10	0.529	0.569	0.598	**0.625**	**0.625**	0.609	0.609
Avg.	0.738	0.764	**0.780**	0.771	0.744	0.708	0.788

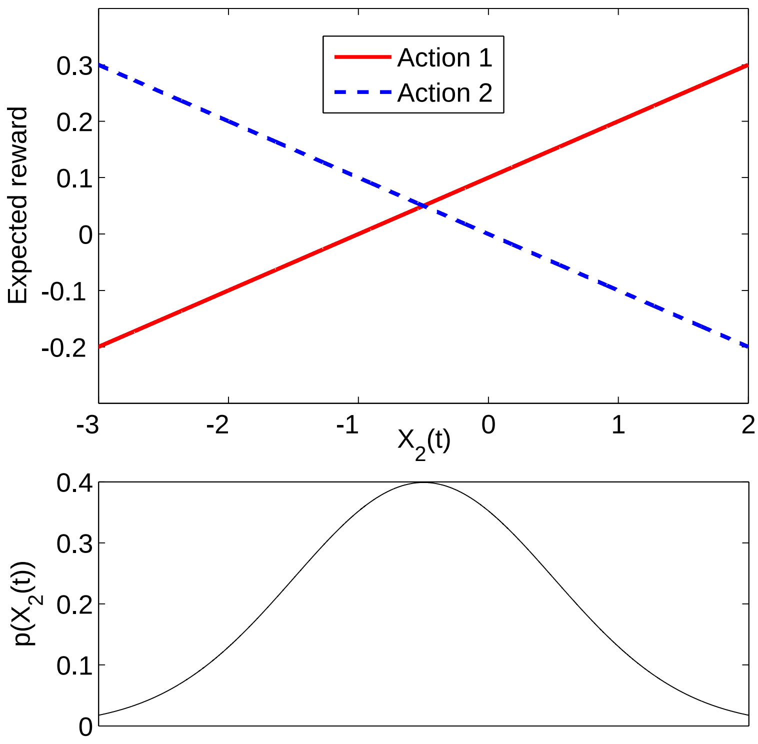


Figure 4.6: Reward functions for a 2-armed problem where the covariate distribution (given below) is centred at the intersection of Action 1 and Action 2 and has variance 1.

One of the key reasons for the strong performance of ϵ-ADAPT is the algorithm's ability to detect when best to explore. This occurs as ϵ-ADAPT explicitly considers the current covariate value when making each decision. To explore this feature in more detail, Figure 4.7 shows the distribution of covariate values over all simulations and separates them into regions of exploration and exploitation for the first and second halves of the game. As expected, the amount of exploration decreases and the correctness of exploitation decisions improve over time. Notice however, that unlike Figure 4.3, ϵ-ADAPT is exploring more for covariate values near the decision boundary. This occurs as ϵ-ADAPT now needs to learn the intercept of the reward plane (which was earlier fixed) and covariate values near the decision boundary are still informative regarding the correct partitioning of the covariate space. This feature is verified in Figure 4.8 which shows the average proportion of times each covariate value is used for exploration (using the same bins

Table 4.8: Comparison of ϵ-ADAPT and optimal ϵ-first with a 2-armed problem

CNR	Off-line (ϵ-first)		On-line (ϵ-ADAPT)	
	Opt. ϵ	Reward	% Optimal	Avg. exp. steps
200	0.03	0.895	101.2%	1.37
100	0.04	0.870	100.9%	2.45
50	0.06	0.827	100.1%	4.05
20	0.09	0.736	97.8%	6.81
10	0.11	0.629	96.8%	8.61

for the covariate as in Figure 4.7) for the two halves of the game. Covariate values near the decision boundary are used much more for exploration than from the tails of the distribution as exploration is not as costly here whilst still being informative. This includes a notion of cost-inclusive exploration into ϵ-ADAPT and explains the potential for higher rewards than the optimal ϵ-first policy.

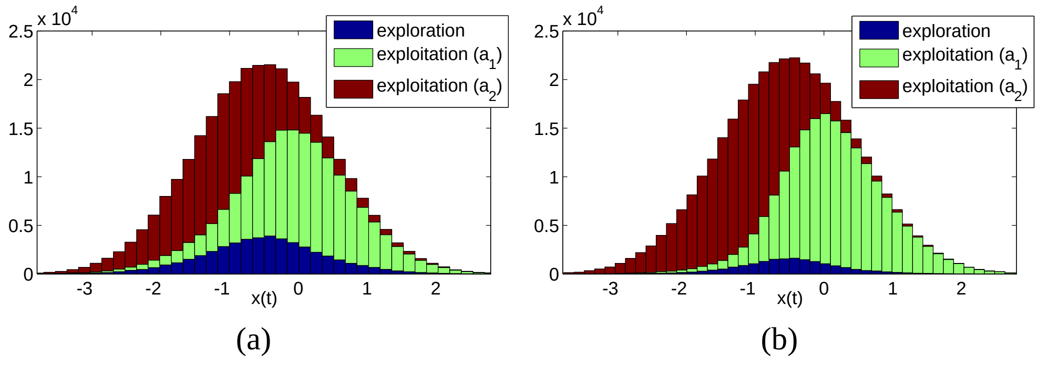
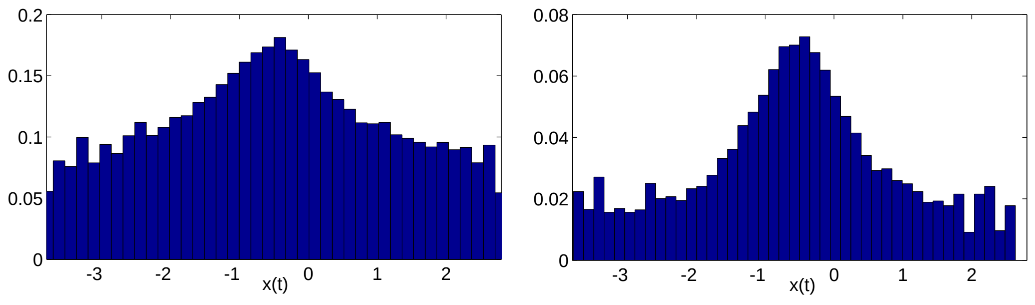


Figure 4.7: Histograms of the covariate distribution partitioned into regions of exploration/exploitation for (a) the first 50 time-steps and (b) the final 50 time-steps, for a 2-armed bandit problem.

- *3-armed problem*

We now study a 3-armed bandit problem where the third action is globally suboptimal (i.e. it is suboptimal for all possible covariate values $x(t)$). This problem is of particular interest as ϵ-first methods do not choose which action should be explored, and will continue to explore an action even if the agent knows it is globally suboptimal. It is of interest to see whether ϵ-ADAPT can do better and learn to not explore suboptimal actions. We keep the same setup as for the two-armed problem and we



(a) (b)

Figure 4.8: Histograms of the average proportion of explorative actions, given a particular covariate value, in (a) the first 50 time-steps and (b) the final 50 time-steps, for a 2-armed bandit problem.

add a third action as shown in Figure 4.9. We repeat our simulations with this third action to check how quickly ϵ-ADAPT can learn to never select this action.

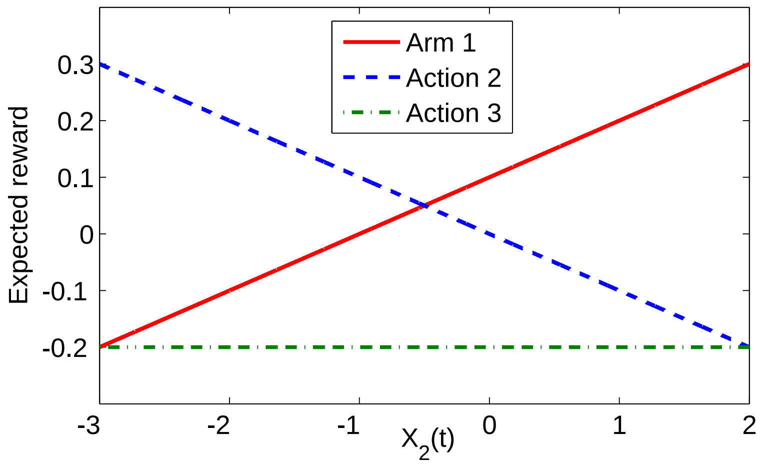


Figure 4.9: Reward functions for a 3-armed problem where the covariate distribution as given in Figure 4.6. Action 3 is globally suboptimal.

To this end, Table 4.9 demonstrates the performance of ϵ-ADAPT with respect to the optimally-tuned ϵ-first policy. As can be seen, the addition of a suboptimal arm that confounds the decision making process has not affected the performance of ϵ-ADAPT with respect to the optimal ϵ-first policy. There is a slight loss in performance for low-noise problems due to the extra round of initialisation required by ϵ-ADAPT (to gain unbiased noise variance estimates), but a slight improvement with high-noise problems where a good exploration policy is particularly important. Moreover, notice that the optimal exploration rates for ϵ-first are lower than with the 2-armed problem (cf. Table 4.8) as exploration is more costly (if performed randomly), leaving the other 2 actions under-explored. Conversely, ϵ-ADAPT has performed marginally more exploration than in the 2-armed problem, as there are more actions and hence more indices to construct. This suggests that this exploration is performed more intelligently than with ϵ-first.

Table 4.9: Average rewards with a 3-armed problem

CNR	Off-line (ϵ-first)		On-line (ϵ-ADAPT)		
	Opt.ϵ	Reward	Reward	% Optimal	Avg. exp.
200	0.00	0.893	0.891	99.8%	1.40
100	0.00	0.867	0.869	100.3%	2.56
50	0.02	0.835	0.835	100.0%	4.63
20	0.04	0.757	0.747	98.7%	8.80
10	0.07	0.675	0.654	96.9%	12.13

To investigate this further, Figure 4.10 shows that the suboptimal action has been selected infrequently for exploration (even for a high-noise problem) and Figure 4.11 shows that the rate with which the suboptimal action is selected decreases rapidly over time, irrespective of the CNR value. This demonstrates that ϵ-ADAPT has learnt to adapt to the problem faced and correctly select *which actions* to explore, whilst also correctly controlling the overall amount of exploration. This allows more exploration to be performed, particularly for high-noise problems, in such a way that is less costly than the random exploration of ϵ-first and ϵ-greedy policies.

- *10-armed problem*

So far we have demonstrated the ability of ϵ-ADAPT to partition the covariate space into regions where each action is optimal, and also to learn and identify if any actions are globally suboptimal. Every bandit problem in higher dimensions (whether that is in the number of actions or covariates) will be an extension of the above mentioned examples. Nonetheless, to check whether ϵ-ADAPT is indeed robust to high-dimensional covariates and a large number of actions, we now test ϵ-ADAPT

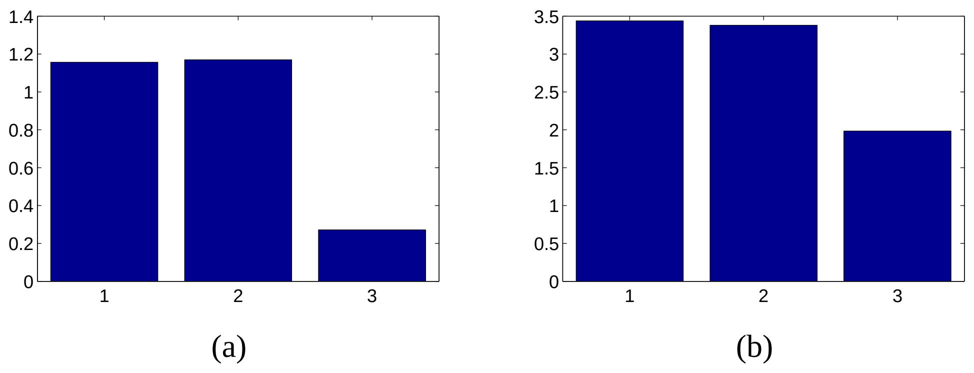


Figure 4.10: The average number of times each action is selected for exploration by ϵ-ADAPT where (a) CNR = 100 and (b) CNR = 20, for a 3-armed problem.

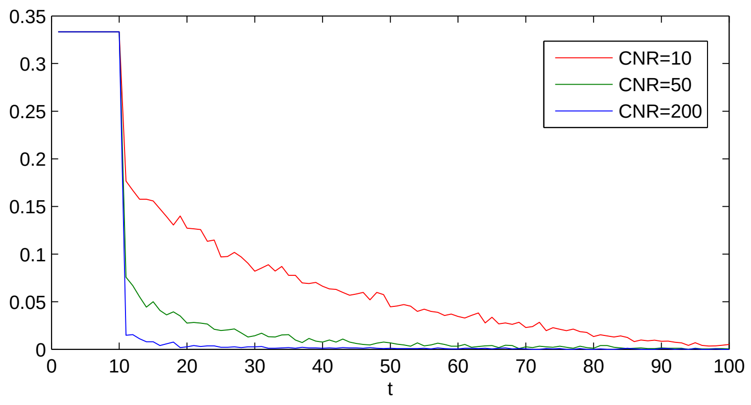


Figure 4.11: The average frequency of selecting the globally suboptimal action (Action 3) over time for various CNR values with a 3-armed problem. The initial rate of 1/3 is the average rate during the initialisation period.

for a 10-armed problem with a 10-dimensional covariate. We extend the length of the game to 200 iterations, to allow for the initialisation period of length 100.

Table 4.10 shows results for ϵ-ADAPT compared with the optimal ϵ-first policy for a range of CNR values. The optimal value of ϵ is close to zero for all problems – this is a feature of problems with a high-dimensional covariate (Pavlidis et al., 2008a), as the extensive side information and long initialisation period render additional exploration to be of no particular value. ϵ-ADAPT explores more as the algorithm explores intelligently and learns not to explore suboptimal actions or with covariate values far from decision boundaries. Moreover, ϵ-ADAPT yields rewards that are again consistently close to the optimally-tuned ϵ-first policy.

Table 4.10: Average rewards with a 10-armed problem

CNR	Off-line (ϵ-first)		On-line (ϵ-ADAPT)		
	Opt.ϵ	Reward	Reward	% Optimal	Avg. exp.
50	0	0.721	0.709	98.3%	8.9
20	0.01	0.675	0.671	99.4%	11.0
10	0.01	0.651	0.647	99.4%	12.7
5	0.02	0.616	0.601	97.6%	14.5

4.3.2 Bandit Problem without Covariates

In this section we check how ϵ-ADAPT performs in the standard bandit problem, with no side information. This allows ϵ-ADAPT to be compared with the POKER algorithm (Section 2.2.5) and the UCB policy (Section 2.2.3), which are the only other methods that are free of exploration parameters – but have no obvious extension to the bandit with covariates problem.

- *2-armed problem*

We first test ϵ-ADAPT for a simple 2-armed bandit problem (length 100) with normally distributed rewards (with means 0.5 and 1 and variance σ_η^2). Table 4.11 displays results for 10,000 repeats, where we also include results for various ϵ-first policies, the POKER algorithm and UCB1-Normal – a UCB approach specifically designed for normally distributed rewards (see Equation (2.10)). ϵ-ADAPT is the only approach that outperforms all ϵ-first policies when rewards are averaged and also outperforms the POKER algorithm and the UCB policy for each value of the noise variance – which is a particularly desirable result given that ϵ-ADAPT works for a broader range of problems. The poor performance of UCB1-Normal is attributed to the fact that any action is explored if it has been selected less than 8log(t) times at time t – for a 2-armed problem this means exploration effectively occurs for at least the first 66 iterations (and for an even longer period for problems with more actions). This renders this policy of limited use in finite-time problems and is hence most useful for its desirable asymptotic properties.

Table 4.12 provides a comparison with the optimal ϵ-first policy and the POKER algorithm for each value of σ_η^2 used. Performance degrades more rapidly with the

Table 4.11: Average rewards for a 2-armed problem with no covariates

σ_η^2	$\epsilon = 0$	0.02	0.05	0.1	0.15	0.20	UCB1-Normal	POKER	ϵ-ADAPT
0.05	**0.967**	0.957	0.929	0.880	0.830	0.780	0.237	0.961	0.962
0.1	0.917	**0.932**	0.923	0.879	0.830	0.779	0.217	0.930	0.948
0.2	0.827	0.869	**0.892**	0.870	0.827	0.778	0.191	0.873	0.911
0.5	0.645	0.716	0.778	**0.801**	0.785	0.754	0.183	0.736	0.800
1	0.501	0.572	0.653	0.691	**0.697**	0.684	0.160	0.597	0.698
Avg.	0.772	0.809	**0.835**	0.824	0.794	0.755	0.181	0.819	0.859

POKER algorithm as the noise increases, whereas ϵ-ADAPT yields rewards that are still within 95% of the optimally tuned ϵ-first policy, even for high noise problems. Both on-line algorithms correctly learn to explore more as the noise increases, but POKER does this to a lesser extent, and with no more than 2 exploration steps the algorithm performs similarly to an ϵ-first policy with $\epsilon = 0.02$. ϵ-ADAPT, however, explores to a more optimal level as the agent's high levels of uncertainty are driving exploration directly.

Table 4.12: Comparison of optimal exploration rates for a 2-armed problem with no covariates

σ_η^2	Off-line (ϵ-first)		On-line (POKER)		On-line (ϵ-ADAPT)	
	Opt.ϵ	Reward	% Opt	Avg. exp.	% Opt	Avg. exp.
0.05	0.00	0.967	99.4%	0.65	99.4%	0.77
0.1	0.02	0.932	99.8%	0.89	101.7%	1.09
0.2	0.05	0.892	97.9%	1.27	102.2%	1.82
0.5	0.10	0.801	91.8%	1.69	99.9%	3.44
1	0.17	0.698	85.5%	1.95	96.9%	4.80

- *5-armed problem*

We now test ϵ-ADAPT for a 5-armed bandit problem to test the algorithm's ability to correctly choose the most important actions to explore when there is no side information available. This time we bound the rewards in the interval $[0, 1]$, which allows for a comparison with the other UCB policies constructed by Auer et al. (2002). These policies only require an initialisation period of k rounds, which is much shorter than with UCB1-Normal, and will therefore perform much better in finite-time problems. In fact, it is noted in Auer et al. (2002) that UCB1-Tuned is the best performing UCB policy in all experiments, so we restrict our attention to this policy in this section.

One of the best known distributions bounded in the interval $[0, 1]$ is the beta distribution and we choose to use this distribution to draw the rewards from each of the 5 actions. We consider 4 different problems, where the 5 actions always have expected rewards of 0.3, 0.4, 0.5, 0.6 and 0.7 respectively, but with each problem we increase the variance of the beta distributions, which makes the learning problem harder. These reward distributions are displayed in Figure 4.12.

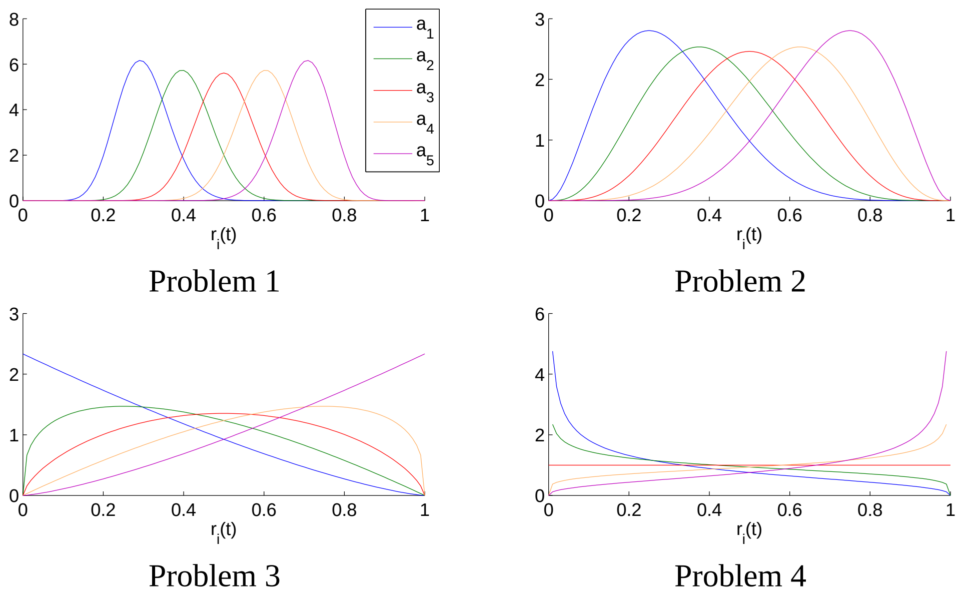


Figure 4.12: 4 different 5-armed bandit problems where each action is beta-distributed. The parameters are set such that the expected reward of each action is 0.3, 0.4, 0.5, 0.6 and 0.7 respectively. The variance of each distribution increases with the problem number.

The average rewards of ϵ-ADAPT, POKER and UCB1-Tuned are displayed in Table 4.13, where $T = 100$. Note that we adjust the initialisation of ϵ-ADAPT such that each action is selected once (rather than twice), and the reward variance of each action is then estimated as being equal to the variance of all observed rewards. If a second reward of an action is observed this estimate is then replaced by the sample variance estimate from the past rewards for this action only. This allows for a fair comparison with POKER and UCB1-Tuned which both require initialisation periods of length k.

In the experiments, ϵ-ADAPT and POKER performed close to or better than the optimally tuned ϵ-first policy. Both algorithms benefit from selecting which action should be explored based on both their uncertainty and potential for increased future reward. This is in spite of the fact that neither algorithm assumes a beta distribution, nor rewards bounded in the interval $[0, 1]$. In fact, we have configured ϵ-ADAPT to continue to assume normally distributed rewards, which again shows robustness to misspecified models and explains the slight loss in performance for Problem 4 where the distributions are highly non-normal. UCB1-Tuned however, performs poorly for all experiments, despite having the most restrictive assumptions. The policy over-explores each time and this exploration is not driven by uncertainty – in fact the policy explores marginally more for the easier learning problems, which is not desirable behaviour.

Both ϵ-ADAPT and POKER explore more for the harder problems. On this occasion, POKER has learnt the right level of overall exploration, which is in contrast to Table 4.12, where the algorithm under-explored for a 2-armed problem. This inconsistency is due to the choice of δ_μ in the algorithm (see page 41 for a more detailed discussion), which effectively controls the overall amount of exploration. Although this choice removes the need for an exploration parameter, it can negatively affect finite-time performance for certain types of problem. This feature is not investigated any further in this thesis (as the POKER algorithm is not applicable to dynamic or multi-agent decision problems) and we conclude that POKER and ϵ-ADAPT are comparable in terms of performance for the bandit problem with no covariates – but we reiterate that ϵ-ADAPT is a much more generalisable approach.

Table 4.13: Comparison of policies for a 5-armed problem with no covariates

Problem	Off-line (ϵ-first)		UCB1-Tuned		POKER		ϵ-ADAPT	
Number	Opt.ϵ	Reward	% Opt	Avg. exp.	% Opt	Avg. exp.	% Opt	Avg. exp.
1	0.00	0.937	64.9%	37.49	99.4%	2.01	99.7%	1.39
2	0.00	0.851	70.9%	37.03	101.8%	3.35	102.5%	5.47
3	0.02	0.765	78.4%	35.04	102.5%	5.37	101.2%	10.95
4	0.04	0.735	81.2%	33.31	99.8%	6.00	99.2%	13.00

Figure 4.13 shows the average number of times each action is selected for exploration in Problem 2 and Problem 4 by ϵ-ADAPT. Even for the high-variance problem, action 4 is selected most for exploration – as this is the action that is closest to the optimal action (action 5). Action 5 is also occasionally selected for exploration, which occurs when ϵ-ADAPT has incorrectly found a different action (usually action 4) to be optimal. This shows that the algorithm self-corrects and does not converge to selecting sub-optimal actions as often as other algorithms and policies might.

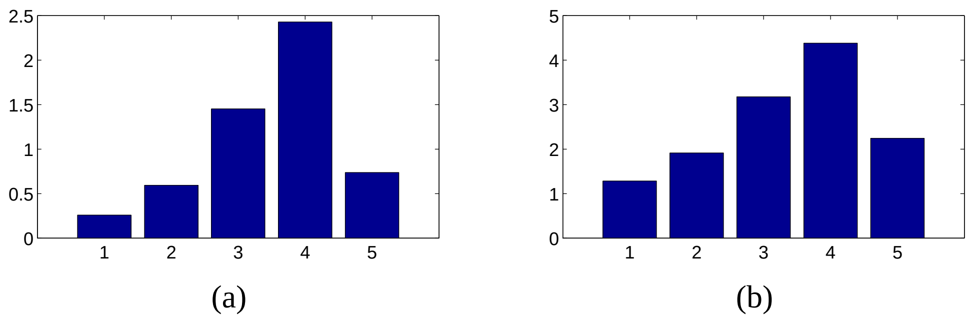


Figure 4.13: The average number of times each action is selected for exploration by ϵ-ADAPT with (a) Problem 2 and (b) Problem 4 for the 5-armed beta-distributed bandit problems.

4.3.3 Real Data

Finally in this section we compare ϵ-ADAPT with the ϵ-first policy and the POKER algorithm using real data. The data is from a real-world *Content Distribution Network* problem (CDN) where an agent must must retrieve data through a network with several sources available (see Vermorel and Mohri (2005) for a more detailed description, and a link to the data source which has been made publicly available). The sources can be viewed as the actions and the delays as rewards (where a small delay yields a high reward and vice-versa). The objective of the agent therefore is to minimise the sum of delays from a series of retrievals.

Table 4.14 displays the average retrieval delay using each algorithm for problems ranging from 2 to 5 arms. We performed 10,000 repeats (game length 100), each time randomly sampling the actions used and then randomly ordering the latencies (such that repeat experiments are not performed). The objective is to minimise the average delay, and all three methods perform relatively equally for this data set with ϵ-ADAPT performing marginally best overall, although the ϵ-first figures given are from the optimally tuned parameter (given in parentheses in the table). Notice that the optimal ϵ value decreases as the number of actions increase, this is because random exploration is more costly. The number of exploration steps performed by ϵ-ADAPT however (also given in the table in parentheses), increases with the number of actions. As demonstrated in Section 4.3.1 with a 3-armed problem and Section 4.3.2 with a 5-armed problem, this is because the algorithm learns to eliminate the worst actions quickly and explore more between the best actions. In summary, ϵ-ADAPT works well with this real data set – despite the algorithm modelling the observation noise as being normally distributed (which is not the case, in fact the data is quite spiky). This again demonstrates that ϵ-ADAPT is robust to misspecified modelling assumptions.

Table 4.14: Comparison of algorithms with real data

No. Actions	ϵ-first	POKER	ϵ-ADAPT
2	38.18 (0.05)	38.21	**38.00** (2.52)
3	37.22 (0.02)	**37.12**	37.27 (5.80)
4	**36.50** (0.01)	36.54	36.70 (8.35)
5	35.02 (0.00)	34.78	**34.64** (11.27)
Avg.	36.73	36.66	**36.65**

4.4 Summary

In this chapter we have constructed ϵ-ADAPT, the first algorithm for balancing exploration with exploitation in an on-line and incremental manner for the bandit with covariates problem. The algorithm is based on the effective and simple ϵ-first policy, but removes the need for an *a priori* fixed exploration parameter, using on-line approximation techniques. We first constructed ϵ-ADAPT for the one-armed bandit problem and then extended this to multiple actions using an indexing approach that shares characteristics with the Gittins indices. We then performed an exhaustive simulation study to show that ϵ-ADAPT is robust to varying numbers of actions, high degrees of noise, high-dimensional covariates and performs well with real data. Our simulation results also show that ϵ-ADAPT is competitive with, and can outperform, optimally tuned off-line policies and performs best overall when results are averaged over experiments, for both the generalised bandit with covariates setting and the commonly studied special case of the standard bandit problem (with no covariates). We note that we have not tested ϵ-ADAPT for non-linear rewards and non-normal covariates, where bootstrapping techniques could be used to regenerate new rewards and covariates. Furthermore, we have not yet placed any theoretical bounds on the finite-time performance of ϵ-ADAPT – this is reserved for future work and discussed further in Chapter 8.

The ϵ-ADAPT algorithm fills an important void in the bandit literature, in that we now have an on-line algorithm that can effectively balance the exploration-exploitation trade-off for the bandit with covariates framework – without any fixed exploration parameters. In the next chapter we make the important next step of extending ϵ-ADAPT for bandit problems with dynamic rewards. We can then revisit Table 2.1 and analyse how ϵ-ADAPT compares with other policies and algorithms and indeed offers one of the most generalisable approaches across all bandit frameworks.

Chapter 5

Adapting Exploration in Dynamic Environments

In the previous chapter we constructed an on-line algorithm for adapting exploration in static bandit problems, ϵ-ADAPT. Most real world decision making problems, however, are likely to change over time – as discussed in Section 2.1.5. Furthermore, many decision problems that are modelled as static problems may in fact have some form of dynamics that are unknown or unexpected *a priori*, which should not be ignored by a reward-maximising agent. Dynamic decision making problems are particularly challenging, however, as not only is predicting future rewards more complicated, but also the optimal balance between exploration and exploitation is closely related with the dynamics of the problem, and will change as the decision problem itself changes.

In this chapter, we construct a dynamic version of ϵ-ADAPT, where the reward functions of each action change over time. In dynamic environments, we require ϵ-ADAPT to *adapt as it plays* and quickly respond to changes in the reward structure that yields significant changes to the optimality of different actions. As in the previous chapter, we do this by capturing the uncertainty surrounding each action, which will automatically increase if the reward process significantly changes, and we then use this uncertainty to drive exploration on-line. In this way, the ϵ-ADAPT method is naturally suited to dynamic problems, but we make several significant changes to the static version of the algorithm for this to work.

Many policies and algorithms presented in Table 2.1 are not suitable for dynamic environments. Several policies, however, including SoftMax (Section 2.2.2) and Reinforcement Learning type approaches (Section 2.2.6), can be used in a dynamic reward setting, by estimating the expected reward of each action using Q-learning (rather than recursive averaging). The learning rate parameter λ (see Equation (2.15)) can be adjusted dependent on the rate of change of the reward process – λ values close to 1 forget past data quickly and weight the reward estimate towards the most recently observed rewards. These approaches, however, are only designed to work for the standard bandit problem, with no side information.

This is in contrast to the ϵ-greedy policy, which has been implemented in a dynamic bandit with covariates framework in Pavlidis et al. (2010). In this study, the coefficients of linear reward functions are assumed to change over time, following an ESTAR process (see Equation (2.4)), such that the optimal partitioning of actions in the covariate space changes over time. The estimated reward coefficients are then estimated using the Recursive Least Squares (RLS) algorithm with adaptive forgetting (Haykin, 2002, p.662). This algorithm learns to weight recent observations more heavily dependent on the *rate of dynamics* (i.e. the speed at which the reward process is changing) – and is hence analogous to Q-learning (with an adaptive learning rate) for the non-covariates setting. We use RLS with adaptive forgetting with ϵ-ADAPT in this chapter, and hence review the algorithm in more detail in Section 5.1.

It is argued in Pavlidis et al. (2010) that ϵ-greedy is a better policy for dynamic bandit problems than ϵ-first or ϵ-decreasing as there is a constant need to explore. In other words, in dynamic environments an agent needs to adapt over time, rather than converge to a fixed decision rule, so constant exploration is required to keep track of any significant changes. The ϵ-greedy policy still requires an *a priori* fixed exploration parameter, so for the same reasons outlined in Chapter 4, we construct an on-line algorithm that removes the need for this exploration parameter. Nevertheless, rather than extending ϵ-ADAPT to dynamic problems by using an on-line approximation of the optimal ϵ-greedy policy, we instead continue to use ϵ-first as the building block of our algorithm. We adjust our approach, however, such that the game is considered on a moving window whose size is dependent on the rate of dynamics. This has the important advantage that ϵ-ADAPT can continue to make near-optimal decisions without having to model the type of dynamics or predict future changes in the reward structure. We also adjust the estimates of the noise variance (which drives exploration) to include the uncertainty created by changes to the reward functions, and not just the amount from observation noise. This allows ϵ-ADAPT to retain the same characteristics as seen in Chapter 4 for the static case, i.e. the algorithm determines *how much*, *when* and *which actions* to explore over time. We detail the ϵ-ADAPT algorithm for dynamic bandits in Section 5.2.

