Spinal Cord Injury Therapy through Active Learning

Thesis by

Thomas Desautels

In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

California Institute of Technology
Pasadena, California

2014
(Defended July 1, 2013)

To Caroline. We made it!

Acknowledgments

There are, of course, many people who deserve to be acknowledged for helping me in my studies and in the completion of this dissertation. Foremost among those who have helped me academically is my advisor, Dr. Joel Burdick, who has been insightful, patient, and wise, providing mentorship in many different ways. I am profoundly indebted to him. Now, if I could only get him to read my emails. I also owe a particular debt to Dr. Andreas Krause, who has offered his considerable expertise on diverse matters in the field of machine learning and who worked closely with me to crack many of the theoretical problems addressed here during a research stay at ETH Zürich.

The experimental work presented here would also not have been possible without the contributions of Dr. Reggie Edgerton of UCLA; he has provided excellent advice and a superb collaborative environment. Among his research group, I would particularly like to recognize Jaehoon Choe and Parag Gad, who carried out the animal experiments described here. Their contributions, both conceptual and in blood, sweat, and wiped-up rat droppings, brought my work to life.

From Caltech, I would also like to thank Yanan Sui, who helped nurse the code through some of the rat experiments, and the rest of the members of the Burdick group, especially Jeff Edlund and Zhao Liu, for many productive discussions. I would also like to extend my thanks to Maria Koeper, the Burdick group's administrative assistant. Without Maria, we would never get anything done. As I move on to new pursuits, I will miss her, her invaluable aid, and her kindness.

Also from Caltech, Dr. Y. C. Tai's input has always been very valuable. His group fabricated the parylene microelectrode arrays; Monty Nandra spent many long hours making these devices. I am also very thankful for the advice and support of Dr. James Beck.

A number of ideas which inspired the work in Chapter 3 are from the work of Dr. Daniel Golovin, with whom Dr. Krause has had a number of collaborations, and to whom I owe a debt of thanks. The theoretical work presented in that chapter was partially supported by SNSF grant 200021_137971, NSF IIS-0953413, and DARPA MSEE FA8650-11-1-7156. My personal support has been derived from NIH/NINDS grant R01NS062009, NIH/NIBIB grant R01 EB007615, CDRF contract ESH1-2012(JB), and the ThinkSwiss Research Scholarship.

Even with all of the aid I received academically, I never would have stayed sane enough to make it through this process without the support of my friends and family. I would particularly like to acknowledge the many contributions of my parents, who set me on the path to learning, drove their sons to the library over and over again, and always tolerated my desire to disassemble everything in sight, even when it placed the house in peril. They have continued to provide advice, love, and support, even now that I am off in the wider world and consequently present less danger to their home and sanity.

No list of those who have supported me would be complete without my wife, Caroline. She has kept me from going off the deep end many a time, and without her ever so subtle urging, I would never have been able to make myself attack this seemingly insurmountable task. Every day, she makes me challenge myself and keep growing. My dear, I love you more than I can say.

Abstract

Therapy employing epidural electrostimulation holds great potential for improving therapy for patients with spinal cord injury (SCI) (Harkema et al., 2011). Further promising results from combined therapies using electrostimulation have also been recently obtained (e.g., van den Brand et al., 2012). The devices being developed to deliver the stimulation are highly flexible, capable of delivering any individual stimulus among a combinatorially large set of stimuli (Gad et al., 2013). While this extreme flexibility is very useful for ensuring that the device can deliver an appropriate stimulus, the challenge of choosing good stimuli is quite substantial, even for expert human experimenters. To develop a fully implantable, autonomous device which can provide useful therapy, it is necessary to design an algorithmic method for choosing the stimulus parameters. Such a method can be used in a clinical setting, by caregivers who are not experts in the neurostimulator's use, and to allow the system to adapt autonomously between visits to the clinic. To create such an algorithm, this dissertation pursues the general class of active learning algorithms that includes Gaussian Process Upper Confidence Bound (GP-UCB, Srinivas et al., 2010), developing the Gaussian Process Batch Upper Confidence Bound (GP-BUCB, Desautels et al., 2012) and Gaussian Process Adaptive Upper Confidence Bound (GP-AUCB) algorithms. This dissertation develops new theoretical bounds for the performance of these and similar algorithms, empirically assesses these algorithms against a number of competitors in simulation, and applies a variant of the GP-BUCB algorithm in closed-loop to control SCI therapy via epidural electrostimulation in four live rats. The algorithm was tasked with maximizing the amplitude of evoked potentials in the rats' left tibialis anterior muscle. These experiments show that the algorithm is capable of directing these experiments sensibly, finding effective stimuli in all four animals. Further, in direct competition with an expert human experimenter, the algorithm produced superior performance in terms of average reward and comparable or superior performance in terms of maximum reward. These results indicate that variants of GP-BUCB may be suitable for autonomously directing SCI therapy.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In a number of problems, including Brain-Computer Interfaces (BCI), deep brain stimulation (DBS), sensory prosthetics, and spinal cord injury (SCI) therapy, complex electronic and computational systems interact with the human central nervous system. An important open question is how to control these engineered systems or agents to produce results which are in some sense optimal, e.g., efficiently decode user intent in BCI or mitigate both tremors and bradykinesia in DBS. This dissertation is concerned with electrical stimulus applied to the spinal cord via multi-electrode arrays. The purpose of this stimulation is to promote the function and rehabilitation of the remaining neural circuitry below the injury, with the goal of enabling the performance of complex motor behaviors, e.g., stepping and standing. To enable the careful tuning of these stimuli for each patient, the electrode arrays which deliver these stimuli have become increasingly more sophisticated, with a corresponding increase in the number of free parameters over which the stimulus must be optimized. Due to the exponential explosion of the sets of possible stimuli, a more rigorous, algorithmic method of selecting stimuli is necessary, particularly in light of the expense and relative inaccessibility of expert hand-tuning. The present work proposes to use a family of recent (GP-UCB, Srinivas et al., 2010) and novel active learning algorithms (GP-BUCB, Desautels et al., 2012, and GP-AUCB) for this purpose. This dissertation develops the GP-BUCB and GP-AUCB algorithms, and bounds their regret (i.e., their sub-optimality over the therapeutic period, as compared with the optimal fixed policy). It compares these novel algorithms with GP-UCB and several competing algorithms in simulation and shows that GP-BUCB and GP-AUCB are competitive with the state of the art. A variant of GP-BUCB was implemented in a closed-loop animal experiment, controlling which epidural stimulating electrodes are used in the spinal cord of an SCI rat; the results obtained are compared with concurrent stimulus tuning carried out by human experimenter. These experiments show that this algorithm is at least as effective as the human experimenter, suggesting that this algorithm can be applied to the more challenging problems of enabling and optimizing complex, sensory-dependent behaviors, such as stepping and standing in SCI patients.

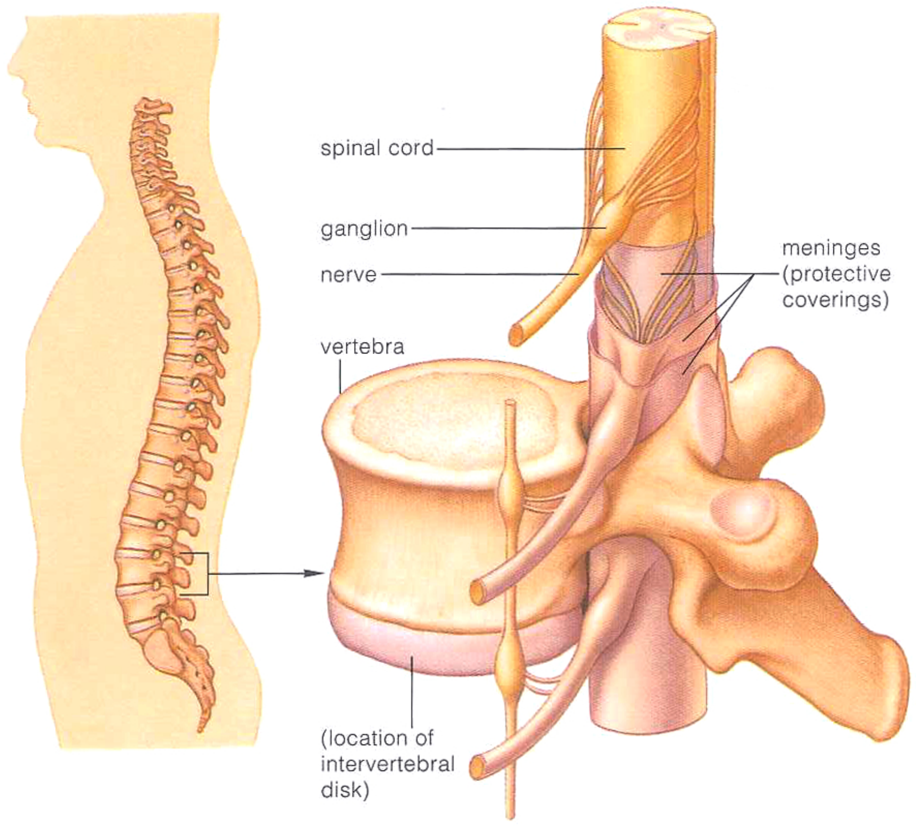

spinal cord
ganglion
nerve
meninges (protective coverings)
vertebra
(location of intervertebral disk)

Figure 1.1: The vertebral column and spinal cord. The vertebral column, shown at left, both provides articulating support for the body and protects the spinal cord. At right, the spinal cord is shown with the associated vertebra, dorsal and spinal roots, dorsal ganglia meninges, and sympathetic nerve ganglia (the chain-like structure parallel to the spinal cord, not labeled). The dura is the outermost of the three meninges; the middle is the arachnoid and the innermost is the pia. Note that the spinal cord is contained within the vertebral canal, such that it is surrounded by fat (not shown) and bone which respectively cushion and protect it. From Starr/Taggart. Biology: The Unity & Diversity of Life w/ CD & InfoTrac, 10E. © 2004 Wadsworth, a part of Cengage Learning, Inc. Reproduced by permission. www.cengage.com/permissions

1.1 Spinal Cord Injury

For the purposes of this dissertation, a *spinal cord injury* (SCI) is defined as a traumatic injury to the spinal cord resulting in a loss of function, especially of the arms or legs. This differentiates SCI from many other disease-related losses in spinal cord or nervous function, such as amyotropic lateral sclerosis (ALS), poliomyelitis, or multiple sclerosis (MS).

Briefly, drawing from the text by Kandel et al. (2012), the spinal cord is both the major connection between the brain and most of the body and a local processing system for many reflexes, as well as more complex behaviors such as locomotion. The spinal cord extends from the brain stem (at the base of the skull) to the first lumbar vertebrae, and has enlargements at the levels which innervate the arms and legs, called the cervical and lumbar enlargements, respectively. Within it, the spinal

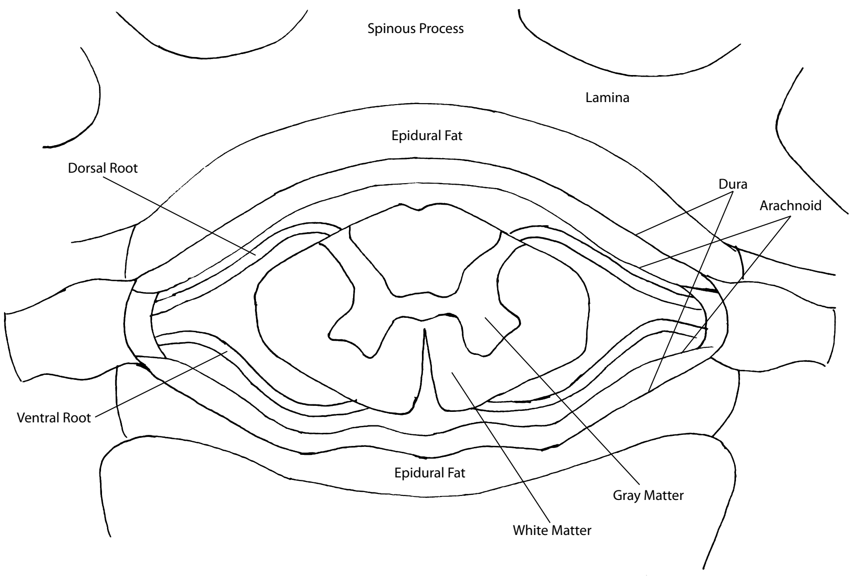


Figure 1.2: Schematic diagram of the the spinal cord in cross-section. The dorsal surface of the body is upward in this figure. The spinal cord is composed of the gray matter and white matter regions in the center of the figure. The epidural electrode array used for EES (see Section 1.2) is implanted in the epidural space, flush with the outside of the dura; this location is approximately where the dorsal of the two labels reading "Epidural Fat" is located. Figure produced with reference to Figure 4.2 of Watson et al. (2008).

cord contains both *gray matter*, composed primarily of the cell bodies of neurons, and *white matter*, composed of tightly packed axons carrying information rostro-caudally; due to extensive myelination of these ascending and descending axons, the white matter does in fact appear lighter in color than the gray matter. When viewed in cross-section, the gray matter is organized in a butteryfly-shape; the dorsal "wings" are referred to as the *dorsal horns*, and the corresponding structures on the ventral side of the cord are called the *ventral horns*. Packed around the periphery of the spinal cord are the well-organized tracts of white matter (see Figure 1.1). The spinal cord, like the brain, is organized into laminae, which are numbered from I (most dorsal) to X (most ventral). Sensory neurons from the body project into the spinal cord through the dorsal roots, and may terminate in a stereotyped fashion in one or more laminae. For the purposes of this dissertation, the most relevant of these projections are those by Aα sensory neurons into or close to the ventral horn, which hosts the motoneurons which control skeletal muscles. Large populations of inhibitory spinal interneurons also reside in the spinal cord and regulate motor activities through interaction with the motoneurons and each other. The motoneurons in turn project out to the body through the ventral roots, which merge with the incoming dorsal roots to form the spinal nerves, which thus carry a mix of afferent and efferent fibers. A cross section of the spinal cord is shown in Figure 1.2. The spinal nerves are designated with the name (e.g., L1) of the vertebra which is below them (cervical) or above them (thoracic, lumbar, and sacral nerves) as they exit the vertebral column, and this name is conferred upon the spinal segments which give rise to the nerve; since spinal nerves remain in the vertebral column for some distance caudal to their origin in the spinal cord, the designation of spinal segements is thus shifted with respect to the designations of the vertebrae. These spinal nerves also innervate well-defined regions of the body called *dermatomes* (for sensation) or *myotomes* (for motor control) (Kandel et al., 2012, pp. 338-340, 357-359, and 488-490). Another thorough treatment of the organization, function, and dysfunction of the spinal cord, can be found in the text and atlas by Watson et al. (2008).

Patients with SCI present different clinical symptoms depending on the location of the injury within the spinal cord, including a variety of syndromes which are symptomatic of damage to different structures within the spinal cord. Sufficiently severe damage to the spinal cord can result in the loss of voluntary control (frequently accompanied by loss of sensation as well) of the legs (paraplegia) or the legs and arms (quadraplegia). The severity of a patient's injury is most commonly assessed on the ASIA (American Spinal Injury Association) scale, as well as by the neurological level of the injury in the spinal cord, diagnosed via the affected dermatomes and myotomes, which correspond in a fixed fashion to spinal levels.

1.2 Epidural Electrostimulation

One technique for SCI therapy is Epidural electrostimulation (EES). EES involves electrically stimulating the spinal cord via an electrode or multi-electrode array placed in the epidural space (see Figure 1.2). Historically, spinal electrostimulation has been applied for a number of purposes, including the the alleviation of chronic pain (Shealy et al., 1967a,b). EES has also been used for the treatment of motor deficits, such as cerebral palsy; this field drove a push toward more complex and capable stimulating devices (for an insider's view of this early history, see Waltz, 1997). Recent leads and electrostimulators (such as the Specify 5-6-5 and RestoreAdvanced, Medtronic, Minneapolis, MN) provide great flexibility in terms of which of many electrodes are active and what stimuli are applied with these electrodes. These increased capabilities allow complex stimuli to be customized post-implantation. A single device can thus accommodate changes in the stimuli as the optimal parameters change with time, as well as variations in surgical placement, injury, and patient-specific needs for symptom alleviation.

Mechanistically, SCI therapy by EES is intended to promote activity, particularly closed-loop activity, of the spinal cord below the site of the injury. This is accomplished by applying a tonic electrical stimulus to activate specific networks and structures in the spinal cord. This stimulus is typically not intended to drive the desired activity directly; rather, stimuli enable the patient's native neural circuits to regulate motor activity according to the sensory environment of the patient, such that the responses, e.g., muscle contractions, are appropriate to the environment and behavior

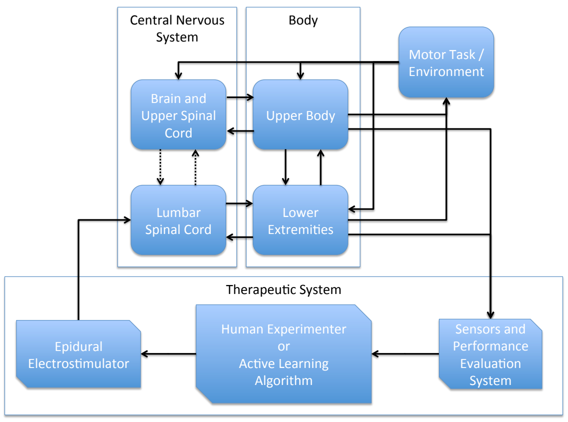

Central Nervous System
Body
Motor Task / Environment
Brain and Upper Spinal Cord
Upper Body
Lumbar Spinal Cord
Lower Extremities
Therapeutic System
Epidural Electrostimulator
Human Experimenter or Active Learning Algorithm
Sensors and Performance Evaluation System

Figure 1.3: A schematic view of SCI therapy through EES. Arrows show the flow of information or action through the composite system. The deficit due to SCI is primarily in terms of disrupted communication between the upper and lower central nervous system (dashed arrows). The intervention employing EES is a modulation of the activity of the lower spinal cord in order to produce better performance in a desired behavior.

of the patient, e.g., weight shifts during standing. Such circuitry does in fact remain intact, if quiescent, below the site of the injury; an autonomously rhythm-generating structure known as a central pattern generator (CPG) is known to exist in a variety of species, including rats and cats, and is thought to exist in humans (Gerasimenko et al., 2008). These neural circuits drive and control complex motor behaviors such as stepping, even in the absence of input from the brain. EES has been applied to stimulate these networks of neurons, enabling stepping and standing after SCI (Harkema et al., 2011; van den Brand et al., 2012). From a control-theoretic perspective, the EES system is not intended to be the controller to the body's plant or process; rather, the EES system modifies the activity of the intrinsic spinal controller or (partially) replaces the absent supraspinal control signal. This scheme is shown in Figure 1.3. In order to make the EES system a higher-level controller for the spinal cord and lower body system, it is necessary to measure the performance of the spinal cord and body (the inner-loop controller) and use these experimental measurements to make decisions about how to change the EES parameters. As the number of electrodes and free parameters increases, however, it is necessary to develop more advanced methods of selecting the stimuli delivered by EES arrays. The motivating problem of this dissertation is optimizing the stimulus patterns for the complex arrays available now and in the near future (e.g., the Medtronic RestoreAdvanced/Specify 5-6-5 system and the parylene-based microelectrodes of Gad et al., 2013, pictured in Figure 4.1).

1.3 Active Learning

Active learning techniques are much as the name indicates; they are algorithms for actively, rather than passively, attempting to learn about a system. In the traditional formulation, the active learner is interacting with an *oracle*, a system which accepts experimental interrogations, each single one of which is called a *query* and returns observations which correspond to the queries. In this fashion, the learner gradually acquires information about expected the response to any query. The element which makes this interaction active, rather than passive, is that the active learner has choices, most typically the choice of which query to submit to the oracle at each opportunity. The active learner makes these choices on the basis of a model of the oracle, constructed based upon the data. Using the choice of which queries to submit, rather than waiting passively for whatever data happens to arrive, the active learner is thus able to acquire the desired information (or simply more information) about the oracle in fewer observations than a passive learner. The text by Settles (2012) gives a thorough treatment of a variety of active learning techniques.

Active learning can be targeted to particular pieces of information about the oracle. One important example is the problem of Bayesian optimization (BO). An interesting approach to this problem is taken by Hennig and Schuler (2012), who also make interesting points regarding the appropriate algorithm design philosophy for this setting. In this case, the active learner is given a finite (but possibly unknown) budget of queries and is tasked with spending these queries to find the action within the available set which yields the maximum value of the *reward*, a measure of the desirability of the oracle's output. After the learning process is complete, the (apparent) optimal action will be chosen and the algorithm will receive this reward. In order to be effective (i.e., probabilistically obtain a high reward), an algorithm must observe the reward values which would be associated with the queries it has so far submitted and then choose future queries which are likely to decrease its uncertainty about where the optimum lies. Choosing queries on-line, rather than *a priori*, allows the algorithm to target these queries to the regions of the set of possible actions which appear promising on the basis of the data being acquired.

An important and closely related problem is the exploitation-exploration tradeoff. If the algorithm receives reward for each and every single query, rather than having distinct search phase in which no rewards are obtained, followed by an exploit phase (as in the BO setting), it is important to choose these queries not simply to learn about the best rewards which may be obtained, but also to obtain high reward at this very moment. This is particularly appropriate in the SCI therapeutic setting, in which each EES stimulus and each interval of training time is valuable and should be spent intelligently. Algorithms for solving this problem have traditionally be explored under the framework of so-called bandits (Robbins, 1952). These algorithms make sequential decisions by trading off exploitation of actions known to yield high reward action with exploration of novel or poorly-understood actions. If these competing imperatives are properly balanced, it can sometimes be demonstrated that the algorithm will converge to the optimal action (i.e., the rate of sub-optimal actions approaches zero) with high probability in the limit of infinite time. Recent work in this field has brought bandits and Bayesian optimization together, yielding algorithms which seek to explore and exploit over very large decision sets, using models of the response function (e.g., the GP-UCB algorithm of Srinivas et al., 2010, which uses Gaussian processes to model the reward funciton). This dissertation continues this line of work, developing the novel algorithms GP-BUCB and GP-AUCB.

1.4 Objective Statement: Major Problem

In order to make a practical, fully-implantable system to deliver epidural electrostimulation which is highly effective for SCI therapy, it is necessary to create, implement, and test a class of active learning algorithms which can:

- Exploit the structure of the epidural spinal stimulation problem, i.e., the anatomical and neurophysiological knowledge of the spinal cord and the lower limbs, as well as the capabilities and construction of the stimulating device, to learn the responses of the patient's spinal cord and muscle activity to epidural electrostimulation;
- Use such a model to choose queries or experimental actions in a way which enables the response function(s) to be learned in a query-efficient manner, due to the expense of individual queries and the combinatorially vast extent of the search space (e.g., 10^7 or more possibilities);
- Also use such a model to perform effective therapy for the patient, as measured by metrics of success available on a per-trial basis; and
- Choose actions and accept observations in batches, or with substantial delay between action initiation and the receipt of the experimental results.

The first property is necessary because, in order for the learning process to be both efficient and tractable, and thus to provide an effective therapy, prior information must be combined with measurements taken for this particular individual. This prior information, largely invariant, structural, and qualitative in nature, is the result of many years of neurophysiological studies and clinical experience and represents a tremendous resource for exploitation by an automated agent. Since it is desirable for this agent to accomplish the same (very difficult) tasks as would normally be performed by experienced experimentalists and clinicians, incorporating this prior information is a crucial first step. The "budget" of experiments, constrained principally by the time required to perform the desired measurements, but also the monetary expense of doing so, will often be several orders of magnitude smaller than the number of potential stimuli; thus, stimuli which will likely be ineffective must be rapidly eliminated from consideration, such that experimental effort is concentrated on stimuli which are more likely to be therapeutically useful. This motivates the second requirement. The third property is required by the fact that the calibration sessions in which the algorithm is run will also constitute a substantial part of the patient's therapy, and indeed, are arguably the most therapeutically useful sessions available due to the (very expensive) presence of highly trained clinicians and therapists. Optimally, all stimuli ever administered (including those delivered by the stimulator as the patient undertakes the tasks of daily living) should be evaluated in terms of their functional performance, such that an algorithm which takes full advantage of this opportunity for experimentation and learning (i.e., is always on) may be preferable. If the algorithm operates continuously, it must treat the therapeutic effectiveness of the stimuli delivered as a substantial component of its decision-making if an effective therapy is to be applied. Further, poor stimuli (i.e., those which produce low reward values, indicative of poor therapeutic performance) may produce high fatigue or confound the results of later experiments. Hence, poor stimulus choices destroy much of the utility of the experimental or therapeutic training session. The fourth property is important, and provides the motivation for much of the theory developed in this dissertation, because it allows much greater flexibility in applications; the requirement of algorithms like GP-UCB that all observations be available before the next action can be selected, and thus that only one action can be pending at any time can prove to be a substantial encumbrance. In the SCI therapy setting, the data processed into the performance metric used by the algorithm are often complex and time-consuming to calculate, resulting in substantial delays between the performance of an experiment and the availability of the assessed performance on that experiment. Motion capture, for example, may take extensive hand annotation to analyze fully, and multi-channel EMG may take several minutes to process into a useful form. However, it is most efficient to assemble an experimental session which consists of an unbroken sequence of requested stimuli; this necessitates either a batch procedure or delayed selection of stimuli.

Further, it is highly desirable that an active learning system have the following additional properties:

- It has rigorous guarantees of behavior, at least under some conditions;
- It is sufficiently modular to enable adaptation to different experimental conditions, e.g. by the revision of structural assumptions, the inclusion or exclusion of stimuli within the decision set, and possibly even modification of the decision rule;
- The predictions made by and the assumptions encoded within the algorithm are human-interpretable; and
- The computations comprising the modeling and action selection steps of the algorithm should

be as efficient as possible, with an eye toward deployment on systems with limited computational power, i.e., miniaturized fully implantable devices.

These secondary specifications also describe important capabilities. Guarantees of performance are an important requirement, as the algorithm's practical performance may be easier to understand in light of these guarantees. Modularity is highly desirable because various components can be interchanged to suit the particular problem at hand. From a practical perspective, modularity is also useful because it enables the re-use of computer code between similar experiments, as well as potentially allowing rigorous comparisons of different modules, e.g., model selection on the Gaussian process kernel functions. The third desire, human-interpretable predictions, is important for both contributing to the body of clinical and neuroscientific literature on the spinal cord, as well as verification of these models by clinical observation and experience. Finally, computational efficiency is important, as the long-range goal of a fully-implantable, autonomous device which administers a dynamic, data-driven therapy requires algorithms which can be run with extremely limited computational resources.

1.5 Contributions

Motivated by the above considerations, this thesis:

- Introduces the GP-BUCB and GP-AUCB algorithms (see Chapter 3), which enable batch or parallel selection of stimuli for epidural electrostimulation, as well as for other general problems;
- Derives theoretical bounds on the regret of a general class of algorithms including the GP-BUCB and GP-AUCB algorithms and also shows that with an easily implemented initialization with a set of finite size, the regret of the GP-BUCB algorithm can be split into two additive terms, such that the time-scaling of the regret is independent of the batch size (also in Chapter 3);
- Successfully tests a derivative of GP-BUCB in a closed-loop SCI therapy therapy setting in a rat SCI model, seeking to maximize a measure of spinal cord interneuronal activity (Chapter 4); and
- Suggests a number of crucial extensions to these algorithms for human studies, as well as further animal experimentation (Chapters 5 and 6).

These contributions demonstrate substantial satisfaction of the major components of the problem specification described in Section 1.4; in particular, the novel algorithms presented here are structured to flexibly incorporate expert knowledge about the structure of the response function over the set of possible stimuli, and were able to elicit a desired motor behavior from four complete

SCI rats in a fashion which, under the chosen reward metric, was at least as effective as the parallel performance of an expert human experimenter. As compared to the existing GP-UCB algorithm (Srinivas et al., 2010), GP-BUCB and GP-AUCB can select batches of experiments, or use knowledge of pending experimental observations to aid in the selection of future experiments. Further, these algorithms make predictions which are readily human-interpretable, are highly modular, and can be computed in closed form, thus requiring (comparatively) little computation.

1.6 Organization

A review of background material relevant to this dissertation, including SCI therapy, Gaussian processes (the major modeling tool used throughout the work), kernel functions (the heart of problem-specific Gaussian process models), and active learning algorithms, follows in Chapter 2. The theoretical properties of GP-BUCB and GP-AUCB are examined in Chapter 3; these results are presented with a series of computational experiments comparing these algorithms with several others. Chapter 4 presents the primary application study of this work, a series of experiments in complete SCI rats. Chapter 5 describes work done toward future experiments in humans, including pilot experiments, along with some suggestions for further improvements. Chapter 6 makes some final conclusions with regard to the present work and also discusses potential extensions of this dissertation's results.

Chapter 2

Background

This chapter reviews background needed to understand this work. Section 2.1 briefly summarizes some important facts about spinal cord injury (SCI), the injury which motivates this work. A discussion of therapeutic approaches for spinal cord injured patients follows in Section 2.2. Section 2.3 reviews the literature on active learning and bandit algorithms, machine learning techniques with the goal of either learning about or optimizing the performance of an unknown system. Gaussian processes (GPs), which will be used to model the responses of an injured spinal cord to electrostimulation, are introduced in Section 2.4. Covariance functions, which are at the heart of specifying a meaningful and problem-tailored GP model, are described in Section 2.5.

2.1 Spinal Cord Injury

Spinal cord injury (SCI) is a medical condition with broad impact. A 2009 survey of 33,000 households estimated that 0.40% of the population of the United States is living with spinal cord injury of some sort (Christopher and Dana Reeve Foundation, 2009). In general, these injuries tend to affect a population which is in the prime of life; the same survey found that over 50% of respondents who indicated they were living with an SCI were under 50 years old and more than 75% of were under 60 years old.

Spinal cord injury has been widely studied in both human patients (e.g., Harkema et al., 2011) and animal models (e.g., van den Brand et al., 2012). Several good reviews exist, including those by Thuret et al. (2006), Edgerton et al. (2006), and Gerasimenko et al. (2008), as well chapter 14 of the text by Watson et al. (2008). Damage resulting from the injury includes the primary damage, e.g., crushing and hemorrhage, directly caused by the insult itself, and secondary damage, including cell destruction and degeneration, which results from the remaining tissue's responses to the primary damage (Watson et al., 2008, p. 209). A field of considerable interest is the acute mitigation of this secondary damage (Zhang et al., 2013).

The primary, long-term result of this damage is a substantial loss of function in terms of motor control and sensation, resulting in impaired mobility and independence. While the symptoms of some patients improve over the first one to one and one-half years after injury, these improvements eventually cease (see Fawcett et al., 2007). The remaining deficits in sensory and motor function are at present generally considered to be largely irreversible, i.e., there is no cure for SCI, though a number of approaches have been developed (e.g., locomotor therapy, see Wernig and Müller, 1992) which have produced gains for some patients. Interestingly, in incomplete injuries and within the general realm of motor control, the degree of recovery in the performance of individual motor tasks may be somewhat independent; this may be due to different levels of supraspinal control exercised in different motor functions, e.g., locomotion versus reaching (Courtine et al., 2005). Beyond loss of motor control and bodily sensation, a number of other problems commonly arise for SCI patients, particularly issues resulting from lack of exercise and from the disruption of the nervous system's internal communications. These deficits can include muscle atrophy and spasticity, as well as potentially life-threatening autonomic problems such as failures of temperature regulation and autonomic dysreflexia. For a discussion of the many and varied autonomic deficits which result from SCI, the symposium proceedings edited by Weaver and Polosa (2006) are an excellent resource. Advances in care have meant that SCI patients who do not die immediately tend to survive for a many years (see John F. Ditunno and Formal, 1994, which contrasts early and late 20th century prognoses for SCI patients), such that therapies which partially alleviate some of their symptoms are highly desirable. Anderson (2004) surveyed 681 SCI patients and found that, among the seven options presented on the survey instrument, a near-majority of quadriplegics believed that recovery of hand and arm function would produce the greatest improvement in their quality of life, while a plurality of paraplegics believed that recovery of sexual function would most greatly improve the quality of theirs. A very substantial number of both populations ranked the item comprised of bladder, bowel, and autonomic dysreflexia as one of their top two potential greatest gains in quality of life. Among paraplegics, walking movement, described by the survey's creator as inclusive of standing and other forms of exercise, also ranked highly, but its share of first or second votes was much lower among quadraplegics. Even given the limitations of the survey, it is striking that bladder, bowel, and autonomic dysreflexia concerns were so important, particularly as compared with walking and mobility.

Current and experimental therapies have begun to deliver this desirable alleviation of secondary symptoms; for example, the initial patient in the epidural electrostimulation studies conducted by our collaborators (Harkema et al., 2011), whose therapy program includes epidural electrostimulation, locomotor training, and stand training, reports that his gains have included improved mental wellbeing, improved bladder and bowel function, some improvement in sensory function, some improvement in sexual function, substantial gains in muscle mass, including gains in the legs, core, and upper body, and better postural control. Note that this patient had already participated in an intensive locomotor training program, and that these gains are relative to his condition after that program. Additionally, this patient has recovered some gross voluntary motor control of his lower limbs; it has been suggested that this control is a result of use-dependent plasticity of spared supraspinal axonal projections (Courtine et al., 2011).

2.2 Existing Therapeutic Approaches

Since no cure currently exists for SCI, current practice focuses on therapy, applied in a variety of approaches. Some techniques attempt to directly create the pattern of muscle activation in the extremities which would ordinarily be associated with the desired activity. Other approaches focus on the spinal cord; Bradbury and McMahon (2006) describe these as attempting either to induce regeneration of damaged axons, repairing the damage to some extent, or to rehabilitate the spinal cord's ability to control the body without addressing the injury itself. A review of a number of approaches is presented in the following sub-sections, including epidural electrostimulation, the approach which is the particular focus of this dissertation.

2.2.1 Functional Electrical Stimulation

Functional Electrical Stimulation (FES, Liberson et al., 1961) attempts to treat the symptoms of paralysis via direct stimulation of the muscles themselves; in FES, electrical stimulators are placed on or within the skeletal muscles and are then activated in a pattern engineered to replicate a desired activity, e.g., the stride cycle. The pattern of muscle activation is directly controlled, such that FES can be used to treat foot drop in hemiplegic patients (Liberson et al., 1961), to aid sit-to-stand transition in paraplegic patients (Kralj and Grobelnik, 1973), or even to produce weight-bearing locomotion in paraplegics (Klose et al., 1997). Applied as an exercise therapy, FES has been shown to confer gains in a variety of cardiorespiratory and metabolic metrics (Davis et al., 2008). However, since FES is an open-loop control method, the stimulation pattern must be carefully designed and/or user-controlled if complex behaviors are desired (e.g., in hand control, as examined by Mangold et al., 2004). Further, the resulting muscle contractions do not respond directly to sensory feedback, an important consideration when considering activities which require significant feedback control, e.g., standing. Another important problem with FES is rapid fatigue; muscle contraction force typically decreases rapidly under FES (see Thrasher et al., 2005, for a discussion of fatigue and the difficulties in mitigating it).

2.2.2 Regenerative Therapies

Another major approach to SCI rehabilitation has been to attempt to induce the spinal cord to repair itself via the introduction of signaling molecules and/or the suppression of endogenous signaling

(Karimi-Abdolrezaee et al., 2012), or to introduce cells, exogenous or autologous, into the injury site which would promote or support regrowth (Coumans et al., 2001; Wu et al., 2012). Often, scaffolds are constructed from biomaterials and used to support regrowth by providing a stable and permissive environment (Pêgo et al., 2012). Regenerative therapies seem promising in the long-term, but have not to date met with substantial success in patients with complete SCI (Thuret et al., 2006). However, there is evidence that, in the case of some incomplete injuries, and even without therapeutic aid beyond exercise, the central nervous system can reroute connections through existing neurons which bypass the injury site to make some small functional gains (Bareyre et al., 2004; Courtine et al., 2008). If these gains can be further improved, they may provide the basis for substantial recovery in the future.

2.2.3 Cord-Rehabilitative Approaches

In contrast to FES and regenerative therapies, cord-rehabilitative approaches do not seek to directly drive the muscles or repair the spinal cord; rather, these approaches take a middle road and attempt to modify the function of the spinal cord in order to produce the desired motor behavior. This avenue of SCI therapy attempts to take advantage of the surviving spinal cord circuitry below the site of injury, which remains viable and adaptable (Edgerton et al., 2006). Specific targets include interneuron networks responsible for reflexes and the central pattern generator (CPG), the region of the spinal cord responsible for generating the overall pattern of muscle activation in walking (see, e.g., Dimitrijevic et al., 1998).

Methods of this type may attempt to use any of a variety of approaches to promote lower spinal cord activity. Pharmaceutical replacement of or substitution for neurotransmitters which would normally be delivered from the higher CNS has been shown to produce substantial gains in stepping performance (Antri et al., 2002). If made practicable by an incomplete motor injury or some other therapeutic approach, physical training is also very useful for recovering function, as it provides the task-appropriate input to which the spinal cord is being trained to respond appropriately (Wernig and Müller, 1992; Engesser-Cesar et al., 2007), which may induce plastic reorganization of the lower spinal cord. In order to reduce the need for human assistance of the patient during activity-based therapy, a number of efforts have concentrated on robotic locomotor training. Cai et al. (2006) considered how the controller which drives a robotic assistance system can affect the therapeutic outcome, showing that an assist-as-needed paradigm which enforced some interlimb coordination outperformed both rote training of the nominal trajectory and an assist-as-needed controller which did not enforce interlimb coordination. Emken et al. (2008) addressed a similar question of robotic gait training and appropriate control algorithms in humans. The phenomenon of learned helplessness, i.e., non-responsiveness to stimuli which cannot be avoided, is present in the rat spinal cord and can be manipulated by both pharmacology and linkage of lower limb position with noxious stimulus (Crown and Grau, 2001). The results of Cai et al. (2006) may provide evidence that variability in the training paradigm is important for avoiding this outcome.