We analyse the applicability of ϵ-ADAPT to dynamic bandit problems by performing a thorough simulation study in Section 5.3. We first test ϵ-ADAPT against various ϵ-greedy policies for drifting reward processes in Section 5.3.1, using the ESTAR framework. Then we construct a new bandit framework in Section 5.3.2, where reward coefficients jump over time to new values at unknown times governed by a Poisson distribution. This tests the robustness of ϵ-ADAPT to suddenly changing environments (as well as the more gradual drift of an ESTAR process). We then show that the dynamic version of ϵ-ADAPT can be successfully applied to static problems (Section 5.3.3), with only a small loss in performance as compared with the static version analysed in the previous chapter. Finally, in Section 5.3.4 we combine all three reward processes (ESTAR/jumps/static) in one decision-making problem. We conclude and discuss future work in Section 5.4.

5.1 The RLS Algorithm with Adaptive Forgetting

In this section we detail the RLS algorithm with adaptive forgetting, which will be used by ϵ-greedy and ϵ-ADAPT to estimate reward coefficients over time with linear reward functions. Furthermore, ϵ-ADAPT uses outputs from this algorithm to control the window sizes of the MC approximations and the on-line estimates of the noise variance (which we outline in Section 5.2). The line-by-line algorithm is given in Appendix B, where the significant step is the update of the $\lambda(t)$ parameter, given by:

$$\lambda(t) = \Big[\lambda(t-1) + \omega\hat{\boldsymbol{\psi}}^T(t-1)\boldsymbol{x}(t)\xi(t)\Big]_{\lambda_-}^{\lambda_+}, \tag{5.1}$$

where,

$$\xi(t) = r(t) - \hat{\boldsymbol{\alpha}}^T(t-1)\boldsymbol{x}(t). \tag{5.2}$$

$\lambda(t)$ is used to effectively place more weight on recent observations. At the extremes, $\lambda(t) \to 1$ corresponds to weighting all observations equally, and the standard RLS algorithm (used in Chapter 4) is recovered by setting $\lambda(t) = 1$; conversely as $\lambda(t) \to 0$ the algorithm places all weight on the most recent observation. In practice, the values that $\lambda(t)$ can take are truncated, with the upper limit set close to unity and the lower limit in the region of 0.8 (Niedzwiecki, 2000) – which is the value we use in this chapter. The quantity $\lambda(t)$ is often referred to as an *exponential weighting factor* or simply as the *learning rate* (Haykin, 2002), but in this thesis we refer to $\lambda(t)$ as a *forgetting factor*, as in Soderstrom et al. (1978) and Sayed (2003).

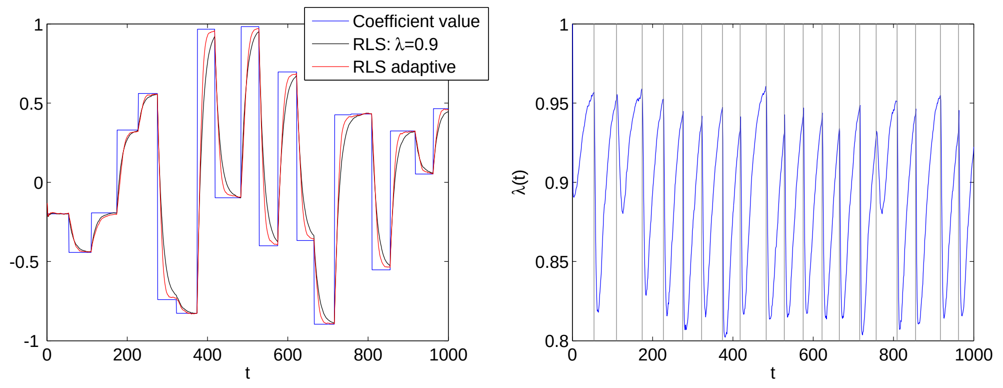
The RLS algorithm with adaptive forgetting propagates $\lambda(t)$ in the direction of the gradient of the one-step-ahead residual error $\xi(t)$ (Anagnostopoulos, 2010) – i.e. large errors in predicting the reward (based on the current coefficient estimates) will shift $\lambda(t)$ towards the lower truncation limit and vice-versa. The rate at which $\lambda(t)$ is adjusted is controlled by the meta-learning rate ω (which should be set close to zero (Haykin, 2002)). We note that removing the need to fix the forgetting factor *a priori* has removed one parameter but created three more: the meta-learning rate ω and the upper and lower truncation limits for $\lambda(t)$. This is still preferable to fixed forgetting however, as the latter two parameters are easily fixed at practical values (as discussed in the previous paragraph) and the meta-parameter ω is much less sensitive than changing a fixed λ value *a priori* (Anagnostopoulos, 2010). Moreover, adapting the forgetting factor on-line allows the RLS algorithm to better track reward coefficients when the optimal forgetting factor changes over time (Anagnostopoulos, 2010) – i.e. when the rate of change of the reward coefficients varies over time, which is particularly useful for unpredictable dynamic environments.

An alternative method for inferring reward coefficients is to perform ordinary recursive least squares over a window of historic time-steps (Haykin, 2002). This method avoids having to adapt forgetting factors on-line, but has the disadvantage that window sizes are fixed *a priori* and are difficult to adapt on-line, which can yield poor results when the rate of dynamics changes over time. Moreover, performing least squares over a window requires storing past data-sets (which is not required in RLS with adaptive forgetting), so this method is not fully-online and can create computational issues with large data sets (for example when there is a large volume of side information).

Another method is to use state-space modelling (Durbin and Koopman, 2001) to try and fit the sequence of reward coefficients to a model that can then predict future values. We choose not to follow this approach however, as we study scenarios where the type of dynamics are unknown and unpredictable. As a result, we do not wish to impose a model on the reward processes and prefer methods that are able to handle all sorts of dynamics using exactly the same algorithm. For the same reasons, we dismiss using particle filter techniques (Gordon et al., 1993).

For these reasons we use RLS with adaptive forgetting in this chapter, and to demonstrate the benefits of using this algorithm, in Figure 5.1 we show its average performance using the Poisson jump framework studied in Section 5.3.2. The forgetting factor $\lambda(t)$ on average takes low values immediately after jumps – as the algorithm learns that it should forget past data quickly. $\lambda(t)$ then incrementally grows until the next jump occurs. As a result, the reward coefficient is (on average) tracked more accurately than by using the optimal rate of fixed forgetting, as demonstrated in the figure. We note that in frameworks such as ESTAR (which we introduced in detail in Section 2.1.5), where the rate of change is constant over time, $\lambda(t)$ will converge to the optimal fixed λ value over time (Anagnostopoulos, 2010).

When applied to the multi-armed bandit problem, the RLS algorithm with adaptive forgetting is applied to each action separately, such that k separate forgetting factors $\lambda_i(t)$ are used. This has the important advantage that actions with different reward dynamics can be modelled separately and estimation is more accurate. Each



(a) (b)

Figure 5.1: (a) Coefficient values from a Poisson jump process with average RLS estimates (using adaptive and constant forgetting) and (b) the corresponding average adaptive forgetting factor (with the jumps indicated by vertical grey lines). The observation noise variance is equal to 0.1 and results are averaged over 10,000 repeats.

set of reward coefficients (and corresponding forgetting factors) are therefore not updated at each time-step, as bandit problems are opaque and rewards from unselected actions are not observed. Therefore, even a decision making problem with a fixed rate of dynamics (such as ESTAR) requires an adaptive forgetting factor – as the sequence of observed rewards for each action is unlikely to be regularly spaced over time. This makes the RLS algorithm with adaptive forgetting even more appropriate for dynamic bandit problems. The alternative, of trying to fill in missing values, is an open problem and beyond the scope of this thesis.

5.2 ϵ-ADAPT for Dynamic Bandit Problems

In this section we show how ϵ-ADAPT can be extended to bandit problems in dynamic environments. Perhaps the most obvious approach would be to design an algorithm that models the dynamics, and then uses this model to generate future rewards in the MC approximation of the ϵ-ADAPT indices $R_{\epsilon f}(T, t, i)$ (similarly to Algorithm 4.4 for the static case); however, we do not take this approach for two key reasons. First, forecasting a dynamic process is often challenging, particularly if we do not want to make any underlying modelling assumptions or impose any additional parameters on the algorithm. Secondly, the opacity of bandit problems means the sequence of observed rewards are usually not regularly spaced over time (as mentioned in Section 5.1), which further complicates the use of any modelling techniques. Instead, we compute the MC approximation of each index over a smaller window and assume the decision problem is static in this window, such that we do not need to explicitly model the dynamics. In Section 5.2.1 we detail how this window size is determined and in Section 5.2.2 we describe other key changes to the algorithm and provide the pseudo-code for ϵ-ADAPT for dynamic bandit problems.

5.2.1 Window Sizes for ϵ-ADAPT

To avoid modelling potentially unpredictable dynamics, ϵ-ADAPT considers a static game over a shorter window at each time-step. Specifically, at each time-step t, ϵ-ADAPT considers a window size of $T^W(t) = T^B(t) + T^F(t)$, where $T^B(t)$ and $T^F(t)$ are the number of past and future time-steps (respectively) used in the MC approximation by ϵ-ADAPT, as shown in Figure 5.2. This approach disregards the impact of time-steps s that are in the distant past or future ($s < t - T^B(t)$ and $s \geq t+T^F(t)$) at time t, which makes sense in dynamic problems, as action choices and observed rewards in these regions are not as relevant to the current decision problem (for suitable values of $T^B(t)$ and $T^F(t)$). Furthermore, this method allows the regeneration of new covariates and rewards to be performed in a static setting (using the current reward coefficient estimates) and avoids any issues of modelling dynamics.

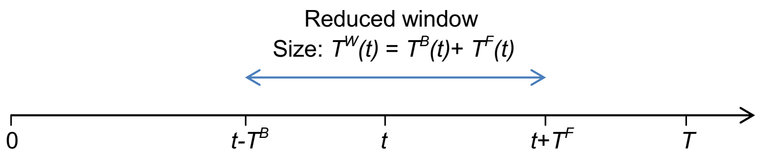


Figure 5.2: Window sizes used by ϵ-ADAPT in the MC approximation.

The main challenge now lies in selecting the window sizes $T^B(t)$ and $T^F(t)$ for the MC approximation of each index. After all, problems that are changing rapidly require shorter windows than problems that are drifting slowly. To avoid introducing any further complicated algorithms or models, we can find appropriate values for $T^B(t)$ and $T^F(t)$ from the outputs from the RLS algorithm with adaptive forgetting, which is otherwise used to estimate reward coefficient values. Specifically, we use the forgetting factors $\lambda_i(t)$ for each action a_i as a metric for how fast the decision making problem is changing overall – low values of $\lambda_i(t)$ suggest high rates of dynamics (and vice-versa). We first introduce the concept of *effective sample size* which is a measure of the effective number of samples being used in the linear regression (if the problem was static and all samples were equally informative). At time t the effective sample size, denoted $ESS_i(t)$, depends on all previous forgetting factors (Niedzwiecki, 2000):

$$\begin{aligned} ESS_i(t) =& \lambda_i(t_i(n_i)) + \lambda_i(t_i(n_i))\lambda_i(t_i(n_i-1)) + \ldots = \sum_{j=1}^{n_i}\prod_{\tau=j}^{n_i}\lambda_i(t_i(\tau)) \\ =& \begin{cases} \lambda_i(t)(1+ESS_i(t-1)) & \text{when action } a_i \text{ is selected;} \\ ESS_i(t-1) & \text{otherwise.} \end{cases} \end{aligned} \tag{5.3}$$

where t_i is the sequence of time-steps for which action a_i has been selected (a total of n_i times).

- *Forward-looking window size*

To set the forward-looking window size, $T^F(t)$, a naïve approach would be to set this value as $\sum_{i=1}^{k} ESS_i(t-1)$, the sum of the effective sample sizes. This approach only makes sense if the effective sample sizes are not going to change in future time-steps, such that the decision made at time t is unlikely to impact any of the coefficient estimates after time $t + \sum_{i=1}^{k} ESS_i(t-1)$. In a dynamic system, however, the rate of dynamics can change and past forgetting factors will have an increasingly smaller impact on future effective sample sizes. Furthermore, in the early stages of the game, the effective sample size will be small and will naturally increase over time (see Equation (5.3)). Consequently, we only make use of the existing forgetting factors $\lambda_i(t)$ to set $T^F(t)$, as this value is the most informative regarding the current rate of dynamics and future effective sample sizes. If the current rates of forgetting were to remain the same in future time-steps then it can be seen from Equation (5.3) that $ESS_i(t) \to 1/(1-\lambda_i(t))$ as $t \to \infty$. We can use this result to set:

$$T^F(t) = \min\left(\sum_{i=1}^{k} \frac{1}{1-\lambda_i(t)}, T-t+1\right), \tag{5.4}$$

where $T^F(t)$ does not exceed the length of the horizon $T-t+1$. This is a much more appropriate choice for $T^F(t)$ as past dynamics are no longer being used to predict future dynamics. Moreover, the required inverse relationship between the window size and the current rate of dynamics is achieved, without any new parameters or additional modelling assumptions. A more sophisticated method could be constructed where future sample sizes are calculated for specific future time-steps (rather than taking $t \to \infty$), but we avoid this approach as it requires consideration of the frequency with which each action is selected and hence involves additional modelling of future dynamics.

- *Backward-looking window size*

The calculation of the backward-looking window size, $T^B(t)$, could also be set to the sum of the effective sample sizes $\sum_{i=1}^{k} ESS_i(t-1)$ – this would be a suitable choice for transparent problems where all rewards are observed and every forgetting factor can be updated at each time-step. For bandit problems, however, this is yet again a poor choice of window size, precisely because of the irregular number of time-steps between each observation and subsequent RLS update. For example, the effective sample size of an action may be large (as the action previously yielded rewards that suggested slow-moving reward coefficients and a forgetting factor that is close to 1), but the action may not have been selected for a long time. In such cases, there is still some uncertainty regarding the future rewards of this action, so it should not be selected too many times during the backward-looking window. In other words, the effective sample size of each action does not mean this action has *actually* been sampled this many times in recent time-steps.

To combat this issue, we count how many times each action has been selected in the window $(t - T^F(t), t-1)$ and then select each action this many times in the MC approximation. We choose to look back over the forward-looking window size so that our two window sizes are consistent in size and respond to the current rate of dynamics (reflected through the $\lambda_i(t)$ values) in the same way. We make an important adjustment however, that reflects the scale of dynamics in recent time-steps. Specifically, the count for each action is re-weighted dependent on the historic $\lambda_i(t)$ values inside the window – this places more weight on actions which have yielded rewards that are not changing as fast and as a result have less uncertainty. ϵ-ADAPT is then more likely to explore actions that have demonstrated recent large changes to their reward structure. We can hence formulate the backward-looking window size as:

$$T^B(t) = \sum_{i=1}^{k} C^w(i) \tag{5.5}$$

where,

$$\begin{aligned} C^w(i) &= \max\left[W\left(ESS_i(t-1) - ESS_i(t - T^F(t))\right), D\right], \\ W &= \frac{T^F(t)}{\sum_{i=1}^{k}\left(ESS_i(t-1) - ESS_i(t - T^F(t))\right)}, \end{aligned} \tag{5.6}$$

where $ESS_i(t-1) - ESS_i(t - T^F(t))$ is the effective sample size within the reduced window, W is a re-weighting constant and D is the minimum number of samples required for unbiased coefficient estimates to exist. Each action is then selected $C^w(i)$ times in the backwards window of the MC approximation. $C^w(i)$ is required to be at least D so that ϵ-ADAPT has estimates of the reward coefficients at the end of the window. This is a reasonable imposition, however, as actions that are only selected D times will have high uncertainty and are likely to be selected for exploration.

Using these values for $T^B(t)$ and $C^w(i)$ in the MC approximation creates two desirable properties. First, actions that have not been selected recently are more likely to be explored by ϵ-ADAPT in future time-steps (as $C^w(i)$ is small). This is key, as actions that have been predictable and selected many time in the past, but have not been selected recently, should continue to be explored. Otherwise ϵ-ADAPT would be incorrectly extrapolating the observed rate of dynamics from many time-steps in the past – which is a naïve approach in dynamic environments. Secondly, ϵ-ADAPT re-weights the values of $C^w(i)$ within the window, such that actions that are suggesting recent fast-changing coefficients are down-weighted and vice-versa. This component allows ϵ-ADAPT to distinguish between actions that have differing rates of dynamics – such that actions which may have "jumped" to new reward coefficient values have down-weighted $C^w(i)$ values and are hence more likely to be explored.

5.2.2 Pseudo-code for ϵ-ADAPT

The ϵ-ADAPT algorithm for dynamic bandit problems is based on the same basic concept as ϵ-ADAPT for static problems. At each time-step, however, the MC approximation occurs on a reduced window $T^W(t) = T^B(t) + T^F(t) \leq T$, as discussed in Section 5.2.1. Indices (now denoted $R_{\epsilon f}(T^B(t), T^F(t), t, i)$) are calculated for each action (with the same intention as that in Section 4.2) and then the action with the highest index value is selected. The full pseudo-code is provided in Algorithm 5.1, with the MC approximation of the indices given in Algorithm 5.2. The significant changes from the static version of ϵ-ADAPT have been denoted in blue type font. We note that ϵ-ADAPT can also be used for dynamic rewards where there is no side information (as in the static case) by setting $x(t) = 1$.

There is one other key difference between the static and dynamic versions of the algorithm: the estimates of the noise variances $\hat{\sigma}_i^2$ for each action. In the static setting, these values are calculated by averaging the squares of the residuals in the regression (as in Equation (3.22)). In the dynamic setting, however, the residual errors will change over time in response to jumps or changes in the rate of drift. As a result, we use an adaptive measure of the noise variance for each action, which is

Algorithm 5.1 ϵ-ADAPT for Dynamic Bandit Problems

```
1: $\boldsymbol{n} = \mathbf{0}$ {Initialise Action count}
2: for $t = 1$ to $T$ do
3:    Observe covariate $\boldsymbol{x}(t)$
4:    Update unknown parameters of covariate distribution
5:    if $t \leq kD$ then
6:       Select action $i$ where $i = 1 + t \bmod k$ {Initialisation}
7:    else
8:       Calculate $T^F(t)$, $T^B(t)$ and $C^w(i)$ $\forall 1 \leq i \leq k$ {from Equations (5.4),
         (5.5) and (5.6) (respectively)} Set $T^W(t) = T^B(t) + T^F(t)$
9:       Generate new Covariates $\boldsymbol{x'}(s)$ $\forall 1 \leq s \leq T^W(t)$
10:      $\boldsymbol{x'}(T^B(t)+1) = \boldsymbol{x}(t)$ {Keep same covariate at time $t$}
11:      Generate new rewards $r'_i(s)$ $\forall 1 \leq i \leq k, 1 \leq s \leq T^W(t)$ using estimated
         reward coefficients $\hat{\boldsymbol{\alpha}}_i$ and estimated noise variance $\hat{\sigma}_i^2(t-1)$
12:      for $i = 1$ to $k$ do
13:         Approximate $R_{\epsilon f}(T^B(t), T^F(t), t, i)$ {Algorithm 5.2}
14:      end for
15:      Select action $i$ ($1 \leq i \leq k$) that maximises $R_{\epsilon f}(T^B(t), T^F(t), t, i)$
16:   end if
17:   Receive reward $r(t) = r_i(t)$
18:   Update $\hat{\boldsymbol{\alpha}}_i$, $\lambda_i(t)$ and $\hat{\sigma}_i^2(t)$
19:   $n_i(t) = n_i(t-1) + 1$ and $n_j(t) = n_j(t-1)$ (for $j \neq i$) {Action counts}
20: end for
```

recursively updated (when the action is selected) as follows:

$$\hat{\sigma}_i^2(t) = \lambda_i(t) \left(\frac{\hat{\xi}_i(t)^2}{n_i} + \frac{n_i - 1}{n_i} \hat{\sigma}_i^2(t-1) \right) + (1 - \lambda_i(t)) \hat{\xi}_i(t)^2, \tag{5.7}$$

where $\hat{\xi}_i(t) = r_i(t) - \hat{\boldsymbol{\alpha}}_i \boldsymbol{x}(t)$ is the residual error at time t. This adaptive measure is equivalent to the approximation of the variance using adaptive Q-learning, with a finite-sample adjustment for low n_i values. The measure is used as it does not require any additional parameters and is consistent with the RLS updates of the reward coefficients. Notice that $\hat{\sigma}_i^2(t)$ is no longer an estimate of the observation noise variance σ_i^2 in Equation (2.3). In fact, this measure ensures that the noise variance estimates used in Algorithm 5.2 track both the observation error and the measurement error resulting from dynamics. As a result recent abrupt changes in

Algorithm 5.2 MC Approximation of $R_{\epsilon f}(T^B(t), T^F(t), t, i)$ index

```
1: for $s = 1$ to $T^B(t) + T^F(t)$ do
2:     if $s \leq T^B(t)$ then
3:         Select an action $i$ such that each action is selected $C^w(i)$ times after $s = T^B(t)$ and receive reward $r'(s) = r'_i(s)$
4:         Update $\bar{\boldsymbol{\alpha}}_i$
5:     else if $s = T^B(t) + 1$ then
6:         Select action $i$ and receive reward $r'(s) = r'_i(s)$
7:         Update $\bar{\boldsymbol{\alpha}}_i$
8:     else
9:         Select action $j$ that maximises $\mathrm{E}\big(r'_j(s)|\bar{\boldsymbol{\alpha}}_j, \boldsymbol{x}'(s)\big)$ $(1 \leq j \leq k)$ and receive reward $r'(s) = r'_j(s)$
10:        Update $\bar{\boldsymbol{\alpha}}_j$
11:    end if
12: end for
13: $R_{\epsilon f}(T^B(t), T^F(t), t, i) = \sum_{s=T^B(t)+1}^{T^W(t)} r'(s)$
```

the reward structure will increase overall levels of exploration. Moreover, if only certain actions have changed (and not others), then exploration will be reserved for these actions. The significance of using these adaptive measures is that ϵ-ADAPT is now more able to determine when and which actions to explore in dynamic environments, which we demonstrate through simulations in the next section.

The ϵ-ADAPT algorithm is robust to different types of dynamics including slow drifts and abrupt jumps in the reward structures. The key reason for this robustness is the fact that ϵ-ADAPT does not attempt to model dynamics, allowing a wide range of dynamics to be handled using the same algorithm. Exploration is driven by uncertainty – through both the noise variance estimates and the size of the windows in the MC approximation. These values are themselves driven by the forgetting factors, which ϵ-ADAPT uses to gauge the current rate of dynamics and influence how past data should be weighted. Furthermore, the dynamic version of ϵ-ADAPT can in fact be readily applied to static problems, as the forgetting factor will stay close to 1. As a result, ϵ-ADAPT considers a large window size (possibly the full window of length T) in the MC approximation (see Equations (5.4) and (5.5)) and in addition the noise variance estimate (Equation (5.7)) becomes equivalent to the estimate used in the static setting. The ϵ-ADAPT algorithm consequently becomes almost identical to the static version of Chapter 4. We test all these claims further in the next section where we run the ϵ-ADAPT algorithm in static, drifting and abruptly changing environments.

5.3 Numerical Results

In this section we test the ϵ-ADAPT algorithm against the ϵ-greedy algorithm for various dynamic multi-armed bandit problems. We compare against ϵ-greedy as this is the only other policy or algorithm that can be extended to dynamic settings and can also include side information (see Table 2.1). First we consider the ESTAR process considered in Pavlidis et al. (2010) where reward coefficients drift around an equilibrium value over time (Section 5.3.1). We then construct a new dynamic framework where reward coefficients jump to new values at unknown times, as governed by a Poisson distribution (Section 5.3.2). We also check whether ϵ-ADAPT for dynamic bandit problems can perform well in static problems, by comparing performance against the results of the previous chapter (Section 5.3.3). Finally, we test a novel 3-armed setting where the rewards of each action follows one of the above mentioned processes (ESTAR, jumps or static), to check whether ϵ-ADAPT can learn to respond to unpredictable and random dynamics and intelligently explore the best actions at the correct time-steps (Section 5.3.4).

5.3.1 ESTAR Process

The ESTAR process, which we introduced in Section 2.1.5, is a drifting process which jumps back to an equilibrium value if the process drifts far in either direction. The ESTAR process has been used to model exchange rates in Kilian and Taylor (2003), for example. This mean-reverting process ensures that the decision problem does not degenerate to one where one action is globally optimal. In our simulations we set $\gamma = 1$ to allow enough drift such that the optimal partitioning of covariates changes throughout the game, but not so much that one action often dominates the covariate space. The parameter σ_v^2 dictates the speed of change and Pavlidis et al. (2010) recommends values in the range $[0, 0.2]$, as higher values are too fast to track (especially as bandit problems are opaque). Otherwise, the frequency of change becomes much faster than the frequency of selection and the sampled process looks like white noise. In our experiments we consider σ_v^2 values of 0.01 and 0.1, to consider both slow and fast changing processes.

Figure 5.3 demonstrates how the optimal partitioning of the covariate space evolves over 50 time-steps with the two σ_v^2 values for a 2-armed problem. The equilibrium value of the coefficients and the covariate distribution are as given in Figure 4.6. When $\sigma_v^2 = 0.01$, the decision boundary drifts slowly over time, but jumps back to the equilibrium value of -0.5 (as at $t \approx 15$) if there is large drift. In contrast, when $\sigma_v^2 = 0.1$, the decision boundary jumps around frequently but occasionally shows periods of slower moving drift. By testing these two ESTAR processes, we examine the robustness of ϵ-ADAPT to different ratios of drifts and jumps in the dynamic bandit problem.

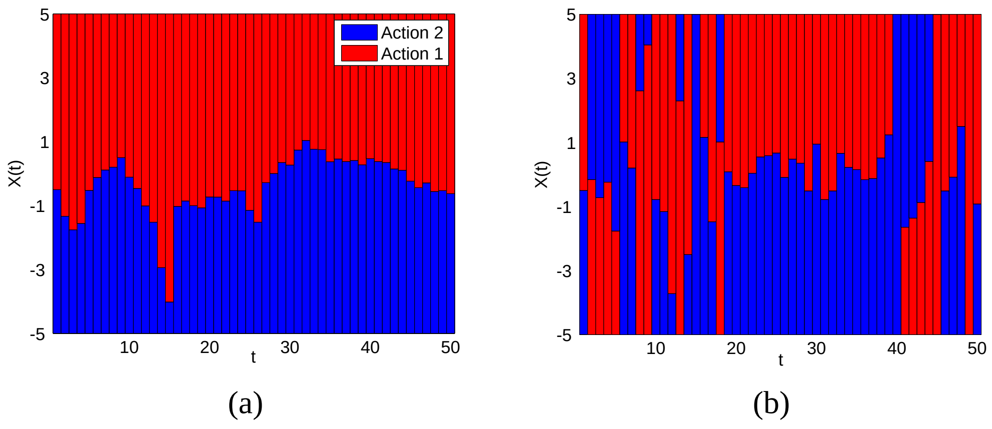


Figure 5.3: The optimal partitioning of covariates in a 2-armed dynamic bandit problem where coefficient values change according to an ESTAR process, where the rate of change is (a) $\sigma_v^2 = 0.01$ or (b) $\sigma_v^2 = 0.1$. The equilibrium coefficient values and the reward functions are the same as those used in Section 4.3.1.

We tested ϵ-ADAPT against various ϵ-greedy policies for the 2 ESTAR processes over 10,000 repeats. We repeated simulations across a range of CNR values[1], where the equilibrium coefficient values and the covariate distribution are the same as those used in the static setting in Section 4.3.1. Throughout this section, we extend the length of game to $T = 1,000$, as this gives the decision making problems enough time to change from their initialised states. The results of the simulations are reported in Table 5.1, where optimal off-line exploration rates are found numerically to 2 decimal places. The performance of each policy/algorithm and the amount of exploration required increases as both σ_v^2 increases and the CNR decreases, as the dynamic decision boundary becomes harder to track. In fact, exploration is even required when there is no observation noise (CNR$= \infty$), as the unknown dynamics still create uncertainty. We include results for ϵ-first to demonstrate the expected poor performance of this policy in dynamic problems. ϵ-ADAPT and ϵ-greedy perform much better in comparison – as both approaches explore both actions throughout the game and adapt to the changing decision boundary more quickly. In fact, ϵ-ADAPT performs marginally better, despite not requiring an optimally tuned exploration parameter – this is attributed to the fact that ϵ-ADAPT will learn on-line to not explore for costly covariate values, and allows on-line approaches to outperform optimally tuned off-line policies.

Table 5.1: Average rewards for the ESTAR Process

σ_v^2	CNR	ϵ-greedy		ϵ-first			ϵ-ADAPT		
		Opt.ϵ	Reward	Opt.ϵ	Reward	% Optimal	Reward	% Optimal	Avg. exp.
0.01	100	0.01	0.506	0.01	0.505	99.6%	0.518	102.4%	6.93
0.01	10	0.04	0.426	0.01	0.422	99.0%	0.423	101.6%	7.88
0.01	1	0.08	0.273	0.05	0.268	98.4%	0.525	101.9%	10.22
0.1	∞	0.06	0.191	0.02	0.185	96.8%	0.196	102.7%	8.45
0.1	100	0.06	0.192	0.02	0.189	98.6%	0.197	102.8%	8.46
0.1	10	0.06	0.183	0.02	0.175	95.7%	0.185	101.5%	8.82

To investigate the strong performance of ϵ-ADAPT further, in Figure 5.4 we display the average forgetting factors and noise variance estimates over time, where CNR$= 10$ (with $\sigma_i^2 = 0.1$) and $\sigma_v^2 = 0.01$. All parameters appear to converge to an equilibrium value, though note that $\hat{\sigma}_i^2$ converges to a value that is higher than

[1]Note that CNR values track changes to the observation noise variance σ_i^2 (as defined in Equation (4.3)) and not the noise variance of the ESTAR process σ_v^2 which determines the rate of dynamics.

the true observed noise variance. As mentioned earlier, this is because the estimate includes the uncertainty from the dynamics and is used to encourage the necessary increased levels of exploration. We display the average rate of exploration over time in Figure 5.5 for the range of CNR values, where $\sigma_v^2 = 0.01$. The average rate quickly converges to a stable value for the entirety of the problem (as the window sizes and noise variance estimates also converge to fixed values), except for the final time-steps as the horizon draws near. Note that the rate of dynamics is constant with an ESTAR process and as a result ϵ-ADAPT has learnt to explore consistently throughout the game. Moreover, the strong performance of the algorithm suggests that the windows sizes used in Algorithm 5.2 are correctly calibrated with the observed rate of dynamics.

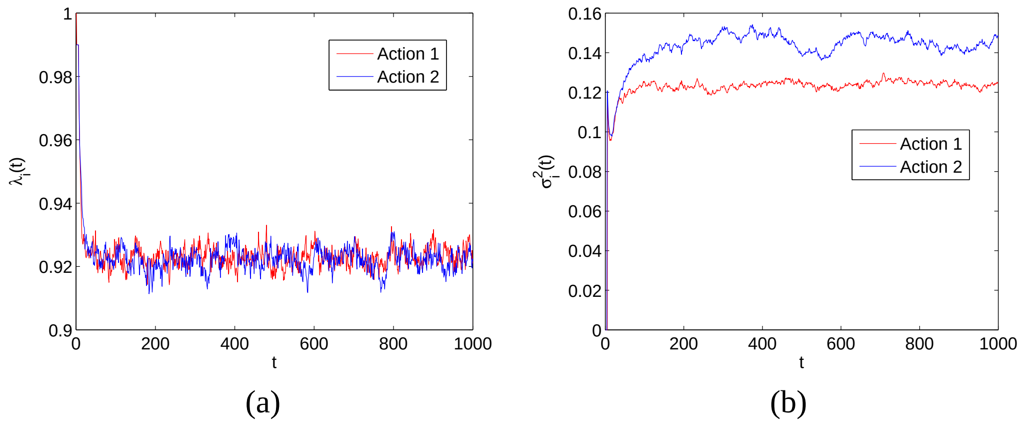


Figure 5.4: The average value of (a) $\lambda_i(t)$ and (b) $\hat{\sigma}_i^2(t)$ over all time-steps where CNR $= 10$ and $\sigma_v^2 = 0.1$, as calculated by ϵ-ADAPT using RLS with adaptive forgetting, for an ESTAR dynamic 2-armed problem.

5.3.2 Poisson Jumps

In this section we construct a different type of dynamic bandit problem, namely one where reward coefficients jump to new values at unknown times, and otherwise remain at the same values. This type of reward process can exist in financial time series for example, where rewards of various instruments are often directly linked to macro-economic factors such as interest rates (Cont and Tankov, 2004), or in climatological data gathered from remote sensors (Jensen et al., 1995). In this section, we select each coefficient uniformly from the interval $[-1, 1]$ and the time between

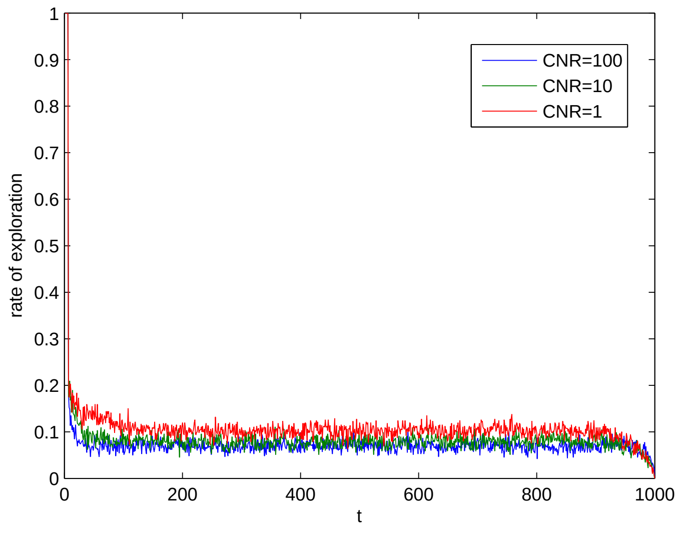


Figure 5.5: The average rate of exploration over time by ϵ-ADAPT for various CNR values where $\sigma_v^2 = 0.01$, for an ESTAR dynamic 2-armed problem.

jumps is drawn from a Poisson(ζ) distribution. The resulting process is displayed in Figure 5.1(a) where we demonstrated that RLS with adaptive forgetting is an appropriate technique for tracking changes to coefficient values. In our simulations in this section we use ζ values of 50 and 200 (and various CNR values) to test the robustness of ϵ-ADAPT to detect the presence of jumps in the data. We fix the timings of the jumps to be the same in all simulations, but only to accurately demonstrate the average behaviour of the algorithm before and after jumps occur – the coefficients of the reward functions are not fixed over the experiments. Table 5.2 displays the results of ϵ-ADAPT against the optimal ϵ-greedy (and ϵ-first) policy over 10,000 repeats. Again ϵ-ADAPT has correctly identified how much to explore (exploring more for problems that have frequent jumps or are noisy) and has outperformed the optimal ϵ-greedy (and ϵ-first) policy. Notice that the optimal ϵ for these two policies is always small, even for high-noise problems – this is because large degrees of exploration are only worthwhile if performed immediately after jumps.

Figures 5.6-5.8 demonstrate why ϵ-ADAPT performs well with this type of reward process. Immediately after jumps occur (as indicated by the grey vertical lines), the average forgetting factor decreases and the average noise variance es-

Table 5.2: Average rewards for the Poisson Jump Process

ζ	CNR	ϵ-greedy		ϵ-first			ϵ-ADAPT		
		Opt.ϵ	Reward	Opt.ϵ	Reward	% Optimal	Reward	% Optimal	Avg. exp.
50	100	0.06	0.674	0.00	0.649	96.3%	0.686	101.8%	4.39
50	10	0.06	0.614	0.00	0.580	94.5%	0.619	100.9%	4.97
50	1	0.07	0.522	0.01	0.479	91.7%	0.525	100.5%	6.49
200	100	0.03	0.848	0.00	0.825	97.3%	0.861	101.5%	2.65
200	10	0.03	0.829	0.00	0.805	97.1%	0.846	102.0%	3.22
200	1	0.03	0.777	0.01	0.757	97.4%	0.787	101.2%	5.29

timate increases (see Figure 5.6). As a result, ϵ-ADAPT calculates indices over shorter windows with more uncertainty – which makes exploration more likely, as shown in Figures 5.7 and 5.8 for the two jump frequencies. Between the jumps, the rate of exploration decreases as ϵ-ADAPT learns the new coefficient values and the uncertainty decreases. ϵ-ADAPT therefore performs better than ϵ-greedy as the algorithm learns when to explore more (rather than exploring at a constant rate), as driven by the varying degrees of uncertainty.

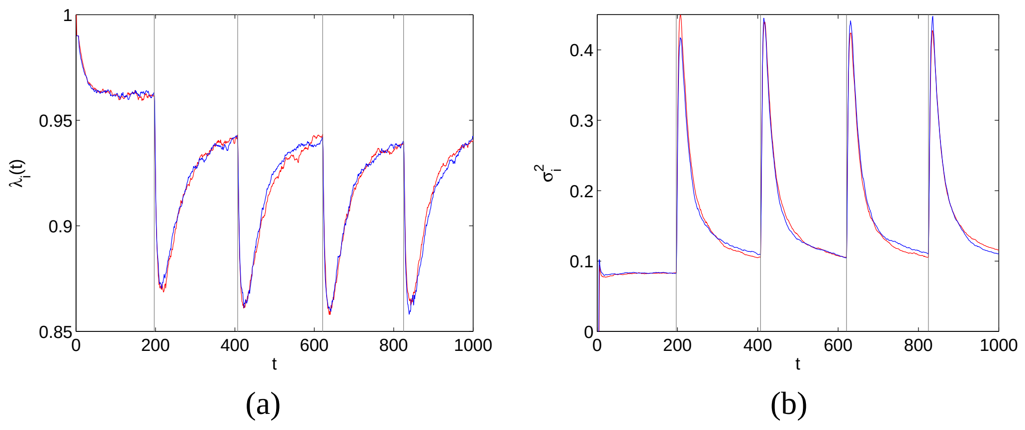


Figure 5.6: The average value of (a) $\lambda_i(t)$ and (b) $\hat{\sigma}_i^2(t)$ over all time-steps where CNR $= 10$ and $\zeta = 200$, as calculated by ϵ-ADAPT using RLS with adaptive forgetting, for a Poisson jump dynamic 2-armed problem.

5.3.3 Static Problems

In this section we check whether the dynamic version of ϵ-ADAPT can be applied to static bandit problems. We repeat all simulations from the 2-armed bandit with

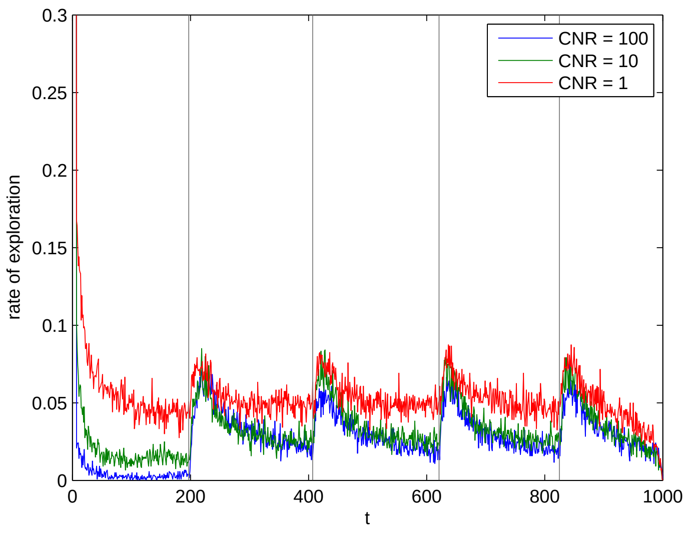


Figure 5.7: The average rate of exploration over time by ϵ-ADAPT for various CNR values where $\zeta = 200$, for a Poisson jump dynamic 2-armed problem.

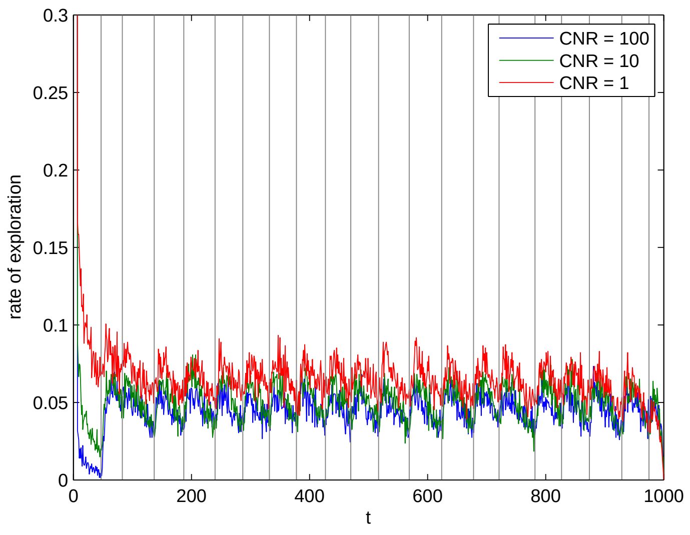


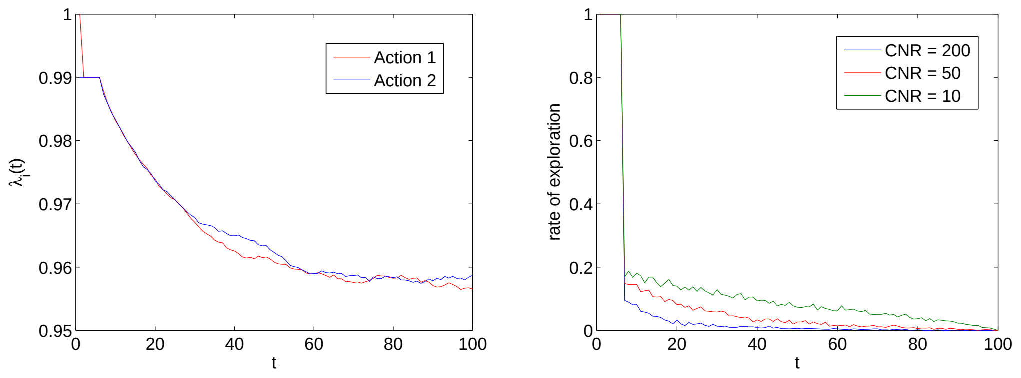
Figure 5.8: The average rate of exploration over time by ϵ-ADAPT for various CNR values where $\zeta = 50$, for a Poisson jump dynamic 2-armed problem.

covariates problem studied in Section 4.3.1, this time including the dynamic version of ϵ-ADAPT. The results are displayed in Table 5.3. As can be seen, there is only a marginal loss in performance with the dynamic version of ϵ-ADAPT which is attributed to the fact that the algorithm must learn to use the full window in the calculation of the indices in Algorithm 5.2 – which will happen if the forgetting factor is as close as possible to 1 (i.e. the upper limit of the forgetting factor truncation).

Table 5.3: Comparison of static and dynamic versions of ϵ-ADAPT with a static 2-armed problem

CNR	Static ϵ-ADAPT		Dynamic ϵ-ADAPT	
	Reward	Avg. exp. steps	% Optimal	Avg. exp. steps
200	0.906	1.37	100.0%	1.26
100	0.878	2.45	100.0%	2.23
50	0.828	4.05	100.0%	3.60
20	0.720	6.81	99.8%	5.85
10	0.609	8.61	99.1%	7.65

To investigate this further, in Figure 5.9 we show the average forgetting factors over time (for the high noise problem, CNR$= 10$) and the average rate of exploration for a range of CNR values. Despite the high levels of observation noise, the forgetting factor does not (on average) deviate far from its initial value of 1, which explains the comparable performance with the static version of ϵ-ADAPT. As a result, the average rate of exploration decays at similar rates as were seen in the static setting.



(a) (b)

Figure 5.9: (a) The average value of $\lambda_i(t)$ for CNR = 10 and (b) the average rate of exploration for a range of CNR values, as calculated by ϵ-ADAPT using RLS with adaptive forgetting, for a static 2-armed problem.

5.3.4 Mixture of Processes

Finally, in this section we test ϵ-ADAPT with a novel 3-armed problem, where we combine the previous considered settings, such that one action has a static reward function and the others have reward coefficients that change according to a Poisson jump ($\eta = 50$) or ESTAR process ($\sigma_v^2 = 0.01$)[2]. We do this to test the robustness of ϵ-ADAPT to scenarios where each action demonstrates different types of dynamics. We display the expected rewards for two cases in Figure 5.10, against various ϵ-greedy and ϵ-first policies, where each case assigns different observation noise variance to each action. In both cases, ϵ-ADAPT performs best overall as it treats each action differently – which happens as the forgetting factors and noise variance estimates are different for each action (as demonstrated earlier in this section). Note that the optimal off-line determined value of ϵ for the ϵ-greedy is high (in both cases), as the combination of ESTAR and Poisson jumps renders the decision problem as highly dynamic.

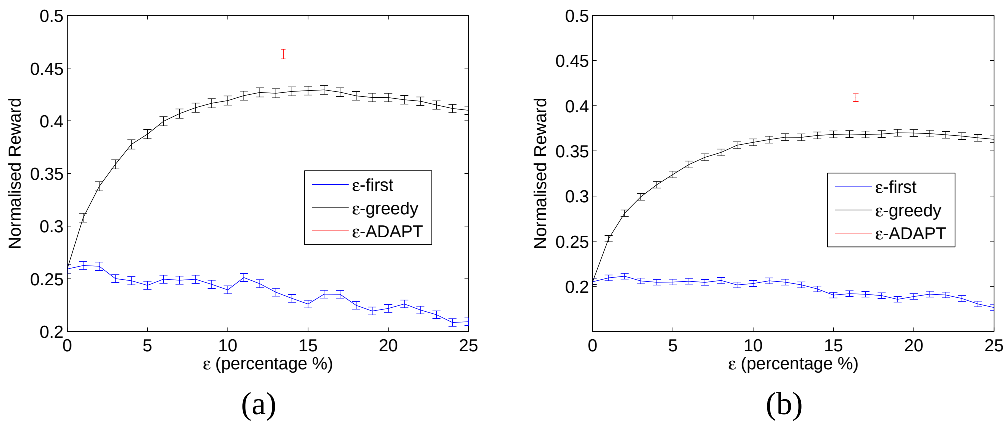


Figure 5.10: Expected rewards for ϵ-ADAPT and various ϵ-greedy and ϵ-first policies where the orderings of the observation noise variances are (a) static $>$ ESTAR $>$ jumps and (b) jumps $>$ ESTAR $>$ static. CNR values (which determine the variance) are fixed at 2.5/10/40 in each figure.

[2]We use the same covariate distribution as shown in Figure 4.6. The equilibrium coefficients of the ESTAR action are as in Action 1 of Figure 4.6. The Poisson Jump action is as defined in Section 5.3.2 and the static action has reward coefficients $\alpha_0 = -0.1$ and $\alpha_1 = 0$. These values are selected such that each action has likely regions of optimality in the covariate space.

5.4 Summary

In this section we have extended the ϵ-ADAPT algorithm to dynamic bandit problems where reward coefficients change over time. The ϵ-ADAPT approach is naturally suited to dynamic problems. Nevertheless, the extension involved several significant changes to the static version – including changing the window size of the MC approximations and changing the estimate of the noise variance, to include the uncertainty of dynamics. These adjustments were performed by using outputs from the RLS with adaptive forgetting factor. Specifically, we used the forgetting factor as an indicator of the rate of dynamics and combined this with sample sizes and observation errors to measure the rate of uncertainty and use this to drive exploration on-line.

Our simulations indicate that ϵ-ADAPT is an effective algorithm for dynamic bandit problems, including processes where reward functions can jump or drift over time (or combinations of the above). In addition, ϵ-ADAPT can be successfully applied to the static bandit problems investigated in the previous chapter. Moreover, ϵ-ADAPT is *not* a tailor made algorithm for specific settings (such as jumps, drifts etc) – exactly the same algorithm has been applied to each decision making problem we have considered, and performance is consistently strong. ϵ-ADAPT makes no attempt to model the dynamics which makes the algorithm robust to all sorts of unpredictable changes in the environment. We consider this to be significant, particularly as several real-world phenomena exhibit unpredictable behaviour that models have been unable to predict or capture (such as financial data or climate models). For dynamic reward processes that can be modelled or predicted, state-space models or particle filters can be used, the implementation of this within ϵ-ADAPT is reserved for future work (see Chapter 8).

We note that in this chapter we have considered dynamic bandit problems where rewards change over time and not the covariate side information. In fact, as noted in Pavlidis et al. (2010), *"a time varying covariate distribution plays no role in the transformation of a static sequential decision making problem into a dynamic problem"*. This is because the optimal partitioning of a covariate space is unaffected by time varying covariates alone. For these reasons we have considered a fixed covariate distribution, though we note that ϵ-ADAPT could be extended to such cases by using adaptive forgetting to draw new covariates (for the MC approximations) based on recent observations only. ϵ-ADAPT could also be extended to problems where new actions arrive or leave over time – a framework studied by Whittle (1981) for the non-covariates setting, which was called *arm-acquiring bandits*. Actions that arrive could be assigned high levels of uncertainty which makes subsequent exploration very likely. We reserve such extensions for future work.

The dynamic version of ϵ-ADAPT does not require any exploration parameters – the parameters used in the RLS algorithm with adaptive forgetting are required for inferring coefficient values, we just happen to exploit outputs from this algorithm to drive exploration on-line. In addition, the computational complexity of ϵ-ADAPT is now bounded linear in time – as the window size of the MC approximations are constrained by the upper truncation limit of the forgetting factors $\lambda_i(t)$. For short-length static problems however, the window size will usually consider the full window (length T), so the resulting algorithm still usually scales quadratically in time. Nevertheless the dynamic version ϵ-ADAPT is actually more computationally efficient than its static counterpart, especially when the rate of dynamics is high.

To bring the findings of the last two chapters together, we insert our static and dynamic version of ϵ-ADAPT into Table 2.1, which cross-compared the existing algorithms and policies in the literature. We display this amalgamated table in Table 5.4. ϵ-ADAPT has filled a void in the bandit literature – the need for an algorithm or policy that is free of exploration parameters and can be applied to settings that are dynamic and/or include side information. We also note that the dynamic version of ϵ-ADAPT could be applied without knowledge of the game-length T, though this will marginally affect performance.

This concludes the analysis of single-agent problems in this thesis. In the following chapters, we investigate the exploration-exploitation trade-off in multi-agent decision making problems. We will study the nature and meaning of exploration in multi-agent domains and attempt to extend the ϵ-ADAPT approach to the various frameworks we consider.

Table 5.4: The applicability of policies and algorithms for the exploration-exploitation trade-off in various bandit problems – including the static and dynamic versions of the ϵ-ADAPT algorithm

Policy/Algorithm	No. of Param	Comp. complex	Covariates	T not required	Unbounded rewards	No distbn. assumptions	Dynamics
Greedy	0	$\mathcal{O}(T)$	✓	✓	✓	✓	✓
ϵ-greedy	1	$\mathcal{O}(T)$	✓	✓	✓	✓	✓
ϵ-first	1	$\mathcal{O}(T)$	✓	✗	✓	✓	✗
ϵ-decreasing	1	$\mathcal{O}(T)$	✓	✓	✓	✓	✗
SoftMax	1	$\mathcal{O}(kT)$	✓	✓	✓	✓	✓[a]
Exp3	1	$\mathcal{O}(kT)$	✗	✓	✗	✓	✓
Interval Estimation	1	$\mathcal{O}(T)$	✓	✓	✓	✗	✓
UCB1/UCB1-Tuned	0	$\mathcal{O}(kT)$	✗	✓	✗	✓	✗
UCB2	1	$\mathcal{O}(kT)$	✗	✓	✗	✓	✗
UCB1-Normal	0	$\mathcal{O}(kT)$	✗	✓	✓	✗	✗
Gittins Indices	0	$\mathcal{O}(k^3T^3)^b$	✗	✗	✓	✗	✗
POKER	0	$\mathcal{O}(kT)$	✗	✗	✓	✗	✗
Q-learning	1[c]	$\mathcal{O}(T)$	✗	✓	✓	✓	✓
Reinforcement Comparison	2[d]	$\mathcal{O}(kT)$	✗	✓	✓	✓	✓
Pursuit Methods	1	$\mathcal{O}(kT)$	✗	✓	✓	✓	✓[a]
ϵ-ADAPT (static)	0	$\mathcal{O}(kT^2)$	✓	✗	✓	✓	✓
ϵ-ADAPT (dynamic)	0	$\mathcal{O}(kT)$	✓	✗	✓	✓	✓

[a] If used with Q-values rather than mean rewards

[b] as proven in Nino-Mora (2010)

[c] $k+1$ with optimistic initial estimates

[d] $k+2$ with Q-learning using optimistic initial estimates

Chapter 6

Exploring in a Multi-Agent Bandit Problem

In the previous chapters we have studied the exploration-exploitation trade-off in single-agent sequential decision making problems, where one agent repeatedly selects between a finite set of actions. For the remainder of this thesis, however, we study the exploration-exploitation trade-off in *multi-agent* sequential decision making problems, where multiple interacting agents must simultaneously choose between actions in the same environment. Multi-agent sequential decision making problems are ubiquitous and have been widely applied to applications as diverse as online auctions (Rogers et al., 2007b), sensor networks (Wang and Cheng, 2008) and disaster management (Ramchurn et al., 2008) amongst many others. In fact, it is likely that most real-world sequential decision making problems are likely to include at least one other decision maker, whose actions cannot be ignored. For these reasons, we investigate the role of exploration-exploitation in the multi-agent domain, and attempt to extend the ϵ-ADAPT algorithm to various important frameworks.

In this chapter we study the impact of communication between agents, to investigate the importance of agents *exploring* their *communication* decisions, as well as their action decisions, in order to learn the best actions to select in an unknown environment. We investigate communication problems because, in many multi-agent scenarios, information relevant to the decision problem is likely to be distributed amongst agents. This information, however, can often be communicated between agents during the decision making process. Consider a disaster management scenario, for example. Emergency service vehicles may have differing information about the locations of injured civilians and the outcomes of different actions are unknown *a priori*, but these agents can communicate such information to achieve a better coordinated solution (at an associated time cost). Agents can therefore learn and explore through communication *and* action decisions.

To study the role of exploration-exploitation in multi-agent communication problems (such as the disaster management scenario), we construct a novel framework which is an extension of the single-agent bandit with covariates problem. Specifically, we construct a multi-armed *decentralised* bandit problem, where each agent controls a (non-overlapping) subset of the available actions. In addition, each agent only observes a subset of the covariate, representing its partial view of the world. Furthermore, we allow agents to initiate communication between themselves (at a cost) exchanging potentially useful covariate values that were previously unknown to the agents.

The communication of information between agents is an important feature of many scenarios modelled by multi-agent systems. Furthermore, the use of the bandit setting allows us to specifically investigate the relationship between communication and the exploration-exploitation trade-off. This is because in a bandit setting the covariate distribution and its relationship to the rewards of each action are unknown and must be learnt over time. Each agent must therefore learn when and which agents to communicate with, to avoid unnecessary and costly communication exchanges. The exploration-exploitation trade-off is hence important to both communication *and* action decisions in multi-agent sequential decision making problems. In this chapter, we show that this trade-off can be effectively balanced using a *double ϵ-greedy* or *double ϵ-first* policy, at the cost of introducing a second exploration parameter. We then go on to show that both exploration parameters can be effectively tuned on-line by extending the ϵ-ADAPT algorithm.

The structure of this chapter is as follows. In Section 6.1 we provide a brief background on multi-agent sequential decision making and communication problems, and the role of the exploration-exploitation trade-off therein. Then in Section 6.2, we introduce the novel multi-agent bandit framework, which allows communication between agents. In Section 6.3 we construct an effective policy for communication and action selection decisions that addresses the exploration-exploitation trade-off. In particular, we propose a novel method of valuing communication between agents, called VOC (Value Of Communication), which finds the best myopic communication decision. We also propose novel exploration policies for this problem called double ϵ-greedy and double ϵ-first, which consider the exploration of communication decisions as well as action decisions. We test the double ϵ-first policy empirically in section 6.4, and discuss the significance of these findings. Finally, in Section 6.5 we extend the ϵ-ADAPT algorithm to this multi-agent framework and demonstrate empirically that exploration of both action and communication decisions can be effectively adapted on-line, without any prefixed exploration parameters, at a slight cost to the reward. Summary remarks follow in Section 6.6.

6.1 Multi-Agent Sequential Decision Making and Communication Problems

Multi-agent sequential decision making problems have been extensively studied in *Markov games* (Littman, 1994; Wang and Sandholm, 2003), otherwise known as *stochastic games* (Shapley, 1953), and also in *Decentralised Markov Decision Processes* (Dec-MDPs) and *Decentralised Partially Observable Markov Decision Processes* (Dec-POMDPs) (Bernstein et al., 2002). These frameworks, where the decision problem changes dependent on the previous state and the subsequent selection of *joint* actions only, can be seen as multi-agent extensions of Markov Decision Processes (MDPs) (Finzi and Lukasiewicz, 2004), which we introduced in Section 2.1.6. Most literature in these frameworks has assumed the reward function is known, so there is no exploration-exploitation trade-off. In other literature, however, the reward function has been treated as unknown, but in these cases the exploration policies used are almost exclusively borrowed from the bandit literature. For example, Hu and Wellman (2003) and Wang and Sandholm (2003) use ϵ-greedy, Carmel and Markovitch (1999) use SoftMax exploration and Littman (1994) combines ϵ-greedy with Q-learning.

The background provided in Chapter 2 therefore also serves as a background to balancing the exploration-exploitation trade-off in multi-agent sequential decision making problems, and many of the algorithms and policies can be applied without any adjustments or changes. A notable addition to the multi-agent literature, however, is *R-MAX* (Brafman and Tennenholtz, 2003), which allocates an optimistic initial estimate to each state and joint action in a stochastic game and updates these estimates until they are assumed to be known (after a sufficient number of visits). This method, which is a multi-agent extension of Q-learning with optimistic initial estimates (Section 2.2.6), ensures under-explored states and joint actions are more likely to be visited in the future. As with Q-learning however, R-MAX requires several parameters that affect finite-time performance, and in addition it is not clear how to extend this algorithm to dynamics or problems with side information. For these reasons we continue to use the simpler and more flexible ϵ-greedy approach as the building block of the off-line and on-line policies constructed in this chapter.

Communication between agents in multi-agent sequential decision making problems has been previously considered in Bayesian games (Gerardi, 2004) which are games where information about the rewards of other agent's actions is unknown or incomplete. The extension to Bayesian games where agents can communicate is based on the idea of *cheap talk* (Farrell and Rabin, 1996), where agents can freely communicate without directly affecting the rewards of the game to each agent. This form of communication is therefore strategic, as agents can attempt to mislead other agents with false information for potential self-benefit (Farrell, 1987). Communication in games has also been considered in network formation games (see Jackson et al. (2003) for a review) where agents must decide whether or not to form links with each other to form a network. In some studies of this problem agents have been allowed to communicate preferences to each other, either at no cost (Aumann and Myerson, 1988) or with a one-off cost (Bala and Goyal, 2000), before any links are made. We note that in this chapter, we consider a different communication framework to that studied in Bayesian games or Network formation games. Specifically, communication is costly and incurred every time information is passed – this is more realistic in various applications, for example in sensor networks (Krause et al., 2006) and disaster management (Ramchurn et al., 2008).

Finally we note that multi-armed bandit problems have been recently studied in a decentralised multi-agent context in Liu and Zhao (2010) and Gai and Krishnamachari (2011). In this framework, a number of distributed agents compete over the set of k actions, but there is no communication, so agents can "collide" and select the same action. In this case, the agents that collide either receive no reward or share the reward from that action in some arbitrary way. The application in mind for this framework is agents contending for opportunistic spectrum access over multiple channels in cognitive radio networks. Our framework in this chapter is very different, as we allow agents to communicate, and do not consider the case of conflicts over actions.

6.2 The Multi-Agent Bandit Framework

Consider a k-armed bandit and let K denote the set of actions, where $|K| = k$. Now consider a set of agents N ($|N| = n$), where each agent a_i[1] controls a disjoint subset C_i of K for $i = 1, \ldots, n$ (i.e. the assignment of actions to agents forms a partition of the set of actions):

$$\bigcup_{i=1}^{n} C_i = K \quad \text{and} \quad C_i \cap C_j = \emptyset \quad \forall i, j, i \neq j.$$

Each action is controlled by one agent only, thus avoiding potential conflicts in action selection decisions. In this version of the bandit problem, each agent a_i can select any number of actions from subset C_i at each time step. Each action $c \in K$ has a reward function $r_c(t)$ based on a p-dimensional covariate $\boldsymbol{x}(t)$ with added observation noise, as used in Chapter 4 (see Equation (4.3)):

[1] In this chapter agents (rather than actions) are denoted a_i.

$$r_c(t) = f_c(\boldsymbol{x}(t), \boldsymbol{\alpha}_c) + \eta_c(t). \tag{6.1}$$

The covariate $\boldsymbol{x}(t)$ is generated from a fixed multivariate distribution, with parameters unknown to the agents. The coefficient vectors $\boldsymbol{\alpha}_c$ for $c = 1, \ldots, k$ are predetermined and also unknown to the agents, and precisely what the agents must learn. We consider fixed coefficient parameters (rather than dynamic) so that we can explicitly focus on the challenges inherent in multi-agent exploration, rather than confounding the decision problem with the difficulties of tracking dynamic decision boundaries (as investigated in the previous chapter).

Each agent a_i only observes a subset $\boldsymbol{y}_i(t)$ of $\boldsymbol{x}(t)$ at time t, representing the agent's partial view of the world. To keep the framework flexible, the subsets of agents' covariate information can be overlapping and there may be covariate information that is not observed by any of the agents at certain time-steps. Agent a_i can request a specific missing covariate value $y^*(t)$ from another agent a_j that has observed this value at a cost denoted $\Pi(y^*(t)|a_i, a_j, \boldsymbol{x}(t), t)$. Agents can request several covariate values from several different agents and the communication cost can be dependent on any function of $\boldsymbol{x}(t)$ or t, which agents are communicating or the number of communications (which can also be limited by bandwidth capacity (Rogers et al., 2005)). This is called the "communication stage" and is an important component of extending bandit problems to realistic multi-agent sequential decision making problems.

The agents are assumed to know the communication cost function (and any bandwidth limitations) *a priori* and to also know which covariate values other agents are observing. Given this, each agent can request unknown covariate values from the least costly agent that has this information. The agent is therefore assumed to know the cost for requesting each subset $\boldsymbol{y}'_i(t) \subseteq \boldsymbol{y}^C_i(t)$ by communication. The set $\boldsymbol{y}^C_i(t)$ is the compliment of $\boldsymbol{y}_i(t)$, but crucially does not contain any covariates that are not observed by any agents (and hence cannot be communicated). The total communication cost is then denoted $\Pi(\boldsymbol{y}_i'(t))$ and is given by:

$$\Pi(\boldsymbol{y}_i'(t)) = \sum_{y^*(t)\in \boldsymbol{y}_i'(t)} \Pi(y^*(t)|a_i, a_{j(y^*(t))}, \boldsymbol{x}(t), t) \tag{6.2}$$

where $a_{j(y^*(t))}$ is the least costly agent from which to acquire $y^*(t)$. We assume this total cost is known for each subset $\boldsymbol{y}_i'(t)$ so that we can explicitly focus on the exploration-exploitation trade-off rather than learning to estimate communication costs on-line. In the simplest case, however, the communication cost is proportional to the number of covariates requested and is independent of $\boldsymbol{x}(t)$ or t and is equal and known to each agent – this would often be the case that the cost of communication is only dependent on the volume and not on the type of information passed.

Agents do not have to communicate 'fully' with other agents and exchange all their covariate information – this is costly and unnecessary (particular when agents share overlapping information). This framework is richer and more general as it allows for partial communication between agents, which is why in our algorithms agent a_i optimises over the set of unknown covariates $\boldsymbol{y}_i^C(t)$ rather than over the set of other agents a_j ($i \neq j$). For applications where agents can only fully communicate, then the agent must then search over a restricted subset of $\boldsymbol{y}_i^C(t)$.