2.2.3.1 Epidural Electrostimulation

Another important cord-therapeutic technique for SCI therapy, and the focus of the applied portions of this dissertation, is epidural electrostimulation. While originally developed for chronic pain therapy (Shealy et al., 1967a,b), spinal electrostimulation can produce complex motor patterns (Dimitrijevic et al., 1998). A variety of methods for delivering the electrical stimulus have been suggested, including penetrating microelectrodes (the major topic of a review by Prochazka et al., 2001). In the animal studies described in Chapter 4, electrostimulation is applied using a non-penetrating electrode array implanted in the epidural space, between the outer membrane of the spinal cord, the dura, and the interior walls of the vertebrae (see Figure 1.2). Epidural spinal cord stimulation has been applied with similar results in spinal and decerebrate cats, spinalized rats, and humans, and when properly configured, can produce walking motions (described in the review by Gerasimenko et al., 2008). Herman et al. (2002) combined epidural electrostimulation with partial body weight support exercise training to produce substantial perceived, functional, and metabolic gains in locomotion in an incomplete quadraplegic patient. More recently, Harkema et al. (2011) have used epidural electrostimulation with training and demonstrated substantial gains in a motor-complete patient in a similar setting. This type of stimulation is believed to activate afferent fibers as they enter the spinal cord through the dorsal nerve roots (Minassian et al., 2007).

2.2.4 Combined Approaches

It is often the case that the therapeutic approaches outlined above can be combined for improved effects. For example, Courtine et al. (2009) examine serotonin agonists and electrostimulation for excitation of the spinal cord in treadmill walking and van den Brand et al. (2012) use electrostimulation, pharmacology, and a compliant robotic assist device, both in SCI rats. Both works show impressive functional gains, with the latter (in animals with two staggered lesions, producing a clinically complete injury, but sparing some bridiging tissue) showing a restoration of voluntary locomotion. As pointed out by Edgerton et al. (2006), it remains an open question as to how to optimally combine individual, disparate therapies into a therapeutic program. In this dissertation, the studies in rats, described in Chapter 4, use both electrostimulation and physical training.

2.3 Active Learning and Bandits

This section provides an overview of active learning and bandit algorithms, slanted toward the work in this dissertation. Intuitively, a learner which asks useful questions should be able to learn more information and use fewer observations doing so than a learner which waits for informative data to arrive by chance. For a view of the traditional field of active learning, i.e., the field of actively querying algorithms which do not obtain reward or suffer regret, the interested reader may refer to the text by Settles (2012). Since this work combines ideas from bandits, Bayesian optimization, and batch selection, a brief review of the literature in each of these areas is included. Much of the material on these three topics is drawn from our recently submitted paper[1], which also provides the material in Chapter 3. Attempts to deal with the problem of time variation as faced by active learning systems are discussed in Section 2.3.4. A review of the literature on control algorithms and learning agents in biological applications follows in Section 2.3.5.

2.3.1 Bandit Algorithms

Many problems have a repeating game structure, in which there exist a set of alternatives and the agent must choose one among these at each round. The agent then receives the reward corresponding to this action. Crucially, only this (possibly noisy) reward is observed, while rewards corresponding to other actions are unrevealed; this suggests that, in order to obtain a good amount of reward, the agent must use a strategy which exploits knowledge of the reward function to obtain high reward, and explores the reward function thoroughly enough to be assured that the action which is apparently best is, in fact, the one which yields the highest reward. The balance between these competing imperatives is referred to as the exploration-exploitation tradeoff. The most crucial division among algorithms in the bandit class is in regard to the types of structural assumptions made about the reward function, i.e., whether the payoffs corresponding to individual actions are somehow related to one another, or if they are totally independent.

2.3.1.1 Classical Setting

Exploration-exploitation tradeoffs have been classically studied in context of the multi-armed bandit problem, in which, from among some finite set of candidate actions, a single action is chosen at each round, and the corresponding (possibly noisy) reward is observed. A recent monograph by Bubeck and Cesa-Bianchi (2012) describes a number of related bandit problems and several algorithms for solving each. Briefly, early work has focused on the case of a finite number of decisions and payoffs that are independent across the arms (Robbins, 1952). In this setting, under some strong assumptions, optimal policies can be computed (Gittins, 1979). Due to the difficulties inherent in doing so, however, a number of heuristic policies have been created. Optimistic allocation of actions according to upper-confidence bounds (UCB) on the payoffs has proven to be particularly effective (Auer et al., 2002). One important feature contributing to the success of UCB algorithms is their

[1]Desautels, Krause, and Burdick, 2013, "Parallelizing Exploration-Exploitation Tradeoffs in Bayesian Global Optimization."

explicit description of the posterior uncertainty about the reward and their direct incorporation of this uncertainty into the decision rule.

2.3.1.2 Making Large Problems Tractable: Structural Assumptions

Recently, approaches for coping with large (or infinite) sets of decisions have been developed. In these cases, since the number of candidate actions is very large compared to the number of actions to be allocated, the reward function cannot be adequately learned if the payoffs are independent. In order to achieve some level of tractability, the dependence between the payoffs associated with different candidate actions must be modeled and exploited. Examples include bandits with linear (Dani et al., 2008; Abernethy et al., 2008) or Lipschitz-continous payoffs (Kleinberg et al., 2008), or bandits on trees (Kocsis and Szepesvári, 2006; Bubeck et al., 2008). Chapter 3 pursues a Bayesian approach to bandits, where fine-grained assumptions on the regularity of the reward function can be imposed through proper choice of the prior distribution (Srinivas et al., 2010).

2.3.2 Bayesian Optimization

The exploration-exploitation tradeoff has also been studied in Bayesian global optimization and response surface modeling, where Gaussian process models are often used due to their flexibility in incorporating prior assumptions about the structure of the payoff function (Brochu et al., 2009). Several bandit-like heuristics, such as Maximum Expected Improvement (Jones et al., 1998), Maximum Probability of Improvement (Mockus, 1989), Knowledge Gradient (Ryzhov et al., 2012), and upper-confidence based methods (Cox and John, 1997), have been developed to balance exploration with exploitation and have been successfully applied in learning problems (e.g., Lizotte et al., 2007). In contrast, the Entropy Search algorithm of Hennig and Schuler (2012) considers the estimate of the location of the optimum at any given time and tries to take the action which will greedily decrease future losses, a less bandit-like and more optimization-focused heuristic. Recently, Srinivas et al. (2010) analyzed GP-UCB, an upper-confidence bound sampling based algorithm for this setting, and proved bounds on its cumulative regret, and thus convergence rates for Bayesian global optimization. This work builds on this foundation and generalizes it to the parallel setting.

2.3.3 Parallel Selection

To enable parallel selection, one must account for the delay between decisions and observations. Most existing approaches that can deal with such delay result in a multiplicative increase in the cumulative regret as the delay grows. Only recently, Dudik et al. (2011) demonstrated that it is possible to obtain regret bounds that only increase *additively* with the delay (i.e., the penalty becomes negligible for large numbers of decisions). However, the approach of Dudik et al. only applies to contextual bandit problems with finite decision sets, and thus not to settings with complex (even nonparametric) payoff functions. In contrast, there has been heuristic work in parallel Bayesian global optimization using GPs, e.g., by Ginsbourger et al. (2010). The state of the art is the *simulation matching* algorithm of Azimi et al. (2010), which uses the posterior of the payoff function at the beginning of the batch to simulate observations that the sequential algorithm would encounter if it could receive feedback during the batch, creating some number of Monte Carlo samples from this posterior over future sequential algorithm behavior. These Monte Carlo samples are then aggregated into a batch of observations which is intended to "closely match" the set of actions which would be taken by the sequential algorithm if it had been run with sequential feedback. To our knowledge, no theoretical results regarding the regret of this algorithm exist. This dissertation compares with this approach in experimental settings in Section 3.6. Azimi et al. (2012b) also propose a very different algorithm which adaptively choses the level of parallelism it will allow. This done in a manner which depends on the expected prediction error between the posterior given the simulated observations in the batch in progress versus the true posterior which would be available assuming the observations had actually been obtained.

Adaptive batch size selection in active learning has also been a topic of some recent interest, e.g., by Chakraborty et al. (2011), who propose a method for selecting batches of key frames from video for classification. Azimi et al. (2012a) recently extended the simulation matching construction to the batch classification setting. Section 4.3 of the text by Settles (2012) discusses a number of methods for batch active learning in non-reward settings.

2.3.4 Active Learning in the Face of Time Variation

Allowing the reward function to vary with time creates some substantial problems for the active learning agent, as uncertainty cannot be said to monotonically decrease in this setting, and thus actions must be allocated to manage this increase in uncertainty, while still doing well with respect to a performance metric. In the bandit setting, Hazan and Kale (2009) develop an algorithm for solving a time-varying bandit problem with bounded total variation, given that the reward is linear over the actions, and bound its regret in terms of this bound on variation. Another bandit algorithm, contextual zooming (Slivkins, 2011), takes advantage of Lipschitz continuity on the space of actions and contexts to bound local variation in the reward. This algorithm then adaptively partitions the space, according to its own actions, such that higher resolution partitions are generated in regions of frequent context arrival and high action frequency. This is particularly advantageous when the contexts (e.g., time) arrive in some relatively benign fashion (e.g., in sequence). Slivkins (2011) is thus able to obtain bounds on the regret in a number of drifting or stochastically changing reward settings, including in cases of spatial constraints on this drift. The problem of how to minimize regret in the case of a dynamic, but structured reward function remains open, however.

2.3.5 Learning Systems and Control Algorithms in Biological Contexts

Some attempts to use algorithmic methods for managing the interaction of therapeutic systems with complex biological systems have been made in the past. Santaniello et al. (2011), for example, modeled the responses of simulated neurons in the ventral intermediate nucleus of the thalamus to deep brain electrical stimulation as a parametric dynamic model, with coefficients fitted online; they then controlled the application of the stimulator to attempt to disrupt tremor-like activity in this population of simulated cells. Another application of interest has been brain-computer interfaces (BCIs). Traditionally, BCIs use fairly simple decoding algorithms, which classify neural activity by comparison to pre-computed, possibly stereotyped patterns corresponding to putative volitional states. An interesting example in SCI rats is presented by Manohar et al. (2011). Before SCI, the animals are first trained to press a button with their hindlimbs on-cue and secondly trained to produce similar activity in the hindlimb motor cortex. After SCI, the reward is again triggered by decoded cortical activity, indicating that this cortical activity is robust to the changes in the motor cortex resulting from SCI. More sophisticated methods for choosing the code-book have been recently employed; Fruitet et al. (2012) developed and tested (Fruitet et al., 2013) in humans a bandit-based algorithm to create personalized BCIs. This algorithm adaptively chooses which action to ask the user to imagine performing, in order to eventually produce good discrimination between this neurological state and the resting state, thus creating a classifier for the state of a volitional "button-press" manifested in the patient's sensorimotor rhythms. While intended to ultimately work with much larger sets of imagined motor actions, these experiments used a menu of three to five possible actions. Gürel and Mehring (2012) use what is essentially an ϵ-greedy bandit as a meta-algorithm for online, continuing calibration of a BCI decoding process, following an initial supervised training stage. Vidaurre et al. (2011) employed a multi-phase calibration of such a system, including the user's immediate feedback responses to the online-decoded intention, in a study with 11 human volunteers. In contrast, the present work in SCI therapy uses more structure over the space of actions (over which the reward function is modeled as a Gaussian process) in order to enable the use of a very large decision set. Further, the goal of the GP-UCB-based algorithms described in this dissertation is the optimization of a reward function *during the course of the experiment*, somewhat different than the optimization of *terminal* classifier performance. This is dependent on (eventually) measuring the reward given by each action, and so the reward must be measurable online. In the BCI setting, this is not possible in the unsupervised phase. Finally, while the unsupervised phase of some BCI algorithms incorporates tracking of the features, the implementation of GP-BUCB in Chapter 4 explicitly models the time-variation of the response function and chooses actions to yield high reward.

2.4 Gaussian Processes

Gaussian processes (GPs) are a flexible model for capturing knowledge about functions from a variety of classes. Rasmussen and Williams (2006) provide an excellent introduction to GPs.[2] In this section and Section 2.5, I present a brief review of relevant parts of GP theory and practice.

Rasmussen and Williams (2006) define a Gaussian process as follows:

Definition 1. *A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.*

Another way of thinking of a Gaussian processes is to describe it as a probability distribution over functions mapping from an arbitrary (possibly continuous) index set $\mathcal{S}$ to $\mathbb{R}$. To denote that such a function $f : \mathcal{S} \to \mathbb{R}$ is drawn from a GP over functions on $\mathcal{S}$, one may write

$$f \sim \mathcal{GP}(\mu(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}')), \tag{2.1}$$

where $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{S}$, $\mu(\boldsymbol{x})$ is the *mean function* and $k(\boldsymbol{x}, \boldsymbol{x}')$ is the *covariance function.* Any element $\boldsymbol{x} \in \mathcal{S}$ corresponds to the identity of a random variable in Definition 1, and the value of any function drawn from the GP at $\boldsymbol{x}$, $f(\boldsymbol{x})$, corresponds to a particular assignment of a value to the random variable identified by $\boldsymbol{x}$. Note that here and in the remainder of the text, we use the notation f to denote a function over $\mathcal{S}$ and $f(\cdot)$ to denote the value of that function at a finite collection of elements in $\mathcal{S}$. A GP is fully specified by the mean function and covariance function; for any collection of elements of the GP, these may be used to define the Gaussian joint distribution over the values of those random variables. For example, on any collection of $n \in \mathbb{N}^+$ elements of $\mathcal{S}$, where this collection of points is described as a column vector, $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t]^T$, the Gaussian joint distribution over this column vector of corresponding values of f is

$$f(\boldsymbol{X}) = [f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_t)]^T \sim \mathcal{N}(\mu(\boldsymbol{X}), K(\boldsymbol{X}, \boldsymbol{X})), \tag{2.2}$$

where $\mu(\boldsymbol{X})$ is the column vector of values of the mean function and $K(\boldsymbol{X}, \boldsymbol{X})$ is the covariance matrix, and where the entries of $K(\boldsymbol{X}, \boldsymbol{X})$ are

$$[K(\boldsymbol{X}, \boldsymbol{X})]_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j), \, \forall\, i, j \leq t.$$

In particular, for any $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{S}$, the covariance of $f(\boldsymbol{x})$ and $f(\boldsymbol{x}')$ is $k(\boldsymbol{x}, \boldsymbol{x}')$, hence the designation "covariance function." Importantly, the existence of a covariance function automatically grants a consistency property, that is, the distribution of any subcollection of random variables from the GP

[2]This text, along with accessible software for using GPs in MATLAB, are provided by the authors at `www.gaussianprocess.org/gpml/`

is identical, whether that subcollection is considered on its own or as part of a larger collection of variables. In much of the analysis in Chapter 3, this dissertation will assume without loss of generality that the mean function $\mu(\boldsymbol{x})$ is zero everywhere in $\mathcal{S}$; as can be seen by an examination of Equations (2.3) and (2.6) below, it is mathematically equivalent to perform regression on deviations from $\mu(\boldsymbol{x})$, expressed as $f(\boldsymbol{x}) - \mu(\boldsymbol{x})$, rather than on the actual value of the function $f(\boldsymbol{x})$, and thus the corresponding change of definitions is preferred for simplicity of presentation and calculation. The key object which defines the structure of the model is then the covariance function, $k(\boldsymbol{x}, \boldsymbol{x}')$, examined in more detail in Section 2.5.

2.4.1 Regression Using Gaussian Processes

In this dissertation, it is of greatest interest to use the GP model to make predictions about $f(\boldsymbol{x}^*)$, the value of a function drawn from the GP at a test point $\boldsymbol{x}^*$, given some finite set of observations $\mathbf{y}$ corresponding to the set $\boldsymbol{X}$. Assuming i.i.d. Gaussian noise on these observations with noise variance σ_n^2, and denoting the size of $\boldsymbol{X}$ as t, the individual observations corresponding to $\boldsymbol{x}_i \in \boldsymbol{X}$ may be written as

$$y_i = f(\boldsymbol{x}_i) + \epsilon_i, \forall i \in \{1, \dots, t\}, \tag{2.3}$$

where $\epsilon_i \sim \mathcal{N}(0, \sigma_n^2)$, $\forall i \in \{1, \dots, t\}$. The joint distribution over the observations $\mathbf{y} = [y_1, \dots, y_t]^T$ and $f(\boldsymbol{x}^*)$ is thus

$$\begin{bmatrix} \mathbf{y} \\ f(\boldsymbol{x}^*) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu(\boldsymbol{X}) \\ \mu(\boldsymbol{x}^*) \end{bmatrix}, \begin{bmatrix} K(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n^2 I & K(\boldsymbol{X}, \boldsymbol{x}^*) \\ K(\boldsymbol{x}^*, \boldsymbol{X}) & k(\boldsymbol{x}^*, \boldsymbol{x}^*) \end{bmatrix} \right). \tag{2.4}$$

Since these variables are jointly Gaussian, the posterior distribution over $f(\boldsymbol{x}^*)$ conditioned on $\mathbf{y}$ is

$$f(\boldsymbol{x}*) \mid \mathbf{y} \sim \mathcal{N}(\mu_t(\boldsymbol{x}^*), \sigma_t^2(\boldsymbol{x}^*)), \tag{2.5}$$

where the posterior mean and variance are denoted $\mu_t(\boldsymbol{x}^*)$ and $\sigma_t^2(\boldsymbol{x}^*)$ and respectively have the forms

$$\mu_t(\boldsymbol{x}^*) = \mu(\boldsymbol{x}^*) + K(\boldsymbol{x}^*, \boldsymbol{X})(K(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n^2 I)^{-1}(\mathbf{y} - \mu(\boldsymbol{X})) \tag{2.6}$$

$$\sigma_t^2(\boldsymbol{x}^*) = k(\boldsymbol{x}^*, \boldsymbol{x}^*) - K(\boldsymbol{x}^*, \boldsymbol{X})(K(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n^2 I)^{-1} K(\boldsymbol{X}, \boldsymbol{x}^*). \tag{2.7}$$

These forms represent the uncertainty over which function from the Gaussian process explains the observations, and capture the marginalization over all functions which could be drawn from the GP; this implicit marginalization is a manifestation of the consistency property. Notice that for $\sigma_n^2 \neq 0$, the matrix inversion used to calculate both the posterior mean and variance is possible even if there are some repeated observations in $\boldsymbol{X}$, such that $K(\boldsymbol{X}, \boldsymbol{X})$ is singular.

Of crucial importance for the algorithms described in Chapter 3 is the observation that while Equation (2.6) is dependent on the observations $\mathbf{y}$, Equation (2.7) is not. This second fact allows an algorithm to use knowledge of how uncertain it *will be* after receiving the observations which are known to be pending. This in turn enables the assembly of batches of experiments which are constructed for simultaneous execution and observation, yet are only redundant to a controlled degree.

2.5 Covariance Functions

The previous discussion has assumed the availability of a covariance function $k(\cdot,\cdot)$, but covariance functions are themselves a topic of significant interest. It should be noted that the terms "kernel function" and "covariance function" are often used interchangeably in the context of GPs; apart from the formal definitions in this section, this dissertation also uses them somewhat flexibly, and usages of "kernel" or "kernel function" should be understood to mean a valid covariance function. In applications, the choice of covariance function is a major opportunity to specify the structure of the problem, as expert knowledge can be used to choose a covariance function which encodes a great deal of problem-specific knowledge. Such choices result in relatively stronger or weaker links between the values of f at various pairs of elements $\boldsymbol{x}, \boldsymbol{x}'$ from within the chosen domain $\mathcal{S}$ of the covariance function (which is thus the domain of functions drawn from the corresponding GP). Further, some regions of $\mathcal{S}$ could be specified to have larger variances than others, encoding the knowledge that some particular region of the space is known to produce more variable behavior. Similarly, in $\mathbb{R}^d$, a covariance function could be constructed to produce draws from the GP which vary more slowly in certain directions than others; this can be very useful, e.g., if the system being modeled is known to be relatively insensitive to one of the variables describing the location in $\mathbb{R}^d$, whereas it is more sensitive to others.

As covariance functions are a crucial topic in understanding GPs, Rasmussen and Williams (2006) also provide a thorough description of this topic. A brief distillation of the relevant details is presented here.

A real *kernel function* k is a function which maps pairs of elements of $\mathcal{S}$ into $\mathbb{R}$, i.e., $k : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$. A kernel which is symmetric in its arguments, i.e., $k(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{x}', \boldsymbol{x})$, is referred to as a *symmetric kernel.* In Euclidean spaces, the properties of stationarity and isotropy are of interest. If k is solely a function of $\boldsymbol{x} - \boldsymbol{x}'$, k is *stationary,* and it is invariant to translations of the inputs. If k is a function of only the (vector) magnitude of this difference, $|\boldsymbol{x} - \boldsymbol{x}'|$, it is *isotropic.*

Covariance functions are a sub-class of symmetric kernel functions. For any kernel function k

and collection $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ of n elements of $\mathcal{S}$, the *Gram matrix* is $K(\boldsymbol{X}, \boldsymbol{X})$, where

$$[K(\boldsymbol{X}, \boldsymbol{X})]_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j), \, \forall i, j \leq n.$$

If for a symmetric kernel k, the Gram matrix is positive semi-definite (i.e., given any vector $v \in \mathbb{R}^n$, $v^T K(\boldsymbol{X}, \boldsymbol{X})v \geq 0$) for all $n \in \mathbb{N}^+$, k is termed *positive semi-definite.* If $v^T K(\boldsymbol{X}, \boldsymbol{X})v = 0$ if and only if $v = 0$, the kernel is termed *positive definite.* If a kernel k is symmetric and positive semi-definite, k is also a valid *covariance function*, and any Gram matrix corresponding to k is referred to as a *covariance matrix.* The covariance function in essence defines similarity between the values of $f(\boldsymbol{x})$ and $f(\boldsymbol{x}')$ for any two elements $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{S}$, and does so by reference to $\boldsymbol{x}$ and $\boldsymbol{x}'$, rather than the function itself. This is useful for a variety of reasons:

- For a finite collection of points $D \in \mathcal{S}$, the covariance matrix $K(D, D)$ of the jointly Gaussian values of f at the elements of D can be precomputed and, conditioned on (noisy, and possibly repeated) observations of elements of D, the posterior over f at D can be computed using this matrix. This fact is useful for the bandit algorithms considered in Chapter 3.

- Complex representations of $\boldsymbol{x}$ and $\boldsymbol{x}'$ in feature spaces, even infinite-dimensional feature spaces, can be encoded by using a kernel (covariance) function which operates (typically very simply) on $\boldsymbol{x}$ and $\boldsymbol{x}'$; this is commonly known in machine learning as the "kernel trick," and is employed to leverage a simple technique into a much more complex and expressive suite of techniques while incurring very little computational expense. In this sense, Gaussian process regression is actually Bayesian linear regression, extended via the kernel trick.

- Perhaps most importantly from a practical perspective, careful choices of the representation of the inputs $\boldsymbol{x} \in \mathcal{S}$, the kernel function, and the corresponding hyperparameters can allow expert knowledge to be encoded into the GP model very simply and intuitively. As an expert works with a system, they might plausibly acquire some intuition of which variables are functionally important and which are less so. It might be that there are many ways to describe the objects in $\mathcal{S}$, but some may be more convenient or meaningful than others; in essence, this is an extension of the kernel trick. The choice of covariance function also is an opportunity for expert intuition to be expressed; linear or squared-exponential kernels imply quite different things about the functions drawn from the corresponding GPs. Similarly, the choice of the hyperparameters of selected kernel function encodes information like the relative sensitivity of the responses to variation in any of the chosen features.

As an example of the last point, in Chapter 4, a variant of the GP-BUCB algorithm is applied to choosing electrical stimuli $\boldsymbol{x}$ for SCI therapy from a set D, using a GP model of the responses. There are discretely many pairs of electrodes which can be chosen to stimulate the spinal cord;

these could be labeled essentially arbitrarily, e.g., by the integers, but there is not an obvious way of doing so without destroying the useful information given by knowing which electrodes are which. The kernel functions in Chapter 4 are chosen such that each element $\boldsymbol{x} \in D$ is represented in terms of the physical locations of the corresponding pair of epidural electrodes; since function within the spinal cord is spatially organized, this spatial representation is an intuitive way to capture the functional similarity of the evoked potentials which will result. The choice of stationary covariance functions over this Euclidian parameterization of the stimulus space also encodes information, in particular, that the uncertainty about how strongly the spinal cord and muscles will respond is the same everywhere, and that this response strength is expected to be of a particular sensitivity to the same movement of the cathode or anode, regardless of where the electrodes are on the cord. Finally, the hyperparameters of the kernel function are chosen to quantify the greater or lesser spatial sensitivity of the evoked potential to the rostro-caudal or lateral movements of each of the anode and cathode. As will be demonstrated in Chapter 4, this series of choices allows the GP model to express meaningful information about the spinal cord and thus gives the algorithm the ability to make intelligent choices about which stimuli to apply.

2.5.1 Reproducing Kernel Hilbert Spaces

A *reproducing kernel Hilbert space* (RKHS) is a Hilbert space of functions over a set $\mathcal{S}$ associated with a particular kernel function. More precisely, an RKHS is defined by Rasmussen and Williams (2006) as follows:

Definition 2. *Let $\mathcal{H}$ be a Hilbert space of real functions f defined on an index set $\mathcal{S}$. Then $\mathcal{H}$ is called a reproducing kernel Hilbert space (RKHS) endowed with an inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ (and norm $||f||_{\mathcal{H}} = \sqrt{\langle f, f \rangle_{\mathcal{H}}}$) if there exists a function $k : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ with the following properties:*

- *for every $\boldsymbol{x}$, $k(\boldsymbol{x}, \boldsymbol{x}')$ as a function of $\boldsymbol{x}'$ belongs to $\mathcal{H}$, and*
- *k has the* reproducing property $\langle f(\cdot), k(\cdot, \boldsymbol{x}) \rangle_{\mathcal{H}} = f(\boldsymbol{x})$.

For any kernel function, there exists a unique RKHS, and for each RKHS, there exists a unique corresponding kernel function (Aronszajn, 1950, Part I, Section 2)

The RKHS can also be viewed as the set of functions

$$\left\{ f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i) \, : \, n \in \mathbb{N}, \, \boldsymbol{x}_i \in \mathcal{S}, \, \alpha_i \in \mathbb{R} \right\}, \tag{2.8}$$

with the associated inner product

$$\langle f, g \rangle_{\mathcal{H}} = \sum_{i=1}^{n} \sum_{j=1}^{n'} \alpha_i \alpha'_j k(\boldsymbol{x}_i, \boldsymbol{x}_j), \tag{2.9}$$

where $g(\boldsymbol{x}) = \sum_{j=1}^{n'} \alpha'_j k(\boldsymbol{x}, \boldsymbol{x}'_j)$. Note that, with the assumption that $\mu(\boldsymbol{x}) = 0, \forall \boldsymbol{x} \in D$ and the definition α_i as the ith entry of the column vector $(K(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n^2 I)^{-1}\mathbf{y}$, where $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, the predictive mean of a GP model, given by Equation (2.6), can be written as a linear combination of kernel evaluations; this is precisely the form of Equation (2.8), and so the RKHS may be regarded as the space of possible posterior means for any collection of data, given that GP regression is conducted with a particular kernel. This is known as the *reproducing kernel map* perspective.

For the purposes of this dissertation, it is most important to note that the RKHS norm of a function f provides a measure of how closely f matches the possible posterior means which would be constructed from a GP model using the corresponding kernel and a finite amount of data, i.e., how well f can be captured by the model; a small value for $||f||_{\mathcal{H}}$ implies that f is much like a linear combination of relatively few copies of the kernel function, whereas a large or infinite value for $||f||_{\mathcal{H}}$ implies that this is not the case. Alternatively, a finite value of $||f||_{\mathcal{H}}$ could be viewed as the rapid decay of the eigenvalues of f with respect to an eigenfunction basis of the RKHS. This use of the RKHS norm to quantify the complexity of a function with respect to a GP model underlies the performance guarantee given by the third case of Theorem 1 in Chapter 3; one expects that it should be easier to make decisions about a function which is well-captured by the model (has a low RKHS norm) than one which is only poorly captured by the model.

For a somewhat different treatment than the above (again, drawn from Rasmussen and Williams, 2006), see Wahba (1990), Sections 1.1 and 1.4.

2.5.2 Stationary Covariance Functions on $\mathbb{R}^d$

A number of stationary functions on $\mathbb{R}^d$ will be used in the remainder of this work.

The first of these is the Squared Exponential covariance function, which may be written as

$$k_{SE}(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 \exp(-1/2 \cdot (\boldsymbol{x} - \boldsymbol{x}')^T \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{x}')), \tag{2.10}$$

where Σ is a symmetric, positive definite matrix in $\mathbb{R}^{d \times d}$. Often, Σ is chosen as $\Sigma = \frac{1}{2} l^2 I$, such that the covariance function is isotropic and can be rewritten as

$$k_{SEiso}(r) = \sigma^2 \exp\left(\frac{-r^2}{l^2}\right), \tag{2.11}$$

where $r = |\boldsymbol{x} - \boldsymbol{x}'|$. Another common choice for Σ is to have a separate lengthscale $l_i > 0$ for each dimension, such that

$$\Sigma_{i,j} = \begin{cases} 0 & i \neq j \\ l_i^2 & i = j \end{cases}, \tag{2.12}$$

such that samples from the GP tend to be rougher in dimensions with small values of l_i and larger in dimensions with large values of l_i. A GP with any version of the squared exponential covariance function is infinitely many times mean square differentiable (i.e, very smooth; see Rasmussen and Williams, 2006, Sections 4.1.1 & 4.2.1 for details).

Another class of covariance functions of interest is the Matérn class. Following Rasmussen and Williams (2006), assuming isotropy, and thus $r = |\boldsymbol{x} - \boldsymbol{x}'|$, the general form of the Matérn covariance is

$$k_{Mat\acute{e}rn}(r) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu} r}{l} \right)^{\nu} K_{\nu} \left(\frac{\sqrt{2\nu} r}{l} \right), \tag{2.13}$$

where ν and l are positive parameters and K_ν is a modified Bessel function. The parameter ν describes the smoothness of samples from the resulting GP; for a given ν, the resulting process is k times mean square differentiable if and only if $\nu > k$ (again, see Rasmussen and Williams, 2006, for details). In the limit as $\nu \to \infty$, the isotropic squared exponential covariance function is recovered from Equation (2.13). From a practical perspective, ν gives a useful means of expressing expert knowledge about the system being modeled and its characteristics, such that it can be chosen to give an appropriate degree of smoothness; small values of ν result in predicted functions which are extremely jagged, whereas larger values give smoother predicted functions. Certain values for ν are commonly used for GP regression, specifically $\nu = 1/2$, $3/2$, and $5/2$; these three values yield a practically useful range of graduated roughness. Again using the isotropic form, these values of ν yield a simplified form of the covariance function consisting of a polynomial in r multiplied by an exponential in r (as do any others such that $\nu = p + 1/2$, where $p \in \mathbb{N}$):

$$k_{\nu=1/2}(r) = \exp\left(-\frac{r}{l}\right), \tag{2.14}$$

$$k_{\nu=3/2}(r) = \left(1 + \frac{\sqrt{3}r}{l}\right) \exp\left(-\frac{\sqrt{3}r}{l}\right), \tag{2.15}$$

$$k_{\nu=5/2}(r) = \left(1 + \frac{\sqrt{5}r}{l} + \frac{5r^2}{3l^2}\right) \exp\left(-\frac{\sqrt{5}r}{l}\right). \tag{2.16}$$

The covariance function in the first case, for which $\nu = 1/2$, is also known as the exponential covariance function; the resulting GP is also known as the Ornstein-Uhlenbeck process (Uhlenbeck and Ornstein, 1930, Section IV). As with other isotropic covariance functions on $\mathbb{R}^d$, it is possible to use a positive definite lengthscale matrix Σ and substitute $\sqrt{(\boldsymbol{x} - \boldsymbol{x}')^T \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{x}')}$ for r, yielding anisotropic covariances of some desired form. As mentioned above, the parameter ν has a relationship to the mean square differentiability of the resulting GP, but for $\nu = p + 1/2$, $p \in \mathbb{N}$, this relationship can be put on firmer footing. It can be shown that, with particular choices of the necessary parameters, the covariance function corresponding to a special case of the steady state of a continuous-time, one-dimensional, auto-regressive process of order p with a Gaussian disruption, i.e., an AR(p) process, is the Matérn covariance with $\nu = p + 1/2$ (Rasmussen and Williams, 2006, Appendix B). Thus, the order 1 AR(p) process corresponds to a special case of the exponential covariance function ($\nu = 1/2$) and order 2 corresponds to a special case of the Matérn covariance with $\nu = 3/2$. Using this fact, the choice of a Matérn covariance for $\nu = 3/2$ can be loosely considered to correspond to modeling the system in question as a second-order linear dynamical system in $\boldsymbol{x}$, such that the "position" and "velocity" are states of the system, but no higher order derivatives. This might also be thought of as specifying the degree of "memory" the system retains about past states.

2.5.3 Non-stationary Covariance Functions on $\mathbb{R}^d$

A non-stationary covariance function on $\mathbb{R}^d$ is one which, while holding the difference between the inputs $\boldsymbol{x} - \boldsymbol{x}'$ constant, changes value with translation of $\boldsymbol{x}, \boldsymbol{x}'$ within $\mathbb{R}^d$. While many others exist, one simple non-stationary covariance function for $\mathcal{S} \in \mathbb{R}^d$ is the linear covariance function, $k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x} \cdot \boldsymbol{x}' + \sigma_0^2$, that is, the dot product of the vectors $\boldsymbol{x}$ and $\boldsymbol{x}'$ with some offset σ_0^2. If this covariance function is used, Gaussian process regression on a collection of data reduces to Bayesian linear regression. By changing from a simple dot product to a product using some positive semi-definite weight matrix Σ_p, one may produce a general linear covariance function, which corresponds to a dot product under a transformation of $\mathbb{R}^d$ by scaling and rotation. Note that scaling $\boldsymbol{x}$ by a constant c produces an increase of $k(\boldsymbol{x}, \boldsymbol{x}')$ by the same factor; linear covariance functions make sense for cases where variability can be expected to increase with distance from the origin or where $\mathcal{S}$ lies entirely on a hypersphere centered on the origin (after any transformation of coordinates implicit in Σ_p), e.g., $|\boldsymbol{x}| = 1, \forall \boldsymbol{x} \in \mathcal{S}$ if $\Sigma_p = I$. Further, since the covariance matrix K corresponding to any finite collection $\boldsymbol{X}$ in $\mathcal{S}$ of size n consists of the matrix of ones, multiplied by σ_0^2, summed with a matrix composed of the product of two $n \times d$ matrices containing the coordinates of these individual locations in $\mathbb{R}^d$, K is of rank no more than $d + 1$. This is consistent with the statement above that a dot product covariance function corresponds to Bayesian linear regression, since the model corresponds to a hyperplane in $\mathbb{R}^{d+1}$.

2.5.4 Constructing Covariance Functions

It is also possible to construct more complicated covariance functions by using compositions of simpler covariance functions. In particular, the sum of two covariance functions is a covariance function, and a sample from the GP corresponding to this covariance function corresponds to a sum of independent samples from the two GPs which correspond to the two original covariance functions. Similarly, the product of two covariance functions is also a covariance function, such that draws from the GP corresponding to the product covariance function can be thought of as being the product of two independent draws from the GPs corresponding to the individual factor covariance functions.

Finally, two covariance functions $k_1(\boldsymbol{x}_1, \boldsymbol{x}'_1)$ and $k_2(\boldsymbol{x}_2, \boldsymbol{x}'_2)$ over different spaces $\mathcal{S}_1$ and $\mathcal{S}_2$ may be combined as either a sum $k(\boldsymbol{x}, \boldsymbol{x}') = k_1(\boldsymbol{x}_1, \boldsymbol{x}'_1) + k_2(\boldsymbol{x}_2, \boldsymbol{x}'_2)$ or product $k(\boldsymbol{x}, \boldsymbol{x}') = k_1(\boldsymbol{x}_1, \boldsymbol{x}'_1) \cdot k_2(\boldsymbol{x}_2, \boldsymbol{x}'_2)$ to form a covariance function k for $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{S}_1 \times \mathcal{S}_2$ via the sum and product methods above. This allows the construction of covariance functions from individual covariance functions over subspaces, e.g., different dimensions of $\mathbb{R}^d$, or even radically different sets; $\mathcal{S}_1$ might be $\mathbb{R}$, whereas $\mathcal{S}_2$ could be nodes in a graph, words in a corpus, or something more exotic.

While it is possible to construct covariance functions which natively represent covariance over a space $\mathcal{S}$ which is not a subset of $\mathbb{R}^d$, another way to construct covariance functions for such $\mathcal{S}$ is to find a mapping $g : \mathcal{S} \to \mathbb{R}^d$ and then use a covariance function $\tilde{k} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ to construct a covariance function, such that $k(\boldsymbol{x}, \boldsymbol{x}') = \tilde{k}(g(\boldsymbol{x}), g(\boldsymbol{x}'))$.

The combination of all of these techniques allows a great deal of flexibility in terms of modeling assumptions; some of this flexibility is used in Chapter 4 and further possibilities are discussed in Chapter 5.

Chapter 3

Theoretical Contributions

3.1 Introduction

Many problems require optimizing an unknown reward function, from which we can only obtain noisy observations. A central challenge is choosing actions that both explore (estimate) the function and exploit our knowledge about likely high reward regions. Carefully calibrating this exploration–exploitation tradeoff is especially important in cases where the experiments are costly in some sense, e.g., when each experiment takes a long time to perform and the time window available for experiments is short. In some such settings, it may be desirable to run several experiments in parallel. By parallelizing the experiments, substantially more information can be gathered in the same time-frame; however, future actions must be chosen without the benefit of intermediate results. One might conceptualize these problems as choosing "batches" of experiments to run simultaneously. The challenge is to assemble batches of experiments which both explore the function and exploit what are currently known to be high-performing regions.

Two key, interrelated questions arise: the *computational* question of how one should efficiently choose, out of the combinatorially large set of possible batches, those that are most effective; and the *statistical* question of how the algorithm's performance depends on the size of the batches (i.e., the degree of informational parallelism). In this chapter, we address these questions by presenting GP-BUCB and GP-AUCB; these are novel, efficient algorithms for selecting batches of experiments in the Bayesian optimization setting where the reward function is modeled as a sample from a Gaussian process prior (or has low norm in the associated Reproducing Kernel Hilbert Space).

In more detail, we provide the following main contributions:

- We introduce GP-BUCB, a novel algorithm for selecting actions to maximize reward in large-scale exploration-exploitation problems while accommodating parallel or batch execution of

The work in this chapter has been submitted to the Journal of Machine Learning Research as Desautels, Krause, and Burdick, "Parallelizing Exploration-Exploitation Tradeoffs In Bayesian Global Optimization", itself a substantial expansion of Desautels et al. (2012), published at ICML 2012.

the actions and the consequent observation of their reward. GP-BUCB may also be used in the setting of a bounded delay between initiation of an action and observation of its reward.

- We also introduce GP-AUCB, an algorithm which adaptively exploits parallelism to choose batches of actions, the sizes of which are limited by the conditional mutual information gained therein; this limit is such that the batch sizes are small when actions are selected for which the algorithm knows relatively little about the reward and batch sizes are large when the reward function is well known for the actions selected. We show that this adaptive parallelism is well-behaved and can easily be constrained using pre-defined limits.
- We prove sublinear bounds on the cumulative regret incurred by algorithms of a general class, including GP-BUCB and GP-AUCB, and provide corollary extensions to each of these two algorithms, thus bounding their rates of convergence.
- For some common kernels, we show that if the problem is initialized by making observations corresponding to an easily selected and provably bounded set of queries, the regret of GP-BUCB can be bounded to a constant multiplicative factor of the known regret bounds of the sequential GP-UCB algorithm. This asymptotic post-initialization guarantee is *independent* of batch size B so long as B grows at most polylogarithmically in T, the number of queries to be selected in total. This implies (near-)linear informational speedup through parallelism.
- We demonstrate how execution of many UCB algorithms, including the GP-UCB, GP-BUCB, and GP-AUCB algorithms, can be drastically accelerated by *lazily evaluating* the posterior variance. This technique does not result in any loss in accuracy.
- We evaluate GP-BUCB and GP-AUCB on several synthetic benchmark problems, as well as two real data sets, respectively related to automated vaccine design and therapeutic spinal cord stimulation. We show that GP-BUCB and GP-AUCB are competitive with state of the art heuristics for parallel Bayesian optimization, and that under certain circumstances, GP-BUCB is competitive with sequential action selection under GP-UCB, despite having the disadvantage of delayed feedback.
- We consider more complex notions of execution cost in the batch and delay settings and identify areas of this cost and performance space where our algorithms make favorable tradeoffs and are therefore especially suitable for employment.

In the remainder of the chapter, we begin by formally describing the problem setting (Section 3.2). In the next section, we describe the GP-BUCB algorithm, present the main regret bound, which applies to a general class of algorithms using an upper confidence bound decision rule, and present corollaries bounding the regret of GP-BUCB and initialized GP-BUCB (Section 3.3). We extend this analysis to GP-AUCB, providing a regret bound for that algorithm, discuss different possible stopping conditions for similar algorithms, and introduce the notion of lazy variance calculations (Section 3.4). We compare our algorithms' performance with each other and with several other algorithms across a variety of problem instances, including two real data sets (Section 3.6). Finally, we present our conclusions (Section 3.7).

3.2 Problem Setting and Background

We wish to make a sequence of decisions (or equivalently, take actions) $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T \in D$, where D is the *decision set*, which is often (but not necessarily) a compact subset of $\mathbb{R}^d$, and the subscript denotes the *round* in which that decision was made; each round is an opportunity for the algorithm to make one decision. For each decision $\boldsymbol{x}_t$, we observe noisy scalar reward $y_t = f(\boldsymbol{x}_t) + \varepsilon_t$, where $f : D \to \mathbb{R}$ is an unknown function modeling the expected payoff $f(\boldsymbol{x})$ for each decision $\boldsymbol{x}$. For now we assume that the noise variables ε_t are i.i.d. Gaussian with known variance σ_n^2, i.e., $\varepsilon_t \sim \mathcal{N}(0, \sigma_n^2),\ \forall t \geq 1$. This assumption will be relaxed later. If the decisions $\boldsymbol{x}_t$ are made one at a time, each with the benefit of all observations $y_1, \ldots, y_{t-1}$ corresponding to previous actions $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-1}$, we shall refer to this case as the *strictly sequential* setting. In contrast, the main problem tackled in this chapter is the challenging setting where $\boldsymbol{x}_t$ may only depend on $y_1, \ldots, y_{t'}$, for some $t' < t - 1$.

In selecting these decisions, we wish to maximize the cumulative reward $\sum_{t=1}^{T} f(\boldsymbol{x}_t)$, or equivalently minimize the cumulative *regret* $R_T = \sum_{t=1}^{T} r_t$, where $r_t = [f(\boldsymbol{x}^*) - f(\boldsymbol{x}_t)]$ and $\boldsymbol{x}^* \in \boldsymbol{X}^* = \operatorname{argmax}_{\boldsymbol{x} \in D} f(\boldsymbol{x})$ is an optimum decision (assumed to exist, but not necessarily to be unique). In experimental design, D might be the set of possible stimuli that can be applied, and $f(\boldsymbol{x})$ models the response to stimulus $\boldsymbol{x} \in D$. By minimizing the regret, we ensure progress towards the most effective stimulus uniformly over T. In fact, the average regret, R_T/T, is a natural upper bound on the suboptimality of the best stimulus considered so far, i.e., $R_T/T \geq \min_t [f(\boldsymbol{x}^*) - f(\boldsymbol{x}_t)]$ (often called the *simple regret*, Bubeck et al., 2009).

3.2.1 The Problem: Parallel or Delayed Selection

In many applications, at time τ, we wish to select a *batch* of decisions, e.g., $\boldsymbol{x}_\tau, ..., \boldsymbol{x}_{\tau+B-1}$, where B is the size of the batch, to be evaluated in parallel. One natural application is the design of *high-throughput* experiments, where several experiments are performed in parallel, but feedback is only received after the experiments have concluded. In other settings, feedback is delayed. In both situations, decisions are selected sequentially, but when making the decision $\boldsymbol{x}_t$ in round t, we can only make use of the feedback obtained in rounds $1, \ldots, t'$, for some $t' \leq t - 1$. Formally, we assume there is some mapping $\mathrm{fb} : \mathbb{N} \to \mathbb{N}_0$ such that $\mathrm{fb}[t] \leq t - 1,\ \forall t \in \mathbb{N}$, and when making a decision at time t, we can use feedback up to and including round $\mathrm{fb}[t]$. If $\mathrm{fb}[t] = 0$, no observation information is available.

This framework can model a variety of realistic scenarios. Setting $B = 1$ and $\mathrm{fb}[t] = t - 1$ corresponds to the non-delayed, strictly sequential setting in which a single action is selected and the algorithm waits until the corresponding observation is returned before selecting the succeeding action. The "simple batch" setting in which we wish to select batches of size B can be captured by setting $\text{fb}[t]_{SB} = \lfloor (t-1)/B \rfloor B$, i.e.,

$$\text{fb}[t]_{SB} = \begin{cases} 0 & : t \in \{1, \dots, B\} \\ B & : t \in \{B+1, \dots, 2B\} \\ 2B & : t \in \{2B+1, \dots, 3B\} \\ & \vdots \end{cases} .$$

Note that in the batch case, the time indexing within the batch is a matter of algorithmic construction, since the batch is built in a sequential fashion, but actions are initiated and observations are received simultaneously. If we wish to formalize the problem of selecting actions when feedback from those actions is delayed by exactly B rounds, the *simple delay* setting, we can simply define this feedback mapping as $\text{fb}[t]_{SD} = \max\{t - B, 0\}$. Note that in both the simple batch and delay cases, $B = 1$ is the strictly sequential case. In comparing these two simple cases for the same value of B, we observe that $\text{fb}[t]_{SB} \geq \text{fb}[t]_{SD}$, that is, the set of observations available in the simple batch case for making the tth decision is always at least as large as in the simple delay case, suggesting that the delay case is in some sense "harder" than the batch case. As we will see, however, the regret bounds for each algorithm presented in this chapter are established based on the maximal number of pending observations (i.e, those which have been initiated, but are still incomplete), which is $B - 1$ in both of these settings, resulting in unified proofs and regret bounds for the two cases.

More complex cases may also be handled by GP-BUCB. For example, we may also be interested in executing B experiments in parallel, but the duration of an experiment may be variable, and we can start a new experiment as soon as one finishes; this translates to having a queue of pending observations of some finite size B. Since we only select a new action when the queue is not full, there can be at most $B - 1$ actions waiting in the queue at the time any action is selected, as in the simple batch and delay cases. Again, the maximum number of pending observations is the key to bounding the regret, though the variable delay for individual observations requires some subtlety with the feedback mapping's specification.

Even in complex cases, one quantity intuitively used in describing the difficulty of the problem instance is the constant B. In this light, while developing GP-BUCB and initialized GP-BUCB in Sections 3.3.4 and 3.3.5, we only assume that the mapping $\text{fb}[t]$ is specified as part of the problem instance (possibly chosen adversarially) and $t - \text{fb}[t] \leq B$ for some known constant B.

3.2.2 Modeling f via Gaussian Processes (GPs)

Regardless of when feedback is obtained, if we are to turn finite numbers of observations into useful inference about the payoff function f, we will have to make assumptions about its structure. In particular, for large (possibly infinite) decision sets D there is no hope to do well, i.e., incur little regret or even simply converge to an optimal decision, if we do not make any assumptions. For good performance, we must choose a regression model which is both simple enough to be learned and expressive enough to capture the relevant behaviors of f. One effective formalism is to model f as a sample from a Gaussian process (GP) prior, as discussed in Section 2.4. Briefly recapitulated, a GP is a probability distribution across a class of – typically smooth – functions, which is parameterized by a kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$, which characterizes the smoothness of f, and a mean function $\mu(\boldsymbol{x})$. For notational convenience, we assume $\mu(\boldsymbol{x}) = 0$, without loss of generality, and we additionally assume that $k(\boldsymbol{x}, \boldsymbol{x}) \leq 1$, $\forall \boldsymbol{x} \in D$. We write $f \sim \mathcal{GP}(\mu, k)$ to denote that we model f as sampled from such a GP. If noise is i.i.d. Gaussian and the distribution of f is conditional on a vector of observations $\boldsymbol{y}_{1:t-1} = [y_1, ..., y_{t-1}]^T$ corresponding to decisions $\boldsymbol{X}_{t-1} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_{t-1}]^T$, one obtains a Gaussian posterior distribution $f(\boldsymbol{x})|\boldsymbol{y}_{1:t-1} \sim \mathcal{N}(\mu_{t-1}(\boldsymbol{x}), \sigma^2_{t-1}(\boldsymbol{x}))$ for each $\boldsymbol{x} \in D$, where

$$\mu_{t-1}(\boldsymbol{x}) = K(\boldsymbol{x}, \boldsymbol{X}_{t-1})[K(\boldsymbol{X}_{t-1}, \boldsymbol{X}_{t-1}) + \sigma_n^2 I]^{-1} \boldsymbol{y}_{1:t-1} \text{ and} \tag{3.1}$$

$$\sigma^2_{t-1}(\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x}) - K(\boldsymbol{x}, \boldsymbol{X}_{t-1})[K(\boldsymbol{X}_{t-1}, \boldsymbol{X}_{t-1}) + \sigma_n^2 I]^{-1} K(\boldsymbol{x}, \boldsymbol{X}_{t-1})^T, \tag{3.2}$$

where $K(\boldsymbol{x}, \boldsymbol{X}_{t-1})$ denotes the row vector of kernel evaluations between $\boldsymbol{x}$ and the elements of $\boldsymbol{X}_{t-1}$, the set of decisions taken in the past, and $K(\boldsymbol{X}_{t-1}, \boldsymbol{X}_{t-1})$ is the matrix of kernel evaluations, where $[K(\boldsymbol{X}_{t-1}, \boldsymbol{X}_{t-1})]_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $\forall \boldsymbol{x}_i, \boldsymbol{x}_j \in \boldsymbol{X}_{t-1}$, i.e., the covariance matrix of the values of f over the set so far observed. Since Equations (3.1) and (3.2) can be computed efficiently, closed-form posterior inference is computationally tractable in a GP distribution via linear algebraic operations.

3.2.3 Conditional Mutual Information

A number of information theoretic quantities will be essential to the analysis of the algorithms presented in this chapter. In particular, the quantity

$$\gamma_T = \max_{A \subseteq D, \, |A| \leq T} I(f; \mathbf{y}_A) \tag{3.3}$$

is the maximum mutual information between the payoff function f and observations $\mathbf{y}_A$ of any set $A \subseteq D$ of the T decisions evaluated up until time T. For a GP, the mutual information $I(f; \mathbf{y}_A)$ is

$$I(f; \mathbf{y}_A) = H(\mathbf{y}_A) - H(\mathbf{y}_A \,|\, f) = \frac{1}{2} \log \left| \mathbf{I} + \sigma_n^{-2} K(A, A) \right|,$$

where $K(A, A)$ is the covariance matrix of the values of f at the elements of the set A, $H(\mathbf{y}_A)$ is the differential entropy of the probability distribution over the set of observations $\mathbf{y}_A$, and $H(\mathbf{y}_A \,|\, f)$ is the corresponding value when the distribution over $\mathbf{y}_A$ is conditioned on f, or equivalently, $f(A)$. The *conditional mutual information* for observations $\mathbf{y}_A$ given previous observations $\mathbf{y}_S$, is defined (for two finite sets $A, S \subseteq D$) as

$$I(f; \boldsymbol{y}_A \mid \boldsymbol{y}_S) = H(\boldsymbol{y}_A \mid \boldsymbol{y}_S) - H(\boldsymbol{y}_A \mid f).$$

The conditional mutual information gain from observations $\boldsymbol{y}_A$ of the set of actions A can also be calculated as a sum of the marginal information gains of each observation in the set; conditioned on $\boldsymbol{y}_S$, and for $A = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T\}$ this sum is

$$I(f; \boldsymbol{y}_A \mid \boldsymbol{y}_S) = \sum_{t'=1}^{T} \log\left(1 + \sigma_n^{-2} \sigma_{t'-1}^2(\boldsymbol{x}_{t'})\right), \tag{3.4}$$

where the term $\sigma_{t'-1}^2(\boldsymbol{x}_{t'})$ is the posterior variance over $f(\boldsymbol{x}_{t'})$, conditioned on $\boldsymbol{y}_S$ and $y_1, ..., y_{t'-1}$. It is important to note that $\sigma_{t'-1}^2(\boldsymbol{x}_{t'})$ is independent of the values of the observations. Since the sum's value can thus be calculated without making the observations (i.e., during the course of assembling a batch), it is possible to calculate the mutual information which will be gained from any hypothetical set of observations.

3.2.4 The GP-UCB approach

Modeling f as a sample from a GP has the major advantage that the predictive uncertainty can be used to guide exploration and exploitation. While several heuristics, such as Expected Improvement (Mockus et al., 1978) and Most Probable Improvement (Mockus, 1989) have been effectively employed in practice, nothing is known about their convergence properties in the case of noisy observations. Srinivas et al. (2010), guided by the success of upper-confidence based sampling approaches for multi-armed bandit problems (Auer, 2002), analyzed the *Gaussian process Upper Confidence Bound (GP-UCB)* selection rule,

$$\boldsymbol{x}_t = \underset{\boldsymbol{x} \in D}{\operatorname{argmax}} \left[\mu_{t-1}(\boldsymbol{x}) + \alpha_t^{1/2} \sigma_{t-1}(\boldsymbol{x}) \right]. \tag{3.5}$$

This decision rule uses α_t, a domain-specific time-varying parameter, to trade off exploitation (sampling $\boldsymbol{x}$ with high mean) and exploration (sampling $\boldsymbol{x}$ with high standard deviation). Srinivas et al. (2010) showed that, with proper choice of α_t, the cumulative regret of GP-UCB grows sublinearly for many commonly used kernel functions. This algorithm is presented in simplified pseudocode as Algorithm 1.

Algorithm 1 GP-UCB

```
Input: Decision set $D$, GP prior $\mu_0, \sigma_0$, kernel function $k(\cdot, \cdot)$
for $t = 1, 2, \ldots, T$ do
    Choose $\boldsymbol{x}_t = \text{argmax}_{\boldsymbol{x} \in D}[\mu_{t-1}(\boldsymbol{x}) + \alpha_t^{1/2}\sigma_{t-1}(\boldsymbol{x})]$
    Compute $\sigma_t(\cdot)$, e.g., via Equation (3.2)
    Obtain $y_t = f(\boldsymbol{x}_t) + \varepsilon_t$
    Perform Bayesian inference to obtain $\mu_t(\cdot)$, e.g., via Equation (3.1)
end for
```

Implicit in the definition of the GP-UCB decision rule, Equation (3.5), is the corresponding confidence interval,

$$C_t^{\text{seq}}(\boldsymbol{x}) \equiv \left[\mu_{t-1}(\boldsymbol{x}) \pm \alpha_t^{1/2}\sigma_{t-1}(\boldsymbol{x})\right], \tag{3.6}$$

where this confidence interval's upper confidence bound is the value of the argument of the decision rule. For this (or any) confidence interval, we will refer to the difference between the uppermost limit and the lowermost, here $w = 2\alpha_t^{1/2}\sigma_{t-1}(\boldsymbol{x})$, as the *width* w. This confidence interval is based on the posterior over f given $\boldsymbol{y}_{1:t-1}$; a new confidence interval is created for round $t+1$ after adding y_t to the set of observations. Srinivas et al. (2010) carefully select α_t such that a union bound over all $t \geq 1$ and $\boldsymbol{x} \in D$ yields a high-probability guarantee of confidence interval correctness; it is this guarantee which enables the construction of high-probability regret bounds. Using this guarantee, Srinivas et al. (2010) then prove that the cumulative regret of the GP-UCB algorithm can be bounded (up to logarithmic factors) as $R_T = O^*(\sqrt{T\alpha_T\gamma_T})$, where α_T is the confidence interval width multiplier described above and γ_T is the maximum mutual information between the payoff function f and the observations $\boldsymbol{y}_{1:T}$. For many commonly used kernel functions, Srinivas et al. (2010) show that γ_T grows sublinearly and α_T only needs to grow polylogarithmically in T, implying that R_T is also sublinear; thus $R_T/T \to 0$ as $T \to \infty$, i.e., GP-UCB is a no-regret algorithm.

Motivated by the strong theoretical and empirical performance of GP-UCB, we explore generalizations to batch and parallel selection (i.e., $B > 1$). One naïve approach would be to update the GP-UCB score, Equation (3.5), only once new feedback becomes available, but this algorithm would simply select the same action at each time step between acquisition of new observations, leading to limited exploration. To encourage more exploration, one may instead require that no decision is selected twice within a batch (i.e., simply rank decisions according to the GP-UCB score, and pick decisions in order of decreasing score, until new feedback is available). However, since f often varies smoothly, so does the GP-UCB score; under some circumstances, this algorithm would also suffer from limited exploration. Further, if the optimal set $\boldsymbol{X}^* \subseteq D$ is of size $|\boldsymbol{X}^*| < B$ and the regret of the every sub-optimal action is finite, the algorithm would be forced to suffer linear regret, since some $\boldsymbol{x} \notin \boldsymbol{X}^*$ must be included in every batch. In short, both of these naïve algorithms are flawed, in part because they fail to exploit knowledge of the redundancy in the observations which will be obtained as a result of their actions.

From the preceding discussion, choosing a diverse set of inputs and while relying upon only outdated feedback presents a significant challenge. In the following, we introduce the Gaussian process - Batch Upper Confidence Bound (GP-BUCB) algorithm, which encourages diversity in exploration, uses past information in a principled fashion, and yields strong performance guarantees. We also extend the GP-BUCB algorithm by the creation of the Gaussian process - Adaptive Upper Confidence Bound (GP-AUCB) algorithm, which retains the theoretical guarantees of the GP-BUCB algorithm, but creates batches of variable length in a data-driven manner.

3.3 GP-BUCB Algorithm and Regret Bounds

We introduce the GP-BUCB algorithm in Section 3.3.1. Section 3.3.2 states the chapter's major theorem, a bound on the cumulative regret of a general class of algorithms including GP-BUCB and GP-AUCB. This main result is in terms of a quantity C, a bound on information used within a batch; this quantity is examined in some detail in Section 3.3.3. Using this examination, Section 3.3.4 provides a corollary, bounding the regret of GP-BUCB specifically. Section 3.3.5 improves this regret bound by initializing GP-BUCB with a finite set of observations.

3.3.1 GP-BUCB: An Overview

A key property of GPs is that the predictive variance at time t, Equation (3.2), only depends on $\boldsymbol{X}_{t-1} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-1}\}$, i.e, *where* the observations are made, but not *which* values $\boldsymbol{y}_{1:t-1} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1}]^T$ were actually observed. Thus, it is possible to compute the posterior variance used in the sequential GP-UCB decision rule, Equation (3.5), even while previous observations are not yet available. To do so, we *hallucinate* observations $\boldsymbol{y}_{\mathrm{fb}[t]+1:t-1} = [\mu_{\mathrm{fb}[t]}(\boldsymbol{x}_{\mathrm{fb}[t]+1}), \ldots, \mu_{\mathrm{fb}[t]}(\boldsymbol{x}_{t-1})]$ for every observation not yet received. A natural approach towards parallel exploration is therefore to replace the sequential decision rule, Equation (3.5), with a decision rule which instead sequentially chooses decisions within the batch as

$$\boldsymbol{x}_t = \operatorname*{argmax}_{\boldsymbol{x} \in D} \left[\mu_{\mathrm{fb}[t]}(\boldsymbol{x}) + \beta_t^{1/2} \sigma_{t-1}(\boldsymbol{x})\right]. \tag{3.7}$$

Here, the parameter β_t has a role analogous to the parameter α_t in the GP-UCB algorithm. The confidence intervals corresponding to this decision rule are of the form

$$C_t^{\mathrm{batch}}(\boldsymbol{x}) \equiv \left[\mu_{\mathrm{fb}[t]}(\boldsymbol{x}) \pm \beta_t^{1/2} \sigma_{t-1}(\boldsymbol{x})\right]. \tag{3.8}$$

The resulting GP-BUCB algorithm is shown in pseudocode as Algorithm 2. This approach naturally encourages diversity in exploration by taking into account the change in predictive variance which

Algorithm 2 GP-BUCB

```
Input: Decision set $D$, GP prior $\mu_0, \sigma_0$, kernel function $k(\cdot, \cdot)$
for $t = 1, 2, \ldots, T$ do
  Choose $\boldsymbol{x}_t = \text{argmax}_{\boldsymbol{x} \in D}[\mu_{\text{fb}[t]}(\boldsymbol{x}) + \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x})]$
  Compute $\sigma_t(\cdot)$
  if $t = \text{fb}[t+1]$ then
    Obtain $y_{t'} = f(\boldsymbol{x}_{t'}) + \varepsilon_{t'}$ for $t' \in \{\text{fb}[t]+1, \ldots, t\}$
    Perform Bayesian inference to obtain $\mu_t(\cdot)$
  end if
end for
```

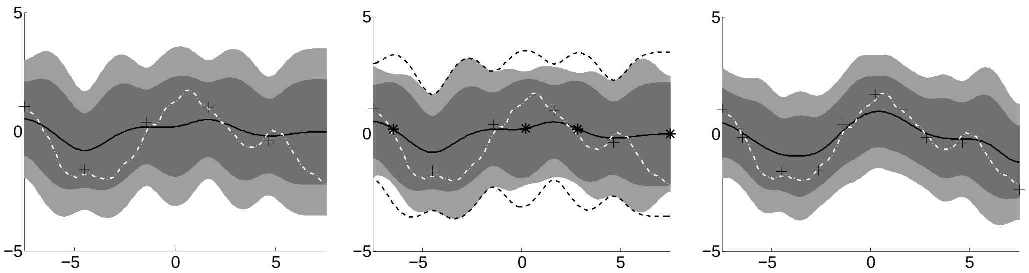
will eventually occur after pending observations: since the payoffs of "similar" decisions have similar predictive distributions, exploring one decision will automatically reduce the predictive variance of similar decisions, and thus their value in terms of exploration. This decision rule appropriately deprecates those observations which will be redundant with respect to pending observations, resulting in a more correct valuation of the action of exploring them.

The disadvantage of this approach appears as the algorithm progresses deeper into the batch. At each time t, the algorithm creates confidence intervals $C_t^{\text{batch}}(\boldsymbol{x})$, the width of which is proportional to $\sigma_{t-1}(\boldsymbol{x})$. This standard deviation is used because it is the standard deviation of the posterior over the payoff f if all observations $\boldsymbol{y}_{1:t-1}$ are available, which enables GP-BUCB to avoid exploratory redundancy. However, doing so conflates the information which is actually available, gained via the observations $\boldsymbol{y}_{1:\text{fb}[t]}$, with the hallucinated information corresponding to actions $\boldsymbol{x}_{\text{fb}[t]+1}$ through $\boldsymbol{x}_{t-1}$. Thus, $\sigma_{t-1}(\boldsymbol{x})$ is "overconfident" about our knowledge of the function. The ratio of the width of the confidence interval derived from the set of actual observations $\boldsymbol{y}_{1:\text{fb}[t]}$ to the width of the confidence interval derived from the partially hallucinated set of observations $\boldsymbol{y}_{1:t-1}$ is given by $\sigma_{\text{fb}[t]}(\boldsymbol{x})/\sigma_{t-1}(\boldsymbol{x})$. This quantity is related to $I(f(\boldsymbol{x}); \boldsymbol{y}_{\text{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\text{fb}[t]})$, the hallucinated conditional mutual information with respect to $f(\boldsymbol{x})$, such that

$$\frac{\sigma_{\text{fb}[t]}(\boldsymbol{x})}{\sigma_{t-1}(\boldsymbol{x})} = \exp\Big(I(f(\boldsymbol{x}); \boldsymbol{y}_{\text{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\text{fb}[t]})\Big). \tag{3.9}$$

This ratio quantifies the degree of "overconfidence" with respect to the posterior as of the beginning of the batch. Crucially, if there exists some constant C, such that $I(f(\boldsymbol{x}); \boldsymbol{y}_{\text{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\text{fb}[t]}) \leq C$, $\forall \boldsymbol{x} \in D$, the ratio $\sigma_{\text{fb}[t]}(\boldsymbol{x})/\sigma_{t-1}(\boldsymbol{x})$ can also be bounded for every $\boldsymbol{x} \in D$. Bounding this ratio of confidence interval shrinkage due to hallucinated information links the hallucinated posterior at round t back to the posterior as of the last feedback, at round fb$[t]$. Using this bound, the algorithm can be constructed to compensate for its overconfidence.

This compensation for overconfidence must strike a delicate balance between the algorithmic requirement of allowing the predictive variance over f to shrink as the algorithm hallucinates observations, thus allowing the algorithm to avoid redundancy, and maintaining the probabilistic inter-



(a) *Beginning of batch 2* (b) *Last decision of batch 2* (c) *Beginning of batch 3*

Figure 3.1: (a): The confidence intervals $C^{\text{seq}}_{\text{fb}[t]+1}(\boldsymbol{x})$ (dark), computed from previous noisy observations $\boldsymbol{y}_{1:\text{fb}[t]}$ (crosses), are centered around the posterior mean (solid black) and contain $f(\boldsymbol{x})$ (white dashed) w.h.p. To avoid overconfidence, GP-BUCB chooses $C^{\text{batch}}_{\text{fb}[t]+1}(\boldsymbol{x})$ (light gray) such that even in the worst case, the succeeding confidence intervals in the batch, $C^{\text{batch}}_{\tau}(\boldsymbol{x})$, $\forall\tau : \text{fb}[\tau] = \text{fb}[t]$, will contain $C^{\text{seq}}_{\text{fb}[t]+1}(\boldsymbol{x})$. (b): Due to the observations that GP-BUCB "hallucinates" (stars), the outer posterior confidence intervals $C^{\text{batch}}_{t}(\boldsymbol{x})$ shrink from their values at the start of the batch (black dashed), but still contain $C^{\text{seq}}_{\text{fb}[t]+1}(\boldsymbol{x})$, as desired. (c): Upon selection of the last decision of the batch, the feedback for all decisions is obtained, and for the subsequent action selection in round t', new confidence intervals $C^{\text{seq}}_{\text{fb}[t']+1}(\boldsymbol{x})$ and $C^{\text{batch}}_{\text{fb}[t']+1}(\boldsymbol{x})$ are computed.

pretation of the confidence interval size, such that high-probability statements can be made about the regret. One satisfactory approach is to increase the width of the confidence intervals (through proper choice of the parameter β_t), such that the confidence intervals used by GP-BUCB are *conservative*, i.e., contain the true function $f(\boldsymbol{x})$ with high probability. More precisely, we require that for all $t \in \{1, 2, \ldots, T\}$ and $\boldsymbol{x} \in D$, $C^{\text{seq}}_{\text{fb}[t]+1}(\boldsymbol{x}) \subseteq C^{\text{batch}}_{t}(\boldsymbol{x})$; that is, the batch algorithm's confidence intervals are sufficiently large to guarantee that even for the last action selection in the batch, they contain the confidence intervals which would be created by the GP-UCB algorithm, Equation (3.6), given $\boldsymbol{y}_{1:\text{fb}[t]}$. Srinivas et al. (2010) provide choices of α_t such that the GP-UCB confidence intervals have a high-probabilty guarantee of correctness $\forall t \geq 1, \boldsymbol{x} \in D$. If it can be shown that $C^{\text{seq}}_{\text{fb}[t]+1}(\boldsymbol{x}) \subseteq C^{\text{batch}}_{t}(\boldsymbol{x})$, $\forall \boldsymbol{x} \in D, t \in \mathbb{N}$, the batch confidence intervals inherit the high-probability guarantee of correctness. By multiplicatively increasing the width of the hallucinated confidence intervals and using the same factor of width increase for all $\boldsymbol{x} \in D$ and $t \in \mathbb{N}$, the redundancy control of the hallucinated observations is maintained, and the required degree of conservatism can simultaneously be achieved. Figure 3.1 illustrates this idea. The main difficulty in using this approach is finding the degree of conservatism required to guarantee the containment of $C^{\text{seq}}_{\text{fb}[t]+1}(\boldsymbol{x})$ by $C^{\text{batch}}_{t}(\boldsymbol{x})$. Sections 3.3.2 and 3.3.4 show that by using a multiplicative factor $\exp(C)$ relative to $\alpha_{\text{fb}[t]}$, where C is a bound on the conditional mutual information which can be gained within a batch, β_t can be chosen such that containment of the sequential confidence intervals within the batch confidence intervals is achieved. This containment follows from Equation (3.14). We further show that, with appropriate initialization, the regret can be made to only mildly increase relative to GP-UCB, providing theoretical support for the potential for parallelizing GP optimization.

3.3.2 General Regret Bound

Our main theorem bounds the regret of GP-BUCB and related algorithms. This regret bound is formulated in terms of a bound C on the maximum conditional mutual information which can be hallucinated with respect to $f(\boldsymbol{x})$ for any $\boldsymbol{x}$ in D, which we assume to be known to the algorithm. We discuss methods of obtaining such a bound in Section 3.3.3. This bound is used to relate hallucinated confidence intervals, used to select actions, and the posterior confidence intervals as of the last feedback obtained, which contain the payoff function f with high probability. This theorem holds under any of three different assumptions about f, which may all be of practical interest. In particular, it holds even if the assumption that f is sampled from a GP is replaced by the assumption that f has low norm in the Reproducing Kernel Hilbert Space (RKHS) associated with the kernel function.

Theorem 1. *Let $\delta \in (0,1)$, γ_t be as defined in Equation (3.3), and α_t be a time-varying exploration-exploitation tradeoff parameter, as in Equation (3.5). Let the variance be bounded, such that $k(\boldsymbol{x},\boldsymbol{x}) \leq 1$, $\forall\, \boldsymbol{x} \in D$. Suppose one of the following sets of assumptions holds:*

1. *D is a finite set, the payoff function f is sampled from a known GP prior with known noise variance σ_n^2, and $\alpha_t = 2\log(|D|t^2\pi^2/6\delta)$.*
2. *$D \subseteq [0,l]^d$ is compact and convex, with $d \in \mathbb{N}$, $l > 0$, and the payoff function f is sampled from a known GP prior with known noise variance σ_n^2. Choose $\alpha_t = 2\log(t^2 2\pi^2/(3\delta)) + 2d\log\left(t^2 dbl\sqrt{\log(4da/\delta)}\right)$, where the constants $a, b > 0$ and $k(\boldsymbol{x},\boldsymbol{x}')$ are such that the following bound holds with high probability on the derivatives of GP sample paths f:*

$$\Pr\left\{\sup_{\boldsymbol{x}\in D} |\partial f/\partial x_j| > L\right\} \leq a e^{-(L/b)^2},\ j = 1,\ldots,d.$$

3. *D is arbitrary and the payoff function f has RKHS norm bounded as $||f||_k \leq M$ for some constant M. The noise variables ε_t form an* arbitrary *martingale difference sequence (meaning that $\mathbb{E}[\varepsilon_t \,|\, \varepsilon_1, \ldots, \varepsilon_{t-1}] = 0$ for all $t \in \mathbb{N}$), uniformly bounded by σ_n. Choose $\alpha_t = 2M^2 + 300\gamma_t \ln^3(t/\delta)$.*

Further suppose there exists a mapping $fb[t]$ which dictates at which rounds new feedback becomes available to the algorithm and a bound $C > 0$ such that, for all $t \in \mathbb{N}$ and all $\boldsymbol{x} \in D$,

$$I(f(\boldsymbol{x}); \boldsymbol{y}_{fb[t]+1:t-1} \mid \boldsymbol{y}_{1:fb[t]}) \leq C, \tag{3.10}$$

where $\boldsymbol{x}_t$ is selected according to Equation (3.7) using $\beta_t = \exp(2C)\alpha_{fb[t]}$ for all $t \in \{1,\ldots,T\}$. Then the cumulative regret is bounded by $O^*(\sqrt{T\gamma_T \exp(2C)\alpha_T})$ with high probability. Precisely,

$$\Pr\left\{R_T \leq \sqrt{C_1 T \exp(2C)\alpha_T\gamma_T}, +2 \quad \forall T \geq 1\right\} \geq 1-\delta$$

where $C_1 = 8/\log(1+\sigma_n^{-2})$.

Proof. The proof of this result is presented in Appendix A.1. $\square$

The key quantity that controls the regret in Theorem 1 is C, the bound in Equation (3.10) on the maximum conditional mutual information obtainable within a batch with respect to $f(\boldsymbol{x})$ for any $\boldsymbol{x} \in D$. In particular, the cumulative regret bound of Theorem 1 is a factor $\exp(C)$ larger than the regret bound for the sequential ($B = 1$) GP-UCB algorithm. Thus, for any practical meaningfulness of the bound, we must be able to define C. The various methods which can be used for selecting C are explored in the following sections.

3.3.3 Suitable Choices for C

The functional significance of a bound C on the information hallucinated with respect to any $f(\boldsymbol{x})$ arises through this quantity's ability to bound the degree of contamination of the GP-BUCB confidence intervals, given by Equation (3.8), with hallucinated information. As background for finding suitable values C, extension of the discussion of Section 3.2.3 on mutual information is required.

Two properties of the mutual information are particularly useful. These properties are monotonicity (adding an element $\boldsymbol{x}$ to the set A cannot decrease the mutual information between f and the corresponding set of observations $\boldsymbol{y}_A$) and submodularity (the increase in mutual information between f and $\boldsymbol{y}_A$ with the addition of an element $\boldsymbol{x}$ to set A cannot be greater than the corresponding increase in mutual information if $\boldsymbol{x}$ is added to $A', A' \subseteq A$) (Krause and Guestrin, 2005).

Using the time indexing notation developed in Section 3.2.1, the property of monotonicity allows the following series of inequalities:

$$I(f(\boldsymbol{x}); \boldsymbol{y}_{\mathrm{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\mathrm{fb}[t]}) \leq I(f; \boldsymbol{y}_{\mathrm{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\mathrm{fb}[t]}) \tag{3.11}$$

$$\leq \max_{A \subseteq D, |A| \leq B-1} I(f; \boldsymbol{y}_A \mid \boldsymbol{y}_{1:\mathrm{fb}[t]}) \tag{3.12}$$

$$\leq \max_{A \subseteq D, |A| \leq B-1} I(f; \boldsymbol{y}_A) = \gamma_{B-1}. \tag{3.13}$$

The first inequality follows from the monotonicity of mutual information, i.e., the information gained with respect to f as a whole must be at least as large as that gained with respect to $f(x)$. The second inequality holds because we specify the feedback mapping such that $t - \mathrm{fb}[t] \leq B$, and the third inequality holds due to the "information never hurts" bound (Cover and Thomas, 1991), which states that the conditional mutual information $I(f; \boldsymbol{y}_A \mid \boldsymbol{y}_S)$ is monotonically decreasing in S (i.e., as elements are added to set S).

Any bound C on the conditional mutual information hallucinated with respect to any $f(\boldsymbol{x})$ during the selection of a batch can be combined with Equation (3.9) to produce the following statement:

$$\frac{\sigma_{\mathrm{fb}[t]}(\boldsymbol{x})}{\sigma_{t-1}(\boldsymbol{x})} = \exp\Big(I(f(\boldsymbol{x}); \boldsymbol{y}_{\mathrm{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\mathrm{fb}[t]})\Big) \leq \exp(C). \tag{3.14}$$

A bound on $I(f(\boldsymbol{x}); \boldsymbol{y}_{\mathrm{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\mathrm{fb}[t]})$ itself or any bound on one of the terms on the right hand side of Equations (3.11), (3.12) or (3.13) is suitable for our purposes.