Agents are assumed to receive covariate values truthfully if they are requested – this is feasible because there is no strategic communication in this framework (as opposed to the "cheap talk" principle discussed in the previous section). The reward function, as given in Equation (6.1), is independent of the actions of all other agents, which is the simplest version of a multi-agent communication problem. The interaction between agents therefore occurs at the communication level, and the action selection problem is essentially still a single-agent problem. This allows us to explicitly consider the impact of communication in this chapter, and its relationship to the exploration-exploitation trade-off. Reward functions that are affected by the actions of other agents are considered in the next chapter, where we study the relationship between exploration-exploitation and game theoretic reasoning.

Returning to the communication problem in this chapter, after the communication stage, each agent a_i must then make the decision as to which actions to select from C_i. The agent only receives rewards from actions that are selected, and zero otherwise. Each agent therefore faces a series of interdependent one-armed bandit problems (which are all tied to the same covariate value). This is called the "action stage". Each agent therefore has a "two-stage" decision process which happens strictly sequentially – although effective policies must consider the impact of one decision on the other. Algorithm 6.1 outlines the sequential decision process each agent follows.

Algorithm 6.1 Two-stage decision process for agent a_i

```
1: for $t$ = 1 to $T$ do
2:     Observe $\boldsymbol{y}_i(t) \subseteq \boldsymbol{x}(t)$
3:     Choose covariates values $\boldsymbol{y}'_i(t) \subseteq \boldsymbol{y}_i^C(t)$
4:     Incur Communication cost $\Pi(\boldsymbol{y}'_i(t))$ {See Equation (6.2)}
5:     Choose actions to select $S_i(t) \subseteq C_i$
6:     Receive reward $r_{a_i}(t)$
7: end for
```

The cumulative reward, $R_{a_i}(T)$, is the sum of the rewards received at each time-step, $r_{a_i}(t)$, which in turn is the sum of all rewards observed at time t minus the communication cost, i.e.:

$$R_{a_i}(T) = \sum_{t=1}^{T} r_{a_i}(t) \quad , \qquad r_{a_i}(t) = \sum_{c \in S_i(t)} r_c(t) - \Pi(\boldsymbol{y}'_i(t)), \tag{6.3}$$

where $S_i(t)$ is the subset of actions selected by agent a_i at time t. The communication cost has been placed in the same currency as the reward, so that each agent only needs to maximise one function (namely $R_{a_i}(T)$), which again allows a clear analysis of the interaction between communication and the exploration-exploitation trade-off. With this reward function, an agent should select an action if the expected reward is positive. This creates a series of interdependent one-armed bandit problems with covariates. The interdependence occurs because the rewards are based on the same covariate $\boldsymbol{x}(t)$ and the benefit of receiving one additional covariate value is shared between all actions, but the communication cost is only incurred once by the agent. Therefore, the various policies and algorithms for the one-armed bandit with covariates problem, introduced in Section 2.2, are applicable to this framework. In particular, the ϵ-greedy and ϵ-first policies are used to construct the double ϵ-greedy and double ϵ-first policies (respectively), which are introduced in more detail in the next section.

Each agent has to learn the reward function parameter $\boldsymbol{\alpha}_i$ despite only observing a subset of the full covariate $\boldsymbol{x}(t)$. There are thus two learning problems for the agents: the estimation of parameters subject to noisy data and a missing value problem. These have to be handled concurrently with reward seeking behaviour and thus increases the challenge that each agent faces. High volumes of communication and action selection lead to faster learning, however this can lead to negative rewards – so the agent faces an exploration-exploitation trade-off. Furthermore, the additional communication decision makes the problem of finding a good policy more subtle, in that communication and action decisions have to be jointly considered. A novel method is therefore needed, as the exploration-exploitation of both action and communication decisions have not previously been considered in the same framework. This is discussed in more detail in the next section.

6.3 A Policy for Action and Communication Decisions

The inclusion of multiple communicating agents to the bandit problem introduces a two-stage decision process for each agent, as outlined in the previous section. In the communication stage, agents choose which missing covariates they would like to observe, and then request these values at a corresponding cost. There are essentially two reasons for an agent to communicate: the myopic gain to an agent's subsequent action decision, and the improved learning of unknown parameters. The myopic gain can be estimated using the *Value Of Communication* (VOC) which is constructed in Section 6.3.1. The agent can also explore communication decisions to speed up the learning of unknown parameters. To this end, we construct *double ϵ-greedy* and *double ϵ-first* policies in Section 6.3.2, which both encourage exploration by communication (where the greedy decision is to pick the optimal myopic communication action using the VOC). Similarly, in the action stage, agents have the same two reasons to select actions: the myopic gain to the reward function and the improved learning of unknown parameters. We show that the optimal myopic action can be found automatically, as part of the VOC, and action exploration forms part of the double ϵ-greedy/ϵ-first policy.

Finally, whether or not an agent has communicated or acted, parameter estimates must be updated from the observations. In a bandit problem with fully observed covariates this could be done using regression for a linear reward function (as in Chapter 4), however in the multi-agent framework an agent must handle missing data during parameter estimation. The agent has two basic choices of how to deal with missing data (Scheffer, 2002). The first is case deletion, which in standard inference problems can be either listwise (deleting an entire case if it contains missing data) or pairwise (cases are only deleted if they contain missing data in the analysis being carried out). The second method is imputation, which involves estimating the missing values dependent on other values that have been observed. In the context of our problem, the agent estimates all the reward coefficients using linear regression, so deletion would have to be listwise and hence this method throws away a lot of data when the agent does not observe the full covariate. For this reason, we use imputation. Specifically, we adopt a maximum likelihood approach and use the Expectation-Maximisation (EM) algorithm (outlined in Section 6.3.3) to update estimated reward coefficients in an effective and computationally efficient way, in the presence of missing data.

We note that in the previous chapter we dismissed using algorithms to predict future rewards from missing data in a dynamic rewards setting. In a static setting, however, dealing with missing data is much simpler as models for dynamics are not required, which is why we use the EM algorithm.

6.3.1 The Value of Communication

Agent a_i observes a subset of covariates $\boldsymbol{y}_i(t)$ at time t. After the communication stage the agent will observe a subset of covariates $\boldsymbol{z}_i(t) = \boldsymbol{y}_i(t) \cup \boldsymbol{y}'_i(t)$. Agent a_i

controls a subset of actions $C_i \subseteq K$ and must decide which action $c \in C_i$ to select. If agent a_i has a reward function given by Equation (6.3) then the agent would select action $c \in C_i$ if $\mathrm{E}(r_c(t)|\boldsymbol{z}_i(t), \hat{\boldsymbol{\alpha}}_c) > 0$, where:

$$\mathrm{E}(r_c(t)|\boldsymbol{z}_i(t), \hat{\boldsymbol{\alpha}}_c) = A + \sum_{x_d(t) \notin z_i(t)} \hat{\alpha}_{c,d} \int x_d(t)\ \mathrm{p}(x_d(t)|\boldsymbol{z}_i(t))dx_d(t)$$
$$A = \hat{\alpha}_{c,1} + \sum_{x_d(t) \in z_i(t)} \hat{\alpha}_{c,d} x_d(t), \tag{6.4}$$

where $r_c(t)$ is given by Equation (6.1) and $\mathrm{p}(x_d(t)|\boldsymbol{z}_i(t))$ is the probability of $x_d(t)$ given $\boldsymbol{z}_i(t)$. Equation (6.4) is the myopic reward to agent a_i for selecting action c at time t. Agent a_i can then find the optimal subset of actions $S_{i,t} \subseteq C_i$ to select at time t using Algorithm 6.2.

Algorithm 6.2 Optimal myopic action for agent a_i at time t

```
1: Observe $\boldsymbol{z}_i(t) = \boldsymbol{y}_i(t) \cup \boldsymbol{y}'_i(t)$
2: for $c \in C_i$ do
3:     Calculate $\mathrm{E}(r_c(t)|\boldsymbol{z}_i(t), \hat{\boldsymbol{\alpha}}_c)$ from Equation (6.4)
4:     if $\mathrm{E}(r_c(t)|\boldsymbol{z}_i(t), \hat{\boldsymbol{\alpha}}_c) > 0$ then
5:         $c \in S_i(t)$
6:     end if
7: end for
8: Select actions $S_i(t) \subseteq C_i$
9: Receive reward $r_{a_i}(t)$ {as given in Equation (6.3)}
```

Before agent a_i communicates, its expected reward at time t is the *Value Of Silence* (VOS) given by:

$$\mathrm{VOS}_{a_i} = \sum_{c \in C_i} \max\left(0, \mathrm{E}(r_c(t)|\boldsymbol{y}_i(t))\right). \tag{6.5}$$

Note that the VOS is bounded below by zero, corresponding to the agent selecting no actions at time t, which occurs when all actions are expected to yield a negative reward.

Agent a_i can observe a subset of covariates $\boldsymbol{y}'_i(t) \subseteq \boldsymbol{y}^C_i(t)$ by communication at a cost $\Pi(\boldsymbol{y}'_i(t))$ (as defined in Section 6.2). If agent a_i knows the joint distribution of $\boldsymbol{x}_t$ then it can find the VOC for the subset $\boldsymbol{y}_i'(t)$:

$$\mathrm{VOC}_{a_i,y_i'(t)} = \sum_{c \in C_i} \mathrm{VOC}_{c,y_i'(t)} - \Pi(\boldsymbol{y}_i'(t)), \tag{6.6}$$

where,

$$\mathrm{VOC}_{c,y_i'(t)} = \int \max\left(0, B + \sum_{x_d(t) \in y_i'(t)} \hat{\alpha}_{c,d} x_d(t)\right) \mathrm{p}(\boldsymbol{y}_i'(t)|\boldsymbol{y}_i(t)) d\boldsymbol{y}_i'(t), \tag{6.7}$$

$$B = \hat{\alpha}_{c,0} + \sum_{x_d(t) \in y_i(t)} \hat{\alpha}_{c,d} x_d(t) + \sum_{x_d(t) \notin z_i(t)} \hat{\alpha}_{c,d} \int x_d(t)\ \mathrm{p}(x_d(t)|\boldsymbol{y}_i(t)) dx_d(t).$$

The VOC calculates whether the probability that the expected reward of an action is positive or negative, after observing $\boldsymbol{y}_i'(t)$. In the instances where this is negative the agent would not select this action and thus receive no reward, but still incur the costs of communication. The VOC is therefore the expected reward to agent a_i at time t, with myopic action selection, if it first requests to observe the covariate values $\boldsymbol{y}_i'(t)$. Agent a_i can maximise this value over all possible subsets $\boldsymbol{y}_i'(t) \subseteq \boldsymbol{y}_i^C(t)$ (not including the empty set, $\boldsymbol{y}_i'(t) = \emptyset$, which is the VOS given in Equation (6.5)), to find the maximum VOC value. The agent then requires this value to be bigger than the VOS, otherwise the agent should not communicate at all. If the communication cost is zero then trivially the maximum VOC would always correspond to choosing the full subset $\boldsymbol{y}_i'(t) = \boldsymbol{y}_i^C(t)$. In effect, each agent must learn to partition the space of observed covariate values into regions where, in each region, a specific subset of unknown covariates is the optimal subset to observe through communication.

Algorithm 6.3 outlines how agent a_i can find the optimal subset of covariates to request by communication using the VOC. The number of possible subsets of $\boldsymbol{y}_i'(t)$ grows exponentially with the size of $\boldsymbol{y}_i^C(t)$, so this search is computationally intensive for large volumes of unobserved side information. In such cases, however, approximations such as forward induction could be used and furthermore, the size of $\boldsymbol{y}'_i(t)$ is likely to be restricted by bandwidth in many applications, which will also restrict the size of this computation.

The solution presented in Algorithm 6.3 is only optimal myopically, as the benefits of exploration have not been factored in. Furthermore, the VOC can only be found precisely with perfect knowledge of the conditional densities of the covariates and the coefficients of the reward function. In reality, these have to be learnt by the agents over time, and hence the VOC can only be approximated on-line. Additional exploration of communication decisions can therefore have a positive effect on the cumulative reward, as this improves future communication decisions (through the increased learning of the VOC boundaries), which in turn will improve future action decisions. Exploration of actions can benefit an agent's reward also, in the same way as with the single-agent bandit problems studied in Chapters 3-5. As a result, in the next section we outline an effective exploration policy, that combines exploration through communication and action decisions.

Algorithm 6.3 Optimal myopic communication decision for agent a_i at time t using the VOC

```
1: Observe $\boldsymbol{y}_i(t) \subseteq \boldsymbol{x}(t)$
2: for all $\boldsymbol{y}'_i(t) \subseteq \boldsymbol{y}^C_i(t)$ do
3:     for all $c \in C_i$ do
4:         Find $\text{VOC}_{c,y'_i(t)}$ {Equation (6.7)}
5:     end for
6:     $\text{VOC}_{a_i,y'_i(t)} = \sum_{c \in C_i} \text{VOC}_{c,y'_i(t)} - \Pi(\boldsymbol{y}'_i(t))$
7: end for
8: Find $\text{VOS}_{a_i}$ {Equation (6.5)}
9: if $\max_{y'_i(t)} \text{VOC}_{a_i,y'_i(t)} > \text{VOS}_{a_i}$ then
10:     Request covariates $\boldsymbol{y}'_i(t)$ by communication
11: else
12:     Do not communicate
13: end if
```

6.3.2 The Double ϵ-greedy and Double ϵ-first Policies

The communication and action policies detailed in the last section are optimal myopic policies, and are thus purely greedy (or exploitative). Nevertheless, in order to improve the accuracy of these decision over time, the agents may have to perform additional exploration to aid their learning. We note, however, that actions by the agents can be explorative as well as exploitative, particularly if the VOC encourages a high volume of communication, or if expected reward calculations encourage a high proportion of actions to be selected. Nonetheless, in a noisy environment with unknown parameters it is likely that additional exploration may benefit the agents. Exploration by communication can be easily increased by requesting more covariate values than the VOC suggests. Exploration by action, similarly, can be increased by selecting additional actions, even though their expected rewards are negative.

To encourage exploration by both communication and action, a double ϵ-greedy policy is proposed, which uses the ϵ-greedy policy separately for both decision processes. We use the ϵ-greedy approach for the same reasons as before: its simplicity, applicability to broad frameworks and its consistent strong performance in finite-time experiments. In the context of this multi-agent framework, we can construct a double ϵ-greedy policy, where covariates that are not selected for communication using the VOC are still requested – each with probability ϵ_1, and actions that are not selected because of their positive estimated expected reward are still selected – each with probability ϵ_2. This policy is formalised in Algorithm 6.4. The optimal parameters, ϵ_1 and ϵ_2, are inter-dependent and will depend on factors such as the communication cost, the degree of noise in the data, and the unknown coefficients of the reward function.

Algorithm 6.4 Double ϵ-greedy Policy

```
1: Set $\epsilon_1$ and $\epsilon_2$
2: **for** $t = 1$ to $T$ **do**
3:     Observe $\boldsymbol{y}_i(t) \subseteq \boldsymbol{x}(t)$
4:     Find optimal subset of covariates $\boldsymbol{y}'_i(t) \subseteq \boldsymbol{y}^C_i(t)$ to request through commu-
       nication using VOC {Using Algorithm 6.3}
5:     Request each covariate $y^*(t)$ that is an element of $\boldsymbol{y}^C_i(t)$, but not an element
       of $\boldsymbol{y}_i(t)$ or $\boldsymbol{y}'_i(t)$, with probability $\epsilon_1$
6:     Find optimal subset of actions to select $S_i(t) \subseteq C_i$ {Using Algorithm 6.2}
7:     Select each action that is in $C_i$, but not in $S_i(t)$, with probability $\epsilon_2$
8:     Update unknown parameters
9: **end for**
```

A similar policy can be devised using the ϵ-first policy. Specifically a double ϵ-first policy, where all covariate values are requested and all actions are selected for the first $\epsilon_1 T$ and $\epsilon_2 T$ iterations (respectively). Afterwards, the agent is greedy and uses the VOC exclusively for communication decisions and maximises expected reward for action decisions. In the static bandit problems studied in Chapters 3 and 4, we stated that ϵ-first would perform better than ϵ-greedy as the agent has more future time-steps to benefit from past exploration. This would not necessarily be the case in the multi-agent bandit problem however, as the agent might gain less myopic value in receiving all covariates for the first $\epsilon_1 T$ iterations for example, rather than having fewer additional covariate values spread throughout the game. For this reason we propose both policies and note that the stronger performing policy is likely to be dependent on the number of actions each agent controls and the dimension of unobserved covariates.

6.3.3 Dealing with Missing Data

The agents have to iteratively update parameter estimates of the reward functions and covariates (Line 8, Algorithm 6.4). With linear reward functions, the coefficients can be updated using recursive least squares estimation (see Appendix B). The agents, however, do not always observe all covariate values, even after communication. This induces a missing value problem and the agent has the choice of imputing these missing values or deleting observations if they contain missing data (as discussed on page 139). Due to the potential high occurrence of missing data, deletion methods are not practical for this framework. We therefore impute the data using a likelihood approach. To this end, an Expectation-Maximisation (EM) algorithm (Dempster et al., 1977) in conjunction with least squares estimation can be used, to iteratively update each agent's parameters. The EM algorithm is a computationally efficient and robust method for dealing with missing data, that can be practically implemented even if the number of agents/variables are high – which is important for the application of multi-agent systems to realistic scenarios.

In more detail, the EM algorithm is a procedure for maximum likelihood inference in the presence of missing data. Starting from an initial guess of the values for the parameter vector, $\theta(0)$, it employs an iterative update step, each time choosing $\theta(i+1)$ to maximise the expected log-likelihood of the observed data, where the expectation is taken over the missing data with respect to the current estimate $\theta(i)$. Once the change in expected log-likelihood is smaller than some pre-defined threshold then the algorithm terminates and missing values have been imputed.

6.4 Numerical Results

In this section we test the framework and proposed policies in a 2-agent version of the problem. We consider the case where each agent controls one action and observes a different covariate value – specifically, the covariate is 3-dimensional (the first dimension is always equal to 1 as in Equation (2.3)) and one agent always observes the second dimension and the other the third. This is perhaps the simplest possible formulation of the multi-agent framework and is considered firstly to illustrate the selection behaviour of the policies and secondly to show that exploration is needed even though the decision problem is relatively simple. The behaviour of systems with more agents and higher-dimensional covariates will share characteristics with this case study, but the detailed study of which is reserved for future work. In the 2-agent problem, the reward function of agent a_i ($i = 1, 2$) is given by:

$$r_i(t) = \sum_{j=1}^{3} \alpha_{i,j} x_j(t) + \eta_i(t), \tag{6.8}$$

The coefficient values $\alpha_{i,j}$ are predetermined and unknown to the agents. The covariate values $\boldsymbol{x}_{2:3}(t)$ are i.i.d. draws from a bivariate normal distribution: $\boldsymbol{x}_{2:3}(t) \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, where the parameters are unknown to the agents (recall that $x_1(t) = 1$). In our experiments we set:

$$\boldsymbol{\mu} = \mathbf{0}, \Sigma = \begin{pmatrix} 1 & 0.4 \\ 0.4 & 1 \end{pmatrix}, \alpha = \begin{pmatrix} 0.1 & 0.3 \\ 0.2 & 0.2 \end{pmatrix},$$

such that the covariate values are weakly correlated and each agent can therefore benefit from accurately learning the VOC. Agent a_i only observes $x_{i+1}(t)$ at each iteration. The noise $\eta_i(t)$ is also normally distributed and i.i.d., with zero mean and variance 0.05 (such that CNR = 20). The length of play considered is 100 iterations, long enough for the agents to start exploiting, but short enough such that the agents must learn quickly and effectively.

6.4.1 Application of the VOC

The optimal myopic communication decision can be found if the agent a_i knows the values of $\alpha_{i,1}$, $\alpha_{i,2}$, $\boldsymbol{\mu}$ and Σ; however the agent must learn these over time. In this 2-agent scenario, the VOC from Equation (6.6) becomes (see Appendix C.1):

$$\begin{aligned}\text{VOC}_{a_i} =& \left(\hat{\alpha}_{i,1} + \hat{\alpha}_{i,i+1}x_{i+1}(t) + \hat{\alpha}_{i,j+1}c_1\right)\Phi\left(-\text{sign}(\hat{\alpha}_{i,j+1})\frac{c_3 - c_1}{\sqrt{c_2}}\right) \\ &+ \hat{\alpha}_{i,j+1}\sqrt{\frac{c_2}{2\pi}}\exp\left(\frac{-(c_3-c_1)^2}{2c_2}\right) - \Pi, \qquad (6.9)\end{aligned}$$

where Π is the communication cost (assumed constant) and,

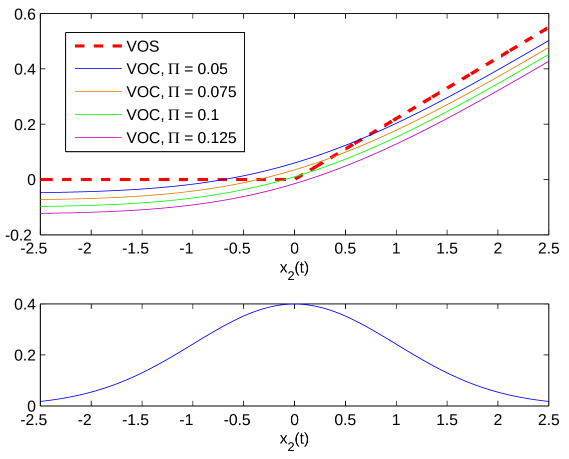
$$\begin{aligned} c_1 =& \hat{\mu}_j + \frac{\hat{\Sigma}_{i,j}}{\hat{\Sigma}_{i,i}}(x_{i+1}(t) - \hat{\mu}_i), \\ c_2 =& \hat{\Sigma}_{j,j} - \frac{\hat{\Sigma}^2_{i,j}}{\hat{\Sigma}_{i,i}}, \\ c_3 =& \frac{-\hat{\alpha}_{i,1} - \hat{\alpha}_{i,i+1}x_{i+1}(t)}{\hat{\alpha}_{i,j+1}}. \end{aligned}$$

for $i = 1, 2$ (where $j = 2, 1$), where $\hat{\Sigma}_{i,j}$ is the $\{i, j\}$th entry of the matrix $\hat{\Sigma}$ (and similarly for other vectors and matrices). $\Phi(x)$ is the cdf of the standard normal distribution and the values c_1 and c_2 are the mean and variance, respectively, of the unknown covariate, $x_{j+1}(t)$, given the known covariate, $x_{i+1}(t)$. The VOC is therefore only dependent on the parameters that agent a_i needs to learn, the observed covariate $x_{i+1}(t)$ and the communication cost. The VOS from Equation (6.5) simply becomes (see Appendix C.1):

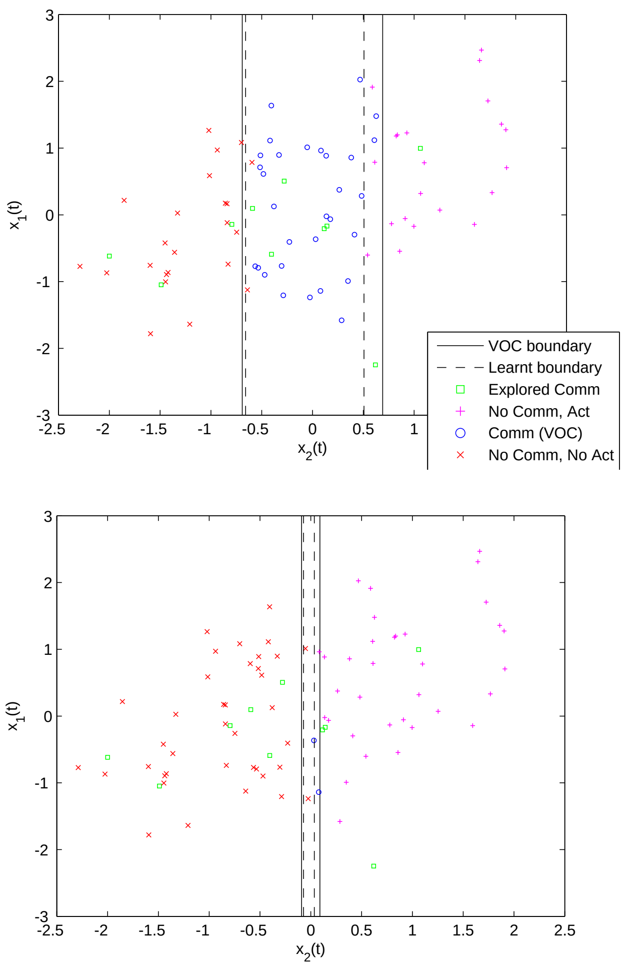
$$\text{VOS}_{a_i} = \max(0, \hat{\alpha}_{i,1} + \hat{\alpha}_{i,i+1}x_{i+1}(t) + \hat{\alpha}_{i,j+1}c_1) \tag{6.10}$$

Agent a_i communicates if the VOC is greater than the VOS. To this end, Figure 6.1(a) displays VOC_{a_1} and VOS_{a_1} for different values of $x_2(t)$ and for various communication costs (which are set as constant values), with the distribution of $x_2(t)$ plotted underneath. There is a region where the VOC is higher than the VOS (except with the highest communication cost) and the agent should communicate when the observed covariate value falls in this region. The agent can find this "region of communication" over time, by learning the unknown parameters correctly. In this region the unknown covariate value will be informative as to whether the expected reward is positive or negative. Conversely, for covariate values outside the region of communication, the optimal action decision is clear enough (as the agent knows whether the reward is likely to be positive or negative) and it is hence not worth incurring the communication cost to verify this. As expected, the region of communication is larger for smaller communication costs.

Figure 6.1(b) demonstrates how the agent, using the double ϵ-first policy, has learnt the region of communication in relation to each communication cost for a particular replication of the 2-armed problem. Furthermore, the agent has made the correct action and communication decisions for most observed covariate values and has therefore learnt to partition the covariate space correctly. The points highlighted by green squares show the points where exploration by communication has occurred (i.e. the agent has been willing to explore outside of the region of communication to aid its learning). For other parameter values the region of communication may not exist (if the covariance between the known and unknown covariate is high for example) or be infinite (if the communication cost is zero for example). Nevertheless, this region does not have to be explicitly found as the VOC only needs to be calculated for the covariate values observed at each iteration.



(a)



(b)

Figure 6.1: (a) VOC_{a_1} and VOS_{a_1} over different values of $x_2(t)$ for varying constant communication costs (with the distribution of $x_2(t)$ plotted below) and (b) the decisions by agent a_1, using a double ϵ-first policy, over 100 iterations with ϵ_1 and ϵ_2 set at 10% with $\Pi = 0.05$ (top) and $\Pi = 0.1$ (bottom), for a 2-armed problem.

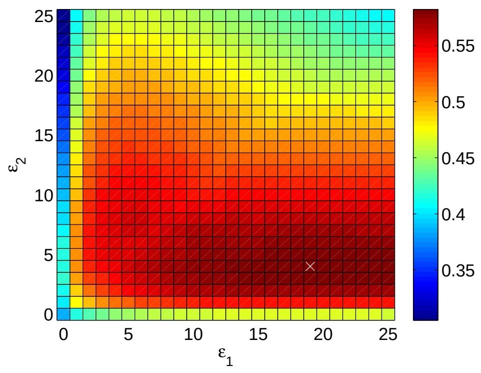
6.4.2 Performance of the Double ϵ-first Policy

In the previous section, we outlined how the agents can approximate the VOC and demonstrated that this approximation can be made accurately with some degree of exploration. In this section, we explore the effect of the double ϵ-first policy on the agent's cumulative reward. We use the double ϵ-first policy for this problem, rather than the double ϵ-greedy policy, as for the 2-dimensional case considered in this section, the double ϵ-first policy will perform better on average. This is because there is only one explorative choice for both communication and action and it is beneficial to do these explorative steps early in the game.

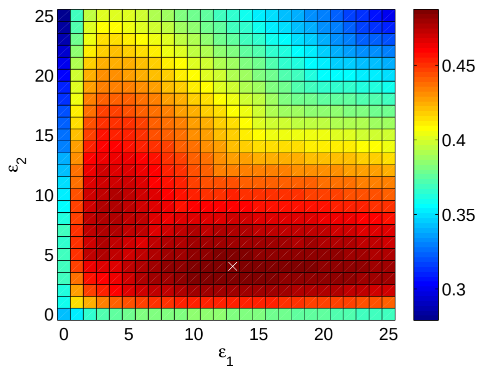
To this end, Figure 6.2 shows the average cumulative reward to agent a_1 (over 10,000 repeats) for various communication costs, using the double ϵ-first policy over a grid of values for ϵ_1 and ϵ_2 ranging between 0 and 25%. The optimal combination of parameters is denoted with a cross. Notice that the agent benefits from exploring by both communication and action – greedy selection in either decision process leads to poor performance. Additionally, there is a correlation between ϵ_1 and ϵ_2 that is dependent on the communication cost. As expected, the amount of optimal exploration by communication (ϵ_1), is inversely related to the communication cost. To a lesser extent, the amount of optimal exploration by action (ϵ_2) is positively correlated with the communication cost; this is due to the fact that a total amount of *global* exploration is required, and as exploration by communication becomes more costly, the agent requires more exploration by action to perform a reasonable amount of learning (and vice-versa).

6.5 Implementing the ϵ-ADAPT Algorithm

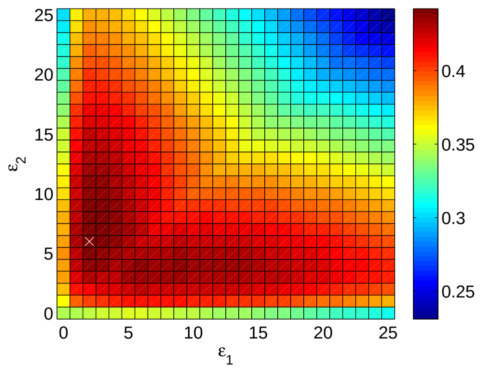
In the previous section we demonstrated that the agents can effectively balance the exploration-exploitation trade-off in this multi-agent framework by using a double ϵ-first policy. This off-line policy requires an additional exploration parameter, however, which further restricts the feasibility of implementing this policy in any practical domain. To this end, in this section we extend the ϵ-ADAPT on-line algo-



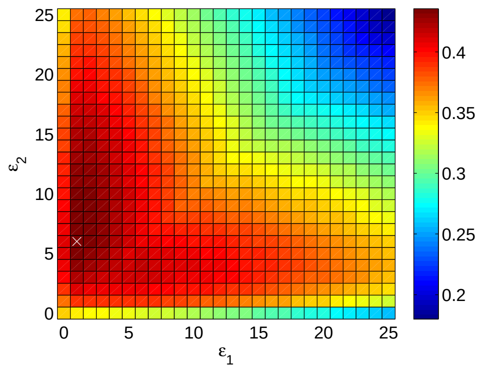
(a)



(b)



(c)



(d)

Figure 6.2: The average cumulative reward to agent a_1, for various communication costs: a) 0.05, (b) 0.075, (c) 0.1 and (d) 0.125, using a double ϵ-first policy, where $0 \leq \epsilon_1, \epsilon_2, \leq 25\%$.

rithm of Chapter 4 to the multi-agent bandit framework, where communication and action decisions are adapted on-line. The main challenge lies in removing the need for both exploration parameters, without incurring too much loss in the overall reward (as compared with the optimally tuned off-line policy). We describe how this can be done in Section 6.5.1 and then report on some simulations in Section 6.5.2.

6.5.1 Adapting Communication and Action Decisions On-Line

As with the double ϵ-greedy/ϵ-first approach (see Algorithm 6.4), we continue to select myopically optimal communication and action decisions (using the VOC). Rather than requesting additional covariates and selecting additional actions with some predefined probability, we now determine this rate of exploration using on-line MC approximations that mimic the decision problem faced. For the single-agent one-armed bandit problem of Chapter 4, we regenerated past covariates and rewards and then simulated the rest of the game to see which policy performed better: greedy selection or one more round of exploration. The likelihood of exploration was driven by the agent's level of uncertainty, which was linked to the amount of noise and the sample size of the unknown action.