3.3.4 Corollary Regret Bound: GP-BUCB

The GP-BUCB algorithm requires that $t - \mathrm{fb}[t] \leq B,\ \forall t \geq 1$, and uses a value C such that, for any $t \in \mathbb{N}$,

$$\max_{A \subseteq D, |A| \leq B-1} I(f; \boldsymbol{y}_A \mid \boldsymbol{y}_{1:\mathrm{fb}[t]}) \leq C, \tag{3.15}$$

thus bounding $I(f(\boldsymbol{x}); \boldsymbol{y}_{\mathrm{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\mathrm{fb}[t]})$ for all $\boldsymbol{x} \in D$ and $t \in \mathbb{N}$ via Inequality (3.12). Otherwise stated, in GP-BUCB, the local information gain with respect to any $f(\boldsymbol{x}), \boldsymbol{x} \in D, t \in \mathbb{N}$ is bounded by fixing the feedback times and then bounding the maximum conditional mutual information with respect to the entire function f which can be acquired by *any* algorithm which chooses any set of $B-1$ or fewer observations. While this argument uses multiple upper bounds, any or all of which may be overly conservative, this approach is still sensible because such a bound C holds for any possible algorithm for constructing batches; it is otherwise quite difficult to disentangle the role of C in setting the exploration-exploitation tradeoff parameter β_t from its role as a bound on how much information is hallucinated by the algorithm, since a larger value of C (and thus β_t) typically *produces* more information gain by promoting exploration under the GP-BUCB decision rule, Equation (3.7).

Since the bound C is related to the maximum amount of conditional mutual information which could be acquired by a set of $B-1$ actions, one expects C to grow monotonically with B; with a larger set of pending actions, there is more potential for explorations which gain additional information. One easy upper bound for the information gained in any batch can be derived as follows. As noted in Section 3.3.3, mutual information is submodular, and thus the maximum conditional mutual information which can be gained by making any set of observations is maximized when the set of observations currently available, to which these new observations will be added, is empty. Letting the maximum mutual information with respect to f which can be obtained by any observation set of size $B-1$ be denoted γ_{B-1} and choosing $C = \gamma_{B-1}$ provides a bound on the possible local conditional mutual information gain for any $t \in \mathbb{N}$ and $\boldsymbol{x} \in D$, as in Inequality (3.13). This choice of bound yields the following Corollary, an extension to Theorem 1:

Corollary 2. *Assume the GP-BUCB algorithm is employed with a constant B such that $t - \mathit{fb}[t] \leq B$ for all $t \geq 1$. Let $\delta \in (0,1)$, and let one of the numbered conditions of Theorem 1 be met. If*

$\beta_t = \exp(2C)\alpha_{fb[t]}$ for all $t \in \{1, \ldots, T\}$, the cumulative regret R_T of the GP-BUCB algorithm can be bounded with probability $1-\delta$ as follows:

$$\Pr\left\{R_T \leq \sqrt{C_1 T \exp(2\gamma_{B-1})\alpha_T \gamma_T} + 2, \quad \forall T \geq 1\right\} \geq 1-\delta,$$

where $C_1 = 8/\log(1+\sigma_n^{-2})$ and γ_{B-1} and γ_T are as defined in Equation (3.3).

While the above result is useful, the choice $C = \gamma_{B-1}$ is not especially satisfying on its own. The maximum information gain γ_{B-1} usually grows at least as $\Omega(\log B)$, implying that $\exp(C)$ grows at least linearly in B, yielding a regret bound which is also at least linear in B. Section 3.3.5 shows that the GP-BUCB algorithm can be modified such that a constant choice of C *independent* of B suffices.

3.3.5 Better Bounds Through Initialization

To obtain regret bounds *independent* of batch size B, the monotonicity properties of conditional mutual information can again be exploited. This can be done by structuring GP-BUCB as a two-stage procedure. First, an *initialization set* D^{init} of size $|D^{\text{init}}| = T^{\text{init}}$ is selected nonadaptively (i.e., without any feedback); following the selection of this entire set, feedback $\boldsymbol{y}^{\text{init}}$ for all decisions in $D^{\text{init}} = \{\boldsymbol{x}_1^{\text{init}}, \ldots, \boldsymbol{x}_{T^{\text{init}}}^{\text{init}}\}$ is obtained. In the second stage, GP-BUCB is applied to the posterior Gaussian process distribution, conditioned on $\boldsymbol{y}^{\text{init}}$.

Notice that if we define

$$\gamma_T^{\text{init}} = \max_{A \subseteq D, |A| \leq T} I(f; \boldsymbol{y}_A \mid \boldsymbol{y}^{\text{init}}),$$

then, under the assumptions of Theorem 1, using $C = \gamma_{B-1}^{\text{init}}$, the regret of the two-stage algorithm is bounded by $R_T = O(T^{\text{init}} + \sqrt{T\gamma_T^{\text{init}}\alpha_T \exp 2C})$. In the following, we show that it is indeed possible to construct an initialization set D^{init} such that the size T^{init} is dominated by $\sqrt{T\gamma_T^{\text{init}}\alpha_T \exp(2C)}$, and – crucially – that $C = \gamma_{B-1}^{\text{init}}$ can be bounded *independently* of the batch size B.

The initialization set D^{init} is constructed via uncertainty sampling: start with $D_0^{\text{init}} = \emptyset$, and for each $t = 1, \ldots, T^{\text{init}}$, greedily determine the most uncertain decision

$$\boldsymbol{x}_t^{\text{init}} = \operatorname*{argmax}_{\boldsymbol{x} \in D} \sigma_{t-1}^2(\boldsymbol{x})$$

and set $D_t^{\text{init}} = D_{t-1}^{\text{init}} \cap \boldsymbol{x}_t^{\text{init}}$. Note that uncertainty sampling is a special case of the GP-BUCB algorithm with a constant prior mean of 0 and the requirement that for all $1 \leq t \leq T^{\text{init}}$, $\text{fb}[t] = 0$, i.e., no feedback is taken into account for the first T^{init} iterations.

Under the above procedure, we have the following key result about the maximum residual information gain γ^{init}:

Lemma 3. *Suppose uncertainty sampling is used to generate an initialization set D^{init} of size T^{init}. Then*

$$\gamma_{B-1}^{init} \le \frac{B-1}{T^{init}} \gamma_{T^{init}}. \tag{3.16}$$

Proof. The proof of this lemma is presented in Appendix A.2. □

Whenever γ_T is sublinear in T (i.e., $\gamma_T = o(T)$), then the bound on $\gamma_{B-1}^{\text{init}}$ given by Inequality (3.16) converges to zero for sufficiently large T^{init}; thus for *any* constant $C > 0$, we can choose T^{init} as a function of B such that $\gamma_{B-1}^{\text{init}} < C$. Using this choice of C in Theorem 1 bounds the post-initialization regret. In order to derive bounds on T^{init}, we in turn need a bound on γ_T which is analytical and sublinear. Fortunately, Srinivas et al. (2010) prove suitable bounds on how the information gain γ_T grows for some of the most commonly used kernels. We summarize our analysis below in Theorem 4. For sake of notation, define R_T^{seq} to be the regret bound of Srinivas et al. (2010) associated with the sequential GP-UCB algorithm (i.e., Corollary 2 with $B = 1$).

Theorem 4. *Suppose one of the conditions of Theorem 1 is satisfied. Further, suppose the kernel and T^{init} are as listed in Table 3.1, and $B \ge 2$. Fix $\delta > 0$. Let R_T be the regret of the two-stage initialized GP-BUCB algorithm, which ignores feedback for the first T^{init} rounds. Then there exists a constant C' independent of B such that for any $T \ge 0$, it holds with probability at least $1-\delta$ that*

$$R_T \le C' R_T^{seq} + 2||f||_\infty T^{init}, \tag{3.17}$$

where C' takes the value shown in Table 3.1.

The results in Table 3.1 are derived in Appendix A.2. Note that the particular values of C' used in Table 3.1 are not the only ones possible; they are chosen simply because they yield relatively clean algebraic forms for T^{init}. Relative to Theorem 1, which depends on B and T through the product $\exp(2C) T \alpha_T \gamma_T$, Theorem 4 replaces this product with a sum of two terms, one in each of B and T; the term $C' R_T^{\text{seq}}$ in Inequality (3.17) is the cost paid for running the algorithm post-initialization (independent of B, dependent on T), whereas the second term is the cost of performing the initialization (dependent on B, independent of T). Notice that whenever $B = O(\text{polylog}(T))$, $T^{\text{init}} = O(\text{polylog}(T))$, and further, note $R_T^{\text{seq}} = \Omega(\sqrt{T})$. Thus, as long as the batch size does not grow too quickly, the term $O(T^{\text{init}})$ is dominated by $C' R_T^{\text{seq}}$ and the regret bounds of GP-BUCB are only a constant factor, *independent of B*, worse than those of GP-UCB.

3.4 Adaptive Parallelism: GP-AUCB

While the analysis of the GP-BUCB algorithm in Sections 3.3.4 and 3.3.5 used feedback mappings fb[t] specified by the problem instance, it may be useful to let the algorithm control when to request

Kernel Type	**Size T^{init} of Initialization Set D^{init}**	**Regret Multiplier C'**
LINEAR: $\gamma_t \leq \eta d \log(t+1)$	$\max \begin{bmatrix} \log(B) \\ \frac{\log \eta + \log d + 2\log(B)}{2\log(B)-1} e\eta d(B-1)\log(B) \end{bmatrix}$	$\exp(2/e)$
MATÉRN: $\gamma_t \leq \nu t^{\epsilon}$	$(\nu(B-1))^{1/(1-\epsilon)}$	e
RBF: $\gamma_t \leq \eta(\log(t+1))^{d+1}$	$\max \begin{bmatrix} (\log(B))^{d+1} \\ \left(\frac{e}{d+1} \frac{\log \eta + (d+2)\log(B)}{2\log(B)-1}\right)^{d+1} \eta(B-1)(\log(B))^{d+1} \end{bmatrix}$	$\exp((\frac{2d+2}{e})^{d+1})$

Table 3.1: Initialization set sizes for Theorem 4.

feedback, and to allow this feedback period to vary in some range not easily described by any constant B. For example, allowing the algorithm to control parallelism is desirable in situations where the cost of executing the algorithm's queries to the oracle depends on both the number of batches and the number of individual actions or experiments in those batches. Consider a chemical experiment, in which cost is a weighted sum of the time to complete the batch of reactions and the cost of the reagents needed for each individual experiment. In such a case, confronting an initial state of relative ignorance about the reward function, it may be desirable to avoid using a wasteful level of parallelism. Motivated by this, we develop an extension to our requirement that $t - \text{fb}[t] \leq B$; we will instead simply require that the mapping $\text{fb}[t]$ and the sequence of actions selected by the algorithm be such that there exists some bound C, holding for all $t \geq 1$ and $\boldsymbol{x} \in D$, on $I(f(\boldsymbol{x}); \boldsymbol{y}_{\text{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\text{fb}[t]})$, the hallucinated information as of round t with respect to any value $f(\boldsymbol{x})$. This requirement on $\text{fb}[t]$ in terms of C may appear stringent, but in actual fact it can be easily satisfied by on-line, data-driven construction of the mapping $\text{fb}[t]$ after having pre-selected a value for C. The GP-AUCB algorithm controls feedback adaptively through precisely such a mechanism.

Section 3.4.1 introduces GP-AUCB and states a corollary regret bound for this algorithm. A few comments on local versus global stopping criteria for adaptivity of algorithms follow in Section 3.4.2.

3.4.1 GP-AUCB Algorithm

The key innovation of the GP-AUCB algorithm is in choosing $\text{fb}[t]$ online, using a limit on the amount of information hallucinated within the batch. Such adaptive batch length control is possible because we can actually measure online the amount of hallucinated information using Equation (3.4), even in the absence of the observations themselves. When this value exceeds a pre-defined constant C, the algorithm terminates the batch, setting fb for the next batch to the current t (i.e., $\text{fb}[t+1] = t$), and waits for the oracle to return values for the pending queries. The resulting algorithm, GP-AUCB, is shown in Algorithm 3. The GP-AUCB algorithm can also be employed in the delay setting, but rather than using the hallucinated information to decide whether or not to terminate the current batch, the algorithm chooses whether or not to submit an action in this round; the algorithm submits an action if the hallucinated information is $\leq C$ and refuses to submit an action ("balks") if the hallucinated information is $> C$.

In comparison to GP-BUCB, by concerning itself with only the batches *actually* chosen, rather than worst-case batches, the GP-AUCB algorithm eliminates the requirement that C be greater than the information which could be gained in *any* batch, and thus makes the information gain bounding argument less conservative; for such a C,

$$I(f(\boldsymbol{x}); \boldsymbol{y}_{\mathrm{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\mathrm{fb}[t]}) \leq I(f; \boldsymbol{y}_{\mathrm{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\mathrm{fb}[t]}) \leq C, \ \forall \boldsymbol{x} \in D, \ \forall t \geq 1,$$

that is, via the monotonicity of conditional mutual information, the information gain locally under GP-AUCB is bounded by the information gain with respect to f as a whole, which is constrained to be $\leq C$ by the stopping condition. Using such an adaptive stopping condition and the corresponding value of β_t, Equation (3.14) can be used to maintain a guarantee of confidence interval correctness for batches of variable length. In particular, the batch length may possibly become quite large as the shape of f is better and better understood and the variance of $f(\boldsymbol{x}_t)$ tends to decrease. Further, if exploratory actions are chosen, the high information gain of these actions contributes to a relatively early arrival at the information gain threshold C and thus relatively short batch length, even late in the algorithm's run. In this way, the batch length is chosen in response to the algorithm's need to explore or exploit as dictated by the decision rule, Equation (3.7), not simply following a monotonically increasing schedule.

This approach meets the conditions of Theorem 1, allowing the regret of GP-AUCB to be bounded for both the batch and delay settings with the following corollary:

Corollary 5. *If the GP-AUCB algorithm is employed with a specified constant value C, for which $I(f; \boldsymbol{y}_{fb[t]+1:t-1} \mid \boldsymbol{y}_{1:fb[t]}) \leq C, \forall t \in \{1, \ldots, T\}$, δ is a specified constant in the interval $(0,1)$, one of the conditions of Theorem 1 is met, and under the resulting feedback mapping fb$[t]$, $\beta_\tau = \exp(2C)\alpha_{fb[\tau]}, \forall \tau \in \{1, \ldots, t\}$, then*

$$\Pr\left\{R_T \leq \sqrt{C_1 T \exp(2C)\alpha_T \gamma_T} + 2, \quad \forall T \geq 1\right\} \geq 1 - \delta$$

where $C_1 = 8/\log(1 + \sigma_n^{-2})$.

Note that it is also possible to combine the results of Section 3.3.5 with Corollary 5 to produce a two-stage adaptive algorithm which can deliver high starting parallelism, very high parallelism as the run proceeds, and a low information gain bound C, yielding a low regret bound.

Despite the advantages of this approach, the value of C is rather abstract and is certainly less natural for an experimentalist to specify than a maximum batch size or delay length B. However, C can be selected to deliver batches with a specified minimum size B_{min}. To ensure this occurs, C can be set such that $C > \gamma_{(B_{min}-1)}$, i.e., no set of queries of size less than B_{min} could possibly gain enough information to end the batch. Further, if we choose C such that $C < \gamma_{(B_{min})}$, it is possible

Algorithm 3 GP-AUCB

```
Input: Decision set $D$, GP prior $\mu_0, \sigma_0$, kernel $k(\cdot,\cdot)$, information gain threshold $C$.
Set $\text{fb}[t'] = 0,\ \forall t' \geq 1,\ G = 0.$
for $t = 1, 2, \dots, T$ do
  if $G > C$ then
    Obtain $y_{t'} = f(\boldsymbol{x}_{t'}) + \varepsilon_{t'}$ for $t' \in \{\text{fb}[t-1], \dots, t-1\}$
    Perform Bayesian inference to obtain $\mu_{t-1}(\cdot)$
    Set $G = 0$
    Set $\text{fb}[t'] = t-1,\ \forall t' \geq t$
  end if
  Choose $\boldsymbol{x}_t = \text{argmax}_{\boldsymbol{x} \in D}[\mu_{\text{fb}[t]}(\boldsymbol{x}) + \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x})]$
  Set $G = G + \frac{1}{2}\log(1 + \sigma_n^{-2}\sigma_{t-1}^2(x_t))$
  Compute $\sigma_t(\cdot)$
end for
```

to select a batch of size B_{min} which does attain the required amount of information to terminate the batch, and thus B_{min} truly can be thought of as the minimum batch size which could be produced by the GP-AUCB algorithm. Often, however, γ_t is not available directly, and cannot be obtained except for combinatorial optimization; in such a case, if B_{min} is very small, this combinatorial optimization may be tractable, and if B_{min} is too large, greedy maximum entropy sampling can be used to bound $\gamma_{(B_{min}-1)}$ from above, allowing the selection of values for C which satisfy the specification on minimum batch size, if with a large degree of conservatism. While relating C to some B_{min} is not the only way to choose the constant C intelligently, doing so gives a clear way to specify C in a more intuitive and relatable way.

It is also possible to choose a very small value for the constant C and produce nearly sequential actions early, while retaining late-run parallelism and producing a very low regret bound. This can be seen if B_{min} is set to 1; following the analysis above, such a C must satisfy the inequalities $\gamma_0 = 0 < C < \gamma_1$, i.e., C can be a very small positive number. Following rearrangement, the regret of GP-AUCB is bounded by Corollary 5 as $R_T \leq exp(C)R_T^{\text{seq}}$, where R_T^{seq} is the bound of Corollary 2 with $B = 1$, the regret of the GP-UCB algorithm. Since for very small C, $\exp(C)$ is nearly 1, the regret bound of GP-AUCB is only a very little more than for GP-UCB. With regard to action selection, choosing C to be a small positive value should result in GP-AUCB beginning its run by acting sequentially, since most actions gain information greater than C. However, the algorithm has the potential to construct batches of increasing length as $T \to \infty$; even assuming the worst case, that all observations are independent and each gains the same amount of information, the batch length allowed with a given posterior is lower-bounded by $B_{max} \geq C/log(1 + \sigma_n^{-2}\tilde{\sigma}^2_{\text{fb}[t]})$ where $\tilde{\sigma}^2$ is the largest variance among the actions selected in the batch. If the algorithm converges toward the optimal subset $\boldsymbol{X}^* \subseteq D$, as the regret bound suggests it will, and $\boldsymbol{X}^*$ is of finite size, then the variances of the actions selected (and thus the denominator in the expression above) can be expected to generally become very small, producing batches of very long length, even for very small

C. Choosing C as a small positive value thus produces the potential for naturally occurring late-run parallelism for very little additional regret relative to GP-UCB.

3.4.2 Local Stopping Conditions Versus Global Stopping Conditions

Both GP-BUCB and GP-AUCB rely upon bounds on the information gain of the hallucinated observations with respect to f as a whole, but Theorem 1 is stated in terms of a bound on the information gain with respect to $f(\boldsymbol{x})$, which must hold $\forall t \geq 1, \forall \boldsymbol{x} \in D$. During the proof of the regret bound, the information gain threshold is a vehicle for ensuring that the confidence intervals do not shrink to too small a ratio, as in Equation (3.14); this enables a choice of β_t which ensures that $C_t^{\text{batch}}(\boldsymbol{x}) \supseteq C_{\text{fb}[t]+1}^{\text{seq}}(\boldsymbol{x})$ for all $t \geq 1$ and $\boldsymbol{x} \in D$. However, as has been stated above, the standard deviation can be calculated on-line, even without the actual observations, thus enabling exhaustive checking of the ratio $\sigma_{\text{fb}[t]}(\boldsymbol{x})/\sigma_{t-1}(\boldsymbol{x})$ for every $\boldsymbol{x} \in D$; assuming this calculation is practical, this strongly suggests that it would be possible to create an algorithm for which the standard deviation ratio is directly checked, rather than being bounded from above through the information gain. Doing so would also imply the existence of some information gain bound $C \geq (f(\boldsymbol{x}); \boldsymbol{y}_{\text{fb}[t]+1:t-1} | \boldsymbol{y}_{\text{fb}[t]})$, $\forall t \geq 1$, $\boldsymbol{x} \in D$, without requiring recourse to the multiple upper bounding arguments used for GP-BUCB and GP-AUCB. This formulation appears attractive because it might enable the algorithm to avoid waiting for early observations when such waiting may be unnecessary, e.g., when the subsequent actions will be additional initialization and not close to previous actions in the batch. This allows the choice of a very small C, e.g., $C = (1+\epsilon)\gamma_1$, such that if the algorithm makes a single observation at a point, and perhaps some other distant points, it will not stop the batch. Due to the small value of C, this translates to a quite small regret bound under Theorem 1.

If we assume a flat prior, such a procedure would tend to create a first batch which thoroughly initializes over the whole set, since most points would be scattered widely, and therefore the local information gain stopping condition would not be met until actions were proposed close together. For practical purposes, it is therefore necessary to introduce a limit on batch size B_{max} such that the first batch is not of a size approaching that of D. By doing so, the tendency is to produce an algorithm which actually tends to *just* produce batches of the maximal size, i.e., $B = B_{max}$ in the simple parallel case, albeit with a tighter regret bound. If nothing else, this tends to offer justification of the common practice with GP-BUCB of setting β_t much smaller than the theory would suggest setting it, such that the algorithm tends to exploit more heavily.

We implement this algorithm, denoting it GP-AUCB Local, and show it in some of the experiments and figures in Section 3.6, along with the Hybrid Batch Bayesian Optimization algorithm (HBBO) of Azimi et al. (2012b). HBBO implements a similar local check on a hallucinated posterior, though this check is expressed in terms of expected prediction error versus the true posterior if all information had been acquired, rather than information gain, and the local stopping condition is only checked at $\boldsymbol{x}_t$, rather than all $\boldsymbol{x}$ in D.

3.5 Lazy Variance Calculations

In this section, we introduce the notion of lazy variance calculations, which may be used to greatly accelerate the computation of many UCB-based algorithms, including GP-UCB, GP-BUCB, and GP-AUCB, without any loss of performance.

Though GP-UCB, GP-BUCB, and GP-AUCB may be implemented as linear algebraic operations and are thus amenable to computational implementation without sampling, the execution time of the algorithms may still be lengthy, particularly as the number of observations becomes larger. The major computational bottleneck is calculating the posterior mean $\mu_{\mathrm{fb}[t]}(\boldsymbol{x})$ and variance $\sigma^2_{t-1}(\boldsymbol{x})$ for the candidate decisions, as required to calculate the decision rule and choose an action $\boldsymbol{x}_t$. The mean is updated only whenever feedback is obtained, and – upon computation of the Cholesky factorization of $K(\boldsymbol{X}_{\mathrm{fb}[t]}, \boldsymbol{X}_{\mathrm{fb}[t]}) + \sigma_n^2 I$ (which only needs to be done once whenever new feedback arrives) – the calculation of the posterior mean $\mu_{\mathrm{fb}[t]}(\boldsymbol{x})$ takes $O(t)$ additions and multiplications. On the other hand, σ^2_{t-1} must be recomputed for every $\boldsymbol{x} \in D$ after every round, and requires solving backsubstitution, which requires $O(t^2)$ computations. For large decision sets D, the variance computation thus dominates the computational cost of GP-BUCB.

Fortunately, for any fixed decision $\boldsymbol{x}$, $\sigma^2_t(\boldsymbol{x})$ is non-increasing in t. This fact can be exploited to dramatically improve the running time of GP-BUCB, at least for decision sets D which are finitely discretized or are themselves finite. The key idea is that instead of recomputing $\sigma_{t-1}(\boldsymbol{x})$ for all decisions $\boldsymbol{x}$ in every round t, we can maintain an upper bound $\widehat{\sigma}_{t-1}(\boldsymbol{x})$, initialized to $\widehat{\sigma}_0(\boldsymbol{x}) = \infty$. In every round, we lazily apply the GP-BUCB rule with this upper bound to identify

$$\boldsymbol{x}_t = \operatorname*{argmax}_{\boldsymbol{x} \in D} \left[\mu_{\mathrm{fb}[t]}(\boldsymbol{x}) + \beta_t^{1/2} \widehat{\sigma}_{t-1}(\boldsymbol{x}) \right]. \tag{3.18}$$

We then recompute $\widehat{\sigma}_{t-1}(\boldsymbol{x}_t) \leftarrow \sigma_{t-1}(\boldsymbol{x}_t)$. If $\boldsymbol{x}_t$ still lies in the argmax of Equation (3.18), we have identified the next decision to make, and set $\widehat{\sigma}_t(\boldsymbol{x}) = \widehat{\sigma}_{t-1}(\boldsymbol{x})$ for all remaining decisions $\boldsymbol{x}$. Minoux (1978) proposed a similar technique, concerning calculating the greedy action for submodular maximization, which the above procedure generalizes to the bandit setting. A similar idea was also employed by Krause et al. (2008) in the Gaussian process setting for experimental design. The lazy variance calculation method leads to dramatically improved empirical computational speed, discussed in Section 3.6.

Locally stopped algorithms (Section 3.4.2) may have stopping conditions which are dependent on the uncertainty at every $\boldsymbol{x} \in D$, but they also benefit from lazy variance calculations. Since the global conditional information gain bounds the local information gain for all $\boldsymbol{x} \in D$, as in Inequality

(3.11), we obtain the implication

$$I(f; \boldsymbol{y}_{\mathrm{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\mathrm{fb}[t]}) \leq C \implies \nexists \boldsymbol{x} \in D : I(f(\boldsymbol{x}); \boldsymbol{y}_{\mathrm{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\mathrm{fb}[t]}) > C$$

that is, that until the stopping condition for GP-AUCB is met, the stopping condition for GP-AUCB Local is also not met, and thus no local calculations need be made. In implementing GP-AUCB Local, we may run what is effectively lazy GP-AUCB until the global stopping condition is met, at which time we transition to GP-AUCB Local. For a fixed maximum batch size B_{max}, it is often the case that local variance calculations become only very rarely necessary after the first few batches.

3.6 Computational Experiments

We compare GP-BUCB with several alternatives: (1) The strictly sequential GP-UCB algorithm ($B = 1$) receiving feedback from each action without batching or delay; (2) Two versions of a state of the art algorithm for Batch Bayesian optimization proposed by Azimi et al. (2010), which can use either a UCB or Maximum Expected Improvement (MEI) decision rule, herein SM-UCB and SM-MEI respectively. Similarly, we compare GP-AUCB against two other adaptive algorithms: (1) HBBO, proposed by Azimi et al. (2012b), which checks an expected prediction error stopping condition and makes decisions using either an MEI or a UCB decision rule; and (2) GP-AUCB Local, a local information gain-checking adaptive algorithm described briefly in Section 3.4.2. We also present some experimental comparisons across these two sets of algorithms.

In Section 3.6.1, we describe the computational experiments which were performed in more detail. Each of the described computational experiments was performed for each data set. These data sets and the corresponding experimental results are presented in Section 3.6.2. Results of the computational time comparisons are reserved to Section 3.6.3. We also briefly highlight the tradeoffs inherent in adaptive parallelism in Section 3.6.4.

3.6.1 Experimental Comparisons

We performed a number of different experiments using this set of algorithms; (1) A simple experiment in the batch case, in which the non-adaptive batch length algorithms are compared against one another, using a single batch length of $B = 5$ (Figure 3.2); (2) A corresponding experiment in the delay case, comparing GP-UCB, GP-BUCB, GP-AUCB, and GP-AUCB Local against one another, using a delay of $B = 5$ (Figure 3.3); (3) An experiment examining how changes in the batch length over the range $B = 5, 10$, and 20 affect performance of the non-adaptive algorithms (Figure 3.4), and a similar experiment where the maximum batch lengths for the adaptive algorithms vary over the same values (Figure 3.5); (4) A corresponding experiment in the delay setting, examining how varying the delay length over the set $5, 10$, and 20 affects algorithm performance (Figure 3.6); and (5) an experiment which examines execution time for various algorithms in the batch case, comparing basic and lazy versions (see Section 3.5) of the algorithms presented (Figure 3.7). All experiments were performed in MATLAB using custom code, which we make publicly available for use. [1]

Comparisons of reward and regret among the algorithms discussed above are presented in terms of their cumulative regret, as well as their simple regret (how close did the points considered ever get to the maximum function value). Execution time comparisons are performed using wall-clock time elapsed since the beginning of the experiment, recorded at ends of algorithmic timesteps. All experiments were repeated for 200 trials, with independent tie-breaking and observation noise for each trial. Additionally, in those experimental cases where the reward function was a draw from a GP (the SE and Matérn problems), each trial used an independent draw from the same GP.

In the theoretical analysis in Section 3.3, the crucial elements in proving the regret bounds of GP-BUCB and GP-AUCB are C, the bound on the information which can be hallucinated within a batch and β_t, the exploration-exploitation tradeoff parameter, which is set with reference to C to ensure confidence interval containment of the reward function. For practical purposes, it is often necessary to define β_t and the corresponding parameter of GP-UCB, α_t, in a fashion which makes the algorithm considerably more aggressive than the regret bound requires. This removes the high-probability guarantees in the regret bound, but often produces excellent empirical performance. On the other hand, leaving the values for α_t and β_t as would be indicated by the theory results in heavily exploratory behavior and very little exploitation. In this chapter, in all algorithms which use the UCB or BUCB decision rules, the value of α_t has been set such that it has a small premultiplier (0.05 or 0.1, see Table 3.2), yielding substantially smaller values for α_t. Further, despite the rigors of analysis explored above in Section 3.3, we choose to set $\beta_t = \alpha_{\text{fb}[t]}$ for the batch and delay algorithms, without reference to the value of C or the batch length B. Taking either of these measures removes the guarantees of correctness as carefully crafted in Section 3.3. However, as is verified by the experiments comparing batch sizes, this is not a substantial detriment to performance, even for large batch sizes, and indeed, the batch algorithms remain generally quite competitive with the sequential GP-UCB algorithm. One experimental advantage of this approach is that (with some limitations necessitated by the adaptive algorithms) the various algorithms using a UCB decision rule are using the same exploration-exploitation tradeoff parameter at the same iteration, including GP-UCB, GP-BUCB, GP-AUCB, and even SM-UCB and HBBO when using the UCB decision rule. This choice enables us to remove a confounding factor in comparing how well the algorithms overcome the disadvantages inherent in the batch and delay settings.

[1]See Appendix E.

3.6.2 Data Sets

We empirically evaluate GP-BUCB and GP-AUCB on several synthetic benchmark problems as well as two real applications. For each of the experimental data sets used in this chapter, the kernel functions and experimental constants are listed in Table 3.2. Where applicable, the covariance function from the GPML toolbox (Rasmussen and Nickisch, 2010) used is also listed by name. For all experiments, $\delta = 0.1$ (see Theorem 1) for UCB-based algorithms and tolerance $\epsilon = 0.02$ for HBBO. Each of the experiments discussed above was performed for each of the data sets described below and their results are presented, organized by experimental comparison (e.g., delay, adaptive batch size, etc.), in the accompanying Figures.

Problem Setting	Kernel Function	Hyperparameters	Noise Variance σ_n^2	Premultiplier (on α_t, β_t)
Matérn	covMaterniso	$l = 0.1$, $\sigma^2 = 0.5$	0.0250	0.1
SE	covSEiso	$l = 0.2$, $\sigma^2 = 0.5$	0.0250	0.1
Rosenbrock	RBF	$l^2 = 0.1$, $\sigma^2 = 1$	0.01	0.1
Cosines	RBF	$l^2 = 0.03$, $\sigma^2 = 1$	0.01	0.1
Vaccine	covLINone	$t_2 = 0.8974$	1.1534	0.05
SCI	covSEard	$l = [0.988, 1.5337, 1.0051, 1.5868]$, $\sigma^2 = 1.0384$	0.0463	0.1

Table 3.2: Experimental kernel functions and parameters.

3.6.2.1 Synthetic Benchmark Problems

We first test GP-BUCB and GP-AUCB in conditions where the true prior is known. A set of 100 example functions was drawn from a zero-mean GP with Matérn kernel over the interval $[0, 1]$. The kernel, its parameters, and the noise variance were known to each algorithm. The decision set D was the discretization of $[0, 1]$ into 1000 evenly spaced points. These experiments were also repeated with a Squared-Exponential kernel. Broadly speaking, these two problems were quite easy; the functions were fairly smooth, and for all algorithms considered, the optimum was found nearly every time, even for long batch sizes or delay lengths. Even for long batch lengths, as in Figures 3.4(a) and 3.4(b), which show substantial disadvantages in the average regret plots, the first batch has essentially all of the information needed to find the optimum, such that minimum regret after receiving the observations in the first batch is essentially zero. Similar sorts of results are present in the delay length experiments, where the adaptive algorithms which balk at spending queries early are able to do very well after receiving only a very few observations.

The Rosenbrock and Cosines test functions used by Azimi et al. (2010) were also considered, using the same Squared Exponential kernel as employed in their experiments, though with somewhat different lengthscales. The Rosenbrock test function shows a very strong skew toward actions near the upper end of the reward range, such that the minimum regret is often nearly zero before the first feedback is obtained. Since under conditions of the same batch length, most algorithms perform very comparably in terms of both average and minimum regret, the most interesting results are in the case concerning delay length changes, Figure 3.6(c). In this figure, it is possible to see that GP-AUCB balks too often in this setting, leading to substantial losses in performance relative to GP-AUCB Local and GP-BUCB. The Cosines test function also shows broadly similar results across specific problem instances, where there is not a tremendous spread amongst the algorithms tested. Because the Cosines function is multi-modal, the average regret seems to show two-phase convergence behavior, in which all algorithms may be converging to a local optimum and then subsequently finding the global optimum. The overly frequent balking by GP-AUCB present in the Rosenbrock test function is also present for longer delays in the Cosines function, as can be seen in 3.6(g).

In both the Rosenbrock and Cosines delay experiments, one reason this behavior may occur has to do with the kernel chosen and how this interacts with the stopping condition, which requires that the information gain (either with respect to the reward function f as a whole or with respect to $f(\boldsymbol{x})$, $\forall \boldsymbol{x} \in D$) be less than a chosen constant C. With a flat prior, both GP-AUCB and GP-AUCB Local initially behave like Greedy Maximum Entropy Sampling (GMES). Since GMES gains a great deal of information globally, GP-AUCB tends to balk; on the other hand, since GMES scatters queries widely, the information gained with respect to any individual reward $f(\boldsymbol{x})$ is small, and so GP-AUCB Local tends not to balk much or at all. Since the information gain is calculated using the kernel used by the algorithm to model the function, misspecification of this kernel's longer-ranged properties may be particularly problematic for GP-AUCB, as opposed to GP-AUCB Local, which is (initially) more dependent on the local properties of the kernel and the assumed noise.

3.6.2.2 Automated Vaccine Design

We also tested GP-BUCB and GP-AUCB on a database of Widmer et al. (2010), as considered for experimental design by Krause and Ong (2011). This database describes the binding affinity of various peptides with a Major Histocompatibility Complex (MHC) Class I molecule, of importance when designing vaccines to exploit peptide binding properties. Each of the peptides which bound with the MHC molecule is described by a set of chemical features in $\mathbb{R}^{45}$, where each dimension corresponds to a chemical feature of the peptide. The binding affinity of each peptide, which is treated as the reward or payoff, is described as an offset IC_{50} value. The experiments used an isotropic linear kernel fitted on a different MHC molecule from the same data set. Since the data describe a phenomenon which has a measurable limit, many members of the data set are optimal; out of 3089 elements of D, 124, or about 4%, are in the maximizing set. In the simple batch experiments, Figures 3.2(h) and 3.2(k), GP-BUCB performs competitively with SM-MEI and SM-UCB, both in terms of average and minimum regret, and converges to the performance of GP-UCB. In the simple delay setting, Figures 3.3(h) and 3.3(k), both GP-BUCB and GP-AUCB produce superior minimum regret curves to that of GP-UCB, while performing comparably in terms of long-run average regret; this indicates that the more thorough initialization of GP-AUCB and GP-BUCB versus GP-UCB may enable them to avoid early commitment to the wrong local optimum, finding a member of the maximizing set more consistently. This is consistent with the results of the non-adaptive batch size comparison experiment, Figures 3.4(h) and 3.4(k), which shows that as the batch size B grows, the algorithm must pay more "up front" due to its more enduring ignorance, but also tends to avoid missing the optimal set entirely. This same sort of tradeoff of average regret against minimum regret is clearly visible for the GP-AUCB Local variants in the experiments sweeping maximal batch size for adaptive algorithms, Figures 3.5(h) and 3.5(k).

3.6.2.3 Spinal Cord Injury (SCI) Therapy

Lastly, we compare the algorithms on a pre-recorded data set of leg muscle activity triggered by therapeutic spinal electrostimulation in spinal cord injured rats. This setting is intended to mimic the on-line experiments conducted in Chapter 4. Much greater detail is given regarding the experimental design in that chapter, but, in brief, the procedure is as follows. From the 3-by-9 grid of electrodes on the array, a pair of electrodes is chosen to activate, with the first element of the pair used as the cathode and the second used as the anode. Electrode configurations were represented in $\mathbb{R}^4$ by the cathode and anode locations on the array. These active array electrodes create an electric field which may influence both incoming sensory information in dorsal root processes and the function of interneurons within the spinal cord, but the precise mechanism of action is poorly understood. Since the goal of this therapy is to improve the motor control functions of the lower spinal cord, the designated experimental objective is to choose the stimulus electrodes which maximize the resulting activity in lower limb muscles, as measured by electromyography (EMG). We used data with a stimulus amplitude of 5V and sought to maximize the peak-to-peak amplitude of the recorded EMG waveforms from the right medial gastrocnemius muscle in a time window corresponding to a single interneuronal delay. Note that in Chapter 4, the muscle chosen is the left tibialis anterior, but the procedure is fundamentally the same. This objective function attempts to measure the degree to which the selected stimulus activates interneurons controlling reflex activity in the spinal gray matter. This response signal is non-negative and for physical reasons does not generally rise above 3mV. A squared-exponential ARD kernel was fitted using experimental data from 12 days post-injury. Algorithm testing was done using an oracle composed of data from 116 electrode pairs tested on the 14th day post-injury.

Like the Vaccine data set, the SCI data set displayed a number of behaviors which indicate that the problem instance was difficult; in particular, the same tendency that algorithms which initialized more thoroughly would eventually do better in both minimum and average regret was observed. This tendency is visible in the simple batch setting (Figures 3.2(i) and 3.2(l)), where GP-UCB was not clearly superior to either GP-BUCB or GP-AUCB; this is surprising because being required to work in batches, rather than one query at a time, might be expected to give the algorithm less information at any given round, and should thus be a disadvantage; this under-exploration in GP-UCB may be a result of the value of the exploration-exploitation tradeoff parameter α being chosen to promote greater aggressiveness across all algorithms. Interestingly, this data set also displays both a small gap between the best and second-best values of the reward function (approximately 0.9% of the range) and a large gap between the best and third-best (approximately 7% of the range). When examining how many out of the individual experimental runs simulated selected $\boldsymbol{x}^* = \operatorname{argmax}_{\boldsymbol{x} \in D} f(\boldsymbol{x})$ on the 200th query in the simple batch case, only 20% of GP-UCB runs choose $\boldsymbol{x}^*$; the numbers are considerably better for GP-BUCB, SM-UCB, and SM-MEI, at 35%, 30.5%, and 36%, but are still not particularly good. If the first sub-optimal $\boldsymbol{x}$ is also included, these numbers improve substantially, to 63.5% for GP-UCB and 84%, 91%, and 96.5% for GP-BUCB, SM-UCB, and SM-MEI. These results indicate that the second-most optimal $\boldsymbol{x}$ is actually easier to find than the most optimal, to a fairly substantial degree. It is also important to place these results in the context of the experimental setting; even assuming that there truly is a difference between these pairs of SCI electrodes, the rewards produced are so close to one another as to likely produce no therapeutic difference between the most optimal and second-most optimal actions. Since all of GP-BUCB, SM-UCB, and SM-MEI more consistently found one of the two best actions in the decision set than GP-UCB, all of them showed strong performance in comparison to GP-UCB.

3.6.3 Computational Performance

Another test of interest across the set of experiments discussed above was the degree to which lazy variance calculations, as described in Section 3.5, reduced the computational overhead of each of the algorithms discussed. These results are presented in Table B.6 and Figure 3.7. Note that for algorithms which appear as both lazy and non-lazy versions, the only functional difference between the two is the procedure by which the action is selected, not the action selection itself; all computational gains are without sacrificing accuracy and without requiring any algorithmic approximations. All computational time experiments were performed on a desktop computer (quad-core Intel i7, 2.8 GHz, 8 GB RAM, Ubuntu 10.04) running a single MATLAB R2012a process.

For all data sets, the algorithms lie in three broad classes: Class 1, comprised of the lazy GP-UCB family of algorithms; Class 2, the non-lazy versions of the GP-UCB family of algorithms, as well as the HBBO UCB and MEI variants; Class 3, consisting of the SM-MEI and SM-UCB algorithms, in both lazy and non-lazy versions. Class 1 algorithms run to completion about one order of magnitude faster than those in Class 2, which in turn are about one order of magnitude faster than those in Class 3. The various versions of the simulation matching algorithm of Azimi et al. (2010), require multiple samples from the posterior over f to aggregate together into a batch, the composition of which is intended to match or cover the performance of the corresponding sequential UCB or MEI algorithm. The time difference between Class 2 and Class 3, approximately one order of magnitude, reflects the choice to run 10 such samples. Within Class 3, our implementation of the lazy version of SM-MEI is slower than the non-lazy version, largely due to the increased overhead of sorting the decision rule and computing single values of the variance; a more efficient implementation of either or both of these elements could perhaps improve on this tradeoff. The lazy algorithms also tend to expend a large amount of computational time early, computing upper bounds on later uncertainties, but tend to make up for this early investment later; this is even visible with regard to the lazy version of SM-UCB, which is initially slower than the non-lazy version, but scales better and, in all six data sets examined, ends up costing substantially less computational time by the 200th query.

3.6.4 Parallelism: Costs and Tradeoffs

Parallelism is motivated by the setting in which each round or opportunity to submit a query is expensive, but the additional marginal cost of taking an action at that round is not very large. It is interesting to consider more precisely what we mean by "expensive" or "not very large." For a given relationship between the individual costs, one can examine which algorithms most effectively trade these costs off against one another. One way to do this is to experimentally measure the costs incurred by several algorithms solving the same problem. In the following discussion, we examine the delay case, in which the algorithm is faced with a choice of which action (if any) to take at each round. A similar examination of cost tradeoffs can be made in the batch case.

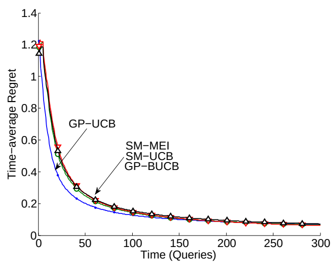
Given N sample runs, a successful algorithm should have a (nearly) monotonically decreasing sample average regret curve, defined as $\bar{r}(T) = \sum_{n=1}^{N} R_{T,n}/T$, where $R_{T,n}$ is the cumulative regret of the nth run at round T. This curve can be inverted to find the first round $\tau(\bar{r})$ in which the average regret is at or below a particular value $\bar{r}$. The sample mean cost of running the algorithm until round $\tau(\bar{r})$ can then be computed. The cost of the nth run is the sum of two contributions, the first for running $\tau(\bar{r})$ rounds of the algorithm, and the second for the actual execution of $a_n(\tau(\bar{r}))$ actions. Parameterizing the relative costs of each round and each action using w, the average cost $C(\bar{r}, w) = (1-w)\tau(\bar{r}) + w \cdot \bar{a}(\tau(\bar{r}))$ corresponding to a particular average regret value $\bar{r}$ can be obtained, where $\bar{a}(\tau(\bar{r})) = \sum_{n=1}^{N} a_n(\tau(\bar{r}))$. Note that $w \in [0,1]$ translates to any constant, non-negative ratio of the cost of a single action to that of a single round. As a technical point, note that this average cost is not exactly equivalent to specifying $\bar{r}$, continuing each individual run until $R_{T,n}/T \leq \bar{r}$, and averaging the individual costs incurred in so doing; such a method, while more intuitive, must deal with the problem that a run may fail to ever attain a specified value of $\bar{r}$, e.g., a run could fail to converge to the optimum. This failure to converge happens with a non-zero probability, and is theoretically treated in Theorem 1 through δ. The post-hoc calculation of $C(\bar{r}, w)$ as proposed here is robust to this case, giving an estimate of the expected cost to run the algorithm until a round in which the expected cumulative average regret is below $\bar{r}$.

Among a set of algorithms, and given a test problem, one can find which among them has the lowest value of $C(\bar{r}, w)$ at any particular point in the $\bar{r}, w$ space. Similarly, for any fixed value of w, it is possible to once more invert the function and plot $\bar{r}_w(C)$; this plot resembles conventional average regret plots, and corresponds to intersection of the set of $C(\bar{r}, w)$ surfaces with the plane at a fixed w. A comparison between three algorithms on the SCI data set, with simple delay $B = 5$, is shown in Figure 3.8. In this scenario, GP-AUCB costs the least through most of the parameter space, due to its tendency to pass in early rounds, when the potential for exploitation is lowest. The advantage changes to the fully sequential algorithm when w is large (i.e., parallelism is expensive), and to GP-BUCB when w is small. Many real-world situations lie somewhere between these extremes, suggesting that GP-AUCB may be useful in a variety of scenarios.

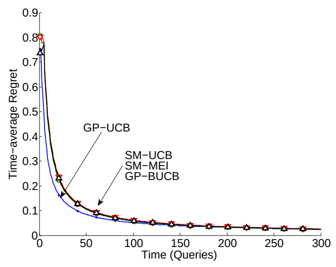
3.7 Conclusions

We develop the GP-BUCB and GP-AUCB algorithms for parallelizing exploration-exploitation tradeoffs in Gaussian process bandit optimization. The analytical framework used to bound the regret of GP-BUCB and GP-AUCB is generalized to include all GP-UCB-type algorithms. We prove Theorem 1, which provides high-probability bounds on the cumulative regret of algorithms in this class, which hold for both the batch and delay setting. These bounds consequently provide guarantees on the convergence of such algorithms. Further, we prove Theorem 4, which establishes a high-probability regret bound for initialized GP-BUCB. This bound scales independently of the batch size or delay length B, if B is constant or polylogarithmic in T. Finally, we introduce lazy variance calculations, which dramatically accelerate computation of GP-based active learning decision rules.

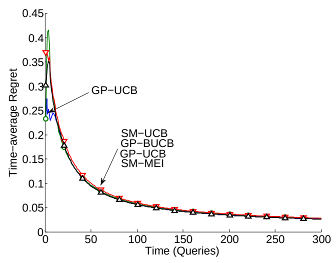
Across the experimental settings examined, GP-BUCB and GP-AUCB performed comparably with state of the art parallel and adaptive parallel Bayesian optimization algorithms, which are not equipped with theoretical bounds on regret. GP-BUCB and GP-AUCB also perform comparably to the sequential GP-UCB algorithm, indicating that GP-BUCB and GP-AUCB successfully overcome the disadvantages of only receiving delayed or batched feedback. With respect to cost to achieve a given level of regret, GP-AUCB appears to offer substantial advantages over the fully parallel or fully sequential approaches. We believe that our results provide an important step towards solving complex, large-scale exploration-exploitation tradeoffs.



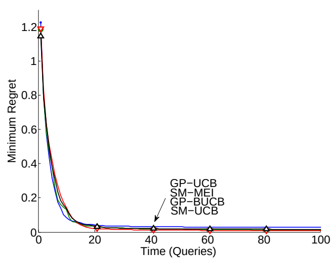
(a) *Matérn: AR*



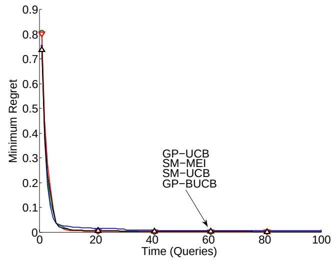
(b) *SE: AR*



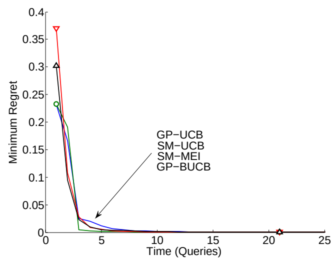
(c) *Rosenbrock: AR*



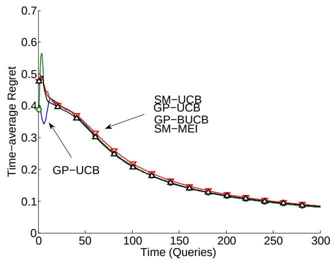
(d) *Matérn: MR*



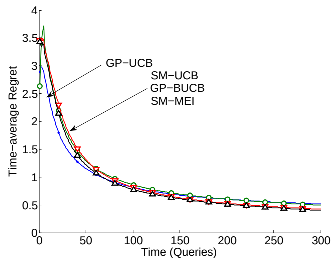
(e) *SE: MR*



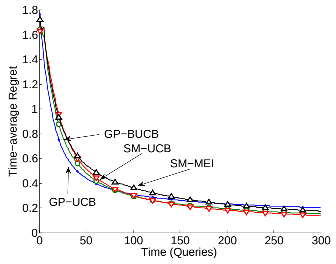
(f) *Rosenbrock: MR*



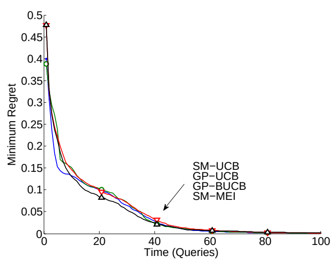
(g) *Cosines: AR*



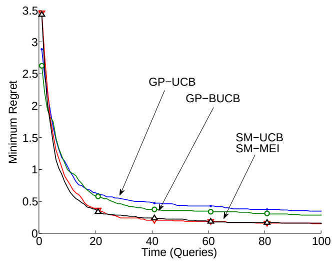
(h) *Vaccine: AR*



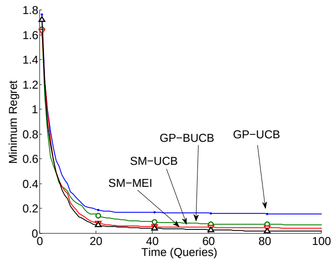
(i) *SCI: AR*



(j) *Cosines: MR*

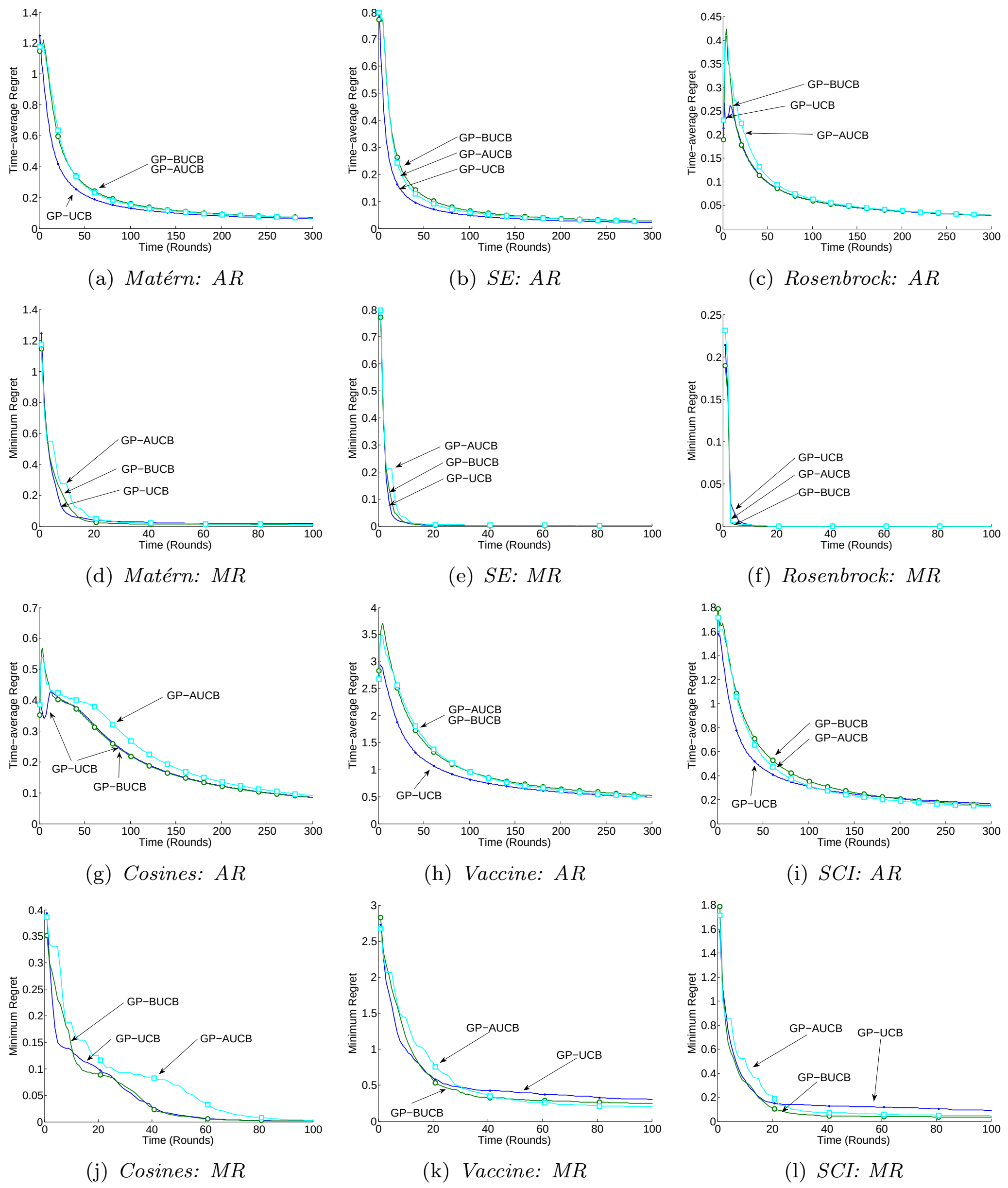


(k) *Vaccine: MR*



(l) *SCI: MR*

Figure 3.2: Time-average (AR) and minimum (MR) regret plots, batch setting, for a batch size of 5.



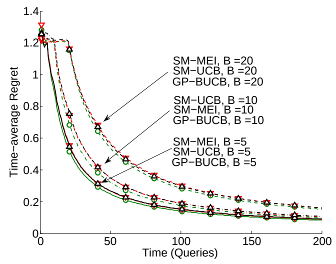
(a) *Matérn: AR* (b) *SE: AR* (c) *Rosenbrock: AR*

(d) *Matérn: MR* (e) *SE: MR* (f) *Rosenbrock: MR*

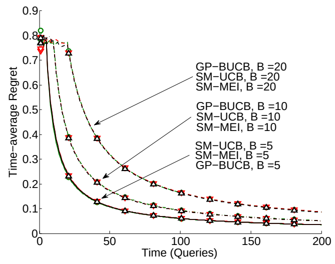
(g) *Cosines: AR* (h) *Vaccine: AR* (i) *SCI: AR*

(j) *Cosines: MR* (k) *Vaccine: MR* (l) *SCI: MR*

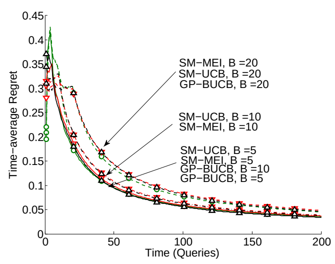
Figure 3.3: Time-average (AR) and minimum (MR) regret plots, delay setting, with a delay length of 5 rounds between action and observation.



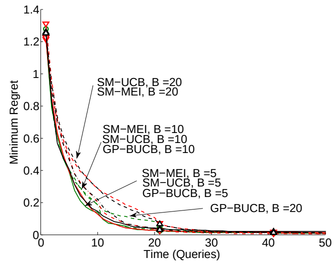
(a) *Matérn: AR*



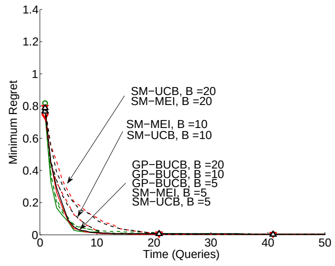
(b) *SE: AR*



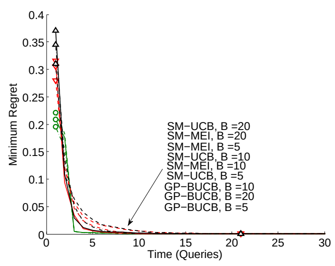
(c) *Rosenbrock: AR*



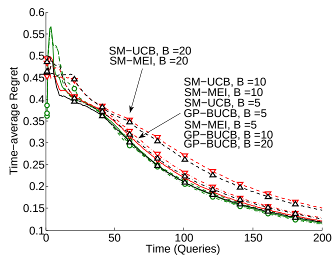
(d) *Matérn: MR*



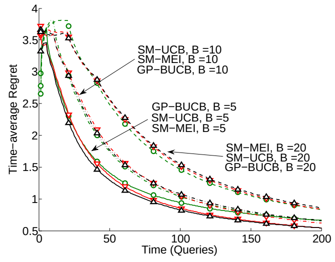
(e) *SE: MR*



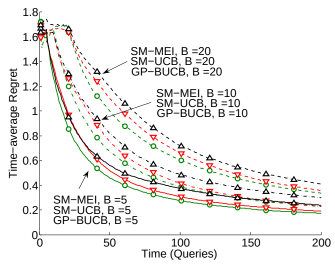
(f) *Rosenbrock: MR*



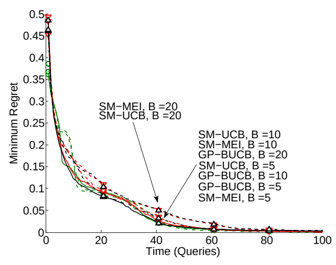
(g) *Cosines: AR*



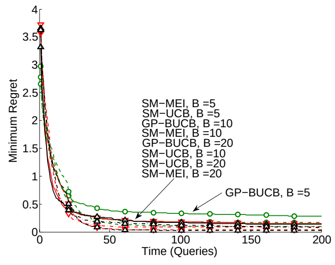
(h) *Vaccine: AR*



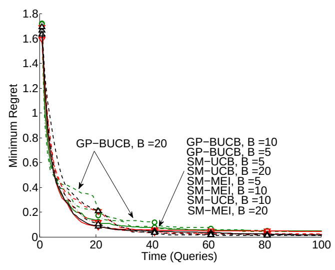
(i) *SCI: AR*



(j) *Cosines: MR*

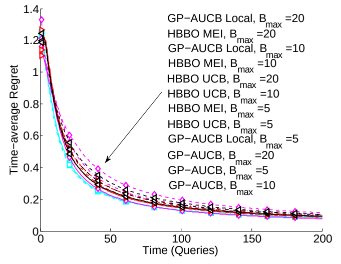


(k) *Vaccine: MR*

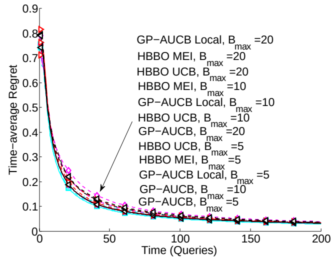


(l) *SCI: MR*

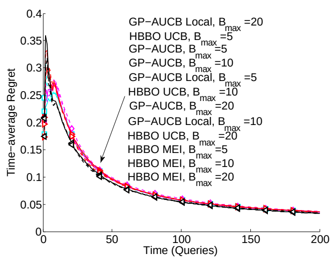
Figure 3.4: Time-average (AR) and minimum (MR) regret plots, non-adaptive batch algorithms, batch sizes 5, 10, and 20.



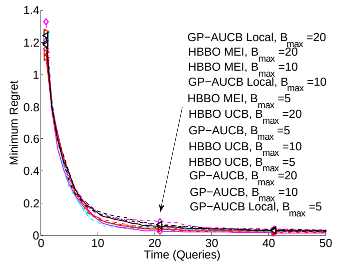
(a) *Matérn: AR*



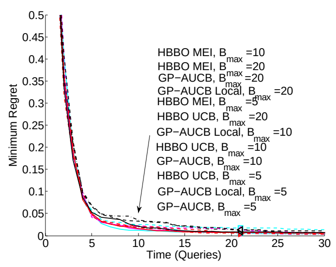
(b) *SE: AR*



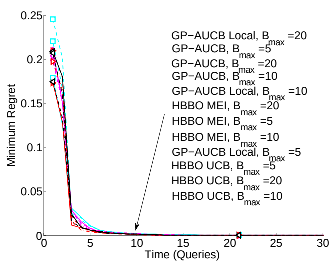
(c) *Rosenbrock: AR*



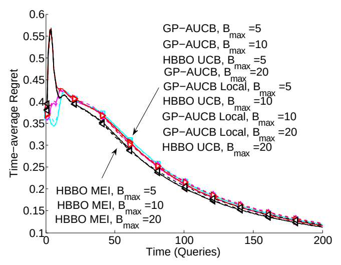
(d) *Matérn: MR*



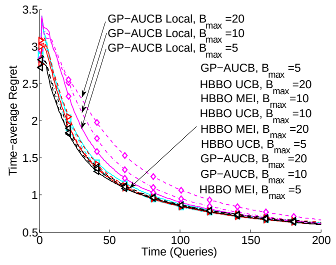
(e) *SE: MR*



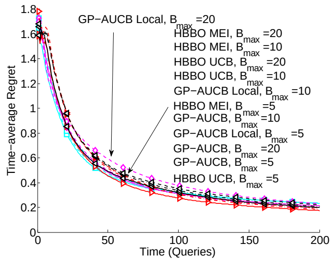
(f) *Rosenbrock: MR*



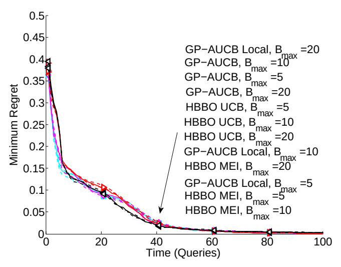
(g) *Cosines: AR*



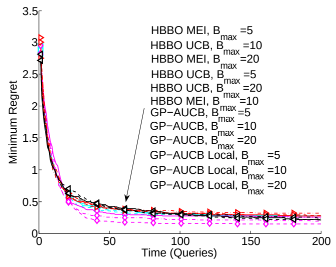
(h) *Vaccine: AR*



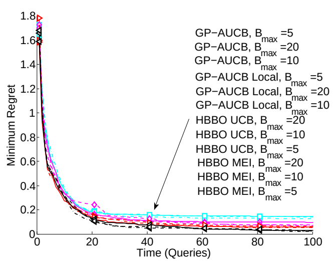
(i) *SCI: AR*



(j) *Cosines: MR*

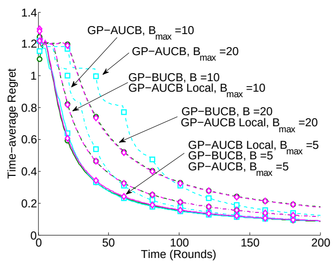


(k) *Vaccine: MR*

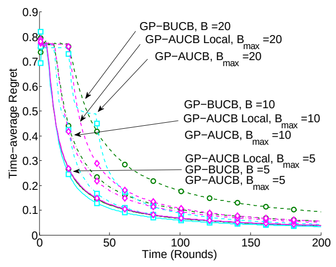


(l) *SCI: MR*

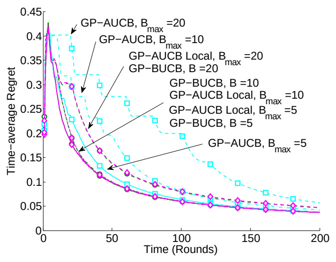
Figure 3.5: Time-average (AR) and minimum (MR) regret plots, adaptive batch algorithms, maximum batch sizes 5, 10, and 20. For the adaptive algorithms, minimum batch size B_{min} was set to 1, as in HBBO. The algorithms tended to run fully sequentially at the beginning, but quite rapidly switched to maximal parallelism.



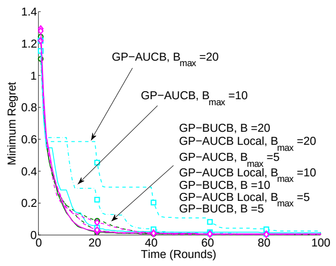
(a) *Matérn: AR*



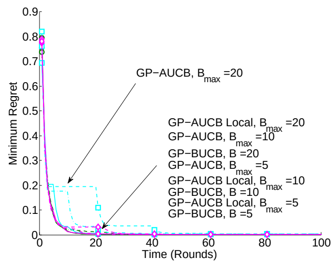
(b) *SE: AR*



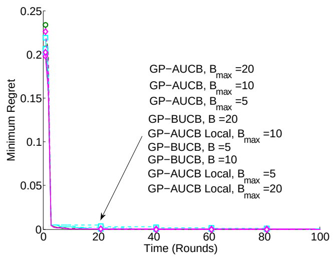
(c) *Rosenbrock: AR*



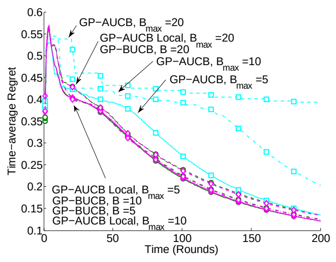
(d) *Matérn: MR*



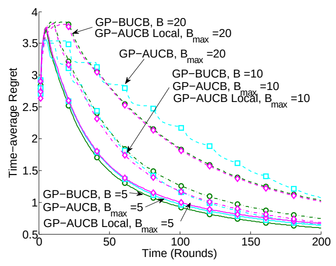
(e) *SE: MR*



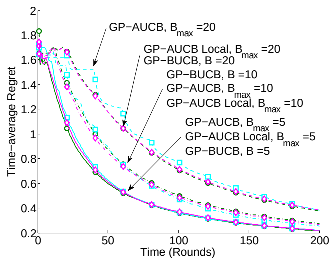
(f) *Rosenbrock: MR*



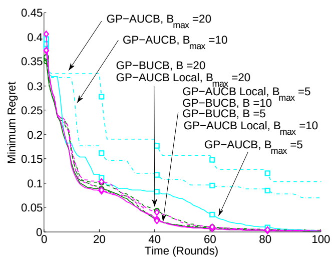
(g) *Cosines: AR*



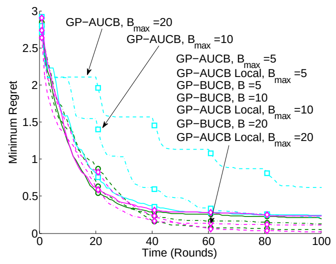
(h) *Vaccine: AR*



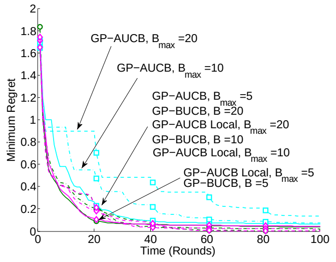
(i) *SCI: AR*



(j) *Cosines: MR*

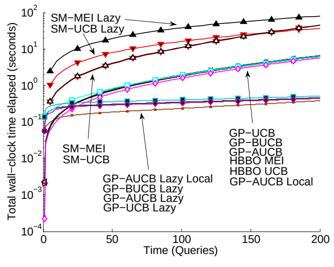


(k) *Vaccine: MR*

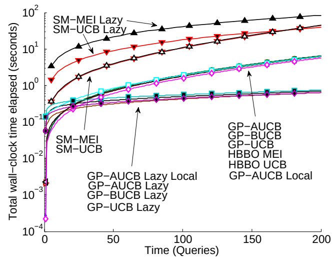


(l) *SCI: MR*

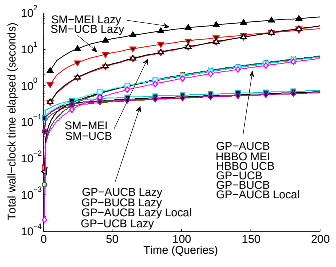
Figure 3.6: Time-average (AR) and minimum (MR) regret plots, delay setting, with delay lengths of 5, 10, and 20 rounds between action and observation. Note that the adaptive algorithms, GP-AUCB and GP-AUCB Local, may balk at some rounds. The time-average regret is calculated over the number of actions actually executed as of that round; this means that the number of queries submitted as of any particular round is hidden with respect to the plots shown, and may vary across runs of the same algorithm.



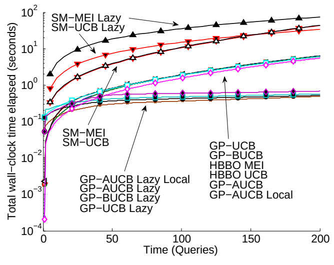
(a) *Matérn*



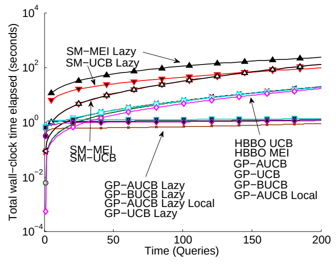
(b) *SE*



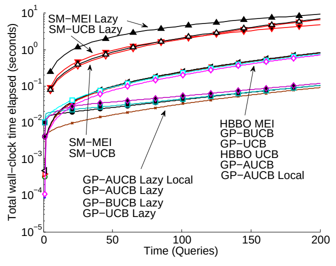
(c) *Rosenbrock*



(d) *Cosines*

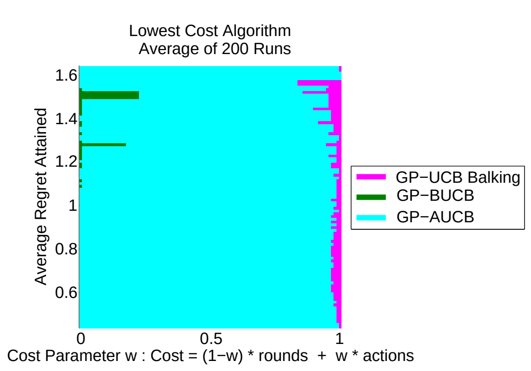


(e) *Vaccine*

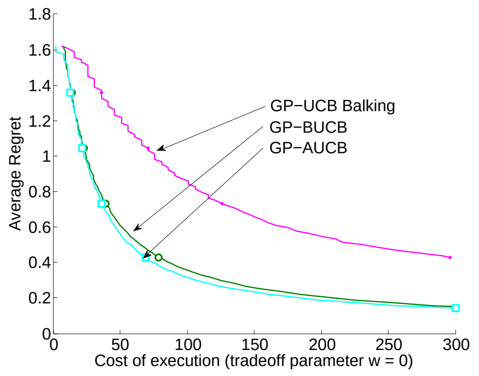


(f) *SCI*

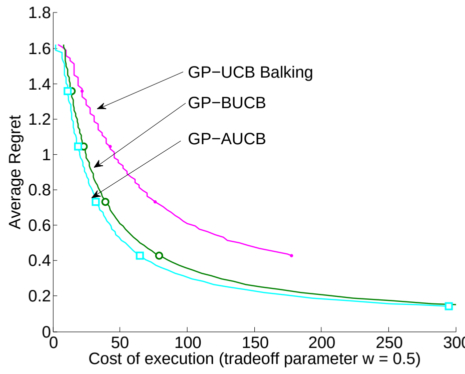
Figure 3.7: Elapsed computational time in batch experiments, B = 5. Note the logarithmic vertical scaling in all plots. Note also the substantial separation between the three groups of algorithms, discussed in Section 3.6.3.



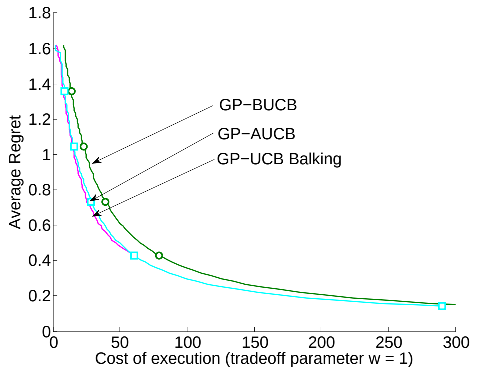
(a) *Lowest-Cost Algorithms*



(b) w = *0*



(c) w = *1/2*



(d) w = *1*

Figure 3.8: Parameterized cost comparison on the SCI data set, simple delay case, $B = 5$. The same experiment, with a different set of algorithms shown, is presented in Figure 3.3(i). Figure 3.8(a): the space of cost tradeoff parameter w and attained average regrets $\bar{r}$ is colored according to which algorithm has the lowest mean cost at the round in which the mean, time-average regret is first $\leq \bar{r}$. The algorithm denoted GP-UCB Balking refuses to submit another query while one is pending, i.e., it is the GP-UCB algorithm obeying the delay constraint of the problem setting. Figures 3.8(b), 3.8(c), and 3.8(d) show $\bar{r}$ as a function of C and correspond to vertical slices through Figure 3.8(a) at the left, center, and right. Since GP-AUCB and GP-UCB Balking pass on some rounds, the terminal cost of GP-AUCB and GP-UCB Balking is possibly < 300.

Chapter 4

Animal Studies

4.1 Introduction

A rigorous and automated method is required to select the stimuli applied by the arrays reviewed in Chapter 1 for SCI therapy. Such a method would allow application of these techniques by non-experts, or even autonomously, and additionally would allow for customization to individual patients and their unique, time-varying responses to the stimuli. This dissertation suggests that a variant of GP-BUCB or a similar GP-based active learning algorithm is suitable for this task.

To show that it would be feasible to use GP-BUCB as a learning system for SCI therapy, a closed-loop implementation of the algorithm in real animals was developed. These experiments represent a first step toward a more complex closed-loop implementation in human patients. A somewhat simplified problem was chosen for demonstrating feasibility. A variant of GP-BUCB was used to control an experiment in which a rat was stimulated using an epidural electrode array and the evoked potential in a muscle was measured via EMG. The goal of this experiment was to maximize the amplitude of the resulting evoked potential. While the evoked potential is not a complex motor behavior, this experiment does have many of the important characteristics of the full SCI therapy problem, in particular that the evoked potential varies with the pattern of active electrodes on the array and over the course of the animal's experimental lifetime, and that evoked potentials are critically dependent on the spinal interneuronal circuitry. Showing that the algorithm can successfully control this activity and additionally learn something about the structure of the spinal cord's responses (considered as a function over the space of active electrode configurations) demonstrates a major step toward a therapeutic implementation.

The simplifications inherent in using evoked potentials present a number of substantial advantages for demonstrating feasibility. First, because the evoked potentials represent a relatively low-level function of the spinal interneuron networks, it is reasonable to suggest that they may be less sensitive to the parameter choices than higher-level motor functions, making the search over stimuli inherently easier, appropriate for a feasibility experiment. Demonstrating that the regression models can indeed capture the important features of an individual muscle response function while using relatively little data means that the responses of single muscles can be modeled effectively using Gaussian processes, plausibly leading to effective models of the high-level behavior based upon combining the predicted responses of multiple, individual muscles. Certainly, it is plausible to create a model which focuses on only high-level phenomena, e.g., user-reported quality of stimuli, but, particularly if physiological monitoring data will form an important component of the response monitoring (desirable in a fully-implanted system), this sort of prediction of high-level quality based upon low-level data is highly desirable. Conversely, if the activity of an individual muscle cannot be effectively captured by a GP, this argues that GPs are inappropriate for modeling the responses of individual muscles and that the system is likely too sensitive to model without exhaustive testing of all potential stimuli (an exponentially large set), suggesting that the full problem is nearly infeasible. The ability to successfully manage a problem like evoked potential optimization using a GP-BUCB-like algorithm is thus a necessary condition for success on the full clinical problem, making this an appropriate first step to applying active learning algorithms to SCI therapy. Second, the EMG recordings arising from a train of stimulus pulses can be temporally separated into the individual responses to each stimulus pulse, meaning that each separate pulse and response may be taken as an individual, independent observation; standing or stepping are much less easily separable into individual "observations." Further, in stepping or standing, it should be expected that successive observations (i.e., blocks of time within a bout, such as strides) would not be independent samples from the same distribution, but instead are highly dependent on their predecessors (e.g., a stumble on stride $n-1$ could reasonably be expected to affect stride n). Third, evoked potentials are naturally expressed as a scalar function of time for each muscle, and easily repeatable scalar measurements (i.e., peak-to-peak amplitude) have already been established for them. In contrast, stepping and standing are complex, high-level behaviors of many muscles, for which no single easily and automatically computed, ordinal measurement has yet been canonically established. There are, however, a variety of measurements which aim to quantify standing performance (Prieto et al., 1996; Santos et al., 2008) or to quantify stepping performance. The latter is generally by either human observation (Basso et al., 1995; Antri et al., 2002) or by automated post-hoc analysis (Fong et al., 2005; Cai et al., 2006). Particularly for measures of locomotor performance based on human-graded observations, it is not clear that the grading scale is ordinal, i.e., while the numerical grade may nominally correspond to quality, these might be more properly thought of as loosely ordered class labels. These label-based grading schemes are often designed to be easy for humans to implement, e.g., using visual features of the stride cycle which are easy to describe semantically, but difficult to describe mathematically, consequently making them very difficult to automate. Further, it is not clear what are "optimal" values of any of these measurements for SCI animals or humans (as opposed to normals), nor is it clear that attaining the nominally optimal value of an individual metric is therapeutically desirable in either the long-term or short-term. It is critical for the practical success of a bandit algorithm that its reward function and the true utility function correspond closely with one another. If this is not the case and the algorithm converges to the maximizer of the reward function, the true therapeutic utility may not be maximized. The creation of a reward metric which is efficiently computable and faithfully matches optima of the therapeutic utility is a highly non-trivial problem in its own right, such that first demonstrating feasibility for an easier criterion is appropriate.