In the multi-agent setting, however, this method of approximation would require repeating past covariates and rewards with missing values (due to the partially-observed side information) and then using embedded EM algorithms at each step of the MC approximation. This is computationally demanding and very problem specific. Furthermore, there is the complication of the two-stage decision process and deciding whether to explore by communication or action. To overcome these issues, we instead perform an MC approximation of a decision problem that is *similar* to the one faced by each agent, in terms of the amount and type of exploration required. Specifically, we capture the effect of increased occurrence of missing values by increasing the noise variance estimates used in the MC approximation, such that missing values create more uncertainty which in turn encourages more exploration. We then do not need to regenerate the problem with partial covariate observations and can instead regenerate with a fully-observed covariate (such that we also do not need to compute VOC values), but with increased reward observation noise. This method of approximation transforms the multi-agent problem into the simplest sequential decision making problem (i.e. a bandit problem) which is then a much easier task for the agent in terms of balancing exploration with exploitation.

The key step is therefore how to calculate the noise variance estimates for each action $c \in C_i$ to incorporate the uncertainty from missing data. We can do this at time t using the following estimate:

$$\begin{aligned}\hat{\sigma}^2_{\eta_c}(t) =& \frac{1}{n_c}\left((n_c-1)\hat{\sigma}^2_{\eta_c}(t-1)+\xi^2\right), \\ \xi =& \left(r_c(t) - \sum_{x_d(t)\in z_i(t)} \hat{\alpha}_{c,d}x_d(t) - \sum_{x_d(t)\notin z_i(t)} \hat{\alpha}_{c,d}\mathrm{E}(x_d(t)|\boldsymbol{z}_i(t))\right),\end{aligned} \tag{6.11}$$

when the action is selected (and kept the same otherwise). $\hat{\sigma}^2_{\eta_c}(t)$ is therefore the estimated *prediction* error given the observed information. The instantaneous prediction error, ξ, will on average be higher when there are more missing covariates ($x_d(t) \notin \boldsymbol{z}_i(t)$), which increases the value of $\hat{\sigma}^2_{\eta_c}(t)$. Note that we did something similar in Chapter 5 with dynamic problems, where we regenerated a similar problem that was static with increased noise variance for highly dynamic problems.

In addition, we saw in the previous section that the optimal rates of exploration by communication and action are interdependent and the balance between the two optimal rates are tied to the communication cost. For this reason, we perform two types of MC approximation: one for exploration by communication and the other for exploration by action. We outline the ϵ-ADAPT algorithm in Algorithm 6.5, with the communication MC approximation given in Appendix C.2 (Algorithm C.1) and the action MC approximation identical to that given in Algorithm 4.2 for single-agent one-armed problems[2]. The MC approximations are similar, but the crucial difference between them is the increased cost of communication, which is explicitly factored in with exploration by communication (line 23, Algorithm C.1).

[2]Note that in Line 5 of Algorithm 4.2, we keep the true covariate at time t. This may be not fully observed in the multi-agent problem, so we draw unknown values conditional on observed values.

Algorithm 6.5 ϵ-ADAPT for a Multi-Agent Bandit Problem

```
1: for $t = 1$ to $T$ do
2:     Observe $\boldsymbol{y}_i(t) \subseteq \boldsymbol{x}(t)$
3:     Find optimal subset of covariates $\boldsymbol{y}'_i(t) \subseteq \boldsymbol{y}_i^C(t)$ to request through communication using VOC {Algorithm 6.3}
4:     Request additional covariates, $\boldsymbol{z}_i^*(t)$, that maximise the function $R_{\epsilon f}(T^*, \boldsymbol{z}'_i(t))$ {Algorithm C.1}
5:     Find optimal subset of actions to select $S_i(t) \subseteq C_i$ {Algorithm 6.2}
6:     Calculate approximations for $R_{\epsilon f}(T^*, 0)$ and $R_{\epsilon f}(T^*, 1/T^*)$ for remaining actions {Algorithm 4.2}
7:     Select actions where $R_{\epsilon f}(T^*, 0) < R_{\epsilon f}(T^*, 1/T^*)$
8:     Update noise variance estimates $\sigma^2_{\eta_c}(t)$ {Equation (6.11)}
9:     Update unknown parameters {Using EM-algorithm with missing values}
10: end for
```

6.5.2 Numerical Results

In this section we repeated the experiments of Section 6.4 to compare ϵ-ADAPT against the double ϵ-first policy, to see how our on-line algorithm performs against an optimally tuned off-line policy. The results are displayed in Table 6.1. ϵ-ADAPT has yielded a high reward that is approximately 95% of the optimal double ϵ-first policy, despite the difficult challenge of removing two exploration parameters. The table also shows that ϵ-ADAPT yields a significant improvement as compared with the greedy policy or when only one decision process is optimally explored (so that there is only one exploration parameter and the other is set to zero). The performance of ϵ-ADAPT could be potentially further improved if missing values were incorporated, but as discussed earlier, this would slow the algorithm down considerably. Furthermore, this would be an *ad hoc* version of ϵ-ADAPT, whereas the algorithm proposed in this section retains the same characteristics as the algorithms used in previous chapters, which demonstrates the algorithm's flexibility.

Table 6.1: Comparison of ϵ-ADAPT and off-line policies for a multi-agent problem

Π	Reward	Opt. Dbl.ϵ-first	$\epsilon_1 = \epsilon_2 = 0$ (greedy)	Opt. ϵ_2, $\epsilon_1 = 0$	Opt. ϵ_1, $\epsilon_2 = 0$
0.05	0.545	93.6%	140.7%	116.6%	129.9%
0.075	0.468	95.9%	136.0%	121.3%	126.5%
0.1	0.427	96.5%	124.2%	120.8%	111.2%
0.125	0.415	95.2%	118.3%	118.3%	101.3%

To gain more insight, Figure 6.3 demonstrates the rate of exploration (by communication and action) over time. Both types of exploration decay over time, as they should, and moreover ϵ-ADAPT explores more by communication for low communication costs (and vice-versa). This demonstrates that ϵ-ADAPT can learn *how* to explore, in addition to when, which action and how much (as demonstrated in previous chapters).

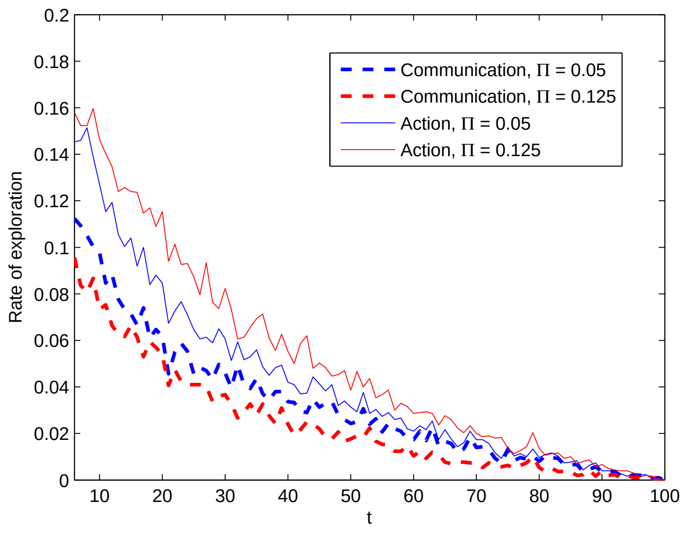


Figure 6.3: The average rate of exploration by ϵ-ADAPT (by communication and action) over time for high and low communication costs, for a 2-armed problem.

6.6 Summary

In this chapter we have proposed a new framework for modelling sequential decision making problems in multi-agent systems. We have extended the multi-armed bandit problem to investigate the exploration-exploitation trade-off in a multi-agent context. Specifically, we have investigated sequential decision making of communication decisions between agents, which is relevant and applicable to many other multi-agent problems.

In more detail, we have constructed novel algorithms for selecting communication and action decisions. The exploitative element of these algorithms involve using the Value of Communication (VOC) to myopically value the optimal communication and action decisions. The explorative element was first designed using a double ϵ-greedy or double ϵ-first policy, which randomly performs more communication or selects additional actions, to benefit the agent's learning. Later we extended the ϵ-ADAPT algorithm (constructed in Chapter 4) to this framework, which removes the need for any prefixed exploration parameters.

In an empirical evaluation of a 2-agent problem, both the double ϵ-first and ϵ-ADAPT methods significantly outperformed the greedy policy. Moreover, both methods, which combine exploration by both communication and action, perform better than doing exploration by one method and not the other. ϵ-ADAPT performs close to the optimal double ϵ-first policy, despite having to remove two prefixed exploration parameters. Furthermore, we have shown that agents can benefit from exploring by communication – agents should hence not communicate with other agents for myopic gain only. This novel framework has therefore developed new ideas about balancing exploration-exploitation in a multi-agent setting where rewards of actions are unknown *a priori*. The framework also includes the possibility of agents communicating, which is central to many real world scenarios modelled by MAS.

This framework is novel in that exploration-exploitation of joint action and communication decisions are considered simultaneously, however the framework is restrictive in that the interaction of agents is constrained to communicating side information – there is no interaction of rewards between agents. We consider this more realistic feature in the next chapter, where we consider 2×2 repeated games with unknown rewards, which introduces game theoretic considerations into the exploration-exploitation trade-off.

Chapter 7

Learning and Exploring in 2-player Repeated Games

In the previous chapter we extended bandit problems to multi-agent systems such that agents could communicate information prior to subsequent action selections. This extension is a *distributed* bandit problem (a phrase first coined by Claus and Boutilier (1998)), where the control of actions and knowledge of side information is partitioned between participating agents in the system. In this framework, however the reward functions of the individual agents are not explicitly affected by the actions of other agents – which is not realistic in a number of multi-agent applications. If individual mobile sensors are controlled by agents in a distributed network, for example, the expected value of one sensor choosing to move to a certain area is dependent on the locations of other sensors (Stranders et al., 2009). Alternatively, the expected reward for an action made by an emergency service vehicle in a disaster is likely to be dependent on the actions of other vehicles (Ramchurn et al., 2008).

To this end, in this chapter we consider a different extended bandit framework where the rewards to each agent are explicitly affected by the actions of others. Specifically, we study a 2-agent problem, where each agent has two available actions, but all expected rewards are unknown *a priori*. This extension of the bandit problem is therefore a 2-player repeated game, which has been extensively studied for general games in Fudenberg and Maskin (1986) and Abreu (1988), amongst many others, and also for specific case games, such as the *Iterated Prisoner's Dilemma* problem in Axelrod (1987), for example. The key difference between this literature and our framework is that we assume expected rewards for different sets of joint actions are unknown *a priori* and must be learnt over time. This is more realistic in several agent applications (as discussed in Chapter 6) and creates an exploration-exploitation trade-off for each agent. Moreover, the study of repeated games with unknown rewards incorporate ideas from both decision theory and game theory, allowing a detailed study of how the exploration-exploitation trade-off should be combined with game theoretic reasoning in finite-time multi-agent problems. Note that in this chapter we refer to action selection policies as *strategies*, to acknowledge the relevant strategic and game theoretic considerations made by each agent.

The repeated games with unknown rewards framework has been well studied (Claus and Boutilier, 1998; Chapman et al., 2011; Babes et al., 2009; Marden et al., 2009) but findings have been restricted to proving convergence to Nash equilibria (defined in Section 7.1) in 2-player games for various homogeneous strategies in self-play. There have been few inroads however in finding strategies that maximise reward in finite time against both homogeneous and heterogeneous opponents, which is the focus of this chapter, in particular as this is more relevant and applicable to real-world multi-agent scenarios. As noted in Claus and Boutilier (1998), the problem of learning rewards in games can also be viewed as a distributed bandit problem – we therefore continue to use bandit exploration policies. We restrict our attention to two-agent, two-action problems (also known as $2{\times}2$ games), to focus on the fundamental relationship between learning, exploration and strategic interaction. We note, however, that this characterisation is still useful in a wider setting (with more agents and actions) as, in particular, many real-world situations can be modelled in this way (Govindan and Wilson, 2010). Specifically, the actions of all opposing agents are represented as the action of one agent and all sub-optimal actions are treated as the alternative action to the optimal.

In more detail, in this chapter we first examine the impact of both agents using non-explorative strategies such as greedy and fictitious play (defined in Section

7.2). Specifically, we show that in the presence of unknown rewards, these strategies can exhibit behaviour quite different from the known rewards setting. We explain this behaviour by proving that non-explorative strategies can converge to non-Nash equilibria for $2{\times}2$ games. We then consider introducing exploration to one agent, initially by using ϵ-greedy, and demonstrate the benefits against a non-explorative opponent. Specifically, we provide simulation results from various motivating examples, to show that an explorative agent can *exploit by exploring* – i.e. gain a higher reward at the expense of the opponent's, with a suitably tuned exploration parameter. This motivates extending the ϵ-ADAPT algorithm to this framework, to see if exploration can be adapted on-line without the need for an *a priori* tuned exploration parameter.

This chapter is structured as follows. Section 7.1 outlines the framework and case study games used in this chapter. Section 7.2 defines two separate types of multi-agent learners: individual and joint-action learners. We also propose several different strategies for selecting actions. Section 7.3 investigates some of our case study games with strategies using no exploration and Section 7.4 provides a proof showing the possible convergence of such strategies to non-Nash equilibria. Sections 7.5 and 7.6 investigate our case study games but this time with the agents using explorative strategies. In Section 7.7 we extend ϵ-ADAPT to this framework and perform some simulations against off-line strategies. Summary remarks follow in Section 7.8.

7.1 Framework and Case Study Games

Agent A and agent B repeatedly play a stage game where they both choose between Action 1 and Action 2 at each time-step, $t = 1, 2, 3, \ldots, T$. Agent $k = \{A, B\}$ receives a reward $r_k(t)$, where:

$$r_A(t) = a(i, j) + \eta(t), \tag{7.1}$$

$$r_B(t) = b(i, j) + \nu(t), \tag{7.2}$$

where $i, j \in \{1, 2\}$ are the actions picked by agents A and B respectively, at time t. $\eta(t)$ and $\nu(t)$ are noise processes which are i.i.d. Gaussian and centred at zero with variance σ_η^2 and σ_ν^2 respectively. The expected reward matrix of the stage game follows Table 7.1. The agents' objective is to maximise their cumulative reward $R_k(T) = \sum_{t=1}^{T} r_k(t)$. The agents observe their own reward and the action selected by the opponent, but not the reward received by the opponent.

		Agent B	
		Action 1	Action 2
Agent A	Action 1	$a(1,1), b(1,1)$	$a(1,2), b(1,2)$
	Action 2	$a(2,1), b(2,1)$	$a(2,2), b(2,2)$

Table 7.1: Expected reward matrix of the 2×2 stage game

Reward matrices where $a(i,j) = -b(i,j)$ $(\forall i, j)$ are referred to as zero sum games (and non-zero sum games otherwise). A *Nash Equilibrium* (Nash, 1951) exists when each agent is playing a strategy such that no individual agent can benefit from a unilateral change to their strategy. In a 2×2 game, for example, $a(1,i) > a(2,i)$ and $b(j,1) > b(j,2)$ ($\forall i, j$) corresponds to a Nash Equilibrium of both agents selecting Action 1 – referred to as a *pure strategy* Nash Equilibrium. On the other hand, when $a(1,1) > a(2,1)$, $a(1,2) < a(2,2)$, $b(1,1) < b(1,2)$ and $b(2,1) > b(2,2)$ (for example), then the Nash Equilibrium is for both agents to play a *mixed strategy* (a randomised strategy where Action 1 is selected with probability p_1 and Action 2 with probability $1 - p_1$). Conversely, $a(1,1) > a(2,1)$, $a(1,2) < a(2,2)$, $b(1,1) > b(1,2)$ and $b(2,1) < b(2,2)$ corresponds to 2 or 3 possible Nash Equilibria: two pure (both agents selecting Action 1 or both agents selecting Action 2) and possibly one mixed (if $a(1,1) > b(1,1)$ and $a(2,2) < b(2,2)$ for example). There is at least one Nash Equilibrium in any 2×2 game (Nash, 1951).

The number and type of Nash equilibria is dependent on the structure of the expected reward matrix, and there are many possible such configurations in a 2×2 game (three of which were detailed in the previous paragraph). For the remainder of this chapter we focus our attention to two well-studied repeated games to illustrate the performance of our various strategies. Together, these games characterise conflicts that frequently arise between reward-maximising agents. In later sections, we will see that both games demonstrate differing sub-optimal and non-Nash behaviour with non-explorative strategies and furthermore, the optimal level of exploration is markedly different due to the underlying reward structure. This motivates the inclusion of both case study games. Note that each case study game represents the expected reward for each joint action – the actual reward received is observed with noise. In addition, the rewards in each case study game are set such that random action selection by both agents yields an expected reward of zero.

- *Case Study Game 1: Matching Pennies*

Consider the following zero-sum stage game, known as matching pennies:

		Agent B	
		Action 1	Action 2
Agent A	Action 1	$1, -1$	$-1, 1$
	Action 2	$-1, 1$	$1, -1$

Agent A wishes to match (both agents selecting the same action) and agent B prefers not to match. There is hence no pure strategy Nash equilibrium, instead the Nash equilibrium is a pair of mixed strategies where both agents select each action with probability 0.5.

- *Case Study Game 2: Prisoner's Dilemma*

We also consider the following non-zero sum stage game, known as prisoner's dilemma:

		Agent B	
		Action 1	Action 2
Agent A	Action 1	$1, 1$	$-2, 2$
	Action 2	$2, -2$	$-1, -1$

The Nash equilibrium is for both agents to "defect" and select Action 2 (a pure strategy Nash equilibrium), despite the fact that this joint action pair is not Pareto efficient (both agents could gain a higher reward by “cooperating” through selecting action 1), which is why each agent faces a dilemma. In a repeated game setting, however, there exist several strategies that can outperform playing the Nash strategy for every stage game (Rogers et al., 2007a, and references therein).

We note that there exist several other interesting case games, such as games with two pure and one mixed strategy Nash equilibria (for example, “Battle of the Sexes” (Dixit et al., 2004)) or games with one pure strategy Nash equilibrium that is Pareto efficient (consider switching the rewards in the prisoner’s dilemma for both agents selecting Action 1 and Action 2), which is in effect a bandit problem. The two case study games, however, characterise the range of possible problems faced (at least in the context of the exploration-exploitation trade-off), as exploration is extremely costly in one game and highly beneficial in the other, as we demonstrate from Section 7.3 onwards.

7.2 On-line Learning Strategies

Before investigating our case study games in more detail, we first outline some strategies that the agents can use in a repeated game setting. The agents do not know the reward structure of the stage game *a priori*. The agents therefore have two distinct learning operations: *estimating* the rewards and *adapting* to the opponent’s strategy. These have to be handled concurrently with reward seeking behaviour, otherwise agents will often select sub-optimal actions. There are two distinguishable forms of multi-agent learning that the agents can use to sequentially estimate and adapt (Claus and Boutilier, 1998). *Independent learners* (ILs) would apply learning in the classical sense, ignoring the existence of the other agent. Conversely, *Joint action learners* (JALs) would make decisions based on their own past actions in conjunction with those of the opponent. JALs are more sophisticated in that they use observations of both the reward received and the action selected by the opponent to learn (thus using all the information provided to the agent). Specifically, JALs estimate rewards in the joint action space and can adapt to the opponent by making inferences on its past history of actions. ILs however, estimate rewards in an individual action space and adapt using this information only.

In various numerical simulations performed in Claus and Boutilier (1998) for cooperative games, ILs were found to perform not much differently from JALs with only slightly slower convergence to a Nash equilibrium. Nevertheless, this convergence was guaranteed by using SoftMax exploration (Section 2.2.2) and it is not clear how performance might differ between ILs and JALs against heterogeneous opponents, particularly in finite time. It is for these reasons that we consider both ILs and JALs in our analysis.

7.2.1 Independent Learners (ILs)

After selecting an action, the agents only observe their own reward and which action the opponent selected. ILs however, choose to ignore the opponent and make inferences based on the reward received only. The decision problem is then analogous to a bandit problem and the estimated expected reward of each action ($\widehat{a}(1)$ and $\widehat{a}(2)$ for agent A) can be simply updated at time t using recursive averaging:

$$\widehat{a}(i) \leftarrow \widehat{a}(i) + \frac{1}{n_i^A(t)} \left(r_A(t) - \widehat{a}(i) \right), \tag{7.3}$$

for $i = 1, 2$ when action i is selected, and similarly for agent B. $n_i^A(t)$ is the number of times agent A has selected action i prior to time t. The agents must then use these estimated rewards to select an action to play at the next iteration. In the absence of any exploration, the obvious way to do this is to adopt a greedy strategy and select the action with the higher valued estimate – we refer to this strategy as *bandit greedy*.

Exploration of actions is however required to guarantee convergence to Nash equilibria (Chapman et al., 2011), but can also improve performance in finite time for bandit problems (as demonstrated in Chapters 3-5). We choose to investigate this in our framework using an ϵ-greedy strategy, for reasons discussed later.

7.2.2 Joint Action Learners (JALs)

Contrary to ILs, JALs learn on the joint action space by observing the actions selected by the opponent. In the 2-player, 2-action game studied here, each agent has 4 running estimates of rewards: $\widehat{a}(i,j)$ and $\widehat{b}(i,j)$ for $i,j = 1,2$. These can again be updated by agent A and B respectively at time t, using recursive averaging:

$$\widehat{a}(i,j) \leftarrow \widehat{a}(i,j) + \frac{1}{n_{i,j}(t)} \left(r_A(t) - \widehat{a}(i,j) \right), \tag{7.4}$$

updated when agent A selects action i and B selects j (and similarly for agent B). $n_{i,j}(t)$ is the number of times joint action $\{i,j\}$ has been selected up to time t.

The agents must use these joint action reward estimates, together with the past history of actions, to select the next action. There are several game-theoretic methods with which this can be done, including fictitious play (Brown, 1951), adaptive play (Young, 1993), regret-matching strategies (Marden et al., 2007) and one-shot Nash (Fudenberg and Maskin, 1986). In this research, we consider the agents using fictitious play as this strategy does not require knowledge of the opponent's rewards (required to calculate one-shot Nash and regret-matching strategies) and is also suitable for a game with a static reward process (an adaptive play strategy would be more naturally suited to dynamic rewards). Furthermore, fictitious play (with known rewards) has been shown to converge to a Nash equilibrium for a variety of games including zero-sum games (Brown, 1951), potential games (Monderer and Shapley, 1996), games that are solvable by iterated elimination of dominant strategies (Nachbar, 1990) and more recently all $2 \times N$ games where an appropriate tie-breaking rule is used to separate actions of equal preference (Berger, 2005).

Games that converge to a Nash equilibrium under fictitious play are said to have the Fictitious Play Property. In the case of convergence towards a mixed strategy equilibrium, fictitious play selects actions deterministically (as opposed to stochastically), but converges to selecting each action at the average frequency as determined by the probability weightings of the mixed strategy equilibrium. Both case study games introduced in Section 7.1 have the Fictitious Play Property. We can define the *fictitious play* strategy for 2×2 games as follows. The agents select the *best response* to the empirical frequency of actions selected by the opponent thus far. Suppose that at time t agent B has selected Action 1 for $n_1^B(t)$ past plays (and Action 2 for $t - n_1^B(t) - 1$). Action 1 is a best response to agent A if and only if:

$$n_1^B(t)a(1,1)+(t-n_1^B(t)-1)a(1,2) \geq n_1^B(t)a(2,1)+(t-n_1^B(t)-1)a(2,2), \quad (7.5)$$

and Action 2 is a best response if the inequality is reversed. The agents, however, do not know the true values of the rewards and instead select the *predicted best response,* using the estimated rewards from Equation (7.4) rather than the unknown true values. Without any exploration, there is no guarantee that fictitious play will converge to a Nash equilibrium, due to the fact that rewards are observed with noise (as we prove in Section 7.4). Claus and Boutilier (1998) use SoftMax exploration with fictitious play for cooperative games and Chapman et al. (2011) use ϵ-decreasing exploration with fictitious play for non-cooperative and potential games – though only Chapman et al. (2011) prove that their strategy converges to a Nash equilibrium for specific games with the correct decay rate for ϵ. As with ILs, we will again use ϵ-greedy exploration and define the ϵ-*FP* strategy as follows:

$$\text{with probability} \begin{cases} 1-\epsilon & \text{select action that is the predicted best response} \\ \epsilon & \text{select action that is the predicted worst response} \end{cases}$$

7.2.3 Strategies for 2$\times$2 Games with Unknown Rewards

We have constructed strategies for both ILs and JALs and also explorative and non-explorative agents. Henceforth we refer to these strategies as they are denoted in Table 7.2. For both ILs and JALs we initialise these strategies with both agents selecting each action once (in a random order) as would often be done in a bandit problem (Auer et al., 2002). The JALs initially estimate the reward of unselected joint actions as the average reward of all selected actions thus far, until these joint actions are eventually selected and the estimates are then replaced with the estimates from Equation (7.4). We therefore do not assume each joint action is sampled once, as this requires coordinated initialisation – although all our results and proofs in this chapter could be extended to deal with this and other initialisation procedures.

Table 7.2: Different types of strategies for on-line learning agents

		On-line Learning	
		Joint Action Leaner (JAL)	Individual Leaner (IL)
Exploration	No	Type I: fictitious play	Type II: bandit greedy
	Yes	Type III: ϵ-FP	Type IV: ϵ-greedy

7.3 Non-Explorative Strategies

In this section we consider agents playing with non-explorative strategies (ie. Type I or Type II strategies from Table 7.2). These strategies would perform well in a setting with known rewards, but we are interested in the impact of no explicit exploration when rewards are unknown. We present simulation results for both case study games, where each stage game is repeated 50 times – this length of game is sufficiently long such that agents can learn in a noisy environment but short enough such that fast learners are rewarded. Note that for simplicity we set the observation noise variance to be equal for each agent in all simulations in this chapter.

7.3.1 Case Study Game 1: Matching pennies

- *Type I: fictitious play vs. Type I: fictitious play*

If both agents use fictitious play with full knowledge of the rewards then the empirical frequencies of both agents converge to the mixed strategy Nash equilibrium. With no prior knowledge and no exploration, however, no such convergence is guaranteed. Figure 7.1 shows the average proportion of times that Action 1 is selected by each agent over the course of the game, for the matching pennies game of length 50 (over 100,000 repeats). Each subplot shows results for games with different noise variances (where top left is the lowest variance and bottom right is the highest).

As can be seen, for low noise variances, the action selection frequencies converge towards the Nash equilibrium without any additional exploration. As the magnitude of the noise increases, however, the agents increasingly play pure strategies. This is due to a lack of exploration. For example, agent B may have observed an unusually low reward for Action 2 in an early round of the game and calculates Action 1 to be the dominant strategy. Action 2 is never revisited to correct this error and the agent is subsequently exploited by agent A (who selects Action 1 to match). This pattern can explain all 4 possible combinations of pure strategies being played.

The initial observations are the most crucial, as this is when reward estimates are furthest from their true values, and can therefore have the biggest impact on the long-term convergence of the strategies. Different initialisation procedures will also effect the long-term convergence of a strategy. Specifically, a longer and more explorative initialisation will result in more games converging to Nash equilibria, for the same strategy, than with a shorter initialisation sequence.

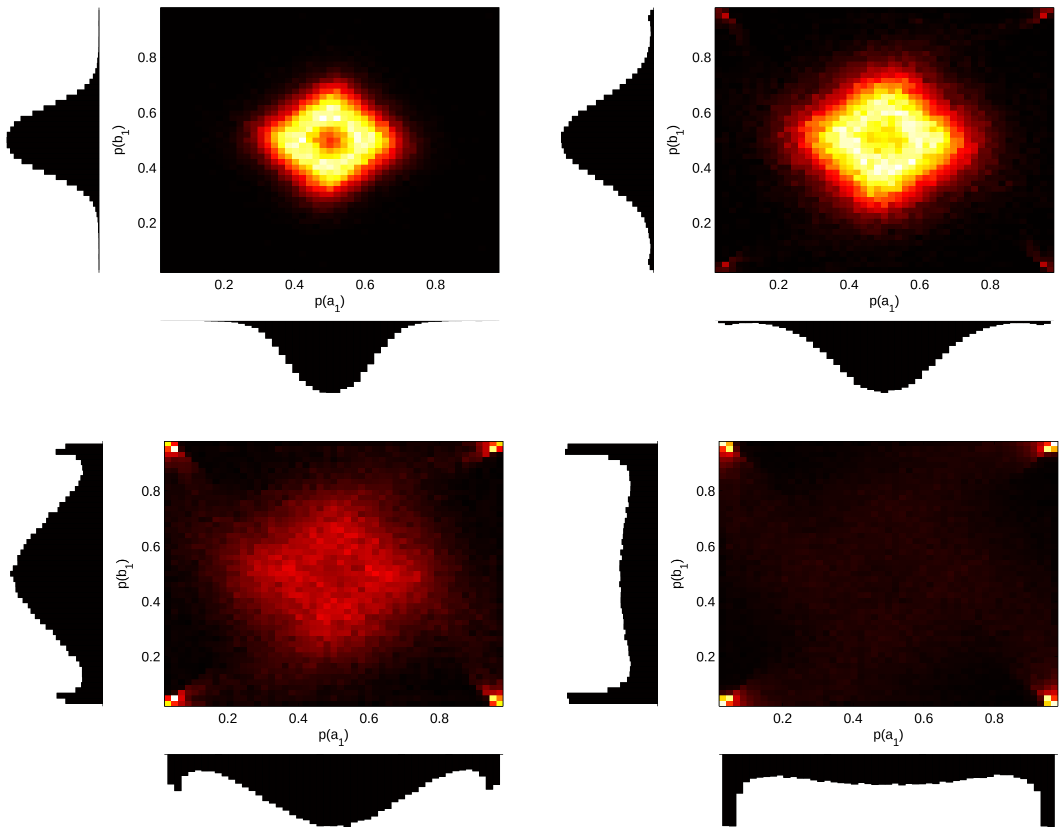


Figure 7.1: Density plots showing the proportion of times Action 1 is selected by agent A ($p(a_1)$) and agent B ($p(b_1)$) over the course of the game in the matching pennies case study game of length 50 over 100,000 repeats for noise variances of 0.25 (top left), 0.5 (top right), 1 (bottom left) and 2 (bottom right). Both agents are JALs using Fictitious Play.

- *Type II: bandit greedy vs. Type II: bandit greedy*

If both agents are ILs using the bandit greedy strategy, then even with full knowledge of rewards, convergence to the Nash equilibrium is not guaranteed. Figure 7.2 (left) displays average rewards from the same setup as Figure 7.1, except both agents are ILs. The noise variance has been deliberately set low and despite this both agents are playing pure strategies – approximately half the games favour agent A and the remainder agent B. The lack of exploration immediately forces one agent to commit to an action first and then the other exploits this choice.

- *Type I: fictitious play vs. Type II: bandit greedy*

In Figure 7.2 (right), agent A is a JAL (using fictitious play) and agent B is an IL. Some games resulted in plays close to the Nash equilibrium but the majority converged to pure strategies where agent A (the JAL) exploits agent B. Learning on the joint action space allows agent A to distinguish between matched and unmatched actions and exploit agent B who plays a bandit problem and often settles on one action. This advocates the use of joint action learning, over independent learning, in the unknown rewards setting.

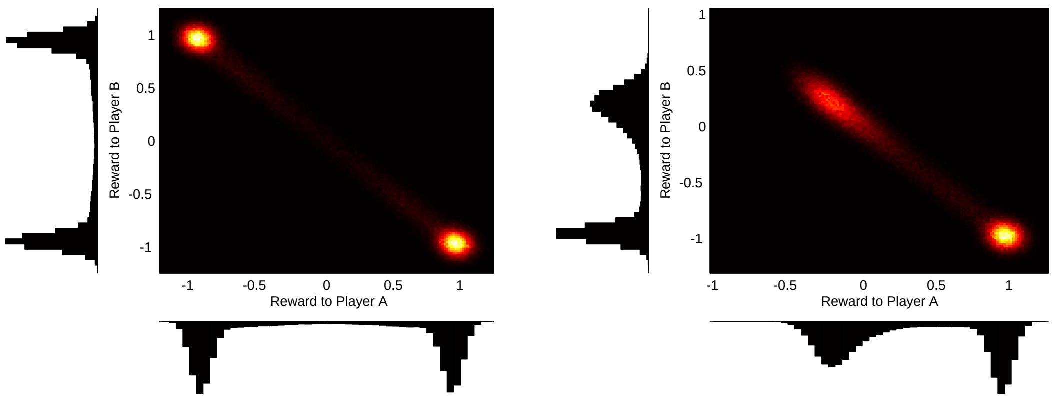


Figure 7.2: Density plots showing the average rewards for agent A and agent B in the matching pennies case study game of length 50 over 100,000 repeats for a noise variance of 0.25, where both agents are ILs (left) and agent A is a JAL and agent B an IL (right).