This chapter focuses on epidural electrostimulation (see Section 2.2.3.1), using flexible, parylene-based electrode arrays, along with simpler wired arrays, in active learning experiments in rats. Section 4.2 lays out the experimental procedures followed in this chapter, with description of the parylene arrays in Section 4.2.2 and the wired arrays in Section 4.2.3. The chosen reward metric, quantifying the evoked potential and thus measuring some aspects of the functional conductivity of the interneuronal network in the spinal cord, is discussed in detail in Section 4.3. Necessary modifications to the GP-BUCB algorithm for this experimental setting are discussed in Section 4.4 and particular choices for the covariance and mean functions are described in detail in Section 4.5. Section 4.6 presents the results of the experiments and a discussion of these results follows in Section 4.7. A few concluding remarks appear in Section 4.8.

4.2 Experimental Methods

Several aspects of the experimental preparations bear detailed discussion; in particular, the preparation of the animals themselves (Section 4.2.1), the parylene-based flexible electrode arrays (Section 4.2.2) and wired arrays (Section 4.2.3) used to deliver the stimulation to the animals' spinal cords, and the basic testing procedures (Section 4.2.4) will all be examined carefully.

4.2.1 Injury, Implantation, and Animal Care

The description below of the surgical and care procedures used in these experiments is largely derived from Gad et al. (2013). These procedures are similar to those developed by the same laboratory for cats and extensively detailed by Roy et al. (1992). All animals used in these studies are adult female Sprague Dawley rats, and approximately 300g in mass at time of implantation. The following procedures are performed on each animal, typically in a single surgery:

1. Partial laminectomy at the T8-T9 vertebral level and complete spinal transection at the T8 spinal segment, including the dura, via microscissors;

2. Placement of gel foam at the site of the transection as a coagulant and separator of the cut ends of the spinal cord;

3. Partial or full laminectomies of some vertebrae (T11, T12, L3, and L4 for animals receiving the parylene arrays, T12, T13, L1, and L2 for those receiving the wired arrays);

4. Implantation of the epidural electrostimulating array (see Section 4.2.3 and Section 4.2.2), inserted using the T11 and L4 laminectomies for the parylene array or the T12 and L2 laminectomies for the wired array, positioned such that the most rostral electrodes are placed in the middle of the T12 vertebral level, and sutured in place to the dura at both the rostral and caudal ends using 8-0 Ethilon sutures;

5. Implantation of one or two ground wires (each composed of 5 braided 0.003 cm gold wires, A-M Systems, Sequim, WA) in the parylene array animals, or one stainless steel, teflon coated wire in the wired array animals (0.304 mm, AS 632, Cooner Wire, Chatsworth, CA). These are placed in the mass of muscles dorsal to the spinal column;

6. Implantation of multi-stranded, Teflon-coated stainless steel EMG wires (AS 632, Cooner Wire) into the bellies of multiple leg muscles, typically left and right tibialis anterior (TA) and left and right soleus (Sol);

7. Attachment of one or two headplug connectors (Amphenol, Wallingford, CT), as required, screwed to the skull and additionally supported with dental cement.

All surgeries are performed under aseptic conditions and with general anesthesia (Isoflurane) delivered via face mask. Analgesia is also provided with buprenex (0.5-1.0 mg/kg, 3 times per day subcutaneously), begun before the end of surgery and continued at least 2 days post-operative. The animals were also treated with Baytril (an antibiotic), administered sub-cutaneously at the end of surgery and at 12 hour intervals thereafter for at least 3 days. The animals recover from anesthesia within incubators and are individually housed both preoperatively and postoperatively, with free access to food and water. The recovery period lasts for one week after surgery (day 7 post-operative, denoted P7), at which point experiments begin. Experiments continue as long as the animal and array both remain viable, or until 6 weeks post-operative (P42). Due to their spinal cord injuries, the animals' bladders must be manually expressed 2-3 times per day and their hind limbs must be manually moved through their range of motion once per day in order to retain joint mobility.

All procedures were in accordance with the National Institutes of Health Guide for the Care and Use of Laboratory Animals and were approved by the Animal Research Committee at UCLA.

4.2.2 Parylene Arrays

The parylene-based microstimulating stimulating array is fabricated partially by MEMS techniques and partially using traditional microelectronics in the laboratory of Dr. Yu-Chong Tai of the California Institute of Technology (see Nandra et al., 2011; Gad et al., 2013). The MEMS portion of

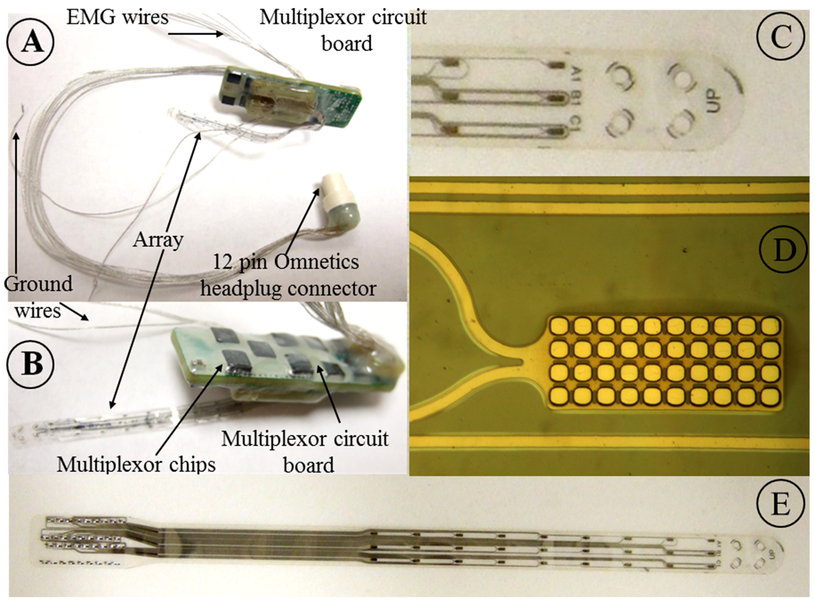

A
EMG wires
Multiplexor circuit board
Array
12 pin Omnetics headplug connector
Ground wires
B
Multiplexor chips
Multiplexor circuit board
C
UP
D
E

Figure 4.1: Parylene Array Device. (A) and (B): The complete implant, including the main circuit board, parylene electrode array, head plug, EMG wires, and ground wires. (C): Detail of the tip of the parylene array. (D): Detail of a single electrode. The bright, roughly square regions on the surface of the electrode are the open spaces allowing contact with the body; the darker regions defining these squares are the surface layer of parylene, constructed so as to prevent delamination. (E): View of the array, the entire parylene and platinum microfabricated portion of the implant. Figure reproduced under Open Access from Gad et al. (2013).

this device consists of a set of platinum electrodes and traces, embedded in a parylene C matrix. The array device is pictured in Figure 4.1. This construction is sized to the spinal cord of a rat (the parylene and platinum section is 59 mm x 3mm, while the circuit board, mounted dorsally on the spinal column, is 33.2 mm x 10.3 mm), is highly biocompatible, and allows the array to conform to the spinal cord as the animal moves. This last capability is important because, to deliver the same effective stimulus in any of a variety of body positions, the array must maintain roughly the same relative position to the spinal cord, i.e., conform to its movements. The device carries 27 electrodes, each 0.2 mm x 0.5 mm, partially covered by a criss-crossed pattern of parylene to prevent delamination. These electrodes are organized into three rostro-caudal columns, labeled “A” on the animal’s left side, “B” on the midline, and “C” on the right. Each column is numbered from 1 to 9 moving caudally; the electrode in the 1 position is at the L2 spinal cord level (T12 vertebral level), the electrode in the 9 position is at the S2 spinal cord level (L2 vertebral level), and the remainder are equally spaced in between. The placement of the array is shown diagrammatically in Figure 4.2. The array is coupled to an implanted circuit board which controls the stimulus, records responses, and communicates with external circuitry through a headplug. A given stimulus is specified by the pairing of a single cathode and single anode from among 29 possible electrodes, the 27 on the array and 2 grounds located distally within the body. Considering only bipolar configurations (i.e.,

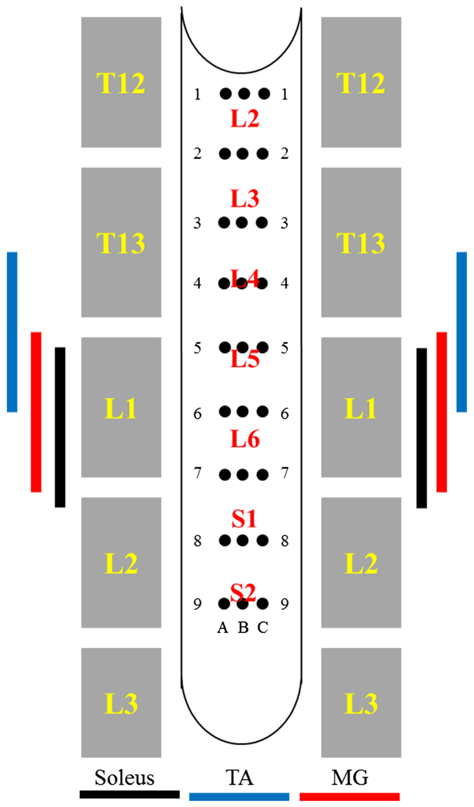


Figure 4.2: Placement of the array device relative to the spinal cord. The gray blocks represent the vertebrae, and are labeled by their conventional numbering scheme. The smaller red text laid over the array denotes the spinal segmental level lying underneath that portion of the array. The colored bars at the far left and right represent the spinal segmental locations of the motor pools of three distal leg muscles, the soleus, medial gastrocnemius, and the tibialis anterior. Figure reproduced under Open Access from Gad et al. (2013).

configurations in which both the cathode and the anode are on the epidural array), there are 702 possible stimulus pairs. Due to the design of the implanted circuit board, 36 of these pairs cannot be stimulated, but the remaining 666 can. The data from two animals tested with this type of array are presented in Section 4.6.2.

4.2.3 Wire-based Spinal Stimulating Arrays

A much simpler array design was also used to test automated stimulus selection. These arrays consist of seven teflon-coated, multi-stranded, stainless steel wires (five are 30 guage, A-M Systems, two are AS 632 wires, Cooner Wire) laid parallel to one another. These wires have openings cut into the insulation at the time of implantation to create exposed electrodes on the surface facing the dura. The wires are then sutured to the dura at the distal end to ensure consistent stimulus location and connected to the headplug on the proximal end. The EMG wires are similarly connected directly to the headplug. This preparation has the significant disadvantage that the number of stimulating electrodes is reduced to seven, placed in locations corresponding to electrodes A1, A4, A9, B2, C1, C4, and C9 on the parylene array. With only seven electrodes, this device has substantially decreased flexibility of stimuli delivered; there are only 42 bipolar configurations which are possible, approximately 6% of the bipolar combinations possible on the parylene array. However, this technology is highly stable in the animals, providing good performance over a long experimental lifetime, something which is not as yet guaranteed with the experimental parylene arrays. The data from two animals tested with this type of array are presented in Section 4.6.1.

4.2.4 Animal Testing Procedures

For testing purposes, the animals are placed in a vest and harness device which supports their body weight. The device is positioned vertically such that the animal is in a bipedal standing position, with both of the hind feet in contact with a custom surface possessing good traction properties. In the closed-loop algorithm experiments, five bipolar pairs of electrodes, possibly with repetition, are selected by the algorithm for each batch. Each of these pairs of electrodes is used to administer a set of stimulus pulses, delivered at 1 Hz. The pulses are delivered in blocks of 10 (or 20, in some animals) individual pulses at each of several voltages. The EMG signals corresponding to evoked potential responses are recorded using an amplifier (A-M Systems) and custom recording software in LabVIEW (National Instruments, Austin, TX), with a sampling rate of 10 kHz on each channel. The data is then processed using custom software in MATLAB (The MathWorks, Inc., Natick, MA) which calculates peak-to-peak amplitudes of the evoked potential within particular delay windows, specified with reference to the onset of the stimulus pulse (see Table 4.2.4). One combination of stimulus voltage and delay window is pre-selected for use by the algorithm in making decisions, and these evoked potentials are recorded into the algorithm's memory. This cycle repeats, where at each opportunity for the algorithm to request experiments, the algorithm's accumulated memory is used to select the five actions constituting the next batch. On a typical testing day, five such batches are selected. A period of the algorithm's conserved memory of experimental observations is denoted a *run*; each run typically lasts several days, and several runs may be performed on the same animal, with the algorithm's memory wiped in between them.

The batch structure allows interleaving batches of algorithm-commanded experiments with batches of human-directed experiments; specifically, our sessions were structured such that the algorithm competed with a human experimenter in alternating batches, but the algorithm and human experimenter were blind to one another's actions and the responses generated. For decision-making purposes, data were not combined from both sources within runs; however, these two sources were combined to make decisions between runs or between animals, e.g., meta-analysis and tuning of hyperparameters.

Response Period	Start (ms)	End (ms)
Early Response	2.0	5.0
Middle Response	4.5	7.5
Late Response	8.5	11

Table 4.1: Post-stimulus latency windows roughly corresponding to zero, one, or several interneuronal delays within the spinal cord. In these experiments, the algorithm only observes the middle response. Note that using the above definitions, there is a small overlap between the early and middle responses.

4.3 Objective Function

In a UCB formalism (see Section 3.2), it is necessary to have a function which describes a notion of "reward" obtained for any particular action available, and which is also gradually learned from the data acquired; in this application, the reward function is chosen to describe the response of the spinal cord and muscles to the stimulus applied. Specifically, we chose to place the animal in a body-weight support harness and measure the peak-to-peak amplitude of the response of the left tibialis anterior muscle (LTA, a dorsiflexor of the foot) to each individual stimulus pulse, in a latency period termed the "middle response" (MR, 4.5 - 7.5 ms post-stimulus), corresponding to responses which likely involve a single interneuron in the spinal cord synapsing directly onto the motor neuron. These measurements were all conducted with a fixed stimulus amplitude of 5 V (7 V in animal 7), and at a stimulus frequency of 1 Hz, such that the response to each individual stimulus pulse was dissociated from its predecessor and successor in time. This response function was chosen because it provides a short-term, measurable surrogate for therapeutic effectiveness by measuring an indicator of interneuronal function. This is necessary because, while the end-goal is to improve a therapeutic outcome (e.g., standing or stepping), the credit assignment problem of associating this long-term therapeutic outcome with the therapy's constituent stimuli would require infeasibly large cohorts of animals. This sort of wholistic policy selection also does not provide feedback for improving the individual patient's therapy at the moment, nor does it provide individualized policies.

Having settled on an immediately measurable surrogate for the utility of therapeutic stimuli, this work uses a UCB-based algorithm to explore and exploit this response function, such that short- and long-term performance with respect to the reward function are appropriately traded off against one another. This is done via selecting actions which individually are expected to gain high reward (i.e., produce a large evoked potential response in the LTA during the MR latency window), yield a substantial amount of information regarding the performance function as a whole, or possibly both. Under a number of assumptions, including the assumption that the response function itself does not change with time, and with carefully chosen parameters, the GP-UCB, GP-BUCB, and GP-AUCB algorithms (discussed in Sections 3.2.4, 3.3, and 3.4) are all guaranteed to converge to a subset of actions which yield maximal reward, given a sufficient number of actions (See Theorem 1 in Section

3.3.2). In the case in question, if the response function were not changing with time, this result would guarantee convergence to the optimal stimulus with respect to the specified response amplitude, given sufficiently many stimuli. In actual fact, however, the response function is non-stationary, requiring an algorithm which can track these changes (i.e., alter its internal model appropriately as the responses alter over time). In order to solve this problem, the present work employs a modified version of the GP-BUCB algorithm. The problem specification, a careful discussion of the algorithmic challenges, including time variation, and specific modifications required to meet these challenges are discussed in Section 4.4.

In animals 2, 5, and 7, the algorithm was compared competitively with a human experimenter in terms of reward. This experimenter's search and exploitation strategy varied over the course of the several months of experimentation. Anecdotally, the procedure used in the later animals for a typical, five batch day, was as follows:

- In each of the first three batches, the experimenter selected the first three actions on the basis of knowledge of the anatomical location of the electrodes with respect to the distal leg motor pools, combined with observations from earlier batches in the day. These first three actions were typically chosen as "variations on a theme," e.g., polarity swaps or small perturbations to anode or cathode position.
- The fourth and fifth actions in each of the first three batches were selected on the basis of the first three actions in that batch; these were usually chosen to be small variations on whichever (if any) of the first three configurations were successful.
- In the fourth and fifth batches, the experimenter selected actions to explore portions of the cord and array which were deemed less likely to produce strong responses.

The human experimenter never selected an action (i.e., pair of electrodes, cathode and anode) more than once on a day; this presents a confounding factor as far as analyzing the competitive performance of the human versus the algorithm, since the two were acting under somewhat different rules, but this was judged to be necessary for scientific purposes. Further, the human was not restricted to use strictly 10 (or 20) pulses for each action, meaning that the human could administer enough pulses to thoroughly assess the results of any configuration, without requiring a repeated action. This may make a per-pulse examination of amplitudes more appropriate. The human experimenter does have the advantage, however, of choosing actions based on feedback obtained within the batch, i.e., does not operate on the strict feedback schedule used by the algorithm. This might be able to prevent the human from wasting time on costly blunders.

4.4 Modifications to the GP-BUCB algorithm

Formally, the GP-BUCB algorithm maintains a Gaussian process posterior over the decision set D, such that for any stimulus $\boldsymbol{x} \in D$, the reward f (i.e., peak-to-peak amplitude of the MR, 5 V stimulus, 1 Hz) for that stimulus is modeled as a normal random variable $f(\boldsymbol{x}) \sim \mathcal{N}(\mu_{\mathrm{fb}[t]}(\boldsymbol{x}), \sigma^2_{t-1}(\boldsymbol{x}))$ when making the tth decision. The algorithm calculates this posterior based on the series of actions $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{t-1}\}$ selected in previous rounds and the observations $\{y_1, y_2, \ldots, y_{\mathrm{fb}[t]}\}$ so far observed. The algorithm then uses this posterior with Equation (3.7) to choose an action $\boldsymbol{x}_t$ for execution in round t. This algorithm is designed such that, with appropriately chosen values of the exploration-exploitation tradeoff parameter β_t, it will converge to the optimal stimulus under the chosen reward criterion for a static (i.e., time-invariant) response function.

Unfortunately, the problem setting does not conform to some of the requirements of the regret bounds currently available for the UCB family of algorithms. Most importantly, the spinal cord's responses are non-stationary. Further, these experiments were required to retain some neuroscientific value in terms of the data collected, rather than simply being an algorithmic exercise. Temporal non-stationarity is discussed in Section 4.4.1 and data value preservation constraints are discussed in Section 4.4.2.

4.4.1 Time Variation of the Reward Function

Of the two deviations from the theoretical setting mentioned above, time-variation is the more fundamental and critical challenge. Both within a session, due to fatigue, and across sessions, the stimulus-response mapping instantiated in the spinal cord is not static, making it essential to consider the variation of the spinal cord's responses in time. These across-session effects are particularly crucial, and may include degradation of the array, changes in its interaction with the surrounding tissue, and, most importantly, the combined effects of recovery and plastic adaptation of the spinal cord. It is the interaction of the algorithm's actions with the course of recovery and spinal plasticity which constitute the essential phenomenon of this therapeutic approach, but these also present a particularly difficult prediction problem for the algorithm's model of the spinal cord's responses. The requirement that the algorithm must model variation of the responses in time means that the algorithm must be able to use information from past observations to predict the current state of the spinal cord. This work chooses to model the response function as a GP which has another free dimension besides the stimulus parameters, the time of stimulation t (in days post-injury). The decision set available to the algorithm at any moment can be thought of as being perpendicular to the t axis in this space, such that the available actions $\boldsymbol{x} \in D$ are constant, but their location in t varies, producing a time-varying decision set D_t. The algorithm must now regress on a function of both stimulus applied $\boldsymbol{x}$ and time t, using data which correspond to some subset of the possible stimuli and which were obtained at times $t' \leq t$. To do so, the covariance function must now be a map $k : (D \times t) \times (D \times t) \to \mathbb{R}$. Critically, the modeling problem becomes one of extrapolating in time, either forward on the order of days (from one session to another) or just a few minutes (from one batch to the next). This stands in sharp contrast to the nominal setting for the UCB family of algorithms, in which the observations of D become increasingly dense as the algorithm runs, such that the problem is more one of interpolation and the posterior uncertainty is non-increasing.

Because t is constantly increasing, the algorithm must cope with a degree of uncertainty which is increasing (for covariance functions which decay monotonically in time, following the intuitive notion that more recent observations are more useful); that is, the information the algorithm has available will be less and less relevant as time goes on, unless it acquires new observations. Since the posterior uncertainty cannot decrease beyond a finite level, dependent on the rate of change of the response function with respect to the arrival of observations, the algorithm will not ever truly "converge" in the sense of having vanishingly small uncertainty as to which action $\boldsymbol{x}$ in the decision set D is the optimal action at time t. This is similar to the well-known results for Kalman filters (Kalman, 1960) and less well-known results for Gaussian Markov processes (see Rasmussen and Williams, 2006, Appendix B), both of which describe non-zero steady-state limits for the uncertainty under a static observation model, given Gaussian disturbances. In the bandit setting, since the observations available for regression (and thus the reduction of uncertainty) and our actions (and thus the reward obtained) are coupled, this may prove additionally problematic.

However, the UCB formalism is quite robust, and the variation of the response function within the course of a day is not so substantial as to be insurmountable. Further, it is possible to draw substantial inference from observations obtained on previous days, since the day-to-day variation of each individual electrode configuration's corresponding response occurs on a slow timescale, often requiring several days for substantial changes. It is thus possible to be guided by previous days' measurements and to learn deviations from previous days during the course of an experimental session. Additionally, from a practical perspective, it is not necessarily important to find the absolute optimal action at any given time, but rather one which has sufficiently high performance to provide useful therapy. Indeed, some variety in the administered stimuli is appropriate; in robotic gait training, in which spinalized rodents are assisted by a robotic device, Cai et al. (2006) showed that an assistance paradigm which allowed some deviation from the nominal limb trajectory, but maintained appropriate inter-limb coordination, performed better than two competing policies which enforced the nominal trajectory or promoted particular individual limb trajectories without regard for inter-limb coordination. This same phenomenon may hold true for epidural electrostimulation, i.e., the application of a highly specific stimulus pattern with the exclusion of all other patterns may result in poorer therapeutic performance than a more diverse set of stimuli. Nevertheless, some notion of regret is a useful means of understanding what the algorithm is doing, and so we will continue to discuss the algorithm in these terms.

4.4.2 Redundancy Control and Repeated Observations

The second substantial deviation from the GP bandit setting in these experiments has to do with repetitions of stimuli during an experimental day. It was desirable to extract information from these animals which also had broader scientific value, but the utility of the data set for other purposes primarily depends on sufficient diversity of measurements. A bandit algorithm, on the other hand, should display some form of convergence behavior, i.e., spend many queries on relatively few elements of the decision set D. The required compromise was that the algorithm was restricted from asking for more than two repetitions (generally) of a given electrode configuration on a given day. The number of allowed repetitions for each experimental run is presented in Table 4.5. Since a testing day for the algorithm typically consisted of 5 batches of 5 requests, on a typical day, a minimum 13 different pairs (i.e., distinct actions $\boldsymbol{x}$) had to be requested, forcing some diversity of exploration. This requirement was particularly stringent for the wired array animals, which had only 42 electrode pairs possible, i.e., $|D| = 42$, and thus the algorithm was required to cover some large portion of D each day. Further, as will be discussed in Section 4.7, the evoked potentials appear to be quite sensitive to the position of the anode; the requirement that at least 13 pairs must be used on each typical testing day meant that, for some anodes, every possible configuration using that anode was tested. This, in turn, meant that the algorithm could be forced out of high-reward portions of the decision set and into regions of much lower reward. This presented a number of challenges in analyzing the data, as, by design, convergence in the conventional sense of taking the same action repeatedly was not possible. This topic is discussed in the context of the wire-based animal results in Section 4.7.1.

Another restriction arose for practical purposes; since the practical cost of setting up an individual experiment in terms of experimental time is essentially constant, regardless of the number of stimulus pulses applied, each experiment requested by the algorithm consisted of many successive stimulus pulses (usually 10 or 20), the number of which was known to the algorithm in advance. This required mechanisms for resolving how many observations had been obtained, versus how many had been requested, which further complicated the measurement of regret. Also, if the algorithm requested an action multiple times in the same batch, these observations were executed together (e.g., if 20 stimulus pulses would ordinarily be delivered for a single request of A1_C9, and this action was requested twice in the same batch, the experiment would be set up once, and 40 stimulus pulses would be delivered successively). On an anecdotal basis, this does not appear to have caused substantial fatigue during this sequence of stimuli, but it does present another mild complication in terms of how to analyze the experimental results.

4.5 Kernel and Mean Functions

A number of substantial difficulties are associated with the tuning of the hyperparameters and the associated model selection problem, particularly in the time dimension. Firstly, the data are collected by a biased observer, which selectively samples in regions of high reward, and only rarely samples in regions of low reward. Further, since GP-BUCB is not fully Bayesian, i.e., it chooses actions using a single set of hyperparameters θ and a single model class M (the kernel and mean functions), the set of actions $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots\}$ selected is itself highly dependent on θ and M. Particularly problematic are the kernel lengthscales, in a fashion analogous to sampling rate in digital signal analysis; if one does not sample sufficiently densely in the dimension one wishes to fit (analogous to sampling at too low a frequency, e.g., less than the Nyquist frequency), one cannot detect the presence of short lengthscales (analogous to high-frequency content), and since the model of the lengthscales present in the response function is precisely what determines what data the algorithm collects, this error could remain undiscovered. It is also possible that unstable behavior might occur in the case where the algorithm is allowed to also adaptively re-fit the hyperparameters, yet the data collection (via GP-BUCB) is not designed to take this procedure into account. Further, note that conditioning the posterior over model classes or hyperparameters on the algorithm used to generate the fitting set is not helpful either; since the interaction of the algorithm with the true system is only through the actions $\boldsymbol{x}$ and observations y (see Figure 4.3), precisely the same data one would use to compute the likelihood, the posterior over the set of hyperparameters Θ or set of model classes $\mathcal{M}$ is independent of the algorithm's internal GP model, given $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots\}$ and $\{y_1, y_2, \dots\}$. This essentially means knowing which model the algorithm used internally does not help if the acquired data are not informative. This leaves the possibility of using strong priors, but in particularly bad cases, this essentially devolves to hand-fitting the model.

Even in light of the above considerations, some attempts to fit the hyperparameters using the conjugate gradient method[1] on the likelihood or posterior were made. These attempts usually incorporated the human-generated data, which tended to be more diverse and did not have the same set of modeling biases, along with the algorithmically-generated data. Fitting was somewhat successful for the spatial lengthscales, but performed very poorly on the time lengthscales. This poor performance with respect to time lengthscales was most likely because the fit was dominated by the short-time changes (e.g., fatigue and queueing bias due to the redundancy control, discussed in Section 4.4.1) within the experimental sessions (on the order of 1-3 hours) rather than the underlying spinal variation taking place on timescales of days.

Guided by the fitting described above for the spatial lengthscales, careful examination of regression performance on data from earlier animals, and intuition, hyperparameters and kernel functions

[1]minimize.m in the GPML toolbox, Rasmussen and Nickisch (2010).

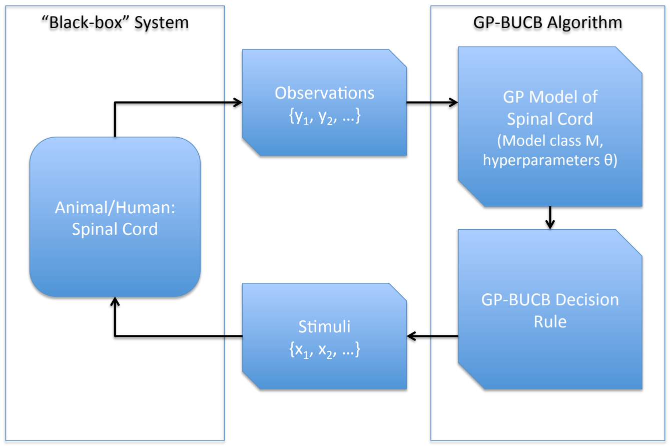


Figure 4.3: Simplified system diagram for the GP-BUCB algorithm interacting with the spinal cord. Given an internal model of the system, fully described by a particular model class M and a particular set of hyperparameters θ, the GP-BUCB algorithm is deterministic, up to tie-breaking. Note that the data likelihood $p(\{y_1, y_2, \dots\}|\{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots\}, M^*, \theta^*)$ is independent of M and θ; thus, poor choices of θ and M could result in the collection of a set of data which is not discriminative between various model classes in $\mathcal{M}$, and knowledge of M and θ will not be of any help. This makes post-hoc (or periodic) model selection challenging.

were hand-selected. The kernel functions and hyperparameters used in the animal experiments are presented in Table 4.5. For the first two animals, a squared-exponential (or RBF) kernel, Equation (2.10), was used, with indepedent lengthscales for each dimension. Unfortunately, due to the strong smoothness assumptions implicit in this kernel (the squared exponential kernel implies the Gaussian process is infinitely mean-square differentiable; see Rasmussen and Williams, 2006, Section 4.1.1), problems resulted from intra-day variations in the responses of individual configurations, as well as from long gaps in testing, e.g., weekends. These effects can be seen in Figure 4.16; on day P34 in animal 2, the posterior was badly mis-specified, causing erratic sampling. For further discussion, see Section 4.7.5. In the third and fourth animals, a hybrid kernel $k_h(\boldsymbol{x}_i, \boldsymbol{x}_j)$ was employed, where

$$k_h(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sigma_1^2 k_m(\boldsymbol{x}_i, \boldsymbol{x}_j) + \sigma_2^2 \delta(i, j), \tag{4.1}$$

k_m is a 3rd order Matérn kernel, Equation (2.15), and $\delta(i,j)$ is the Dirac delta function on the indices i, j of the stimuli, not the stimuli $\boldsymbol{x}_i, \boldsymbol{x}_j$ themselves. In essence, this changes the problem from one which is trying to infer the distribution of $f(\boldsymbol{x})$, $\forall \boldsymbol{x} \in D$ to trying to infer $g(i) = f(\boldsymbol{x}_i) + \eta_i$, $\eta_i \sim \mathcal{N}(0, \sigma_2^2)$, a noisy function. Functionally, this means that the uncertainty over $f(\boldsymbol{x})$ never becomes less than a small, positive value, which is useful because it means that short-time variations in the function are subsumed into this noise term, leaving the overall shape to be captured by the

Animal	Run	Dates	Kernel		Mean		Repetitions
			Function	Hyperparameters	Function	Hyperparameters	Allowed
3 (Parylene)	Run 1a	10/3, 5, & 6	covSEard	$l = [0.2740, 0.4659, 0.3297, 0.6300, 1.2018]$, $\sigma = 0.1244$, $\sigma_n = 0.0268$	constant	$c_0 = 0.0$ mV	2
	Run 1b	10/8	covSEard	$l = [0.2740, 0.4659, 0.3297, 0.6300, 1.2018]$, $\sigma = 0.1244$, $\sigma_n = 0.0268$	constant	$c_0 = 0.1$ mV	1
2 (Wired)	Run 1a	11/26 - 29	covSEard	$l = [0.5480, 0.9317, 0.6593, 1.2599, 2.4034]$, $\sigma = 0.1244$, $\sigma_n = 0.0268$	constant	$c_0 = 0.1$ mV	3
	Run 1b	12/3	covSEard	$l = [0.5480, 0.9317, 0.6593, 1.2599, 2.4034]$, $\sigma = 0.1244$, $\sigma_n = 0.0268$	constant	$c_0 = 0.9$ mV	2
	Run 1c	12/4	covSEard	$l = [0.1147, 46.9592, 0.1949, 5.0130, 0.9114]$, $\sigma = 0.5288$, $\sigma_n = 0.1708$	constant	$c_0 = 0.9$ mV	2
	Run 2	12/5-7 & 10	covSEard	$l = [2.9453, 6.3777, 3.4291, 2.3453, 2.4034]$, $\sigma = 0.7903$, $\sigma_n = 0.1364$	constant	$c_0 = 0.9$ mV	2
	Run 3	12/11-14 & 17	covSEard	$l = [2.9453, 6.3777, 3.4291, 2.3453, 2.4034]$, $\sigma = 0.7903$, $\sigma_n = 0.1364$	constant	$c_0 = 1.4$ mV	2
5 (Wired)	Run 1a	1/31 & 2/1	Hybrid	$l_1 = [1.0000, 2.7183, 1.0000, 2.7183, 2.7183]$, $\sigma_1 = 0.2865$, $\sigma_2 = 0.2231$, $\sigma_n = 0.1353$	Linear	$c_1 = 0.08$ mV/day, $c_0 = -0.2492$ mV	2
	Run 1b	2/6, 7, & 8	Hybrid	$l_1 = [1.0000, 2.7183, 1.0000, 2.7183, 2.7183]$, $\sigma_1 = 0.2865$, $\sigma_2 = 0.2231$, $\sigma_n = 0.1353$	Linear	$c_1 = 0.08$ mV/day, $c_0 = 1.7$ mV	2
	Run 2	2/11-15, 19-22, & 26	Hybrid	$l_1 = [1.0000, 2.7183, 1.0000, 2.7183, 2.7183]$, $\sigma_1 = 0.2865$, $\sigma_2 = 0.2231$, $\sigma_n = 0.1353$	Linear	$c_1 = 0.08$ mV/day, $c_0 = -0.1348$ mV	2
7 (Parylene)	Run 1	3/1, 4, & 5	Hybrid	$l_1 = [1.0000, 2.7183, 1.0000, 2.7183, 2.7183]$, $\sigma_1 = 0.2865$, $\sigma_2 = 0.2231$, $\sigma_n = 0.1353$	Linear	$c_1 = 0.08$ mV/day, $c_0 = 0.2513$ mV	2

Table 4.2: The modeling assumptions, e.g., kernel functions and hyperparameters, used by the algorithm to model the responses of animals in our experiments. Numerical designations of runs indicate memory resets of the algorithm, whereas letter suffixes indicate some change to the hyperparameters. For the linear mean function, the mean prediction at any time t in days post-injury is $m(t) = c_1 t + c_0$.

Matérn kernel. The moderately non-smooth nature of a GP with a Matérn kernel is also useful for preventing overshoots or sensitivity to intra-day variation in responses. Note that, while the choice of the 1st order Matérn kernel k_m implies the Gaussian process f is once mean-square differentiable (i.e., it has some degree of smoothness), the GP with the hybrid kernel, g, is not even mean-square continuous (i.e., samples from it can be extremely rough, though this is additive noise on a somewhat smooth trend; see Section 4.1.1, Rasmussen and Williams, 2006). The hybrid kernel proved to be quite successful and robust, as demonstrated in animal 5 by the continuation of runs over weekends, which had been problematic when using the Squared-Exponential kernel.

4.6 Results

A series of experiments in four rats prepared as in Section 4.2 was carried out at UCLA, including 37 sessions and approximately 1200 actions (670 actions selected by the algorithm). In three animals (3, 5, and 7) the experiment continued up until failure of the electrode array. In the remaining animal (number 2), the experiment terminated to allow analysis of the model selection problems described in Sections 4.5 and 4.7.5. The peak-to-peak amplitudes of the evoked potentials for all stimulus pulses recorded, for all four animals, are presented in Figure 4.4. These plots show substantial variation in the evoked potentials over the course of each experiment. In all four cases, they also show a distribution of evoked potential responses which has peak values similar to those generated by the human experimenter. The close matching between the responses evoked by the human and the algorithm indicates that the majority of the variation in responses was not directly a function of the action selection method (human or machine). Instead, the time variation in the evoked potential amplitudes resulting from each stimulus combination appears to be composed of both fluctuations

Animal	Run	Dates	Total Actions	Unique Stimuli Requested	$r_{algorithm}$	r_{human}
3 (Parylene)	Run 1	10/3, 5, 6, & 8	70	43	0.55993	N/A
2 (Wired)	Run 1	11/26 - 29, 12/3 & 4	80	23	-0.012436	-0.17223
	Run 2	12/5-7 &10	55	23	0.46169	0.021927
	Run 3	12/11-14 & 17	100	29	0.48137	0.28783
5 (Wired)	Run 1	1/31, 2/1, 6, 7, & 8	70	28	0.71533	0.030175
	Run 2	2/11-15, 19-22, & 26	240	33	0.92555	0.51492
7 (Parylene)	Run 1	3/1, 4, & 5	55	40	0.61046	-0.039921

Table 4.3: Actions allocated by the algorithm over the course of the run and the number of unique electrode pairs observed by the algorithm during the run. Also included are correlations between the observed mean reward for each configuration (a single shared value for both the algorithm and the human, calculated by averaging all peak-to-peak amplitudes of the evoked potentials observed by either during the entire course of the run) and the number of pulses each of the algorithm or human received for that configuration. Note that the human experimenter only began to deliberately exploit, rather than explore, the reward function after the second run of animal 2.

on a day-to-day basis and substantial, long-term trends. The latter of these may be due to spinal plasticity. Table 4.3 presents a list of experimental runs, the dates of experimental sessions, the total number of actions initiated by the algorithm in each run, and the number of unique stimuli employed.

In the absence of a ground-truth value for the absolute maximum activation of the chosen muscle at any instant in time, the regret cannot be calculated, and so one may instead consider the reward. This quantity is expressed as $\tilde{w}_\tau = y_\tau$ if calculated for each individual stimulus pulse, where τ indexes in the order of individual stimulus pulses and y_τ is the corresponding observation of the response's peak-to-peak amplitude. If the reward is calculated on an action by action basis (i.e., in the fashion in which the algorithm or human makes decisions and requests actions, each composed of several stimulus pulses), the reward for action t is

$$w_t = \sum_{\tau=\tau_{min}(t)}^{\tau_{max}(t)} y_\tau,$$

where $\tau_{min}(t)$ and $\tau_{max}(t)$ are respectively the indices of the first and last pulses for action t. Results are presented in both by-pulse and by-action forms. Reward may also be examined in terms of the maximum value observed so far,

$$\tilde{W}_m(\tau') = \max_{\tau \le \tau'}(\tilde{w}_\tau)$$
$$W_m(T) = \max_{t \le T}(w_t),$$

where the maximum is found by comparison among the by-pulse stimulus responses observed ($\tilde{W}_m$)

or among the average responses observed for each individual action (W_m). Note that, at any given time, $\tilde{W}_m \geq W_m$, since the by-pulse maximum amplitude is more sensitive to random variation between pulses administered for a single action than is the average of these individual measurements. The maximum reward is analogous to the minimum regret (see Section 3.2). This quantity in some sense describes the thoroughness of the search over the stimulus space, since a high maximum reward value implies that high-performing stimuli have been found, and thus could conceivably be exploited. This quantity explicitly ignores the number of times high-performing stimuli are visited, so strategies like random search may perform well in terms of maximum reward, while not delivering effective therapy during the same period of time. The reward may also be examined in terms of the average reward so far observed,

$$\begin{aligned} \tilde{W}(\tau') &= \frac{1}{\tau'} \sum_{\tau=1}^{\tau'} \tilde{w}_\tau \\ W(T) &= \frac{1}{T} \sum_{t=1}^{T} w_t, \end{aligned}$$

analogous to the average regret (see Section 3.2). This quantity measures how well the algorithm has traded off exploration and exploitation against one another; a high value for the average reward implies that the algorithm has spent most of its time choosing actions which perform well, yet also explored thoroughly enough to find these high-performing stimuli. In terms of these reward measures, superior performance of one algorithm or method relative to another at a given time index T or τ' is observable as a larger value of that method's maximum and/or average reward plot. Both the per-pulse and per-action results are used in the presentation of results below. The presentation of the results of the animal studies is divided into two; the results for the wired array animals appear in Section 4.6.1 and those of the parylene array animals in Section 4.6.2.

4.6.1 Wire-Based Array Animals: Results

Figures 4.4(a) and 4.4(b) show the responses for all stimuli administered to the wire-based array animals. Maximum and average regret for individual runs are presented in Figures 4.5, 4.6, and 4.7, for animal 2, and Figures 4.8 and 4.9 for animal 5. In both animals, testing included 15 experimental days, with 235 and 310 actions selected by the algorithm, respectively. Although the regions of the stimulus space which yielded strong responses were quite different in size between the two animals (discussed in Section 4.7.2), the algorithm learned the response function well in both. Apart from the final day of animal 2's experiment, tracking by the GP model (i.e., responsiveness to the time-variation in the response function) was sufficient to enable effective decision-making. Both of these animals also exhibited an increase in the level of responsiveness over the course of the experiments. The qualitative shape of this change in responsiveness may be a function of the recovery post-injury, a function of the spinal plasticity, or both.

4.6.2 Parylene Microarray Animals: Results

Figures 4.4(c) and 4.10 (animal 3) and Figures 4.4(d) and 4.11 (animal 7) show the maximum and average reward results of these experiments. Though much briefer than the experiments in the wire-based array animals (4 days and 70 actions in animal 3 and 3 days and 55 actions in animal 7, rather than 15 days and over 200 actions in the wire-based array animals), the algorithm also found high-performing stimuli in the parylene array animal experiments and exploited them.

4.6.3 Computational Performance

The algorithm was implemented in the MATLAB programming language and run on one of two machines (a MacBook Pro, 2.2 GHz quad-core i7 processor, 8 GB RAM running Mac OS 10.6 and MATLAB R2012a; or an AMD Athlon 64 X2 Dual Core 3800+, 3 GB RAM machine running Ubuntu 12.04 and MATLAB R2011b). Recordings were processed and decisions were made by the algorithm approximately within the time required for the human experimenter to perform a batch of tests, i.e., in minutes, with few exceptions (generally errors in data processing). After having completed the computationally intensive analysis of the raw data, typical times to compose a new batch of actions using the GP-BUCB algorithm were on the order of seconds. Under the most severe conditions which occurred during the experiments discussed in this chapter, with $n = 4432$ individual evoked potential observations available at the end of animal 5's second experiment, computing a hypothetical new batch of five actions to begin the following day took 142.045 seconds of CPU time on the Ubuntu machine described above. The majority of this time was spent on the five executions of the Cholesky decomposition (more than 50 seconds) and five evaluations of the entire kernel matrix (also more than 50 seconds); both of these operations could be changed to execute only once per batch, taking advantage of structural characteristics of the Cholesky decomposition and kernel matrix to append rows and columns with the addition of each observation, rather than completely recalculating, thus saving substantial computational time. In terms of memory consumption, the largest objects in memory are the kernel matrices and the associated Cholesky decomposition results, which are $n \times n$ in size, where n is the number of observations.

Since n is expected to grow with time and both computational time required and memory consumption are critically dependent on n, one reasonable method for controlling computational load would be to "forget" (i.e., delete or not pass to the GP-BUCB algorithm) observations which were redundant by virtue of many more recent observations of the same configuration having been made. In the case mentioned, this could potentially reduce n significantly. It may be reasonable to impose a hardware-based cap on n instead or in parallel. Another reasonable measure would be to treat average responses to actions as the observations to be fed into the algorithm, rather than individual evoked potential responses. This could potentially reduce n by a factor of $m = 10$ or 20, depending on how many pulses typically correspond to each action, potentially reducing computational time by a factor of as much as m^3, since computation of the Cholesky decomposition scales as $O(n^3)$.

4.7 Discussion

As there were relatively few animals involved in these experiments and the pieces of information which are to be extracted from them are relatively complex, careful dissection of the results from these experiments is necessary. The wire-based array experiments are addressed in Section 4.7.1, with a particular focus on inter-animal comparisons in Section 4.7.2. Section 4.7.3 deals with the experiments in the animals with parylene-based arrays. Finally, Section 4.7.5 focuses on the results obtained with respect to appropriate kernel functions and hyperparameters.

4.7.1 Wire-based Array Animals

Because of the long experimental lifetime of the preparation, the wired array experiments allow exploration of how the algorithm copes with time-variation in the response function. The time prediction problem is inherently one of extrapolation, rather than interpolation. The algorithm must maintain enough uncertainty over how the response function will evolve over time, such that the changes which occur are not unexpectedly large, while conversely limiting this uncertainty such that the model makes strong enough predictions to guide exploitative behavior. In general, the algorithm was successful in terms of future performance prediction; in animal 5, run 2, for example, the algorithm showed strong queueing behavior, i.e., it first selected what were both predicted and proven to be the best stimuli, before being forced by the repetition limit to apply different stimuli. This same run also demonstrated that the algorithm is capable of predicting forward in time over long intervals with no observations, e.g., weekends, as examined in detail in Figure 4.16; after three days with no testing, the algorithm showed the same strong queueing behavior and strong performance (Figure 4.12) as on P31. This problem is discussed more thoroughly in Section 4.7.5.

The major difficulty in analyzing the data from these animals is that the total number of actions the algorithm can request on a given day (typically 25, in 5 batches of 5 actions) is of the same order as 42, the size of the decision set D. Several individual criticisms follow from this fact:

- Assuming that the response function does not vary tremendously in character from day to day, exhaustive testing can be performed in two full days of experiments. Indeed, if a very simple model is considered in which k configurations are selected uniformly at random, without replacement, and there are two classes of stimuli (n of which produce a response, and the remainder of which do not), probabilities of finding an effective stimulus in a given number of experiments may be calculated. Considering a situation in which two batches are used for searching ($k = 10$) and there is a non-negligible response if and only if a particular anode is picked ($n = 6$), the probability of successfully finding a responsive configuration is over 82%. For $k = 15$, this increases to over 94%, and for $k = 25$ (i.e., a full testing day), over 99%. This may imply that the problem could be solved without needing to make recourse to Gaussian processes as a model of the response function and that a classical bandit algorithm, which does not consider covariance between stimuli, would be sufficient, given modification for capturing time-variation. In particular, this suggests that simply finding a configuration which responds strongly is an insufficient measure of relative success in this setting.

- Because of the restriction on repetitions of any single stimulus, it might be difficult to differentiate between any of the many possible learning algorithms which have approximately the same performance, since those algorithms would most likely choose similar actions.

- It may be that this restriction on repetitions, combined with the small search space, means that the algorithm is forced to collect a sufficient set of data each day to track the time-evolution of the reward function, regardless of the kernel and hyperparameters chosen.

Each of these criticisms is addressed in turn. To the exhaustive testing argument, one may make two counter-arguments with support from the experimental results. First, the GP algorithm employed in this work appears to have effectively employed the structure of its model of the response function. In all five runs on wired array animals, the algorithm found high-performing stimuli in the first day. As discussed above, this is not particularly surprising given the relative size of the search space. In cases where the number of batches was fairly small (e.g., 2, as for both animals on their first day), which typically occurred while the animal initially became acclimated to the experiment, this result at least provides some weak evidence that the algorithm's initial search was well-structured for finding high-performing stimuli. Stronger support for the effectiveness of the algorithm's search may be derived from the fact that the algorithm avoided visiting many of the configurations in the stimulus space. A major difference between classical bandit algorithms and bandit algorithms on structured payoff functions is that, while classical bandit algorithms must eventually visit every action in the decision set, a structured model of the payoff function allows an algorithm to avoid doing so. As shown in Table 4.3, the number of unique stimuli chosen in the wired experiments over the length of any given run was substantially smaller than the size of the decision set; even in the second run on animal 5, in which testing lasted for 10 sessions and 240 actions, only 33 out of 42 configurations were ever selected by the algorithm. This strongly suggests that the structure of the GP model of the reward function enabled the algorithm to avoid exhaustive testing, while still effectively modeling the reward for untested configurations. Additionally, since the algorithm is competitive with the human experimenter in terms of the maximum reward so far observed, while typically maintaing a better average reward, it follows that the algorithm thoroughly searches the space for the highest reward regions (as demonstrated by the maximum reward), while simultaneously exploiting the gathered data in a more effective fashion. These observations argue that the algorithm is making an effective exploration-exploitation tradeoff, founded on a model which captures the system's behavior.

In response to the second potential criticism, regarding limitations placed on the algorithm, it is true that there is a strong upper limit to the potential performance of the algorithm under these conditions, and that the number of repetitions allowed for any individual stimulus within a daily session strongly influences this constraint. However, the degree to which the algorithm was able to effectively allocate its actions, in spite of the constraints, can be examined. Table 4.3 shows that there was a strong correlation between the number of pulses observed for a given combination of electrodes (a measure of the number of actions allocated to that combination of electrodes for both the human and the algorithm) and the whole-run average response to that combination (retrospectively computed by averaging every peak-to-peak, MR response amplitude observed from both the human and machine experiments in that run). Note that this measure of response strength does not include any notion of normalization of each day's responses; thus, given that the responses generally increased in amplitude over the course of the experiment, it may be expected that poor-performing stimuli would be visited early on and then never revisited, contributing to a low amplitude mean response for such configurations. Conversely, high-performing configurations should be visited consistently and often. It may be, however, that this influence may be mitigated by the combination of data from both agents, along with the limit on repetition within a day, which enforced query diversity. In the case in which the reward function does not change with time, it should be expected that an algorithm which performs well should allocate more queries to actions which provide higher reward. While the relationship between reward and number of pulses observed should not be linear, especially not as the number of observations becomes very large and the function is well known, computing the correlation coefficient provides a gross measure of this discrimination. The correlation coefficient of greater than 0.9 in the case of animal 5, run 2 can be taken as particularly strong evidence that the algorithm is capable of selecting queries in an appropriate priority ordering. In this particular animal, the strong saw-tooth shape of the average reward in Figure 4.9(b) also shows queueing behavior; the best performing stimuli were consistently selected first every day (causing the upswing of the saw-tooth), followed by some collection of other stimuli, most of which were low-performing (causing the down-swing), but which received dramatically fewer queries allocated to them than the highest performing stimuli. One alternate explanation for this saw-tooth behavior is rapid fatigue; if the result of stimulation is a general decrease in response amplitude over the course of a testing day, a set of stimuli could be identified as high-performing solely because they were selected early in the session, and others could be identified as low-performing due to being selected later and fatigue thus having set in by this time. However, this explanation is not consistent with the data obtained. First, the difference between weak and strong responses is more dramatic than might be expected from a gradual fatigue process, as seen in Figure 4.12(a). Second, the case where the same stimulus (i.e., pair of electrodes) is applied multiple times on a given day can give insight into the relative importance of generalized fatigue. Examining stimuli which resulted in substantially non-zero average response (an average of at least 0.2 mV over the day), the difference ΔV between the average response amplitudes at successive applications is essentially independent of the length of time between the two applications ($r = 0.0101$, $n = 142$ in data from animal 5). The differences in the sizes of the responses to successive applications of the same stimulus are small ($\Delta V = -0.0879 \pm 0.4878$ mV), even over long periods of time; for the 35 such intervals of one hour or more, the differences were -0.0537 ± 0.4295 mV. Were generalized fatigue a substantial factor, it would be reasonable to expect that long inter-application intervals would correspond with large, negative values of ΔV, since these long intervals would imply a substantial difference in time of application within the course of the experimental session, and thus the two instances would have occurred at substantially different points in the fatigue process.

To the third criticism, that the diverse set of data which the algorithm is forced to collect enables better tracking of time variation than would otherwise be possible, a nuanced answer must be given. Clearly, the constraint on repetitions will tend to produce more visits to poorly-performing stimuli than might otherwise occur. Occasional observations of these stimuli confirm that these stimuli remain poorly performing, thus contributing to tracking. Even without the constraint on repetitions, however, these stimuli would likely be revisited by the algorithm eventually due to the chosen kernels' description of temporally distant observations as relatively independent. This argues that the enforced diversity of the current paradigm may not cause a qualitative change in this respect. Additionally, because it is observed that most stimuli which perform poorly continue to perform poorly, re-visiting stimuli which were poor in the past is very unlikely to produce a surprisingly high reward; this argues that the practical value of good tracking on these configurations is relatively small, meaning that the experimental constraints may not cause too large of a performance gain in this respect. With regard to stimuli which are strongly responsive, but are not the absolute best on a given day, it may be that the diversity enforced by the constraint on repetitions may be a substantial aid to the algorithm's tracking; once the algorithm has "converged" to a subset of stimuli which produce strong responses, decisions among these could be strongly influenced by tracking. However, this better tracking may not actually confer a useful advantage with respect to reward because, due to the same constraint, many rankings of this subset of high-performing stimuli will result in the same actions being chosen (up to permutations in ordering).

Day post-injury	r	p	Number of configurations in common
15	0.7885	0.1130	5
20	0.4361	0.1564	12
21	0.6863	0.0197	11
22	0.5812	0.0144	17
23	0.6778	0.0055	15
24	0.2817	0.5405	7
28	0.5051	0.0100	25
29	0.6551	0.0032	18
30	0.4937	0.0270	20
31	0.5242	0.0802	12

Table 4.4: Days in both animals 2 and 5 in which some configurations were tested in common, combining human- and algorithm-commanded experiments for each animal. On many days, strong correlations were present between the responses for a given pair of electrodes across the two animals.

4.7.2 Cross-animal Comparisons

Due to the stability of the wired array implants used in animals 2 and 5, it was possible to collect a large amount of data from these two animals over a substantial period of time. These measurements allow comparison of the performance of individual pairs of electrodes across the two animals. One way to do this is to consider cases in which the same pair of electrodes (e.g., A1_A9) was tried in both animals on the same day post-injury. Some such comparisons are shown in Table 4.4 and Figure 4.15. While from only two animals, these data demonstrate that there is some fairly substantial repeatability between animals with respect to the strength of evoked potential elicited by a given stimulus on a given day post-injury. This appears to be particularly true for the highest-performing stimuli in animal 5, those with anode A9. It is interesting to note that animal 2, while also highly responsive to stimuli using A9 as an anode, had a larger set of effective stimuli than animal 5 (see Figure 4.15).

4.7.3 Parylene Array Animals

The analysis of data from the parylene array animals is primarily of interest as a means of assessing how well the algorithm can search the space D of electrode combinations for effective stimuli $\boldsymbol{x}$. Because this space is quite large (666 elements) relative to the number of combinations which can be tested on a given day (no more than 25), this is expected to be a challenging problem, requiring strong assumptions regarding the shape of the reward function $f(\boldsymbol{x})$ (i.e., the stimulus-response mapping) over the search space.

Both of the parylene array experiments were fairly brief; in both cases, the devices ceased to function at approximately two weeks post-injury. By comparison, the experimental lifetimes of both wired animals were roughly five weeks post-injury. Thus, it was particularly important for the algorithm to make intelligent choices with the few actions it had. It should be noted that the amplitudes of the peak-to-peak responses observed were somewhat smaller in these animals than in animal 5, though they were approximately the same as in animal 2.

In spite of the brevity of these experiments, it appears that it is possible to make some assertions about the actions of the algorithms in these cases. In both animals, after finding a configuration (C8_A9 in animal 3, C6_A9 in animal 7) which produced strong responses, the algorithm moved sharply to exploit this configuration, allocating many double queries to its neighbors. Both of these configurations include an anode at the left, caudal corner of the array (the same region which worked well in both of the wired animals), which indicates that this could plausibly be the region of strongest response (and thus highest reward) in the space. Though not all such configurations were effective, the algorithm does appear to have been able to explore and exploit the structure of the response function appropriately. Of particular note is the fact that the algorithm was able to overcome the flat prior it had been given and find this consistent anatomical pattern, even with so few queries; the key observation which allowed this exploitative behavior occurred in only the 3rd batch for animal 3, and the 6th batch for animal 7. The fact that this search was so efficient and could be effectively exploited by selecting neighboring configurations argues that the structure of the GP model is providing a benefit over what would be possible with conventional bandit algorithms.

While no cross-comparison to a human experimenter was made in animal 3, in animal 7, interleaved batches were selected by the human and algorithm. Possibly due to anatomical prior information, the human experimenter was able to effectively find strongly responding stimuli on the first day of testing in animal 7, despite only having two batches. However, later in the experimental period, in both animals 3 and 7, the algorithm found multiple high-performing stimuli. In the final day of animal 7, for example, the responses were similar in magnitude to those produced from the human-commanded experiments, and there were more successful stimuli. Figure 4.14 shows this result.

4.7.4 Therapeutic Relevance

While it is very difficult to disentangle the long-term effects of the generally intensive training protocol, the specific stimuli chosen by the human, and those chosen by the algorithm, it is possible to examine the immediate responsiveness of the rat's spinal cord and muscles to those stimuli. This work has assumed that stimuli which elicit strong responses are therapeutically useful, whereas those which produce little to no muscle activity are not; using a threshold twitch strength to explicitly divide stimuli in such a fashion may provide insight into which agent is better at usefully allocating therapeutic actions. This all-or-nothing division of stimuli into effective or not stands in contrast to the notion of reward, used extensively in the preceding discussion. In particular, reward is sensitive to outliers, possibly to a degree which is not reflective of actual therapeutic performance, since fine differences in twitch strength between stimuli which produce strong responses can have effects on the

Animal and Run	Agent	Threshold (V)							
		0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0
3 (Parylene)	Algorithm	0.229	0.157	0.100	0.043	0.000	0.000	0.000	0.000
	Human	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
2 (Wired), Run 1	Algorithm	0.662	0.450	0.263	0.000	0.000	0.000	0.000	0.000
	Human	0.440	0.253	0.067	0.013	0.000	0.000	0.000	0.000
2 (Wired), Run 2	Algorithm	0.945	0.727	0.545	0.091	0.000	0.000	0.000	0.000
	Human	0.720	0.520	0.280	0.040	0.000	0.000	0.000	0.000
2 (Wired), Run 3	Algorithm	0.910	0.850	0.790	0.560	0.350	0.110	0.020	0.000
	Human	0.828	0.697	0.626	0.424	0.192	0.081	0.040	0.000
5 (Wired), Run 1	Algorithm	0.657	0.543	0.357	0.214	0.186	0.129	0.057	0.000
	Human	0.564	0.418	0.273	0.127	0.109	0.055	0.018	0.000
5 (Wired), Run 2	Algorithm	0.779	0.642	0.442	0.308	0.229	0.167	0.142	0.075
	Human	0.685	0.477	0.285	0.153	0.106	0.081	0.068	0.026
7 (Parylene)	Algorithm	0.255	0.091	0.036	0.000	0.000	0.000	0.000	0.000
	Human	0.043	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Table 4.5: Proportion of stimuli yielding satisfactory responses, at any of several different thresholds for therapeutic twitch strength. The algorithm produced a higher proportion of responses above each achieved threshold, for each animal tested competitively, with the exceptions of Animal 2, Run 1, 2.0V and Run 3, 3.5V.

average or maximum reward measures used above, but may not be of any functional relevance. A comparison based on thresholded average twitch amplitude elicited by actions is presented in Table 4.5. Note that the algorithm was able to allocate its actions such that a higher proportion of them received responses satisfying nearly all of the thresholds examined; if, as posited above, therapeutic effectiveness is a function of the proportion of stimuli which elicit a sufficiently strong response, then this result indicates that the algorithm is more effective in this regard than was the human expert's chosen strategy.

4.7.5 Kernels and Hyperparameters

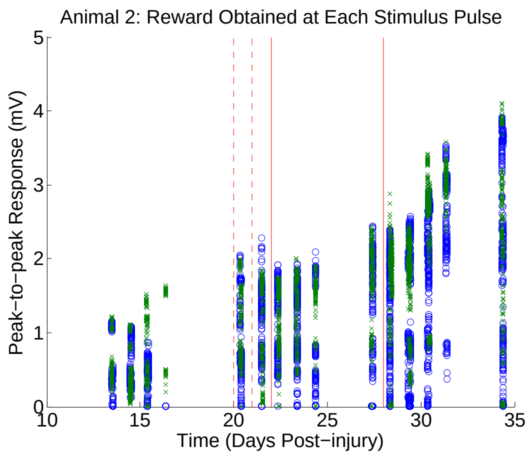
For the Gaussian process model to capture the responses of the muscles faithfully, yet allow rapid learning, it is crucial that the kernel function, mean function, and their respective hyperparameters be well chosen, such that the Gaussian process prior matches fairly well with the structure of the data actually measured. If these modeling choices are made well, the algorithm will likely perform well, but poor choices can have pathological behavior, especially with respect to the time variation of the responses. Figure 4.16(a), showing the GP posterior at the beginning of day P34 of animal 2's experiment, provides a good example of what can happen under poor modeling assumptions; the GP model gave predictions which produced a query pattern unreflective of the structure of the response function. This performance is clearly dependent on the properties of the kernel function and mean function with regard to changes over time, but there exist kernels and mean functions for which this problem can be resolved, as demonstrated in Figure 4.16(b).

Since active learning experiments using GP models of the spinal cord's responses had not been conducted before, it was critical to periodically re-examine the kernel, mean, and hyperparameters. After possible re-fitting during each animal's first run, subsequent hyperparameter re-fitting and/or kernel re-selection in that animal were accompanied by a wipe of the algorithm's memory; Table 4.5 and Figure 4.4 show these fitting events and memory wipes in more detail. It is important to note that the choice of when to re-fit or wipe the algorithm's memory was made by the human inspection, rather than by the algorithm; in this way the algorithm was in some sense protected from making prolonged, catastrophic errors. This does introduce some bias into the data, because the algorithm was not allowed to act nonsensically, but this was necessary to preserve the value of the experiments. Further, this did not confer a substantial advantage upon the algorithm over the human, because the human experimenters are constantly making just such checks for sensibility, e.g., using the responses elicited by the chosen stimuli to detect failures of the stimulating hardware. Many of these sorts of checks could and should be built into eventual implanted stimulators, but were beyond the scope of this work.

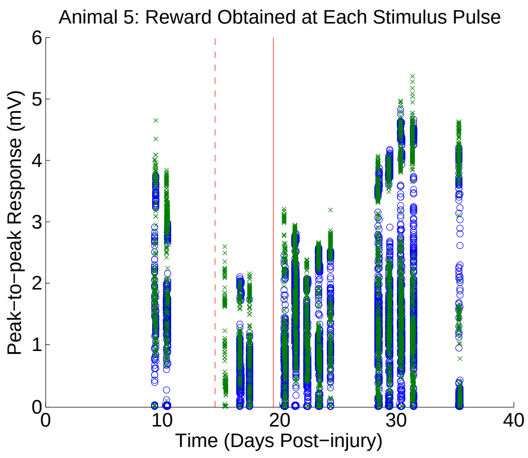
While the hybrid kernel, Equation (4.1), was successful for the experiments on animals 5 and 7, argument can be made that other kernels might be more appropriate. Two possiblities are fairly strongly suggested by the experience in these four animals. First, the noise term in the hybrid kernel could be replaced by a covariance term which gives covariance only between measurements of the same configuration on the same day, but otherwise treats observations as independent; this amounts to an assumption that there exists a hidden additive variable for each configuration on each day. Second, it may be that some linear kernel in time is reasonable, provided the algorithm has some "forgetting" of very old observations; this would somewhat account for the fact that, for anatomical reasons (which are thus the same from day to day), non-responsive configurations tend to remain non-responsive over the course of the animal's experimental lifetime.

4.8 Conclusions

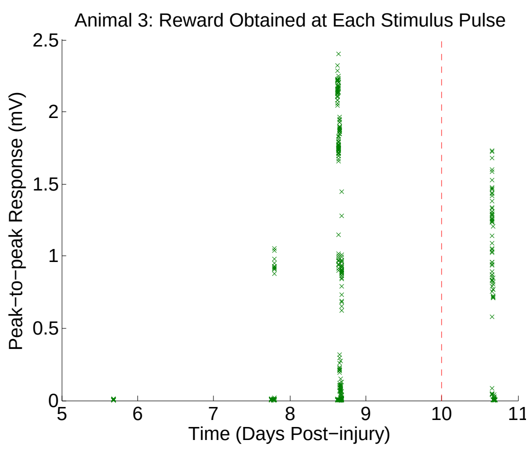
From the four animals examined, it can be concluded that the algorithm is effective for selecting stimuli to maximize a simple experimental objective, consisting of the evoked potential amplitude during the middle response period (4.5-7.5 ms latency with respect to stimulus pulse onset), over the set of pairs of stimulating electrodes. Further, from the three animals in which comparisons with a human expert were considered, the algorithm achieves action-averaged reward which is superior to that achieved by the human experimenter, while matching the human experimenter's performance in terms of maximum reward. This indicates that the algorithm is able to allocate more actions to exploiting high-performing stimuli while still matching the human experimenter's effectiveness in finding the best stimuli. These results provide a strong indication that the GP model used by the algorithm effectively captures the variation of evoked potentials, that the algorithm's decision rule effectively trades off exploration and exploitation, and that doing so enables the algorithm to provide effective stimulation in terms of maximizing evoked potentials. While this problem is substantially simpler than the related problem of maximizing standing or stepping performance, the capability demonstrated in these animal experiments, effectively modeling and exploiting the responses of individual muscles, is critical for fine-tuning these high-level behaviors.



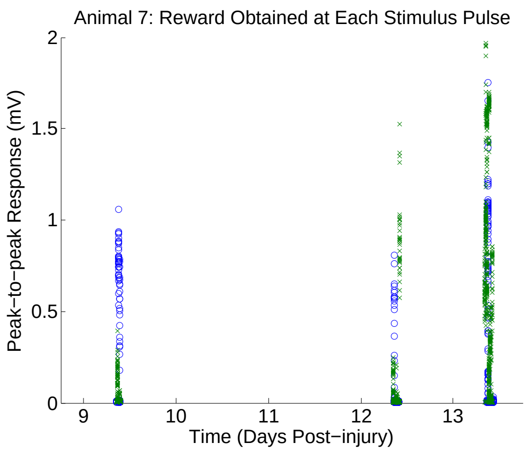
(a) *Animal 2 (November)*



(b) *Animal 5 (February)*



(c) *Animal 3 (October)*



(d) *Animal 7 (March)*

Figure 4.4: Peak-to-peak amplitude (mV, reward) of all individual stimulus pulses for each animal, combined across runs. Blue circles: evoked potentials obtained by the human experimenter; Green 'x': those observed by the algorithm. Solid red lines denote a wipe of the algorithm's memory (thus dividing separate runs), with or without changes to the hyperparameters, while dashed lines denote changes to the hyperparameters without a memory wipe. The human experimenter and algorithm were not privy to each other's actions or the resulting rewards. They (typically) executed their actions as interleaved batches of experiments.

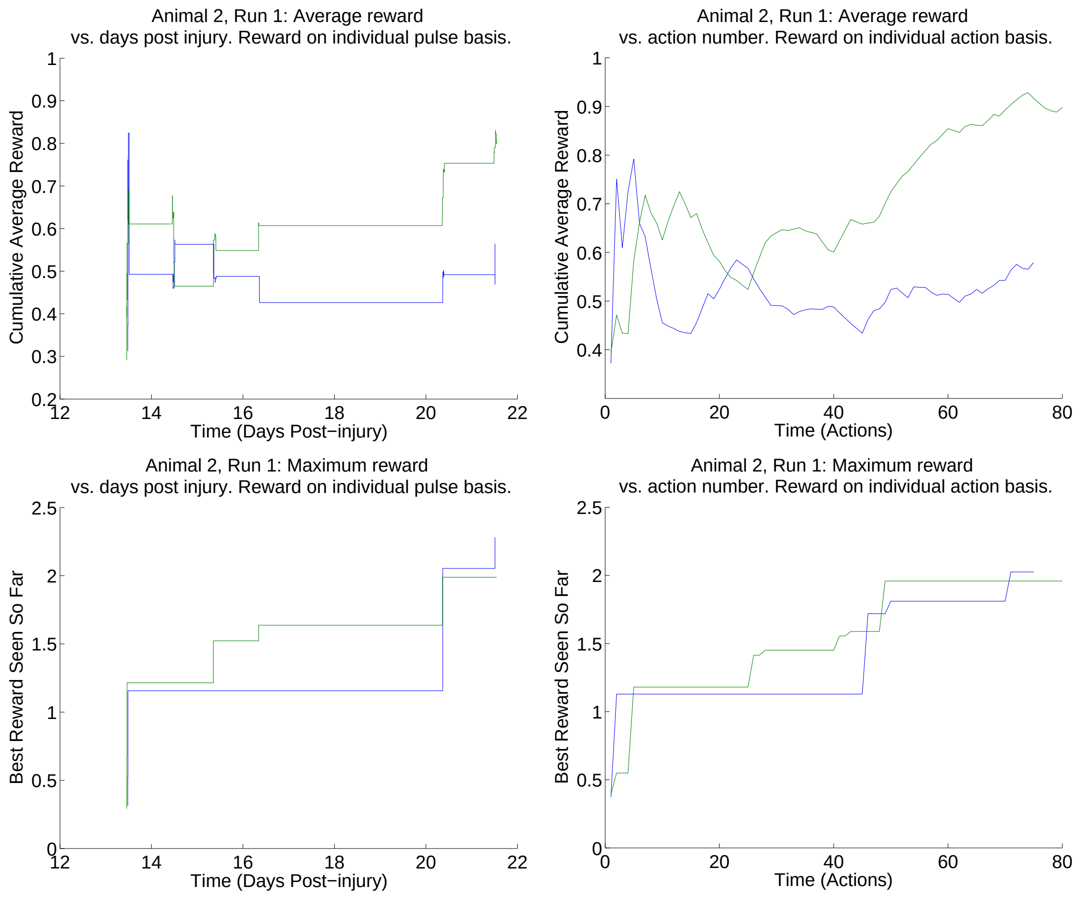


Figure 4.5: Human experimenter's (Blue) and Algorithm's (Green) reward (Peak-to-Peak amplitude of MR evoked by a 5V stimulus at 1Hz, in mV) in Run 1 of animal 2, the first wired array animal and second animal tested. This was the first competitive experiment between the algorithm and human. Average reward measures the algorithm's tendency to consider the reward generated by *every* action, rather than just the best; the behavior of the best action vs. time is captured by the maximum reward. Note that the algorithm's average reward is typically superior to that of the human experimenter, while the algorithm also maintains superior or competitive maximum reward, as of similar time or action index.

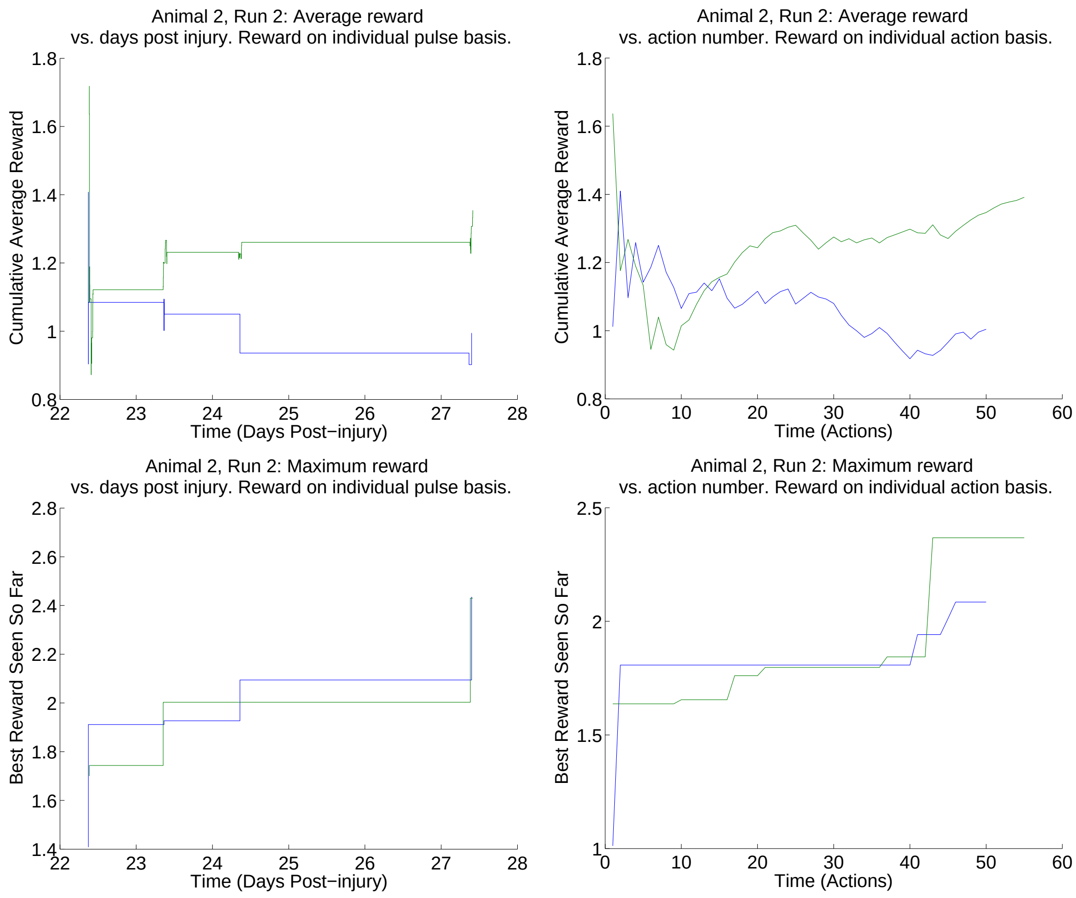


Figure 4.6: Human experimenter's (Blue) and Algorithm's (Green) reward (Peak-to-Peak amplitude of MR evoked by a 5V stimulus at 1Hz, in mV) in Run 2 of animal 2. The algorithm's average reward is again typically superior to that of the human experimenter, while maintaining competitive maximum reward, as of similar time or action index.