7.3.2 Case Study Game 2: Prisoner's Dilemma

- *Type I: fictitious play vs. Type I: fictitious play*

For the prisoner's dilemma game, fictitious play would immediately converge to both agents defecting with known rewards, as this is a dominant action and hence is a best response regardless of the frequency of the opponent's actions. Figure 7.3 displays average rewards, in the unknown rewards setting, for 2 fictitious players playing the prisoner's dilemma game of length 50 (for 100,000 repeats), with low noise variance (left) and high variance (right).

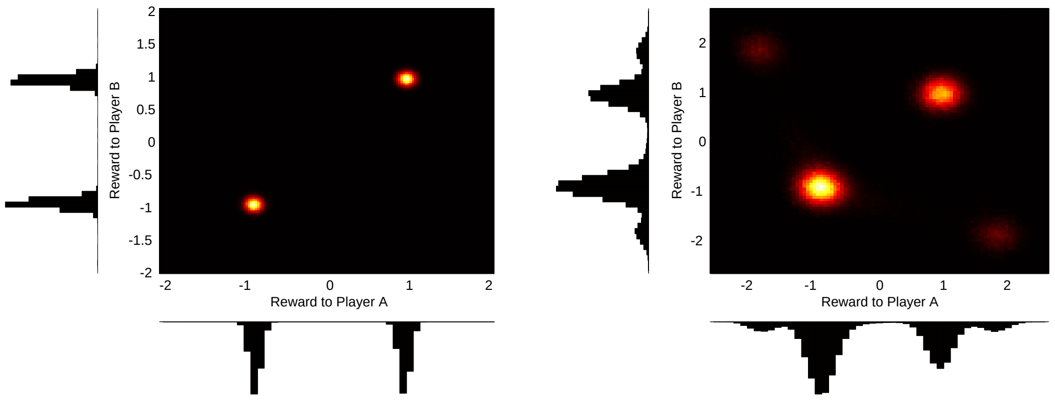


Figure 7.3: Density plots showing the average rewards for agent A and agent B in the prisoner's dilemma case study game of length 50 over 100,000 repeats for noise variances of 0.25 (left) and 2 (right). Both agents are JALs using Fictitious Play.

Notice that although around half of the games have converged to the agents defecting and receiving the Nash equilibrium reward of -1, other games have both agents "cooperating" and selecting the dominated pure strategy. This happens by chance from the noisy reward estimates. When the noise variance is high some games also converge to cooperate/defect. This non-Nash convergence is again due to the lack of exploration of the joint action space, which has resulted in incorrectly calculated best responses, especially when the noise variance is high. This selection of dominated strategies has actually resulted in a higher expected reward to each agent than if the rewards were known, due to the Pareto optimality of cooperating.

In particular, the clustering of rewards around $(1, 1)$ has improved the expected reward from -1 (with known rewards) to -0.01 when the noise variance is low and -0.19 when the noise variance is high.

- *Type II: bandit greedy vs. Type II: bandit greedy*

Figure 7.4 (left) displays the same results for two ILs using bandit greedy (left). For this game setting, the ILs have performed similarly to JALs with almost identical convergence characteristics – although the larger clusters of average reward values suggest that the convergence has been slightly slower (this was also found to be the case in Claus and Boutilier (1998)).

- *Type I: fictitious play vs. Type II: bandit greedy*

In Figure 7.4 (right) we show results for a JAL against an IL (right). On this occasion, the IL has performed as well as the JAL with very similar action selection behaviour. This is because, for this reward structure, the problem is more like a bandit problem – with one action dominating the other. Explicit knowledge of the joint action space is therefore of no particular benefit if the agent is indifferent between the actions of the opponent and trying to calculate a best response.

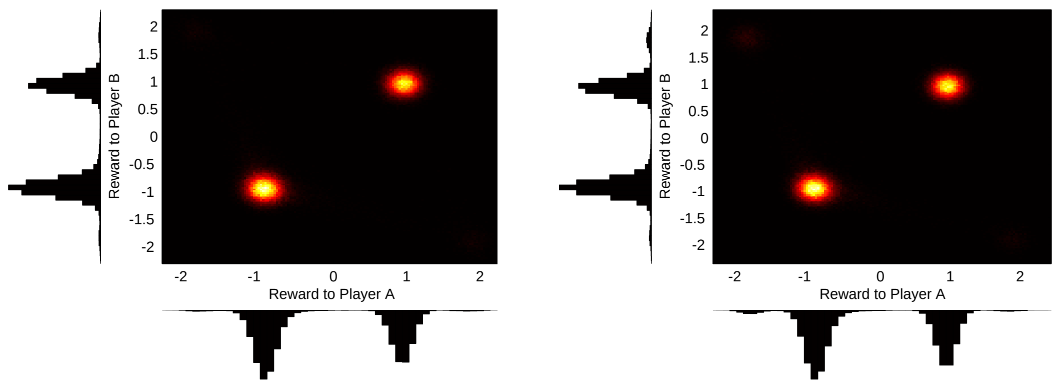


Figure 7.4: Density plots showing the average rewards for agent A (R_A) and agent B (R_B) in the prisoner's dilemma case study game of length 50 over 100,000 repeats for noise variance of 0.25, where both agents are ILs (left) and agent A is a JAL and agent B an IL (right).

7.4 Convergence of Non-Explorative Strategies

2×2 games, with the genericity assumption[1], can have either 0, 1 or 2 pure strategy Nash equilibria, 0 or 1 mixed strategy Nash equilibria and at least 1 Nash equilibrium (of any kind) overall (Dixit et al., 2004). In the previous section we demonstrated that, with unknown rewards, two fictitious players will not necessarily converge to playing a Nash equilibrium strategy in finite time. A number of simulated games resulted in both agents sticking to a non-Nash pure strategy. In fact, we prove the possible convergence of all four pure strategy profiles, at least two of which are non-Nash, in Theorem 7.1.

Theorem 7.1 *In a 2×2 game with a priori unknown rewards (and with the genericity assumption) there is a non-zero probability that the joint strategies of two agents using a fictitious play strategy will converge to any one of the 4 pure strategy profiles – at least two of which are non-Nash.*

Proof Consider, without loss of generality, both agents converging to Action 1. There is a probability greater than 0 that $\widehat{a}(2,1) < a(1,1)$ and $\widehat{b}(1,2) < b(1,1)$ at a certain time-step, as the noise is unbounded. It suffices to prove that there is a probability greater than 0 such that $\widehat{a}(1,1) > \widehat{a}(2,1)$ (and $\widehat{b}(1,1) > \widehat{b}(1,2)$) perpetually. In this formulation, the estimate $\widehat{a}(1,1)$ is an average of i.i.d Gaussian samples, $X_i \sim \mathcal{N}(a(1,1), \sigma_\eta^2)$. Suppose that $\delta = a(1,1) - \widehat{a}(2,1)$ where $\delta > 0$. We are therefore trying to prove that:

$$\prod_{j=1}^{\infty} P\left(\frac{\sum_{i=1}^{j} X_i}{j} > a(1,1) - \delta\right) > 0 \quad \text{for} \quad \delta > 0, \tag{7.6}$$

which after some rearranging is equivalent to proving that if:

$$Y_i \overset{\text{iid}}{\sim} \mathcal{N}(\delta, 1), \quad W_j = \sum_{i=1}^{j} Y_i, \tag{7.7}$$

[1]The genericity assumption (Pruzhansky, 2003) states that an agent has a preferred action for every fixed action of the opponent, specifically $a(1,1) \neq a(2,1)$, $a(1,2) \neq a(2,2)$, $b(1,1) \neq b(1,2)$ and $b(2,1) \neq b(2,2)$.

then,

$$\left(\prod_{j=1}^{\infty} P(W_j > 0)\right) > 0 \quad \text{for} \quad \delta > 0. \tag{7.8}$$

The process W_j is a random walk, or a discretised Brownian motion with positive drift δ. We have from Chang (1999) that the probability of continuous Brownian motion W_t (with positive drift δ) never falling below zero for $1 \leq t \leq \infty$ is $e^{-2\delta^2} > 0$. The probability for the discretised version is therefore bounded below by this. We have hence proved that there is a probability greater than 0 such that $\widehat{a}(2,1) < \widehat{a}(1,1)$ perpetually. The proof for $\widehat{b}(1,2) < \widehat{b}(1,1)$ follows by symmetry. By this logic, there is a probability greater than zero that both agents perpetually choose Action 1 as their best response (see Equation (7.5)) and hence there is a non-zero probability that both agents converge to Action 1. By symmetry, convergence to any 4 pure strategy combinations is possible. ∎

Theorem 7.1 is concerned with asymptotic properties of infinite-length games, which is not of direct relevance in finite-time problems, as studied in this thesis. Nevertheless, Theorem 7.1 helps to explain the finite-time behaviour of non-explorative strategies that we demonstrated in Section 7.3. This non-Nash and potential suboptimal finite-time performance leads us to consider explorative strategies, which we construct and investigate in detail in the following sections.

7.5 Explorative vs. Non-Explorative Strategies

In Sections 7.3 and 7.4 we analysed ILs and JALs with no exploration and proved that convergence to non-Nash pure strategies is possible. Now we consider the impact of introducing explorative actions. As defined in Table 7.2, we consider an ϵ-FP strategy for JALs (Type III) and an ϵ-greedy strategy for ILs (Type IV). First we consider only agent A selecting explorative actions for our two case study games, to see whether a non-explorative agent can be easily exploited to gain a higher reward.

7.5.1 Case Study Game 1: Matching pennies

- *Type III: ϵ-FP vs. Type I: fictitious play*

Figure 7.5 displays simulated results for JALs (i.e. ϵ-FP against fictitious play) for the matching pennies game of length 50 (with a relatively high noise variance of 1). In the left figure, agent A selects the predicted worst response 10% of the time and in the right figure 30%. In both cases agent A has received a higher reward than agent B (there are more matched actions than unmatched actions), despite this being a zero-sum symmetric game. Agent A has exploited agent B for two key reasons:

- The agents have no longer both converged to pure strategies where actions are not matched and agent A receives a low reward. This is the first benefit of exploration to agent A – the agent is no longer exploited by the opponent as the exploration causes agent A to learn that this is not a best response (compare with Figure 7.1 (bottom left) where agent A is occasionally exploited).

- Agent B, in the absence of any exploration, is still sometimes playing a pure strategy and agent A has learnt to exploit this (by matching) for close to $(1-\epsilon)\%$ of plays and gain a high reward. Moreover, notice that agent B selects pure strategies more often when $\epsilon = 0.3$. This feature can be attributed to the fact that agent A is playing a more mixed strategy (due to the added exploration) which gives agent B rewards close to 1 (rather than -1) more often. Consequently, the exploration of agent A prevents agent B from switching action as its predicted best response action is less likely to change.

The second benefit of exploration is of particular interest – exploiting agent B too often, such that it consistently receives a low utility, is more likely to make the agent switch action. Therefore agent A benefits from a high exploration parameter even if it has learnt the expected reward values. This feature can be viewed as agent A explicitly managing its mixed strategy to maintain long term rewards. Higher values of ϵ come at the cost of less frequent exploitation, but has the benefit that the opposing agent is easier to exploit. This exploitation yields a high utility to the explorative agent – far greater than the Nash equilibrium utility of 0.

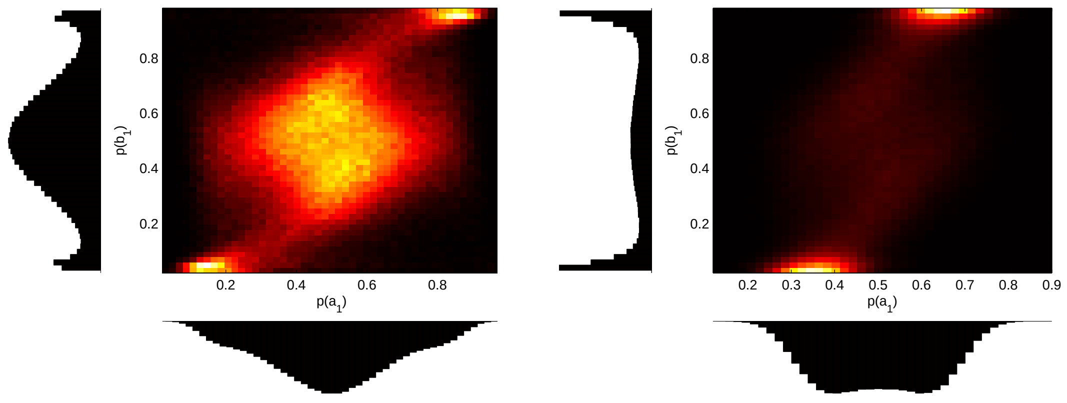


Figure 7.5: Density plots showing the proportion of times Action 1 is selected by agent A ($p(a_1)$) and agent B ($p(b_1)$) in the matching pennies case study game of length 50 over 100,000 repeats for a noise variance of 1, where agent B is a fictitious player and agent A is playing ϵ-FP with ϵ=0.1 (left) and ϵ=0.3 (right).

To explain these results in more detail, the benefit of explicitly managing a mixed strategy can be derived from the theoretical reasoning used in Section 7.4 (where we proved non-Nash convergence of non-explorative strategies). In particular, notice that there is a positive probability that a non-explorative agent never switches action and continuously selects a pure action. In such instances this positive probability for agent B is greater if the expected utility of the pure action is kept, with a higher likelihood, above a required level (which is $a(1,1) - \delta$ in Equation (7.7)) – which can be done by agent A exploring and selecting the suboptimal action. The trade-off, however, is exploring too much such that agent A is penalised and agent B is rewarded (above the Nash expected reward) despite continuously playing this pure action.

Figure 7.6 (left) displays the expected reward to agent A for $\epsilon \in [0, 1]$, for a selection of noise variances, for the game of length 50. Agent A has benefited from exploring by having a positive reward in a symmetric zero-sum game, when $0 < \epsilon < 0.5$. It is easy to see that $\epsilon = 0.5$ will yield an expected reward of 0 for all noise variances as agent A selects each action exactly 50% of the time. For $\epsilon > 0.5$, agent A selects the predicted worst response more often than not and thus allows agent B to take a positive reward. The optimal value of ϵ is larger for small noise variances, which initially appears counter-intuitive. After all, lower noise variance corresponds to an easier learning problem. It must be remembered, however, that the only way agent A can gain a positive reward is to keep agent B on a pure strategy. For low noise variances this is hard to do (see Figure 7.1 (top left)), as the opponent quickly learns the correct best response of a mixed strategy. Agent A therefore has to keep its strategy very mixed in order to keep agent B's predicted response on the pure action. Conversely, large variances make agent B's predicted best responses more erroneous. Agent A can afford to exploit this more often and hence gain a higher expected reward, by playing with a smaller ϵ value.

Figure 7.6 (right) displays the optimal value of ϵ for a range of noise variances, along with the corresponding average expected rewards. The pattern emerges that higher noise variances, correspond to lower optimal ϵ, which in turn correspond to higher potential rewards. Note that the optimal value of ϵ is of course unknown to the agent *a priori* which motives the extension of ϵ-ADAPT to this framework in Section 7.7.

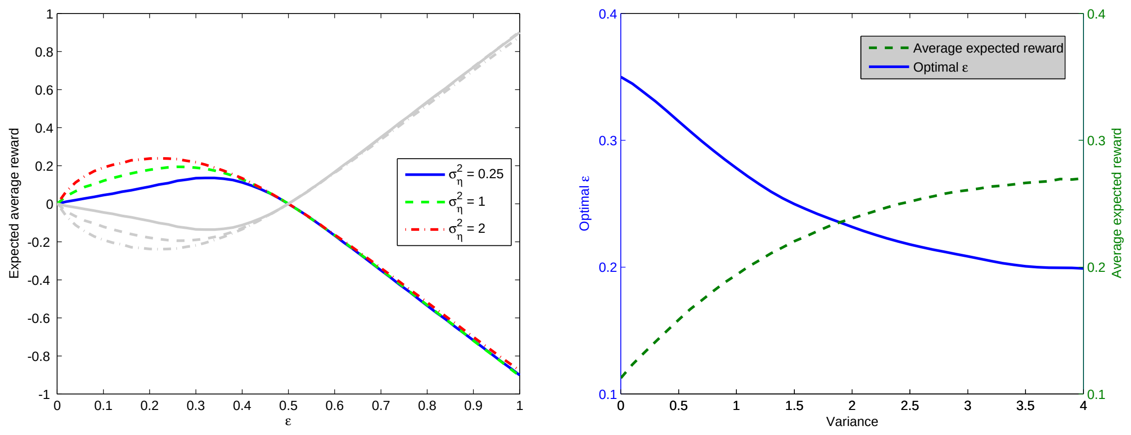


Figure 7.6: $\epsilon/$-FP against fictitious play in the matching pennies case study game of length 50. (left) The Average expected reward to agent A and agent B (shaded) for the range of ϵ values for different noise variances and (right) Optimal ϵ and corresponding average expected reward over a range of noise variances.

- *Type IV: ϵ-greedy vs. Type II: bandit greedy*

Figure 7.7 (left) considers the same matching pennies game except with ILs, where agent A uses ϵ-greedy (with $\epsilon = 0.1$) and agent B uses bandit greedy with no exploration. Agent A has again learnt to not be exploited with unmatched pure strategies and has learnt to often exploit agent B. The introduction of this exploration, however, has now introduced some convergence of the empirical action frequencies to the mixed strategy equilibrium (compare with Figure 7.2 (left), although note that this convergence can only be seen on the histograms on each axis). Figure 7.7 (right) displays the expected reward to agent A for $\epsilon \in [0, 1]$, for a selection of noise variances. The properties of the results are similar to Figure 7.6 (left) for JALs in that the explorative agent can exploit with $0 < \epsilon < 0.5$. For ILs however, the optimal ϵ is smaller and less dependent on the noise variance. This lower value can be attributed to the fact that the greedy strategy learns more slowly than fictitious play and hence agent B can be exploited at a higher frequency without forcing the agent to change action.

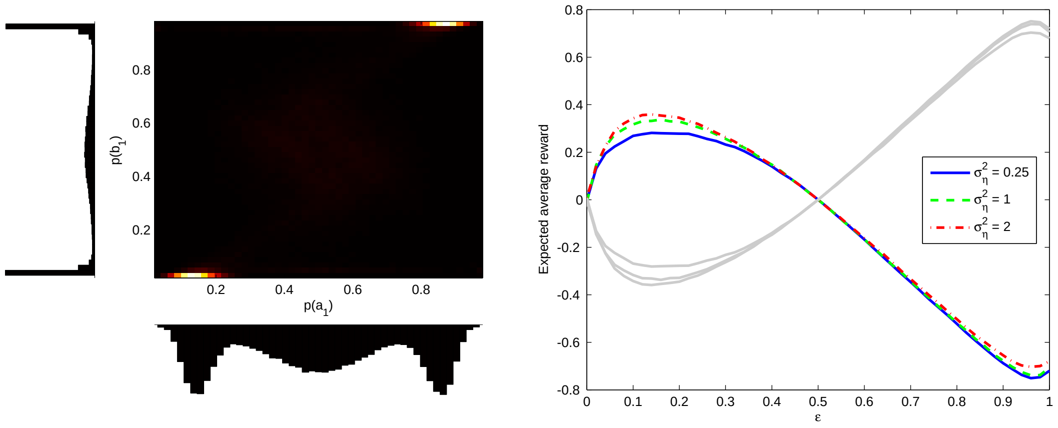


Figure 7.7: (left) Density plot showing the proportion of times Action 1 is selected by agent A ($p(a_1)$) and agent B ($p(b_1)$) in the matching pennies case study game of length 50 over 100,000 repeats for a noise variance of 0.25, where agent B is playing bandit greedy and agent A is playing ϵ-greedy with ϵ=0.1. (right) Average expected reward to agent A and agent B (shaded) for the full range of ϵ values with different noise variances.

- *Type III: ϵ-FP vs. Type II: bandit greedy and Type IV: ϵ-greedy vs. Type I: fictitious play*

Finally, we show results for the same setup except in Figure 7.8 (left) agent A uses ϵ-FP and agent B uses bandit greedy and (right) agent A uses ϵ-greedy and agent B uses fictitious play. ϵ-FP already exploits bandit greedy when $\epsilon = 0$ (see Figure 7.2 (right)) and only benefits from a non-zero ϵ when the noise variance is high – so exploration is not always required to maximise reward. ϵ-greedy, however, is able to recover the deficit when $\epsilon = 0$ and exploit the fictitious player for certain values of $\epsilon < 0.5$. It can be concluded from these results that exploiting a non-explorative fictitious player requires careful management of the agent's mixed strategy (and hence a large ϵ) to prevent the opponent from switching strategies. In contrast, a bandit greedy strategy can be exploited with less exploration, as the opponent here is an IL and is therefore slower to learn that it should switch action. Nevertheless with high noise variance, a small amount of exploration will always benefit an agent, as this will help reduce the high initial estimation error of the reward estimates.

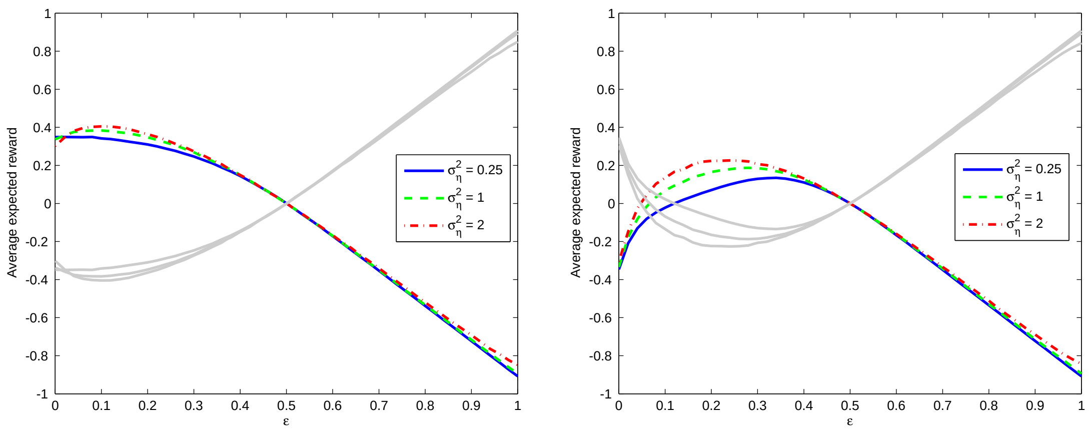


Figure 7.8: Average expected reward to agent A and agent B (shaded), in the matching pennies case study game, for the range of ϵ values for different noise variances where (left) agent A uses ϵ-FP and agent B uses bandit greedy and (right) agent A uses ϵ-greedy and agent B uses FP.

7.5.2 Case Study Game 2: Prisoner's Dilemma

- *Type III: ϵ-FP vs. Type I: fictitious play and Type IV: ϵ-greedy vs. Type II: bandit greedy*

We now briefly return to the prisoner's dilemma example. Figure 7.9 displays results for ϵ-FP against fictitious play and ϵ-greedy against bandit greedy. In both scenarios, agent A has a better reward than agent B, for low values of $\epsilon > 0$. For JALs the reward is maximised with $\epsilon = 0$ and for ILs with $\epsilon \approx 0.1$. These lower optimal ϵ values (compared with matching pennies) can be attributed to the fact that agent B cannot be kept on the dominated strategy (cooperate) by playing a mixed strategy. In addition, the prisoner's dilemma is an unusual type of game where the Nash equilibrium is Pareto dominated – so learning and playing the true best response quickly results in lower rewards.

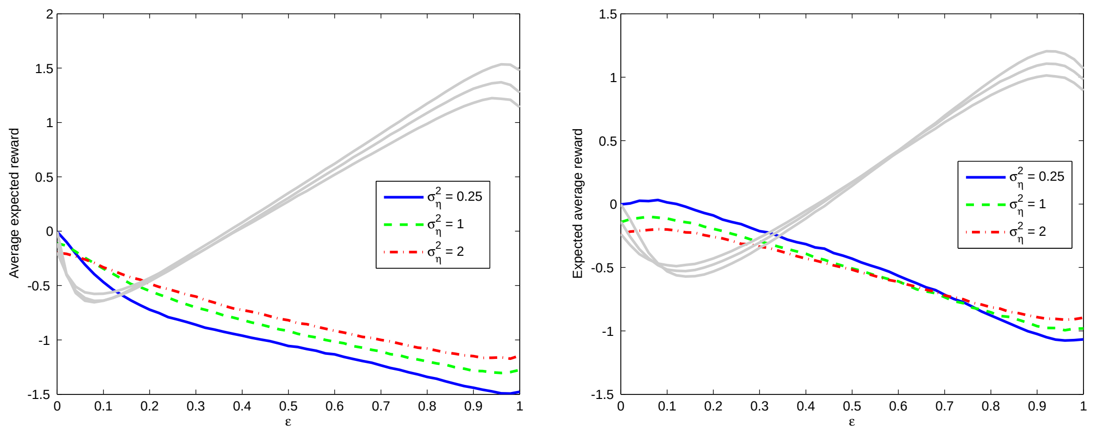


Figure 7.9: Average expected reward to agent A and agent B (shaded), in the prisoner's dilemma case study game, for the range of ϵ values for different noise variances where (left) agent A uses ϵ-FP and agent B uses fictitious play and (right) agent A uses ϵ-greedy and agent B uses bandit greedy.

7.5.3 Summary of Results

We have seen that exploring the action space, by playing the action that is estimated to perform worst, can in fact be beneficial to the agent's reward in finite time. An explorative agent can outperform a non-explorative agent as it learns the rewards from the action space more quickly and can then exploit the opponent, particularly if the opponent is stuck playing non-Nash strategies. The optimal value of ϵ, the exploration parameter, is quite varied depending on the structure of the game and the noise variance. In particular, if the game has a mixed strategy Nash equilibrium, then the agent can *exploit by exploring*, i.e. explicitly manage its mixed strategy, with a high ϵ value, to maintain long term rewards. Nevertheless, even with only pure strategy Nash equilibria, a small amount of exploration can still allow an agent to exploit a non-explorative learner.

7.6 Explorative Strategies

In this section we consider both agents using explorative strategies. As documented in Chapman et al. (2011), a suitable exploration strategy will ensure convergence to a Nash equilibrium in certain games. The exploration parameter has to decay at a suitable rate, such that each action is infinitely explored but also action selections are greedy in the limit, i.e. the probability the correct best response is selected tends to 1 as $t \to \infty$. Such action selection strategies are called *greedy in the limit with infinite exploration (GLIE)* (Singh et al., 2000). Chapman et al. (2011) prove that fictitious play with ϵ-decreasing exploration (ϵ decaying at rate 1/t) will converge to a Nash equilibrium in any game with the fictitious play property. In contrast, Claus and Boutilier (1998) argue (without proof) that both ILs and JALs (using SoftMax exploration) will converge to a Nash equilibrium in any cooperative game.

Our explorative strategies, ϵ-FP and ϵ-greedy, guarantee infinite exploration for $\epsilon > 0$, but are not greedy in the limit as the exploration parameter remains constant. We deliberately keep this parameter constant to maximise reward in finite time – refer to the previous section where we showed that an explorative agent can maximise reward by explicitly managing its mixed strategy profile throughout the game. In addition, decaying ϵ in finite time requires an additional decay parameter or function. Only when both agents explore and the game is sufficiently long does decaying ϵ makes sense, as the joint action space becomes thoroughly explored over time.

Nevertheless, even without a decaying exploration parameter, there is fast convergence towards the Nash equilibrium. See, for example, Figure 7.10 where two JALs (left) both using ϵ-FP (Type III) and two ILs (right) using ϵ-greedy (Type IV) both converge towards the mixed Nash equilibrium in the matching pennies game (Case study game 1). Without exploration (see Figure 7.1 (bottom left) and Figure 7.2 (left)), there exists only occasional convergence to the mixed Nash for JALs and none for ILs. Note that, as expected, the convergence of ILs appears slower than with JALs: there is clear evidence in each corner of the plot, that the agents are learning at a slower rate to move away from pure strategies and towards the mixed strategy Nash equilibrium.

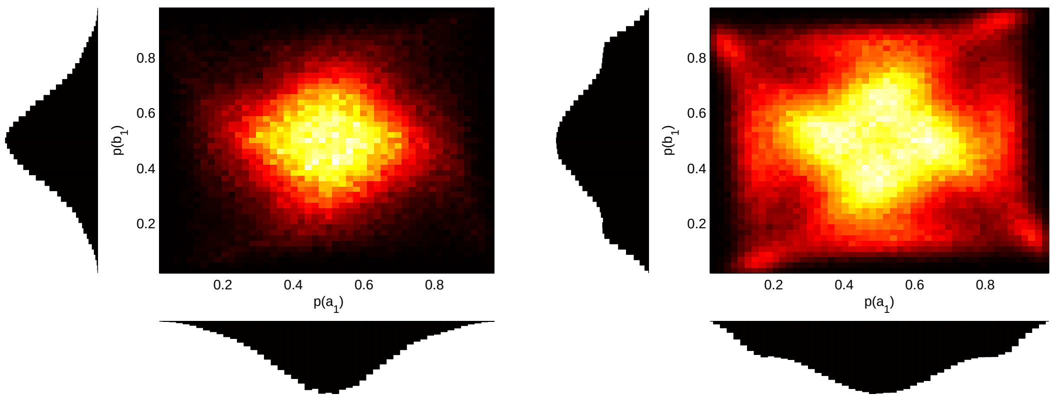


Figure 7.10: Density plots showing the average rewards for agent A and agent B in the matching pennies case study game of length 50 over 100,000 repeats for a noise variance of 1, where both agents are JALs using ϵ-FP (left) and both agents are ILs using ϵ-greedy (right). ϵ=0.1 for all strategies.

The optimal value of ϵ for a certain strategy is dependent on the strategy type and parameter values used by the opposing agent. In Figure 7.11 we display the average cumulative reward to agent A over a grid of ϵ_A and ϵ_B values, where ϵ_A refers to the exploration parameter used by agent A (and similarly for B). Each subplot corresponds to a different pair of strategies or a different case study game. In the matching pennies game, the optimal value of ϵ is high for both ILs and JALs and generally above the ϵ value of the opponent – exploration is not costly in this game where mixed strategies perform best. In contrast, in the prisoner's dilemma game, the optimal value of ϵ is much closer to 0 (but is now higher with ILs) and usually below the ϵ value of the opponent – exploration is costly here as non-dominant action selection can often be exploited by a less explorative agent.

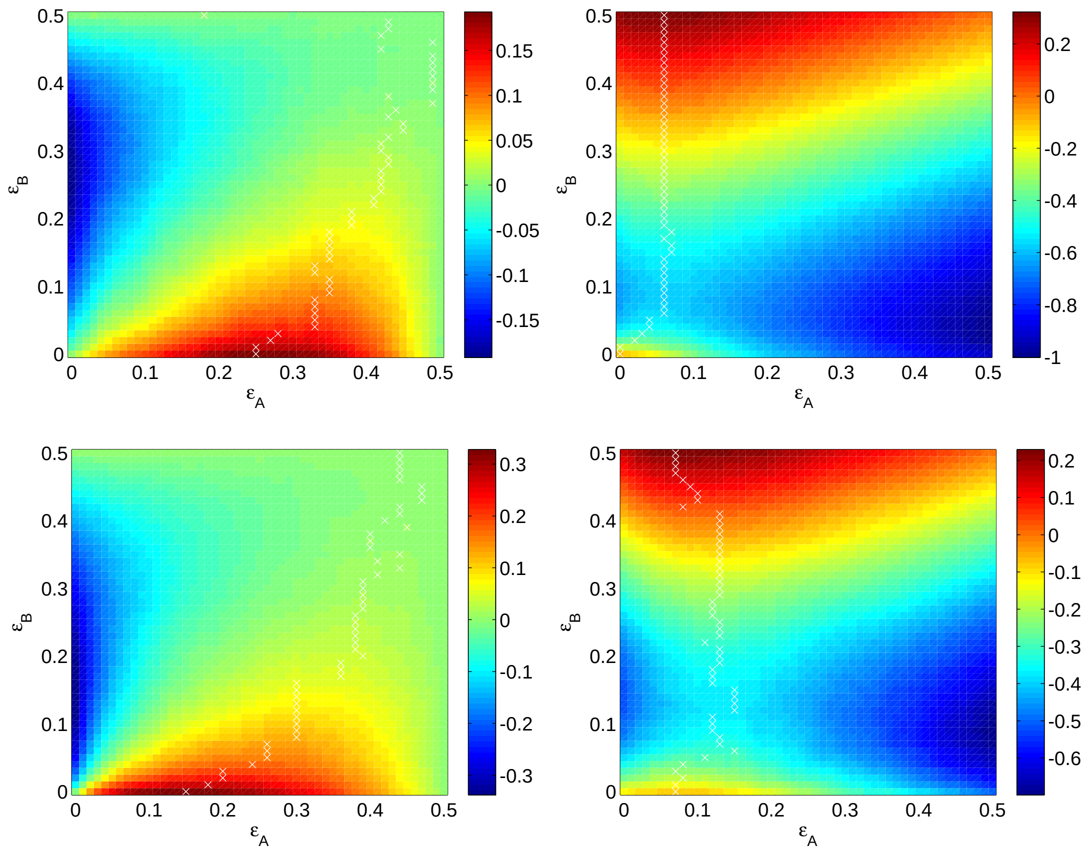


Figure 7.11: Average cumulative rewards (over 10,000 repeats) to agent A over a grid of ϵ_A and ϵ_B values, where (top figures) both players are using ϵ-FP and (bottom) both players are using ϵ-greedy and the case study game is (left figures) matching pennies and (right) prisoner's dilemma. The optimal value of ϵ_A, given a value of ϵ_B, is denoted by a star. In all figures, the game length is 50 and the noise variance is 1.

7.7 ϵ-ADAPT in Repeated Games

In this chapter we have demonstrated that exploration of actions can benefit an agent's total reward in 2×2 repeated games with unknown rewards. The optimal amount of exploration, however, is dependent on several factors other than the amount of observation noise: such as the structure of the game (in terms of types and number of Nash equilibria) and the type of strategy and degree of exploration used by the opponent. In addition, the relationship between the amount of observation noise and the optimal amount of exploration is not necessarily positively correlated, as is typically the case in single-agent bandit problems (see Chapters 3-5). The relationship, in fact, was found to be negatively correlated for the matching pennies game (see Section 7.5) for reasons related to the *exploitability* of the opposing agent. Taking all these features together, the optimal rate of exploration (with an ϵ-greedy strategy) was found to range from anywhere between 0% and 50%.

In this section we propose adapting the degree of exploration on-line using the ϵ-ADAPT approach constructed in earlier chapters. An adaptive on-line approach to exploration is particularly useful in this framework, due to the sensitivity of optimal exploration rates with respect to the type of repeated game and opposing agent strategies encountered. Adapting exploration on-line using an IL approach can be immediately performed using Algorithm 4.3 for multi-armed bandit problems. This is because independent learners ignore the presence of other agents and the problem can be effectively treated as a single-agent bandit problem. We have seen in previous sections, however, that JALs always perform better than ILs in finite-time problems as JALs make use of observing the action selected by the opponent. For this reason, we construct ϵ-ADAPT to learn rewards on the joint action space and explicitly consider the sequence of actions selected by the opposing agent.

The optimal amount of exploration is dependent on several factors such as the game structure and opponent strategy, but these are unknown *a priori*, and moreover, opponent rewards are *not* observed throughout the game. It is therefore extremely challenging to gauge appropriate rates of exploration without knowing whether the opponent will exploit explorative strategies (as with the prisoner's dilemma game) or can be exploited with a mixed strategy (as with matching pennies). We construct ϵ-ADAPT, however, without violating the assumption that opponent rewards are unobserved and instead make use of the sequence of actions selected by the opponent to model its future behaviour.

7.7.1 The ϵ-ADAPT Algorithm

ϵ-ADAPT constructs an index based approximation for each action a_i at time t. In addition to regenerating past and future rewards in the MC approximation, the algorithm also simulates the future behaviour of the opponent, to find the best action to select at time t. Specifically, prior rewards are first regenerated for each joint action $n_{i,j}(t)$ times (the number of times each action has been selected thus far), action a_i is then selected at time t, and then future actions are selected using fictitious play. Future opponent actions are drawn stochastically, where Action 1 is drawn with probability $\tilde{p}_B(t)$ (and Action 2 with probability $1 - \tilde{p}_B(t)$). The resulting ϵ-ADAPT algorithm for 2×2 games with unknown rewards follows Algorithms 7.1 and 7.2.

Algorithm 7.1 ϵ-ADAPT for 2×2 Games with Unknown Rewards

```
1:  $n_{i,j}(0) = 0 \; \forall i,j$ {Initialise Action count}
2:  for $t = 1$ to $T$ do
3:      if $t \leq 2$ then
4:          Select each action once {Initialisation}
5:      else
6:          Generate new rewards $r'_{i,j}(s) \; \forall i,j$ and for $1 \leq s \leq T$ using estimated
            reward coefficients $\hat{a}(i,j)$ and estimated noise variances $\hat{\sigma}^2(i,j)$
7:          for $i = 1$ to $2$ do
8:              Approximate $R_{fp}(T,t,i)$ {Algorithm 7.2}
9:          end for
10:         Select action $i$ ($1 \leq i \leq 2$) that maximises $R_{fp}(T,t,i)$
11:     end if
12:     Observe opponent action $j$ and corresponding reward $r_A(t)$ {Equation (7.1)}
13:     Update $\hat{a}(i,j)$, $\hat{\sigma}^2(i,j)$ and $\tilde{p}_B(t)$
14:     $n_{i,j}(t) = n_{i,j}(t-1) + 1$ ($n_{k,l}(t) = n_{k,l}(t-1)$ for $\{k,l\} \neq \{i,j\}$) {Action
        counts}
15: end for
```

The key differences with the ϵ-ADAPT algorithm for multi-armed bandits (Algorithms 4.3 and 4.4) are denoted in blue. The first of these is the variance estimate of joint action $\{i,j\}$ (Line 13 of Algorithm 7.1). This value cannot be estimated using sample estimates when $n_{i,j}(t) < 2$, but can be instead estimated using the variance of all observed rewards – this is an idea borrowed from Vermorel and

Algorithm 7.2 MC Approximation of $R_{fp}(T, t, i)$ index

```
1:  $\tilde{n}_b(j) = 0$ (for $j = 1, 2$)
2:  for $s = 1$ to $T$ do
3:      if $s < t$ then
4:          Select joint action $\{i, j\}$ such that each action is selected $n_{i,j}(t)$ times
            when $s = t$ and receive reward $r'(s) = r'_{i,j}(s)$
5:      else if $s = t$ then
6:          Sample action $j$ for Agent B (Action 1 with probability $\tilde{p}_B(t)$)
7:          Select action $i$ and receive reward $r'(s) = r'_{i,j}(s)$
8:      else
9:          Sample action $j$ for Agent B (Action 1 with probability $\tilde{p}_B(t)$)
10:         Select action $i$ that maximises $\tilde{a}(i,1)\tilde{n}_b(1) + \tilde{a}(i,2)\tilde{n}_b(2)$ {Fictitious play
            estimate of best response} and receive reward $r'(s) = r'_{i,j}(s)$
11:     end if
12:     Update $\tilde{a}(i, j)$
13:     $\tilde{n}_b(j) = \tilde{n}_b(j) + 1$
14: end for
15: $R_{fp}(T, t, i) = \sum_{s=t}^{T} r'(s)$
```

Mohri (2005) which allows sample estimates to be calculated without full initialisation. This is particularly useful in this multi-agent framework as the agent has only partial control of the joint action space and cannot guarantee seeing a particular joint action without the cooperation of the opposing agent. This method therefore allows ϵ-ADAPT to calculate the noise variance of each joint action separately, which means that rarely selected joint actions have high uncertainty and ϵ-ADAPT is more likely to try and explore these actions.

The other key difference to ϵ-ADAPT is the explicit consideration of future actions that might be selected by the opponent. This is particularly evident in lines 6-10 of Algorithm 7.2, where ϵ-ADAPT is predicting future actions of the opposing agent (lines 6 and 9) and then using fictitious play to select future actions dependent on the opponent's past action choices (line 10). The future actions of the opposing agent are drawn stochastically, where Action 1 is selected with probability $\tilde{p}_B(t)$. $\tilde{p}_B(t)$ could be selected in a number of ways. In the simplest case this value could be set to equal the frequency of times Action 1 has been selected thus far: $\tilde{p}_B(t) = (n_{1,1}(t) + n_{2,1}(t))/(t-1)$, but even if the opposing agent is a fictitious player (or using another stationary strategy), this measure will often not predict future actions of the opponent well. This is because a learning agent can select actions very differently at various stages of the game, as the agent's knowledge of the rewards constantly changes over time. It is therefore more appropriate to use an adaptive measure to predict future opponent actions. Specifically, we weight previous action choices more heavily using a forgetting factor λ, such that:

$$\tilde{p}_B(t) = \frac{I_1(t) + \lambda I_1(t-1) + \lambda^2 I_1(t-2) + \ldots}{\left(\frac{1-\lambda^t}{1-\lambda}\right)}, \tag{7.9}$$

where $I_1(t)$ is an indicator function that is equal to 1 if Action 1 is selected at time t and 0 otherwise. This probability can also be found recursively:

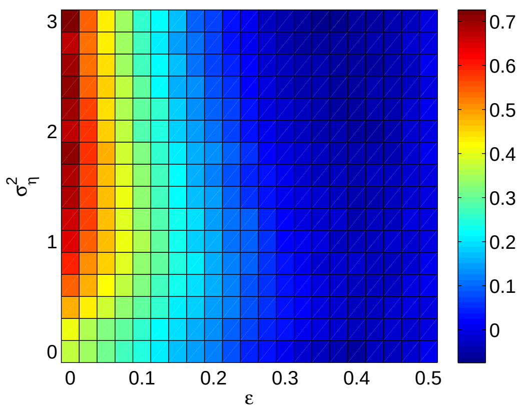
$$\tilde{p}_B(t) = \frac{1}{1-\lambda^t}\left((1-\lambda)I_1(t) + \lambda(1-\lambda^{t-1})\tilde{p}_B(t-1)\right). \tag{7.10}$$

As $t \to \infty$ this recursion approaches $\tilde{p}_B(t) = (1-\lambda)I_1(t) + \lambda\tilde{p}_B(t-1)$, which is identical to the belief update used in geometric fictitious play (Fudenberg and Levine, 1998). The version given in Equation (7.10) is therefore equivalent to geometric fictitious play with a finite-sample adjustment. Overall, this estimate is appropriate for opponent strategies that are both stationary and adaptive – this is because in the unknown rewards setting, agents can change their behaviour throughout the learning process.

We note that λ could be adapted on-line, as performed in Chapter 5 for dynamic bandit problems and in Smyrnakis (2010) for geometric fictitious play, but in our experiments we keep λ constant at 0.8 so that we can directly focus on the issue of adapting exploration on-line. Finally, we note that ϵ-ADAPT could directly replace the fictitious play estimate in the MC approximation (Line 10, Algorithm 7.2) with something more sophisticated, such as stochastic fictitious play (Fudenberg and Levine, 1998) or generalised weakened fictitious play (Leslie and Collins, 2006). In our experiments, however, we test ϵ-ADAPT against an agent using fictitious play (with ϵ-greedy exploration). For this reason we use the fictitious play estimate in the MC approximation, so that we can test for the improvements from adapting exploration on-line, and not the improvements from better strategic action selection.

7.7.2 Numerical Results

In this section we test ϵ-ADAPT against a range of ϵ-FP strategies for both the matching pennies and prisoner's dilemma case study games. Recall from Section 7.6 (and in particular Figure 7.11) that optimal rates of exploration can vary markedly dependent on the case study game, the opponent strategy and the observation noise variance. For this reason, we test ϵ-ADAPT for each case study game over a grid of noise variances and opponent ϵ values. We repeat each experiment 10,000 times and the results are displayed in Figure 7.12. ϵ-ADAPT has gained a better reward than its ϵ-FP counterpart for large regions of the parameter grid. In the other regions the rewards are comparable, with ϵ-ADAPT rarely yielding a reward that is smaller than the opponent. Note that differences are plotted rather than ratios as rewards can be negative.



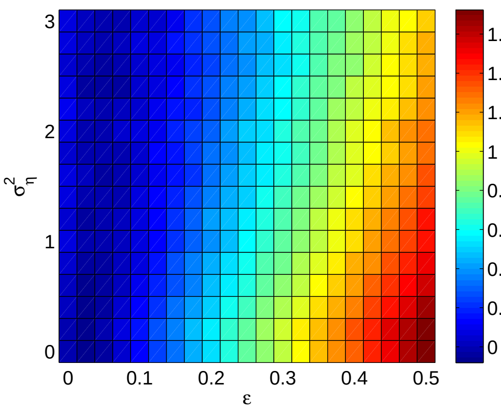


Figure 7.12: Average difference to the cumulative reward (over 10,000 repeats) of ϵ-ADAPT against the ϵ-FP strategy over a grid of ϵ_B values and noise variances, where the case study game is (left) matching pennies and (right) prisoner's dilemma. In both figures the game length is 50.

To investigate this further, in Figure 7.13 we plot the average number of explorative steps performed by ϵ-ADAPT in the same set of experiments for each case study game. An explorative step here refers to time-steps where the ϵ-ADAPT algorithm chooses to select the predicted worst response in the MC approximation. The overall amount of exploration is generally higher in the matching pennies game, as it should be, as exploration is less costly when mixed strategies cannot be exploited as easily. Moreover, for the matching pennies game, the amount of exploration decreases as both the noise increases and the value of ϵ decreases, again as it should (see Figures 7.6 and 7.11). For high noise variance and low values of ϵ, the opponent is more likely to be on a pure strategy and hence easier to exploit. ϵ-ADAPT learns to exploit the opponent more often (and explore less) in such cases. This happens because the value for $\tilde{p}_B(t)$ is likely to be close to 0 or 1 and ϵ-ADAPT learns that selecting the predicted worst response is costly.

For the prisoner's dilemma game, the amount of exploration increases with noise variance, as the presence of a dominant strategy makes this problem more like a bandit problem. The relationship between the amount of exploration and the value of ϵ, however, is dependent on the noise variance and the attributed behaviour of the opponent. Overall, ϵ-ADAPT has learnt to explore the correct amount dependent on the noise variance, the structure of the game and the type of opponent faced. ϵ-ADAPT is therefore able to learn *who* to explore against, in addition to how much, when and which action.

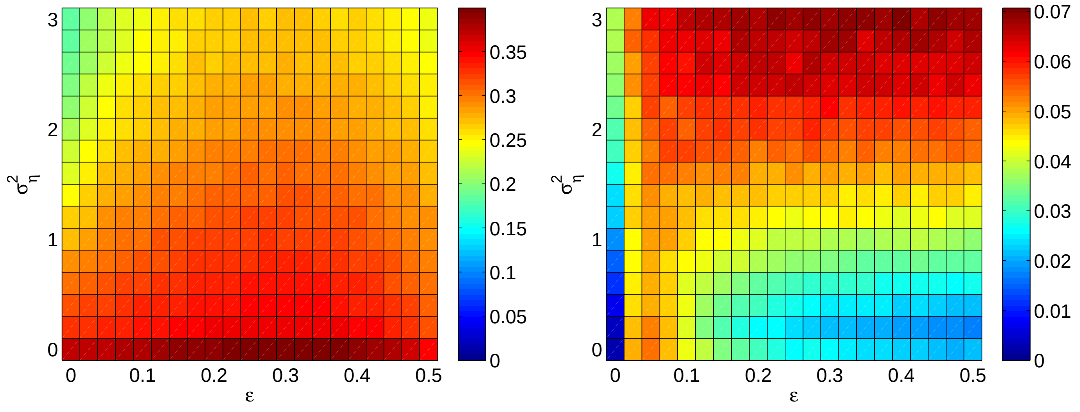


Figure 7.13: Average number of explorative steps performed by ϵ-ADAPT (over 10,000 repeats) against the ϵ-FP strategy over a grid of ϵ_B values and noise variances, where the case study game is (left) matching pennies and (right) prisoner's dilemma. In both figures the game length is 50.

7.8 Summary

In this chapter we have studied 2-player repeated 2×2 games where expected rewards are unknown *a priori*. The agents must learn as they play, and hence must simultaneously estimate rewards and adapt to the opponent. We investigated two fundamental learning techniques: Individual Learning (IL) and Joint Action Learning (JAL). Both ILs and JALs, when used with suitable exploration strategies, have been shown in Claus and Boutilier (1998) and Chapman et al. (2011) to converge to a Nash equilibrium for certain games. In this chapter, however, we demonstrated and proved that ILs (using the greedy strategy) and JALs (using fictitious play) with no exploration, have no such guarantee of converging to a Nash equilibrium in any game – and hence are often suboptimal strategies. We then constructed exploration strategies, based on the ϵ-greedy strategy from bandit problems, and showed that an agent can sometimes use this strategy to exploit a non-explorative opponent. We found surprisingly high optimal values of ϵ, for games with a mixed strategy Nash equilibrium, as the agent could *exploit by exploring* – or in other words explicitly manage its mixed strategy, to keep its opponent on a favourable pure strategy, and hence maintain long term rewards. In other games, however, the optimal exploration rate was close or equal to zero.

The wide-ranging optimal exploration rates motivated the extension of ϵ-ADAPT to this framework, such that exploration could be adapted on-line without the need for an *a priori* fixed exploration parameter. By predicting future opponent action selections on-line, ϵ-ADAPT was able to find a near-optimal exploration rate and perform better than most explorative fictitious play strategies. The ϵ-ADAPT algorithm is therefore applicable in a multi-agent context and is able to consider the presence of other agents in balancing the exploration-exploitation trade-off on-line. This work could be extended by considering unknown classes of opponents (beyond fictitious play) that can be learnt over time by an agent, along with the reward function. This further complicates the exploration-exploitation trade-off however, and we reserve such extensions for future work (see Chapter 8).

Chapter 8

Conclusions

In this chapter, we present an overview of the contributions of this thesis towards understanding the exploration-exploitation trade-off in sequential decision making problems. First, in Section 8.1, we summarise the research undertaken and the main results from each chapter. Then, in Section 8.2, we identify key directions for future work that arise from the findings in this thesis.

8.1 Summary of Results

We have studied the exploration-exploitation trade-off in several important sequential decision making problems, all of which are useful extensions of the classic multi-armed bandit problem. Chapters 3-5 focused on single-agent problems, and Chapters 6 and 7 considered the role of exploration in multi-agent systems. In particular, we have studied scenarios where:

Chapters 3 and 4: The agent observes side information that helps identify the optimal action at each time-step, known as the bandit with covariates problem

Chapter 5: The agent must consider rewards that are changing over time

Chapter 6: Multiple-agents can communicate missing side information

Chapter 7: Agents must consider the actions of other agents to identify the optimal exploration strategy

In each instance, we have studied the role that the exploration-exploitation trade-off plays in terms of maximising reward in finite time, which was the first objective that we outlined at the beginning of this thesis. First of all, we note that all the problems considered are connected with the idea that the agent has additional information other than the observed rewards (such as side information or the actions selected by other agents) which should not be ignored and plays a vital role in terms of the optimal balance between exploration and exploitation. Further to this, the main results and insights we have gained in this direction can be summarised as follows:

- For the simplest bandit with covariates problem, we proved that the optimal exploration rate in finite-time problems is zero for the ϵ-greedy policy, but non-zero for the ϵ-first policy (beyond trivial game lengths).

- For more general static bandit problems, we proved that the optimal on-line ϵ-first policy can be computed on-line by considering the exploration decision in the next time-step *only*. This result is particularly significant as it forms the key reason behind the computational tractability of our new on-line algorithm, ϵ-ADAPT.

- On the other hand, exploration is required throughout a dynamic bandit problem, but the overall amount is dependent on the type and rate of dynamics, i.e. changes such as smooth drifts or abrupt jumps.

- In multi-agent communication problems, agents can benefit from exploring communication decisions, as well as action selection decisions. The optimal balance between the two is dependent on the communication cost.

- In 2-player repeated games with unknown rewards, an agent can *exploit by exploring* in games with mixed strategy Nash equilibria, but exploration is much more costly in games with dominant strategies. We found the optimal amount of exploration to be highly dependent on the reward structure and type of opponent faced.

Our second objective was to develop practical and implementable algorithms for the various frameworks studied. To this end, we have used the results and insights highlighted above to construct an on-line algorithm, ϵ-ADAPT, which can approximate the optimal exploration decision at each time-step for each decision problem. The algorithm is based on theoretical properties of the ϵ-first policy, which has been found to perform consistently strongly (often best of all) in a variety of empirical studies in the literature.

The need for constructing the ϵ-ADAPT algorithm was motivated in Chapter 2, where we conducted an extensive review of existing policies and algorithms used to balance exploration and exploitation in bandit problems. Specifically, the only policy that was found to be flexible enough to be used in all the frameworks considered in this thesis is the ϵ-greedy policy, but this policy requires an exploration parameter to be tuned *a priori*, the optimal value of which is highly variable and unknown to the agent. ϵ-ADAPT, on the other hand, is free of exploration parameters, and extends naturally to a range of sequential decision making problems. This is because with ϵ-ADAPT exploration is driven by *uncertainty* – the more unsure ϵ-ADAPT is about the rewards from an action, the more likely that this action is explored.

In the case of dynamic rewards, for example, ϵ-ADAPT would have high uncertainty about actions that suddenly start yielding different rewards, and this action is then explored more. In this way the algorithm is able to detect *which action* to explore. After implementing and testing ϵ-ADAPT for all frameworks we considered, we demonstrated that the algorithm is also able to learn *how much* to explore, *when* to explore (in the presence of side information), *how* to explore (when exploration by communication is also available) and finally *who* to explore against (in the case of repeated games). We tested ϵ-ADAPT against optimally tuned ϵ-first and ϵ-greedy policies, and found the performance to be consistently close to the better-performing off-line policy for each framework. We have therefore constructed a robust algorithm for adapting exploration on-line in sequential decision making problems which can achieve strong finite-time performance in both single and multi-agent problems.

To summarise, in this thesis we have investigated the fundamental role that the exploration-exploitation trade-off plays in terms of maximising finite-time reward in sequential decision making problems. Not only is it important that the overall amount of exploration performed by an agent is at some required level, but the presence of side information and other agents in a decision problem means exploration needs to be performed at the right times and with the correct actions. To this end, we have developed the first on-line algorithm that can approximate these optimal exploration decisions on-line, without the need for a prefixed exploration parameter.

8.2 Future Work

There remain a number of open issues to be addressed beyond those considered in this thesis. First and foremost, the next step is to construct theoretical performance bounds for the finite-time performance of the ϵ-ADAPT algorithm, to establish the overall robustness of the algorithms to different problem setups. Finite-time bounds have been given little attention in sequential decision making problems in general, with most theoretical analysis restricted to finding optimal asymptotic properties. A notable exception is the class of Upper Confidence Bound (UCB) algorithms (which we reviewed in Section 2.2.3), which bounds finite-time performance for static bandit problems with no side information, where rewards are bounded in the interval $[0, 1]$. Further theoretical evaluations and findings have also been found in restricted settings in Cesa-Bianchi and Lugosi (2006). In our case, we would like to bound the performance of ϵ-ADAPT in more general problems with unbounded rewards where side information is present, as this is more useful in applications. Due to the MC approximations used by ϵ-ADAPT, together with the presence of unbounded rewards and side information, constructing theoretical finite-time bounds is particularly challenging. To combat this challenge, however, we can bound performance (under expectation) against that of the optimal ϵ-first policy. This can be done by calculating the mean and variance of the MC approximations and then using these values to measure the accuracy of selecting optimal ϵ-first actions on-line.

Other useful directions of future work include:

Arm acquiring bandits and infinite-action problems: As discussed in Section 2.1.1, arm acquiring bandits study scenarios where new actions arrive throughout the decision problem. ϵ-ADAPT naturally extends to such scenarios as a new action will be treated with high uncertainty and explored more soon after arrival. It is less clear, however, how a decision problem changes if new agents arrive over time, but again ϵ-ADAPT could be constructed to explore this agent more in such circumstances (whether that is by communication or action selection). The case of infinite actions is interesting, as ϵ-ADAPT does not naturally extend to this approach due to the indexing approach used for each action. A simple but naïve solution might be to discretise the action space to make the problem finite action, but this is not always possible and balancing the exploration-exploitation trade-off in such circumstances is still an open problem.

State-space modelling: Using state-space modelling techniques to predict future rewards in dynamic bandit problems. Although we dismissed using this approach in Chapter 5, as we wanted to consider unpredictable dynamics, it would still be useful to extend ϵ-ADAPT to scenarios where dynamics can be fitted to a certain model, as will often be the case with climatological or financial data for example. In such cases, the model can be used to regenerate future rewards in the MC approximation by ϵ-ADAPT, rather than assuming the decision problem is static in some window, which will lead to more accurate MC approximations. Rewards are not always observed, however, which induces a missing value problem, the solution of which is still very much an open area of research.

Modelling unknown adversaries: In repeated games with unknown rewards an agent could attempt to learn the type of opponent faced, in order to develop a better counter-strategy. This means the agent has an exploration-exploitation trade-off whilst trying to learn about opponents and not just through learning the reward function. This feature could also be studied in games with more than 2 agents, which can introduce strategic interactions such as *tacit collusion* – where agents agree to team up and exploit other agents, without explicitly communicating such preferences.

This has already been investigated in a known rewards setting in Sykulski et al. (2010b) and Munoz de Cote et al. (2010) and applied to a 3-player game known as the *Lemonade Stand Game* – extending these results to the unknown rewards setting can combine this work with the findings in this thesis and further investigate the relationship between game theory and the exploration-exploitation trade-off.

Applications: Finally, ϵ-ADAPT can be further tested with real-world data sets (in addition to that performed in Section 4.3.3 for a data-retrieval problem) from wireless sensor network problems, multi-target tracking assignments and web-based advertising problems for example, where we can implement the dynamic bandit version of ϵ-ADAPT. Furthermore, the multi-agent frameworks that we have studied, can be tested (together with the off-line and on-line policies we constructed) against real-data from mobile sensor problems and on-line auctions.

Appendix A

Derivations and Proofs for One-Armed Bandit Framework

A.1 Derivation of ϵ-greedy Expected Reward

Derivation of Equation (3.11)**:** Consider the case $\alpha > \beta$:

$$\begin{aligned}
r_g(t) = F_{\epsilon g}(t,\epsilon)&\left(\int_{-\infty}^{0}\frac{\alpha x(t)}{\sqrt{2\pi\sigma_x^2}}\mathrm{e}^{\left(-\frac{x(t)^2}{2\sigma_x^2}\right)}\mathrm{d}x(t)+\int_{0}^{\infty}\frac{\beta x(t)}{\sqrt{2\pi\sigma_x^2}}\mathrm{e}^{\left(-\frac{x(t)^2}{2\sigma_x^2}\right)}\mathrm{d}x(t)\right)\\
+(1-F_{\epsilon g}(t,\epsilon))&\left(\int_{-\infty}^{0}\frac{\beta x(t)}{\sqrt{2\pi\sigma_x^2}}\mathrm{e}^{\left(-\frac{x(t)^2}{2\sigma_x^2}\right)}\mathrm{d}x(t)+\int_{0}^{\infty}\frac{\alpha x(t)}{\sqrt{2\pi\sigma_x^2}}\mathrm{e}^{\left(-\frac{x(t)^2}{2\sigma_x^2}\right)}\mathrm{d}x(t)\right)\\
= F_{\epsilon g}(t,\epsilon)\sqrt{\frac{\sigma_x^2}{2\pi}}&\left(-\alpha\int_{0}^{\infty}\frac{x(t)}{\sigma_x^2}\mathrm{e}^{\left(-\frac{x(t)^2}{2\sigma_x^2}\right)}\mathrm{d}x(t)+\beta\int_{0}^{\infty}\frac{x(t)}{\sigma_x^2}\mathrm{e}^{\left(-\frac{x(t)^2}{2\sigma_x^2}\right)}\mathrm{d}x(t)\right)\\
+(1-F_{\epsilon g}(t,\epsilon))\sqrt{\frac{\sigma_x^2}{2\pi}}&\left(-\beta\int_{0}^{\infty}\frac{x(t)}{\sigma_x^2}\mathrm{e}^{\left(-\frac{x(t)^2}{2\sigma_x^2}\right)}\mathrm{d}x(t)+\alpha\int_{0}^{\infty}\frac{x(t)}{\sigma_x^2}\mathrm{e}^{\left(-\frac{x(t)^2}{2\sigma_x^2}\right)}\mathrm{d}x(t)\right)\\
&= |\alpha-\beta|\sqrt{\frac{\sigma_x^2}{2\pi}}(1-2F_{\epsilon g}(t,\epsilon)).
\end{aligned}$$

Note that $\int_0^\infty \frac{x(t)}{\sigma_x^2}\exp\left(-\frac{x(t)^2}{2\sigma_x^2}\right)\mathrm{d}x(t) = 1$, as $\frac{x(t)}{\sigma_x^2}\exp\left(-\frac{x(t)^2}{2\sigma_x^2}\right)$ is the pdf of a Rayleigh distribution (defined on $x(t) \in [0,\infty)$). By symmetry the result holds for $\beta > \alpha$ also.

A.2 Expansion of Binomial Coefficients in Theorem 3.1

Derivation of Equation (3.19)**:** $\sum_{n=1}^{t} F'(n)G(n) =$

$$
\begin{aligned}
&= \sum_{n=1}^{t-1} \left(\frac{1}{2}\right)^{t-2} \binom{t-2}{n-1} \left(\frac{q-n}{q-1}\right) \left(1-(1+\epsilon)^{n-1}(1-\epsilon)^{t-n}\right) F(1) \\
&+ \sum_{n=1}^{t-1} \left(\frac{1}{2}\right)^{t-2} \binom{t-2}{n-1} \left(\frac{n-1}{q-1}\right) \left(1-(1+\epsilon)^{n-1}(1-\epsilon)^{t-n}\right) F(q) \\
&= \sum_{n=1}^{t-1} \left(\frac{1}{2}\right)^{t-2} \binom{t-3}{n-1} \left(1-(1+\epsilon)^{n-1}(1-\epsilon)^{t-n}\right) F(1) \\
&+ \sum_{n=2}^{t-1} \left(\frac{1}{2}\right)^{t-2} \binom{t-3}{n-2} \left(1-(1+\epsilon)^{n-1}(1-\epsilon)^{t-n}\right) F(q) \\
&+ \frac{t-q-1}{q-1} \sum_{n=2}^{t-1} \left(\frac{1}{2}\right)^{t-2} \binom{t-3}{n-2} \left(1-(1+\epsilon)^{n-1}(1-\epsilon)^{t-n}\right) \left(F(q)-F(1)\right) \\
&= \frac{1}{2} \sum_{n=1}^{t-1} \left(\text{bin}\left(n-1, t-3, \frac{1}{2}\right) - (1-\epsilon)^2 \text{bin}\left(n-1, t-3, \frac{1}{2}(1+\epsilon)\right)\right) F(1) \\
&+ \frac{1}{2} \sum_{n=2}^{t-1} \left(\text{bin}\left(n-2, t-3, \frac{1}{2}\right) - (1-\epsilon^2) \text{bin}\left(n-2, t-3, \frac{1}{2}(1+\epsilon)\right)\right) F(q) \\
&+ \frac{1}{2} \frac{t-q-1}{q-1} (F(q)-F(1)) \times \\
&\times \sum_{n=2}^{t-1} \left(\text{bin}\left(n-2, t-3, \frac{1}{2}\right) - (1-\epsilon^2) \text{bin}\left(n-2, t-3, \frac{1}{2}(1+\epsilon)\right)\right) \\
&= \frac{1}{2}(1-(1-\epsilon)^2)F(1) + \frac{1}{2}(1-(1-\epsilon^2))F(q) \\
&+ \frac{1}{2} \frac{t-q-1}{q-1} (1-(1-\epsilon^2))(F(q)-F(1)).
\end{aligned}
$$

Appendix B

The RLS Algorithm with Adaptive Forgetting

Start with initial values $\hat{\boldsymbol{\alpha}}(0) = \mathbf{0}$, $\boldsymbol{P}(0) = \delta \boldsymbol{I}$, $\lambda(0) = 1$, $\boldsymbol{S}(0) = \mathbf{0}$ and $\hat{\boldsymbol{\psi}}(0) = \mathbf{0}$ (where δ is a large positive number and $\boldsymbol{I}$ is the identity matrix). Then for $t = 1, 2, \ldots,$ compute:

$$\boldsymbol{k}(t) = \frac{\lambda^{-1}(t-1)\boldsymbol{P}(t-1)\boldsymbol{x}(t)}{1 + \lambda^{-1}(t-1)\boldsymbol{x}^T(t)\boldsymbol{P}(t-1)\boldsymbol{x}(t)}, \tag{B.1}$$

$$\xi(t) = r(t) - \hat{\boldsymbol{\alpha}}^T(t-1)\boldsymbol{x}(t), \tag{B.2}$$

$$\hat{\boldsymbol{\alpha}}(t) = \hat{\boldsymbol{\alpha}}(t-1) + \boldsymbol{k}(t)\xi(t), \tag{B.3}$$

$$\boldsymbol{P}(t) = \lambda^{-1}(t-1)\boldsymbol{P}(t-1) - \lambda^{-1}(t-1)\boldsymbol{k}(t)\boldsymbol{x}^T(t)\boldsymbol{P}(t-1), \tag{B.4}$$

$$\lambda(t) = \Big[\lambda(t-1) + \omega\hat{\boldsymbol{\psi}}^T(t-1)\boldsymbol{x}(t)\xi(t)\Big]_{\lambda_-}^{\lambda_+}, \tag{B.5}$$

$$\begin{aligned}\boldsymbol{S}(t) =& \lambda^{-1}(t), \left[\boldsymbol{I} - \boldsymbol{k}(t)\boldsymbol{x}^T(t)\right] \boldsymbol{S}(t-1) \left[\boldsymbol{I} - \boldsymbol{x}(t)k^T(t)\right] \\ & \lambda^{-1}(t)\boldsymbol{k}(t)\boldsymbol{k}^T(t) - \lambda^{-1}(t)\boldsymbol{P}(t),\end{aligned} \tag{B.6}$$

$$\hat{\boldsymbol{\psi}}(t) = \left[\boldsymbol{I} - \boldsymbol{k}(t)\boldsymbol{x}^T(t)\right] \hat{\boldsymbol{\psi}}(t-1) + \boldsymbol{S}(t)\boldsymbol{x}(t)\xi(t). \tag{B.7}$$

Computing Equations (B.1-B.4) with $\lambda = 1$ corresponds to the standard RLS algorithm used in Section 4 for static regression problems. Equations (B.5-B.7) are required to adapt the forgetting factor $\lambda(t)$ over time. The bracket followed by λ_- and λ_+ in Equation (B.5) indicates truncation.

Appendix C

Addition Material for the Multi-Agent Bandit Problem

C.1 Derivation of VOC and VOS for 2-armed Problem

From Equation (6.6) it follows that (for $\hat{\alpha}_{i,j+1} > 0$):

$$\begin{aligned}
&\text{VOC}_{a_i} + \Pi \\
&\qquad = \int \max\left(0, \hat{\alpha}_{i,1} + \hat{\alpha}_{i,2}x_2(t) + \hat{\alpha}_{i,3}x_3(t)\right) \mathrm{p}(x_{j+1}(t)|x_{i+1}(t))dx_{j+1}(t), \\
&\qquad = \int_{c_3}^{\infty} \left(\hat{\alpha}_{i,1} + \hat{\alpha}_{i,2}x_2(t) + \hat{\alpha}_{i,3}x_3(t)\right) \mathrm{p}(x_{j+1}(t)|x_{i+1}(t))dx_{j+1}(t), \\
&\qquad = \left(\hat{\alpha}_{i,1} + \hat{\alpha}_{i,i+1}x_{i+1}(t)\right) \int_{c_3}^{\infty} \mathrm{p}(x_{j+1}(t)|x_{i+1}(t))dx_{j+1}(t) \\
&\qquad + \hat{\alpha}_{i,j+1} \int_{c_3}^{\infty} x_{j+1}\mathrm{p}(x_{j+1}(t)|x_{i+1}(t))dx_{j+1}(t),
\end{aligned}$$

where $c_3 = \frac{-\hat{\alpha}_{i,1} - \hat{\alpha}_{i,i+1}x_{i+1}(t)}{\hat{\alpha}_{i,j+1}}$. Now,

$$\mathrm{p}(x_{j+1}(t)|x_{i+1}(t)) \sim \mathcal{N}(c_1, c_2), \qquad \begin{cases} c_1 = \hat{\mu}_j + \frac{\hat{\Sigma}_{i,j}}{\hat{\Sigma}_{i,i}}(x_{i+1}(t) - \hat{\mu}_i) \\ c_2 = \hat{\Sigma}_{j,j} - \frac{\hat{\Sigma}^2_{i,j}}{\hat{\Sigma}_{i,i}} \end{cases}$$

Therefore it follows that:

$$
\begin{aligned}
&\text{VOC}_{a_i} + \Pi \\
&\quad = (\hat{\alpha}_{i,1} + \hat{\alpha}_{i,i+1} x_{i+1}(t) + \hat{\alpha}_{i,j+1} c_1) \int_{c_3}^{\infty} \mathrm{p}(x_{j+1}(t) | x_{i+1}(t)) dx_{j+1}(t) \\
&\quad + \hat{\alpha}_{i,j+1} \int_{c_3}^{\infty} (x_{j+1} - c_1) \mathrm{p}(x_{j+1}(t) | x_{i+1}(t)) dx_{j+1}(t), \\
&\quad = (\hat{\alpha}_{i,1} + \hat{\alpha}_{i,i+1} x_{i+1}(t) + \hat{\alpha}_{i,j+1} c_1) \left[1 - \Phi \left(\frac{c_3 - c_1}{\sqrt{c_2}} \right) \right] \\
&\quad + \hat{\alpha}_{i,j+1} \int_{c_3 - c_1}^{\infty} u \mathrm{p}(u | x_{i+1}(t)) du, \\
&\quad = (\hat{\alpha}_{i,1} + \hat{\alpha}_{i,i+1} x_{i+1}(t) + \hat{\alpha}_{i,j+1} c_1) \left[1 - \Phi \left(\frac{c_3 - c_1}{\sqrt{c_2}} \right) \right] \\
&\quad + \hat{\alpha}_{i,j+1} \int_{c_3 - c_1}^{\infty} u \frac{1}{\sqrt{2\pi c_2}} \exp \left(- \frac{u^2}{2c_2} \right) du, \\
&\quad = (\hat{\alpha}_{i,1} + \hat{\alpha}_{i,i+1} x_{i+1}(t) + \hat{\alpha}_{i,j+1} c_1) \left[1 - \Phi \left(\frac{c_3 - c_1}{\sqrt{c_2}} \right) \right] \\
&\quad + \hat{\alpha}_{i,j+1} \sqrt{\frac{c_2}{2\pi}} \exp \left(- \frac{(c_3 - c_1)^2}{2c_2} \right) du,
\end{aligned}
$$

and similarly for $\hat{\alpha}_{i,j+1} > 0$, to get Equation (6.9). The VOS can be found more immediately from Equation (6.5):

$$
\begin{aligned}
\text{VOS}_{a_i} &= \max \left(0, \mathrm{E}(\hat{\alpha}_{i,1} + \hat{\alpha}_{i,i+1} x_{i+1}(t) + \hat{\alpha}_{i,j+1} x_{j+1}(t) | x_{i+1}(t)) \right), \\
&= \max \left(0, \hat{\alpha}_{i,1} + \hat{\alpha}_{i,i+1} x_{i+1}(t) + \hat{\alpha}_{i,j+1} c_1 \right).
\end{aligned}
$$

Note that the VOC and VOS implemented in all algorithms use on-line estimates of $\boldsymbol{\alpha}$, $\boldsymbol{\mu}$ and Σ, as the true values are unknown.

C.2 MC approximation of Optimal Exploration by Communication

Algorithm C.1 On-line MC approximation of $R_{\epsilon f}(T^*, \boldsymbol{z}_i'(t))$

```
1: Inputs: $T^* = T - t + 1$ , $\boldsymbol{n}$ (no. of times each action selected prior to time $t$),
   sample estimates ($\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\sigma}}_\eta^2, \ldots$),
2: for all $\boldsymbol{z}_i'(t) \notin \boldsymbol{y}_i^C(t) \cap \boldsymbol{y}_i'^C(t)$ do
3:   Denote additional communication cost as $\Pi(\boldsymbol{z}_i'(t))$
4:   for $s = 1$ to $T^* + \max(\boldsymbol{n})$ do
5:     Generate $\boldsymbol{x}'(s)$ {New Covariate}
6:     if $s = \max(\boldsymbol{n}) + 1$ then
7:       for $d = 1$ to $p$ do
8:         if $x_d(t) \in \boldsymbol{z}_i'(t)$ then
9:           $x_d'(s) = x_d(t)$ {True covariate value kept at time $t$, for observed
             covariate values only}
10:        else
11:          Draw $x_d'(s)$ conditional on $\boldsymbol{z}_i'(t)$
12:        end if
13:      end for
14:    end if
15:    for all $c \in C_i$ do
16:      if $\max(\boldsymbol{n}) - n_c \leq s \leq \max(\boldsymbol{n})$ or $E[r_c'(s)|\boldsymbol{x}'(s), \bar{\boldsymbol{\alpha}}_c] > 0$ then
17:        Select action $c$ and receive reward $r_c'(s)$. $c \in S_i'$
18:        Update $\bar{\boldsymbol{\alpha}}_c$ using EM with missing covariates
19:      end if
20:    end for
21:    $r_{a_i}'(s) = \sum_{c \in S_i'(t)} r_c'(s)$
22:  end for
23:  $R_{\epsilon f}(T^*, \boldsymbol{z}_i'(t)) = \sum_{s=\max(n)+1}^{T^*+\max(n)} r_{a_i}'(s) - \Pi(\boldsymbol{z}_i'(t))$ {MC approximation}
24: end for
25: $R_{\epsilon f}(T^*, \boldsymbol{z}_i^*(t)) = \max_{\boldsymbol{z}_i'(t)} R_{\epsilon f}(T^*, \boldsymbol{z}_i'(t))$
```

The benefit of exploration (by communication) occurs in lines 16 and 18, where if the agent observes more covariates then the agent can make better action decisions and update $\bar{\boldsymbol{\alpha}}_c$ with fewer missing covariate values. The cost of exploration occurs in line 23, where increased communication has an increased cost which is removed from the cumulative reward function $R_{\epsilon f}(T^*, \boldsymbol{z}_i'(t))$.

References

777OnlineSlots. Slots faq. http://www.777onlineslots.com/slots_faq.html, 2011.

N. Abe, A.W. Biermann, and P.M. Long. Reinforcement learning with immediate rewards and linear hypotheses. *Algorithmica*, 37(4):263–293, 2003.

M. Abramowitz and I.A. Stegun. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. Dover Publications, 1965.

D. Abreu. On the theory of infinitely repeated games with discounting. *Econometrica: Journal of the Econometric Society*, 56(2):383–396, 1988.

C. Anagnostopoulos. *A statistical framework for streaming data analysis*. PhD thesis, Department of Mathematics, Imperial College London, 2010.

P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(3):397–422, 2003.

P. Auer, N. Cesa-Bianchi, Y. Freund, and R.E. Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 322–331, 1995.

P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.

P. Auer, N. Cesa-Bianchi, Y. Freund, and R.E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2003.

R.J. Aumann and R.B. Myerson. Endogenous formation of links between players and of coalitions: An application of the Shapley value. *The Shapley Value: Essays in Honor of Lloyd S. Shapley*, pages 175–191, 1988.

R. Axelrod. The evolution of strategies in the iterated prisoners dilemma. *Genetic Algorithms and Simulated Annealing*, pages 32–41, 1987.

R. Azoulay-Schwartz, S. Kraus, and J. Wilkenfeld. Exploitation vs. exploration: Choosing a supplier in an environment of incomplete information. *Decision Support Systems*, 38 (1):1–18, 2004.

M. Babes, M. Wunder, and M. Littman. Q-learning in two-player two-action games. *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems*, 2009.

V. Bala and S. Goyal. A noncooperative model of network formation. *Econometrica*, 68 (5):1181–1229, 2000.

J.S. Banks and R.K. Sundaram. Switching costs and the Gittins index. *Econometrica*, 62 (3):687–694, 1994.

R. Bellman. A Markovian decision process. *Journal of Applied Mathematics and Mechanics*, 6:679–684, 1957.

U. Berger. Fictitious play in 2$\times$n games. *Journal of Economic Theory*, 120(2):139–154, 2005.

D.S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27(4): 819–840, 2002.

D.A. Berry. A Bernoulli two-armed bandit. *The Annals of Mathematical Statistics*, 43(3): 871–897, 1972.

D.A. Berry and B. Fristedt. *Bandit problems*. Chapman and Hall London, 1985.

BettingCorp.com. Slots - random fun or pre-programmed teasers? http://www.bettingcorp.com/casino-games-strategies/slots101/, 2009.

A. Beygelzimer, J. Langford, L. Li, L. Reyzin, and R.E. Schapire. Contextual bandit algorithms with supervised learning guarantees. *Proceeding of the 13th International Conference on Artificial Intelligence and Statistics*, 2011.

A. Blum, V. Kumar, A. Rudra, and F. Wu. Online learning in online auctions. *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 202–204, 2003.

R.I. Brafman and M. Tennenholtz. R-max – a general polynomial time algorithm for near-optimal reinforcement learning. *The Journal of Machine Learning Research*, 3(2):213–231, 2003.

G.W. Brown. Iterative solution of games by fictitious play. *Activity Analysis of Production and Allocation*, 13(1):374–376, 1951.

D. Carmel and S. Markovitch. Exploration strategies for model-based learning in multi-agent systems: Exploration strategies. *Autonomous Agents and Multi-agent Systems*, 2 (2):141–172, 1999.

N. Cesa-Bianchi and P. Fischer. Finite-time regret bounds for the multiarmed bandit problem. *Proceedings of the 15th International Conference on Machine Learning*, pages 100–108, 1998.

N. Cesa-Bianchi and G. Lugosi. *Prediction, learning, and games*. Cambridge University Press, 2006.

J. Chang. *Lecture notes on stochastic processes*. Yale University, 1999.

A.C. Chapman, D.S. Leslie, A. Rogers, and N.R. Jennings. Convergent learning algorithms for potential games with unknown noisy rewards. *In submission*, 2011.

H. Chernoff. Sequential models for clinical trials. *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, 4:805–812, 1967.

C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multi-agent systems. *Proceedings of the 15th National Conference on Artificial Intelligence*, pages 746–752, 1998.

M.K. Clayton. Covariate models for Bernoulli bandits. *Sequential Analysis*, 8(4):405–426, 1989.

R. Cont and P. Tankov. *Financial modelling with jump processes*. Chapman & Hall, 2004.

D.R. Cox and N.J.H. Small. Testing multivariate normality. *Biometrika*, 65(2):263–272, 1978.

F. Daly, D.J. Hand, M.C. Jones, A.D. Lunn, and K.J. McConway. *Elements of statistics*. Addison Wesley, 1995.

R.W. Dearden. *Learning and planning in structured worlds*. PhD thesis, University of British Columbia, 2000.

A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

A.K. Dixit, S. Skeath, and J. Repcheck. *Games of strategy*. Norton New York, 2004.

J. Durbin and S.J. Koopman. *Time series analysis by state space methods*. Oxford University Press, 2001.

B. Efron and RJ Tibshirani. An introduction to the bootstrap. 1993.

E. Even-Dar, S. Mannor, and Y. Mansour. PAC bounds for multi-armed bandit and Markov decision processes. *Proceedings of the 15th Annual Conference on Computational Learning Theory*, pages 255–270, 2002.

J. Farrell. Cheap talk, coordination, and entry. *The RAND Journal of Economics*, 18(1): 34–39, 1987.

J. Farrell and M. Rabin. Cheap talk. *The Journal of Economic Perspectives*, 10(3):103–118, 1996.

A. Finzi and T. Lukasiewicz. Relational Markov games. *Logics in Artificial Intelligence*, pages 320–333, 2004.

A.D. Flaxman, A.T. Kalai, and H.B. McMahan. Online convex optimization in the bandit setting: Gradient descent without a gradient. *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 385–394, 2005.

D. Fudenberg and D.K. Levine. *The theory of learning in games*. The MIT Press, 1998.

D. Fudenberg and E. Maskin. The folk theorem in repeated games with discounting or with incomplete information. *Econometrica*, 54(3):533–554, 1986.

Y. Gai and B. Krishnamachari. Decentralized online learning algorithms for opportunistic spectrum access. *Arxiv preprint arXiv:1104.0111*, 2011.

D. Gerardi. Unmediated communication in games with complete and incomplete information. *Journal of Economic Theory*, 114(1):104–131, 2004.

J. Ginebra and M.K. Clayton. Response surface bandits. *Journal of the Royal Statistical Society, Series B*, 57(4):771–784, 1995.

J.C. Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society. Series B*, 41(2):148–177, 1979.

J.C. Gittins and D.M. Jones. A dynamic allocation index for the discounted multiarmed bandit problem. *Biometrika*, 66(3):561–565, 1979.

N.J. Gordon, D.J. Salmond, and A.F.M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE proceedings. Part F. Radar and Signal Processing*, 140 (2):107–113, 1993.

S. Govindan and R. Wilson. A decomposition algorithm for N-player games. *Economic Theory*, 42(1):97–117, 2010.

T. Graepel, J.Q. Candela, T. Borchert, and R. Herbrich. Web-scale bayesian click-through rate prediction for sponsored search advertising in microsofts bing search engine. In *Proceedings of the 27th International Conference on Machine Learning*, pages 13–20. Citeseer, 2010.

V. Haggan and T. Ozaki. Modelling nonlinear random vibrations using an amplitude-dependent autoregressive time series model. *Biometrika*, 68(1):189, 1981.

J. Hardwick, C. Page, and Q.F. Stout. Sequentially deciding between two experiments for estimating a common success probability. *Journal of the American Statistical Association*, 93(444):1502–1511, 1998.

S. Haykin. *Adaptive filter theory*. Prentice Hall, 2002.

A.O. Hero, K. Kastella, D. Castanon, and D. Cochran. *Foundations and applications of sensor management*. Springer, 2006.

J. Hu and M.P. Wellman. Nash Q-learning for general-sum stochastic games. *The Journal of Machine Learning Research*, 4:1039–1069, 2003.

T. Ishikida. *Informational aspects of decentralized resource allocation*. PhD thesis, University of California, Berkeley, 1992.

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

M.O. Jackson, G. Demange, S. Goyal, and A. Van Den Nouwel. A survey of models of network formation: Stability and efficiency. *Group Formation in Economics: Networks, Clubs and Coalitions*, 2003.

J.R. Jensen, K. Rutchey, M.S. Koch, and S. Narumalani. Inland wetland change detection in the Everglades Water Conservation Area 2A using a time series of normalized remotely sensed data. *Photogrammetric Engineering and Remote Sensing*, 61(2):199–210, 1995.

L.P. Kaelbling. *Learning in embedded systems*. MIT Press, 1993.

S.M. Kakade, S. Shalev-Shwartz, and A. Tewari. Efficient bandit algorithms for online multiclass prediction. *Proceedings of the 25th International Conference on Machine Learning*, pages 440–447, 2008.

I. Karatzas. Gittins indices in the dynamic allocation problem for diffusion processes. *The Annals of Probability*, 12(1):173–192, 1984.

L. Kilian and M.P. Taylor. Why is it so difficult to beat the random walk forecast of exchange rates? *Journal of International Economics*, 60(1):85–107, 2003.

R. Kleinberg. Nearly tight bounds for the continuum-armed bandit problem. *Advances in Neural Information Processing Systems*, pages 697–704, 2005.

R. Kleinberg, A. Slivkins, and E. Upfal. Multi-armed bandits in metric spaces. *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pages 681–690, 2008.

L. Kocsis and C. Szepesvári. Bandit based Monte-Carlo planning. *Lecture Notes in Computer Science*, 4212:282–293, 2006.

A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg. Near-optimal sensor placements: Maximizing information while minimizing communication cost. *Proceedings of the 5th International Conference on Information Processing in Sensor Networks*, pages 2–10, 2006.

V. Krishnamurthy and R.J. Evans. Hidden Markov model multiarm bandits: A methodology for beamscheduling in multitarget tracking. *IEEE Transactions on Signal Processing*, 49 (12):2893–2908, 2001.

P. Kumar and T. Seidman. On the optimal solution of the one-armed bandit adaptive control problem. *IEEE Transactions on Automatic Control*, 26(5):1176–1184, 1981.

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

J. Langford and T. Zhang. The epoch-greedy algorithm for contextual multi-armed bandits. *Advances in Neural Information Processing Systems*, pages 1096–1103, 2007.

D.S. Leslie and EJ Collins. Generalised weakened fictitious play. *Games and Economic Behavior*, 56(2):285–298, 2006.

A. Lew. Canonical greedy algorithms and dynamic programming. *Control and Cybernetics*, 35(3):621–643, 2006.

L. Li, W. Chu, J. Langford, and R.E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, pages 661–670. ACM, 2010.

Z. Li and C.H. Zhang. Asymptotically efficient allocation rules for two Bernoulli populations. *Journal of the Royal Statistical Society, Series B*, 54(2):609–616, 1992.

M.L. Littman. Markov games as a framework for multi-agent reinforcement learning. *Proceedings of the 11th International Conference on Machine Learning*, pages 157–163, 1994.

K. Liu and Q. Zhao. Distributed learning in multi-armed bandit with multiple players. *IEEE Transactions on Signal Processing*, 58(11):5667–5681, 2010.

T. Lu, D. Pál, and M. Pál. Contextual multi-armed bandits. *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pages 485–492, 2010.

R.D. Luce. *Individual choice behavior*. Wiley New York, 1959.

W.G. Macready and D.H. Wolpert. Bandit problems and the exploration/exploitation trade-off. *IEEE Transactions on Evolutionary Computation*, 2(1):2–22, 1998.

J.G. March. Exploration and exploitation in organizational learning. *Organization Science*, 2(1):71–87, 1991.

J.R. Marden, G. Arslan, and J.S. Shamma. Regret based dynamics: Convergence in weakly acyclic games. *Proceedings of the 6th International Conference on Autonomous Agents and Multiagent Systems*, pages 1–8, 2007.

J.R. Marden, H.P. Young, G. Arslan, and J.S. Shamma. Payoff based dynamics for multi-player weakly acyclic games. *SIAM Journal on Control and Optimization*, 48(1):373–396, 2009.

N. Meuleau and P. Bourgine. Exploration of multi-state environments: Local measures and back-propagation of uncertainty. *Machine Learning*, 35(2):117–154, 1999.

G.E. Monahan. A survey of partially observable Markov decision processes: Theory, models, and algorithms. *Management Science*, 28(1):1–16, 1982.

D. Monderer and L.S. Shapley. Potential games. *Games and Economic Behavior*, 14(1): 124–143, 1996.

E. Munoz de Cote, A.M. Sykulski, A.C. Chapman, and N.R. Jennings. Automated planning in repeated adversarial games. *Proceedings of the 26th International Conference on Uncertainty and Artificial Intelligence*, pages 376–383, 2010.

J.H. Nachbar. Evolutionary selection dynamics in games: Convergence and limit properties. *International Journal of Game Theory*, 19(1):59–89, 1990.

J. Nash. Non-cooperative games. *The Annals of Mathematics*, 54(2):286–295, 1951.

G. Neumann, M. Pfeiffer, and W. Maass. Efficient continuous-time reinforcement learning with adaptive state graphs. *Proceedings of the 18th European Conference on Machine Learning*, pages 250–261, 2007.

M. Niedzwiecki. *Identification of time-varying processes*. Wiley Chichester, UK, 2000.

J. Nino-Mora. Computing a classic index for finite-horizon bandits. *INFORMS Journal on Computing, Articles in Advance*, pages 1–14, 2010.

D.G. Pandelis and D. Teneketzis. On the optimality of the Gittins index rule for multi-armed bandits with multiple plays. *Mathematical Methods of Operations Research*, 50 (3):449–461, 1999.

S. Pandey, D. Agarwal, D. Chakrabarti, and V. Josifovski. Bandits for taxonomies: A model-based approach. *Proceedings of the SIAM International Conference on Data Mining (SDM)*, pages 216–227, 2007.

N.G. Pavlidis, D.K. Tasoulis, N.M. Adams, and D.J. Hand. Dynamic multi-armed bandit with covariates. *Proceedings of the 18th European Conference on Artificial Intelligence*, pages 777–779, 2008a.

N.G. Pavlidis, D.K. Tasoulis, and D.J. Hand. Simulation studies of multi-armed bandits with covariates. *Proceedings of the 10th International Conference on Computer Modeling and Simulation*, pages 493–498, 2008b.

N.G. Pavlidis, N.M. Adams, D. Nicholson, and D.J. Hand. Prospects for bandit solutions in sensor management. *The Computer Journal*, 53(9):1370–1383, 2010.

V. Pruzhansky. On finding CURB sets in extensive games. *International Journal of Game Theory*, 32(2):205–210, 2003.

S.D. Ramchurn, A. Rogers, K. Macarthur, A. Farinelli, P. Vytelingum, I. Vetsikas, and N.R. Jennings. Agent-based coordination technologies in disaster management. *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems: demo papers*, pages 1651–1652, 2008.

H. Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58:527–35, 1952.

A. Rogers, E. David, and N.R. Jennings. Self-organized routing for wireless microsensor networks. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 35(3):349–359, 2005.

A. Rogers, R.K. Dash, S.D. Ramchurn, P. Vytelingum, and N.R. Jennings. Coordinating team players within a noisy iterated prisoners dilemma tournament. *Theoretical Computer Science*, 377(1-3):243–259, 2007a.

A. Rogers, E. David, N.R. Jennings, and J. Schiff. The effects of proxy bidding and minimum bid increments within eBay auctions. *ACM Transactions on the Web*, 1(2):9, 2007b.

D. Rosenberg, E. Solan, and N. Vieille. Social learning in one-arm bandit problems. *Econometrica*, 75(6):1591–1611, 2007.

M. Rothschild. A two-armed bandit theory of market pricing. *Journal of Economic Theory*, 9(2):185–202, 1974.

J. Sarkar. One-armed bandit problems with covariates. *The Annals of Statistics*, 19(4): 1978–2002, 1991.

A.H. Sayed. *Fundamentals of adaptive filtering*. Wiley-IEEE Press, 2003.

J. Scheffer. Dealing with missing data. *Research Letters in the Information and Mathematical Sciences*, 3(1):153–160, 2002.

G. Shani, R.I. Brafman, and S.E. Shimony. Model-based online learning of POMDPs. *Proceedings of the 16th European Conference on Machine Learning*, pages 353–364, 2005.

L.S. Shapley. Stochastic games. *Proceedings of the National Academy of Sciences*, 39(10): 1095–1100, 1953.

S. Singh, T. Jaakkola, M.L. Littman, and C. Szepesvari. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning*, 38(3):287–308, 2000.

M. Smyrnakis. *Game-theoretical approaches to decentralised optimisation*. PhD thesis, Department of Mathematics, University of Bristol, 2010.

T. Soderstrom, L. Ljung, and I. Gustavsson. A theoretical analysis of recursive identification methods. *Automatica*, 14(3):231–244, 1978.

R. Stranders, A. Farinelli, A. Rogers, and N.R. Jennings. Decentralised coordination of mobile sensors using the max-sum algorithm. *Proceedings of the 21st International Joint conference on Artificial Intelligence*, pages 299–304, 2009.

A.L. Strehl, C. Mesterharm, M.L. Littman, and H. Hirsh. Experience-efficient learning in associative bandit problems. *Proceedings of the 23rd International Conference on Machine Learning*, pages 889–896, 2006.

M.J.A. Strens. A bayesian framework for reinforcement learning. In *Proceedings of the 17th International Conference on Machine Learning*, pages 943–950. Morgan Kaufmann Publishers Inc., 2000.

R.S. Sutton. *Temporal credit assignment in reinforcement learning*. PhD thesis, University of Massachusetts Amherst, 1984.

R.S. Sutton and A.G. Barto. *Reinforcement learning: An introduction*. MIT press, 1998.

A.M. Sykulski, N.M. Adams, and N.R. Jennings. On-line adaptation of exploration in the one-armed bandit with covariates problem. *Proceedings of the 9th International Conference on Machine Learning and Applications*, pages 459–464, 2010a.

A.M. Sykulski, A.C. Chapman, E. Munoz de Cote, and N.R. Jennings. EA^2: The winning strategy for the inaugural lemonade stand game tournament. *Proceedings of the 19th European Conference on Artificial Intelligence*, pages 209–214, 2010b.

M.A.L. Thathachar and P.S. Sastry. A new approach to the design of reinforcement schemes for learning automata. *IEEE Transactions on Systems, Man and Cybernetics*, 15(1):168–175, 1985.

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

J.N. Tsitsiklis. A short proof of the Gittins index theorem. *The Annals of Applied Probability*, 4(1):194–199, 1994.

D. Van Dijk, T. Teräsvirta, and P.H. Franses. Smooth transition autoregressive models - a survey of recent developments. *Econometric Reviews*, 21(1):1–47, 2002.

M.P. Van Oyen, D.G. Pandelis, and D. Teneketzis. Optimality of index policies for stochastic scheduling with switching penalties. *Journal of Applied Probability*, 29(4):957–966, 1992.

J. Vermorel and M. Mohri. Multi-armed bandit algorithms and empirical evaluation. *Proceedings of the 16th European Conference on Machine Learning*, pages 437–448, 2005.

C.C. Wang, S.R. Kulkarni, and H.V. Poor. Arbitrary side observations in bandit problems. *Advances in Applied Mathematics*, 34(4):903–938, 2005a.

C.C. Wang, S.R. Kulkarni, and H.V. Poor. Bandit problems with side observations. *IEEE Transactions on Automatic Control*, 50(3):338–355, 2005b.

T.Y. Wang and Q. Cheng. Collaborative event-region and boundary-region detections in wireless sensor networks. *IEEE Transactions on Signal Processing*, 56(6):2547–2561, 2008.

X.F. Wang and T. Sandholm. Reinforcement learning to play an optimal Nash equilibrium in team Markov games. *Advances in Neural Information Processing Systems*, pages 1603–1610, 2003.

C.J.C.H. Watkins. *Learning from delayed rewards*. PhD thesis, Cambridge University, 1989.

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

M.L. Weitzman. Optimal search for the best alternative. *Econometrica: Journal of the Econometric Society*, 47(3):641–654, 1979.

P. Whittle. Multi-armed bandits and the Gittins index. *Journal of the Royal Statistical Society, Series B*, 42(2):143–149, 1980.

P. Whittle. Arm-acquiring bandits. *The Annals of Probability*, 9(2):284–292, 1981.

P. Whittle. Restless bandits: Activity allocation in a changing world. *Journal of Applied Probability*, 25A:287–298, 1988.

M. Woodroofe. A one-armed bandit problem with a concomitant variable. *Journal of the American Statistical Association*, 74(368):799–806, 1979.

M. Woodroofe. Sequential allocation with covariates. *Sankhyā: The Indian Journal of Statistics, Series A*, 44(3):403–414, 1982.

Y. Yang and D. Zhu. Randomized allocation with nonparametric estimation for a multi-armed bandit problem with covariates. *Annals Of Statistics*, 30(1):100–121, 2002.

H.P. Young. The evolution of conventions. *Econometrica*, 61(1):57–84, 1993.