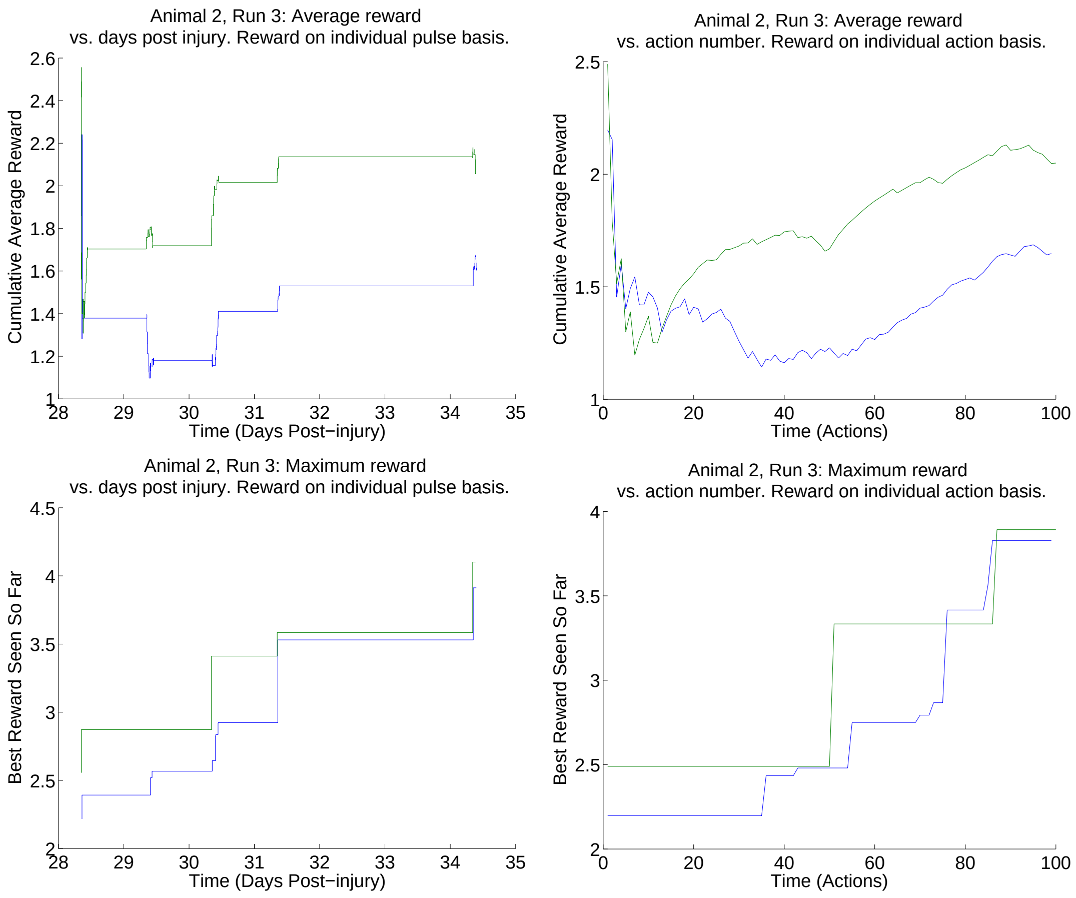


Figure 4.7: Human experimenter's (Blue) and Algorithm's (Green) reward (Peak-to-Peak amplitude of MR evoked by a 5V stimulus at 1Hz, in mV) in Run 3 of animal 2. Once again, the algorithm's average reward is typically superior to that of the human, and the algorithm's maximum reward is competitive or better.

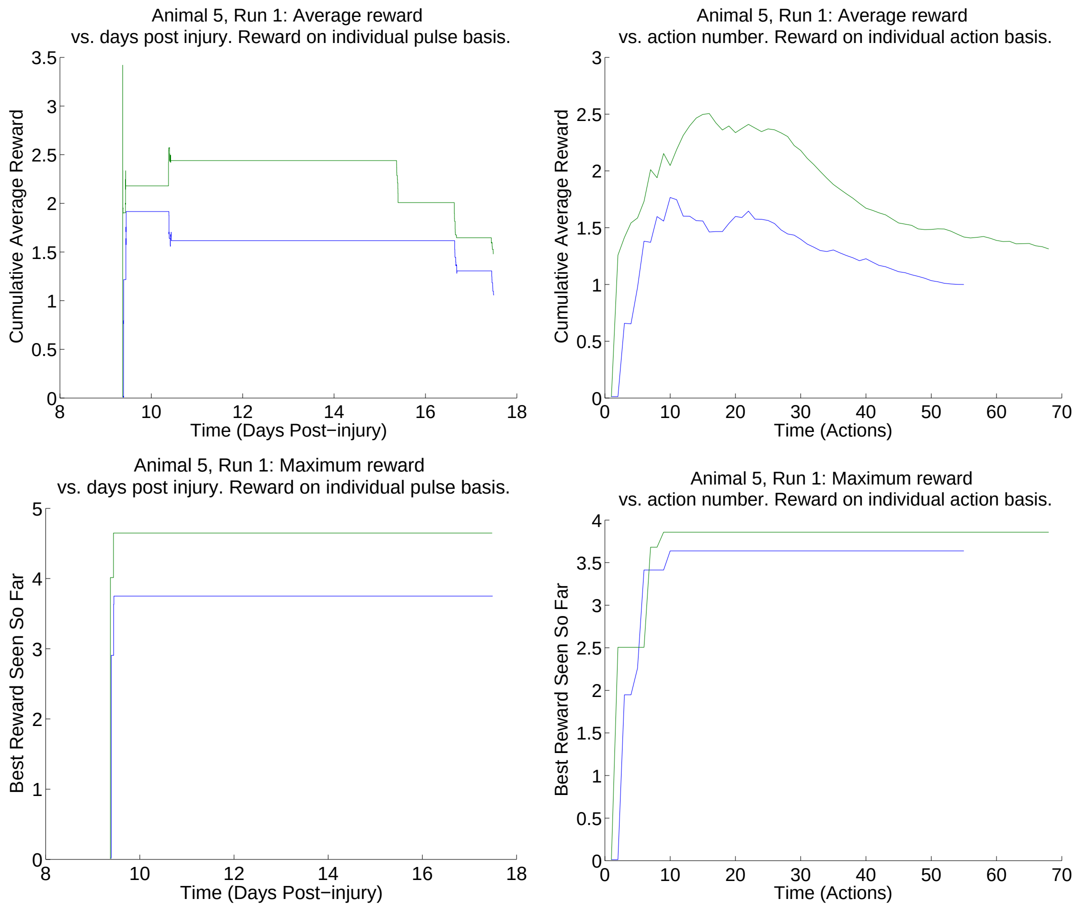


Figure 4.8: Human experimenter's (Blue) and Algorithm's (Green) reward (Peak-to-Peak amplitude of MR evoked by a 5V stimulus at 1Hz, in mV) in Run 1 of animal 5. For animal 5, 20 pulses were delivered per action. The algorithm shows substantially stronger performance than the human experimenter in both average and maximum reward. Note that the human experimenter did not execute any experiments on the 15th day post-injury (P15), whereas the algorithm was able to execute three batches, or 15 stimuli. Alternate versions of the action-based reward plots are presented in Appendix C. P15-17 all showed substantially reduced evoked potential amplitudes relative to days P9-10, as visible in Figure 4.4(b), producing the decline in average reward apparent in (a) and (b).

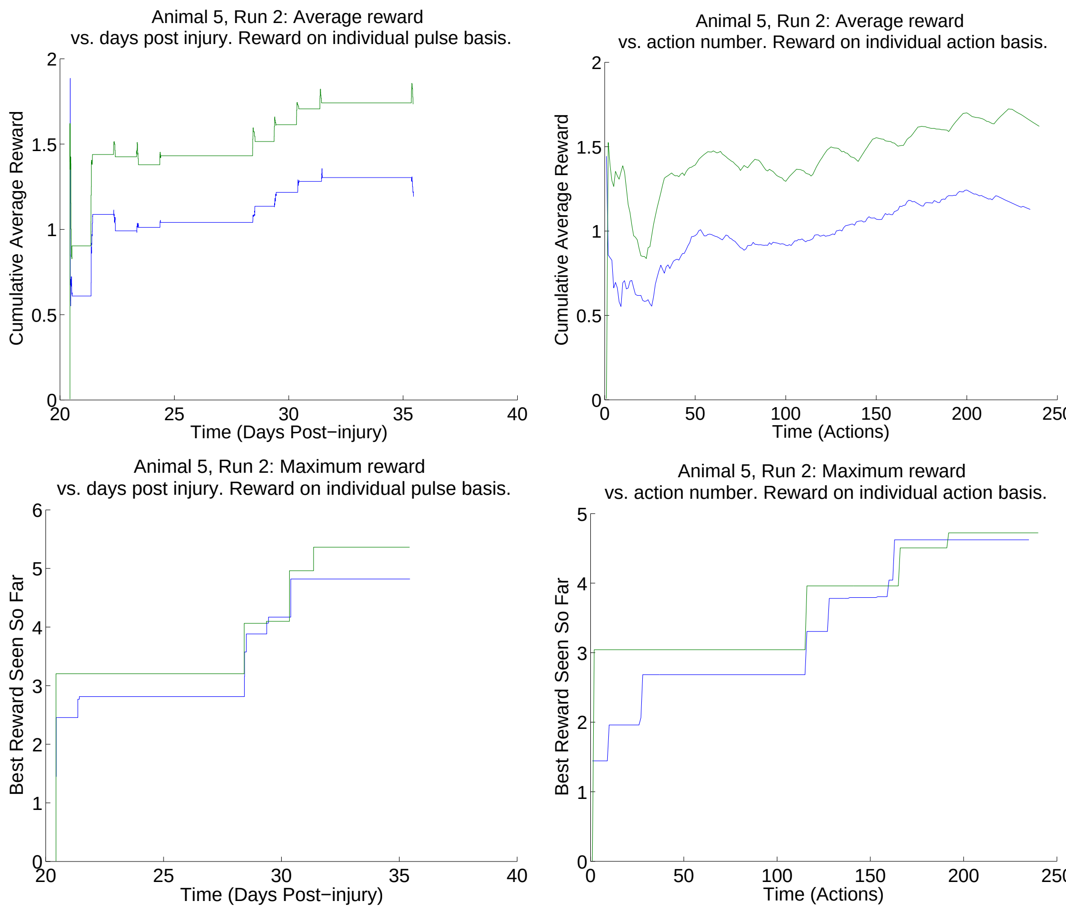


Figure 4.9: Human experimenter's (Blue) and Algorithm's (Green) reward (Peak-to-Peak amplitude of MR evoked by a 5V stimulus at 1Hz, in mV) in Run 2 of animal 5.

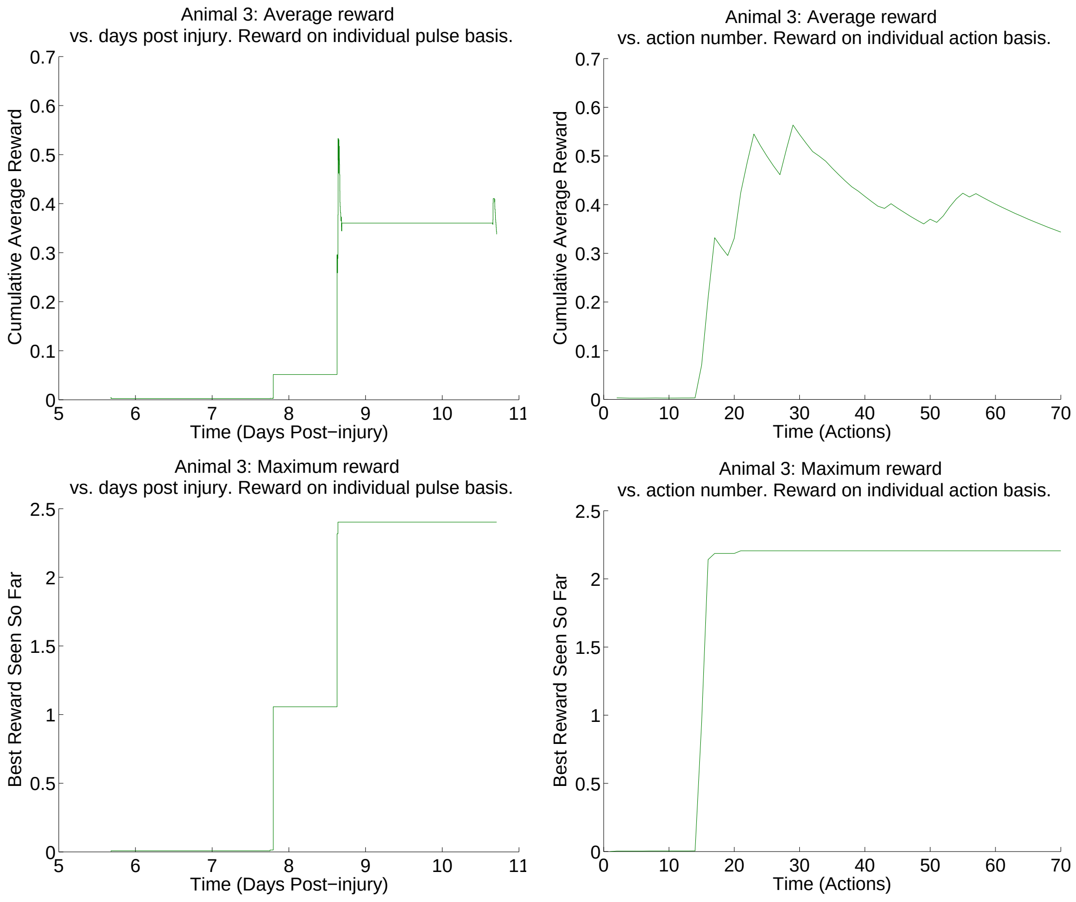


Figure 4.10: Algorithm's reward (Peak-to-Peak amplitude of MR evoked by a 6V stimulus at 1Hz, in mV) in animal 3, the first parylene array animal and the first fully-closed-loop experiment conducted in this work. In this experiment, the first conducted, only one batch of actions (5 individual stimulus combinations) was conducted on day P5, producing the apparently poor performance until part way through the second session, on the evening of P7.

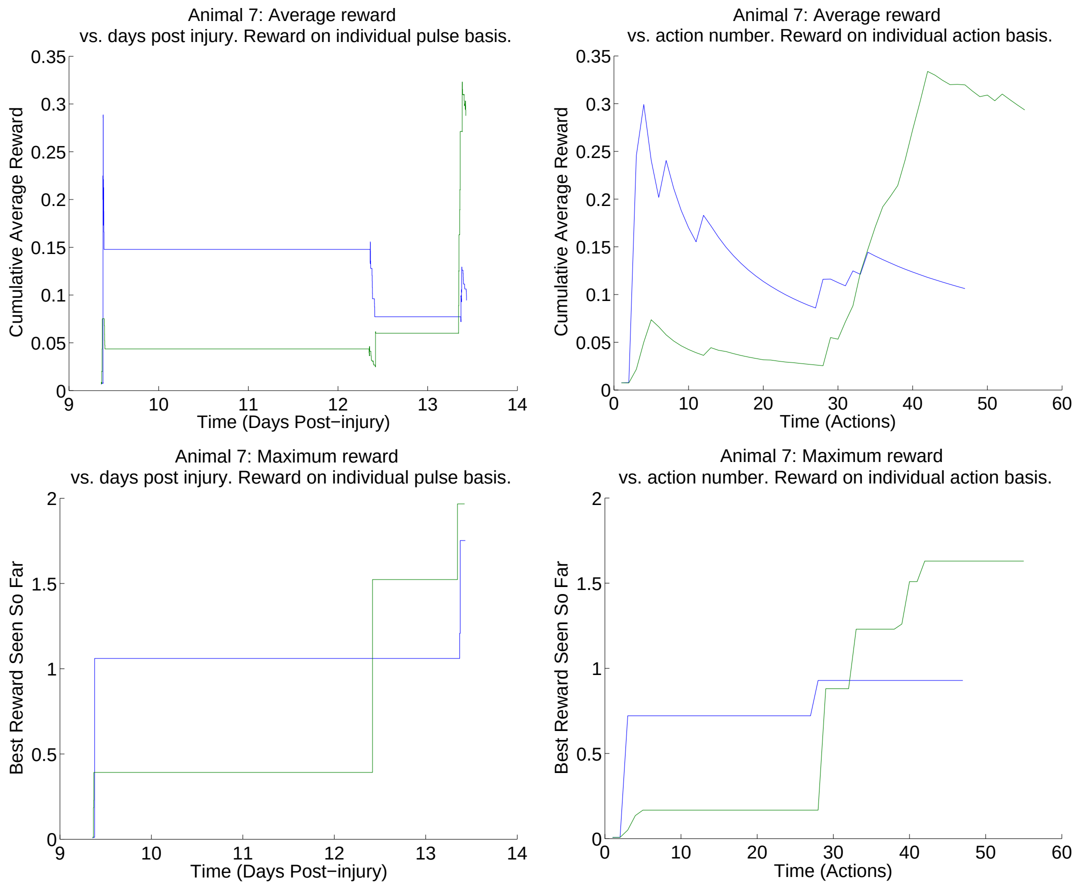


Figure 4.11: Human experimenter's (Blue) and Algorithm's (Green) reward (Peak-to-Peak amplitude of MR evoked by a 7V stimulus at 1Hz, in mV) in animal 7, the second parylene array animal. For animal 7, 20 pulses were delivered per action. Note that on day P9, both the algorithm and human were limited to two batches, due to the animal's unfamiliarity with the training paradigm. This meant that the algorithm only received feedback from the first batch for the purpose of making decisions on P9. Similarly, on P12, testing ended before the fourth batch by the human experimenter. Alternate versions of the action-based plots, which compensate for these missed actions, are presented in Appendix C. While the human experimenter found three configurations on the first day which produced relatively strong responses, the algorithm did not find any such responses. This not surprising, given the flat prior of the algorithm and the size of the search space (666 pairs) as compared to the 10 stimuli administered. The algorithm first found a relatively strong response (from configuration C6_A9) within the last batch of day P12. On P13, the algorithm made choices to heavily exploit neighbors of this configuration throughout batches 1, 2, and 3. During batch 5, and to a certain extent, during batch 4, the algorithm resumed exploration of configurations which bore little resemblance to C6_A9.

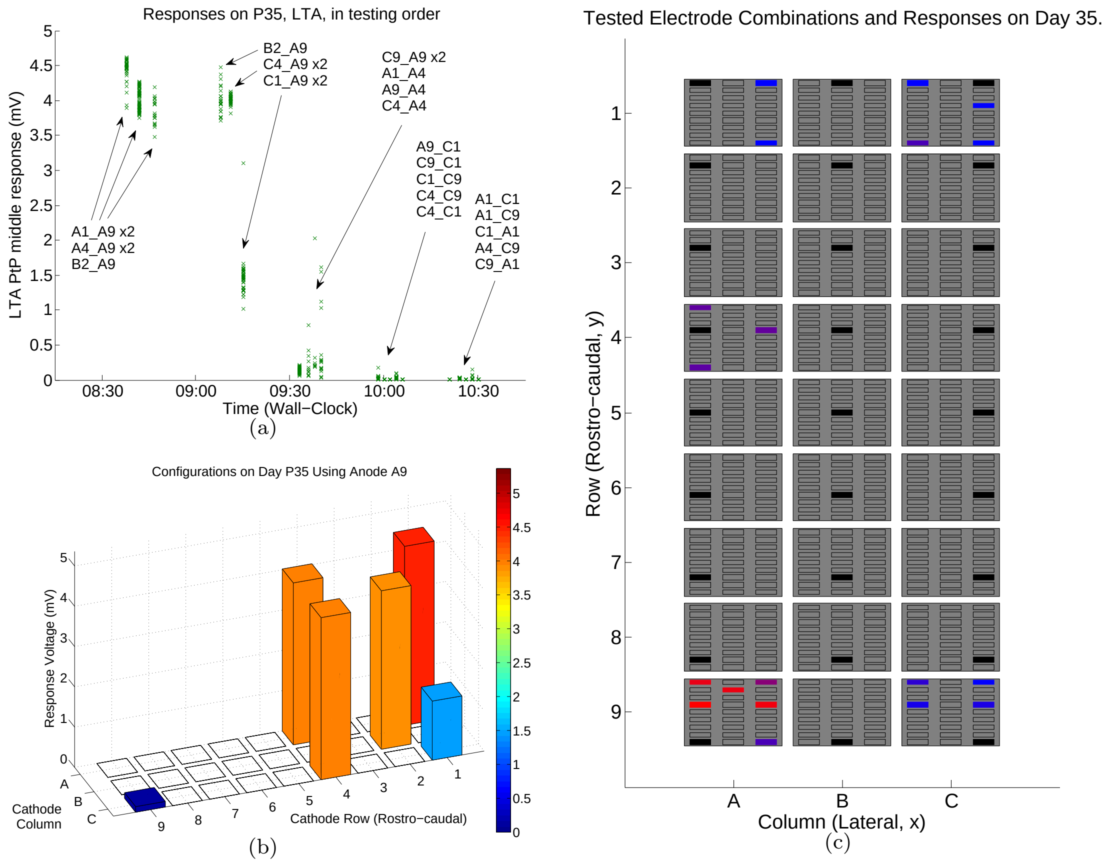


Figure 4.12: Left TA middle responses (mV, peak-to-peak amplitude) to algorithmically requested stimuli, animal 5, run 2, day P35. (a): Responses shown according to the time of commencement. Each point corresponds to a single evoked potential, i.e., a stimulus pulse and response. Note that the algorithm requests stimulus configurations in batches of 5, possibly including double repeats of some actions, and that the algorithm chose stimuli in the approximate order of their efficacy. When all six options (A1, A4, B2, C1, C4, and C9) to pair as cathodes with anode A9 are exhausted, the algorithm is then forced to use other anodes. The sensitivity of the responses to changing the anode suggests that in future experimental preparations, greater density of potential anodes near A9 is required. (b): Responses to configurations with A9 as the anode, averaged over all individual pulses for each configuration. Both color and bar height designate amplitude of response, in mV. The rostral cathodes generally paired effectively with A9 as an anode, while the combination C9_A9 was ineffective. This may suggest that the broad rostro-caudal region of stimulation is important, or that a small, but crucial target for stimulation is rostral to the 9 row and caudal to the 4 row. (c): Relative strength of pulse-averaged peak-to-peak responses, shown with respect to spatial location of the stimulus on the cord. The large boxes show anode location and the smaller boxes within the anode boxes correspond to cathode location; the extreme lower left box corresponds to (b). Red corresponds to the strongest responses seen on P35, blue to the weakest (nearly 0 mV), and purple to intermediate response strength. This pattern shows a relatively diverse search, combined with exploitation of configurations with A9 as the anode. Note that configurations using C9 (on the right side of the spinal cord, and extreme caudal end of the array) and a rostral cathode failed to elicit strong responses; the middle response in the TA appears to be quite sensitive to the lateral location of the anode as well.

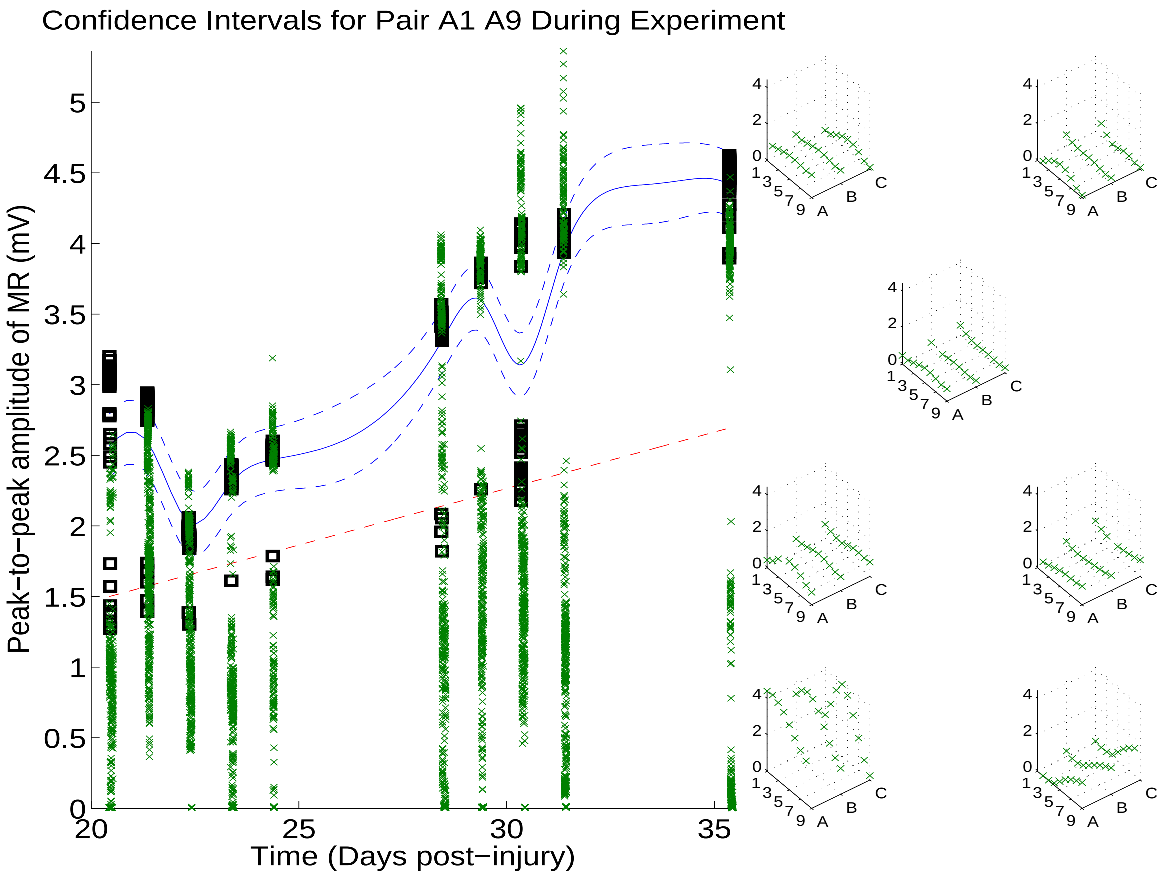


Figure 4.13: Retrospective over the entirety of run 2 for animal 5, focusing on algorithmic predictions for the stimulus pair A1_A9 during the experiment. These retrospective plots were used as part of the process for hand-fitting the covariance functions. The left panel shows the mean (solid) and ±1 standard deviation confidence intervals (dashed) for the Gaussian process posterior over the response function f(A1,A9,t), with respect to the time t post-injury. A black square denotes an observation of a response evoked by the pair of interest, A1_A9, and a green 'x' denotes an observation of a pulse for any other configuration. These other configurations may be more or less "distant" from A1_A9 in terms of their covariance under the specified kernel function. The prior mean of the entire GP with respect to time (which is invariant to the stimulus configuration) is shown as a red, dashed line. The right panel shows the predicted spatial mean function 5-6 minutes after the last observation, roughly when another batch could have begun. Each subplot corresponds to a possible anode. These anodes are A1, A4, and A9 descending the left column, B2 in the center column, and C1, C4, and C9 in the right column. Within each subplot, the isometric views show variation of the predicted mean peak-to-peak response (vertical axis, mV) over the cathode location (rostro-caudal on the left side, lateral on the right). All cathodes (i.e., A1-A9, B1-B9, C1-C9) are shown for ease of visual assessment, even though this animal has implanted with a wired array and thus only those locations listed above for the anodes were available.

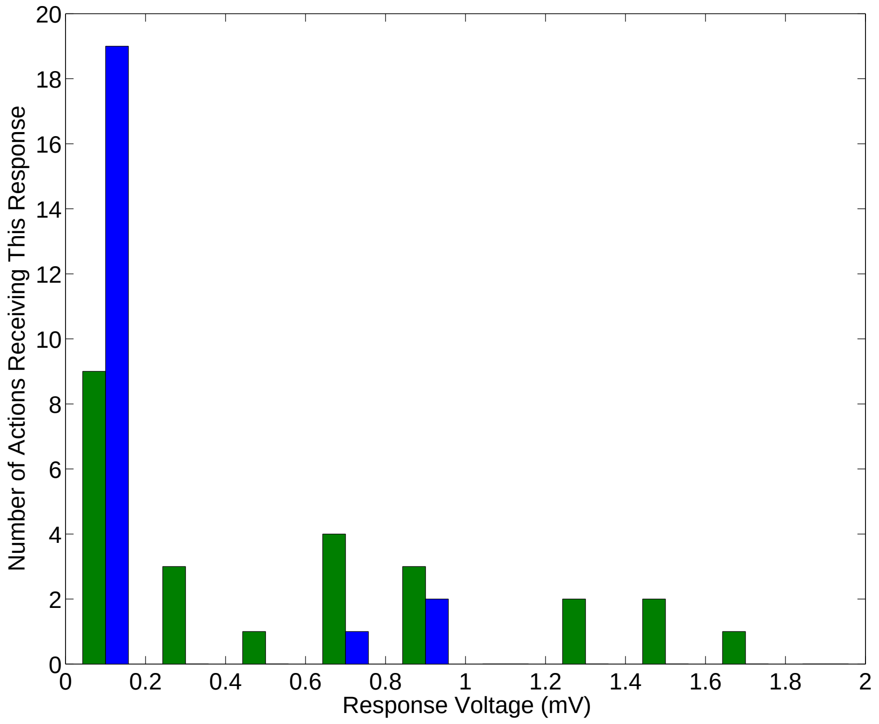
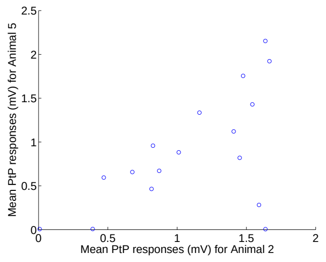
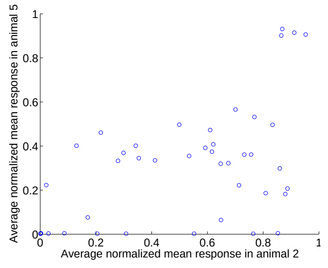


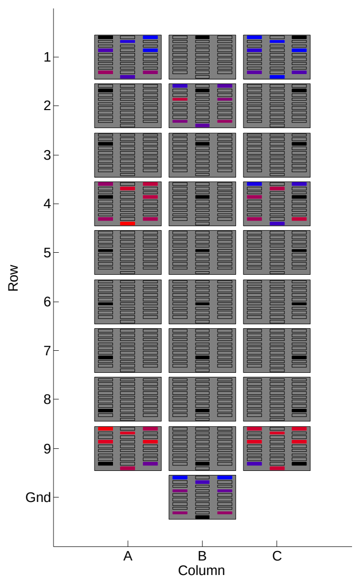
Figure 4.14: Rewards (peak-to-peak evoked potential amplitudes during the MR period) obtained by the algorithm and the human experimenter on day P13 of animal 7's experiment. The algorithm initiated 25 actions (17 unique pairs of electrodes) and the human initiated 23 actions (all unique). Note that the algorithm devoted substantially more actions to exploiting strongly-responding stimuli, and found several such stimuli.



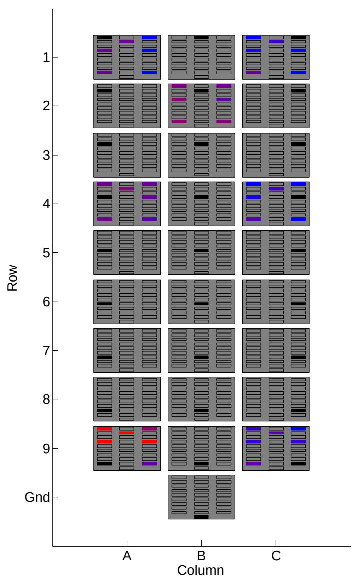
(a) P22



(b) Mean of Each Day's Normalized Mean

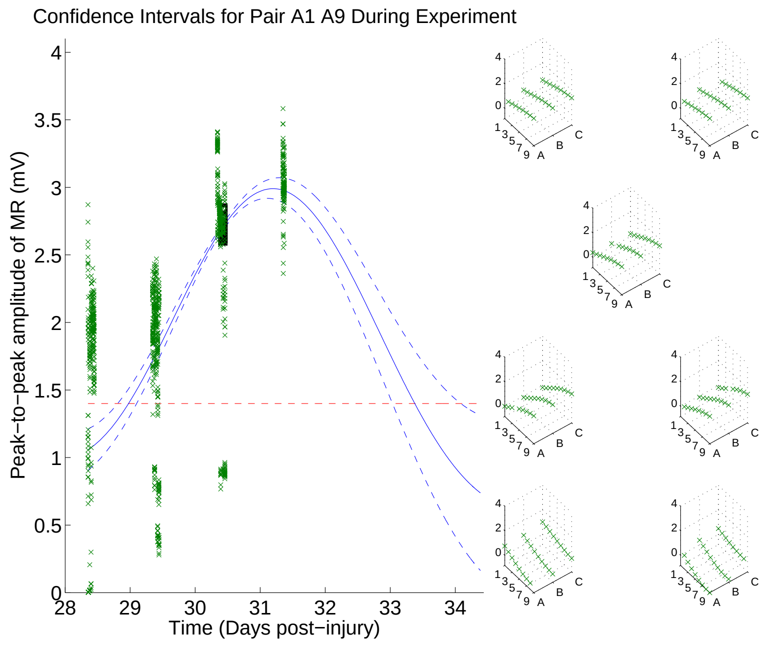


(c) Animal 2

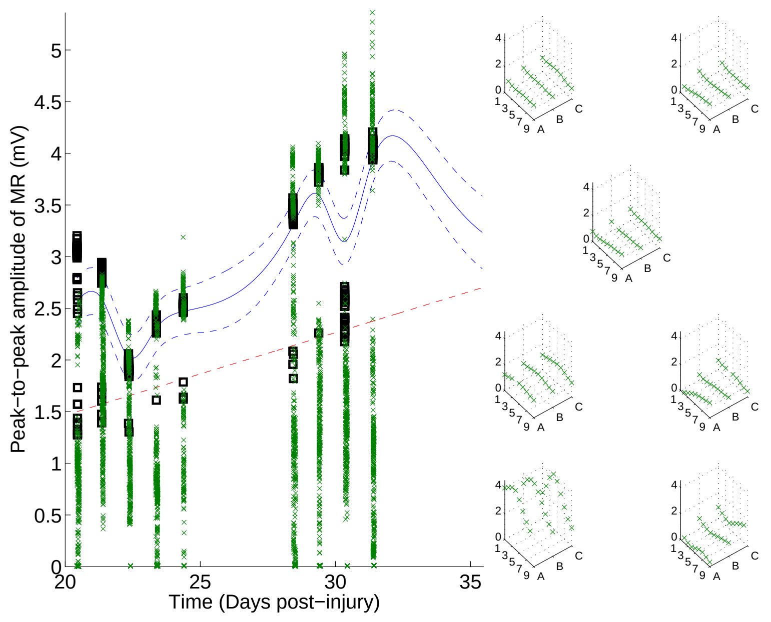


(d) Animal 5

Figure 4.15: Comparison of stimuli and responses observed in the two wired array animals, combining human- and algorithm-commanded data for each animal. (a): On many days, response amplitudes showed similar patterns across animals 2 and 5. The absolute amplitudes of responses in animal 2 were somewhat smaller than those in animal 5 (see Figure 4.4). For most stimuli, there was qualitative agreement in response strength, excepting those for which animal 2 had much stronger responses than in animal 5. (b): For any day on which at least 15 distinct stimuli were present from each of animal 2 and animal 5, normalized mean responses were computed for every pair of electrodes tested that day. For each configuration which was tested on any such day, the mean across such days of the normalized mean response was computed; all bipolar stimuli fell within this set. The mean normalized mean responses are shown. The correlation coefficient for these distributions is $r = 0.5151$; with the removal of the cluster of outliers in the upper right (those with A9 as the anode), this decreases to $r = 0.3510$. (c) & (d): Mean normalized mean response amplitudes for each configuration, from both animal 2 (c) and from animal 5 (d), shown with respect to location on the array; color corresponds to mean normalized mean response strength, where red is close to 1 and blue is close to 0, major box represents anode location, and minor box represents cathode location. Configurations present on any eligible day are shown, regardless of whether or not they were tested on that day in both animals. The subset of highly excitable configurations for animal 2 is strikingly broader than that for animal 5. Note that configurations including the ground wire (i.e., monopolar stimuli) were tested by the human experimenter in animal 2, and are shown using the bottom-most box in each representation of the array (large representation for anode or small representation for cathode).



(a) Animal 2, run 3



(b) Animal 5, run 2

Figure 4.16: The consequences of undesirable kernel characteristics; posterior predictions. The left panel for each subfigure focuses on A1_A9. See Figure 4.13 for more details. (a): The posterior predictions over the stimulus space as of the beginning of day P34 in animal 2, run 3 (based on data acquired on P31 and earlier) demonstrate a severe undershoot in the posterior predictions relative to actual performance. This is a consequence of the smoothness characteristics of the squared-exponential kernel, the time-lengthscale used, and the low noise assumed. The poor spatial predictions (particularly evident in the lower left subplot of the right panel, representing the posterior mean over configurations using A9 as an anode) caused erratic sampling, visible as broadly distributed rewards at the right of Figure 4.4(a). (b): Using the hybrid kernel described in Section 4.5, the posterior predictions at the start of day P35 in animal 5, run 2 (based on data from P31 and earlier) do not display this same pathology, and thus strong queueing behavior was present, as shown in Figure 4.13.

Chapter 5

Toward Human Studies

5.1 Organization

The ultimate goal of this line of research is to develop a new methodology for automating human SCI therapy. This chapter lays out a roadmap of issues which must be resolved before full-scale human applications can be realized and provides suggestions for extending the approach of Chapter 4 to human experiments and therapy. Section 5.2 provides an overview of the existing human SCI therapy experiments to which this work could be applied. The experimental framework for the pilot studies executed so far is laid out in Section 5.3. Discussions of several specific technical issues and the solutions employed follow in Section 5.4. Section 5.5 gives results of the pilot experiments and discusses the significance of these results for the continued development of this approach. Finally, a number of extensions to the current techniques are discussed in Section 5.6.

5.2 Prior Human Experiments

The following discussion is based upon the work of our collaborators at the Frazier Rehab Institute and University of Lousiville in Louisville, Kentucky (see Harkema et al., 2011). Three major different types of experiments under multi-electrode stimulation are undertaken at Frazier Rehab: supine, standing, and stepping experiments. These experiments are carried out in SCI humans with paraplegia, each implanted with a RestoreAdvanced neurostimulator (Medtronic, Minneapolis, MN) connected to a Specify 5-6-5 electrode array (Medtronic) positioned over the lumbar enlargement of the spinal cord. This dual-component device was originally developed for chronic pain therapy, but has been adapted to this application.

The supine experiments involve the participant lying in a supine position as the electrical stimulus is changed to any of a variety of configurations in the stimulus space; the free parameters include the combination of electrodes used as cathodes and anodes, and the voltage, frequency, and pulse-width of the stimulus. This experiment is intended to "map out" the spinal cord's motor pools with respect to the active electrodes. In many ways, this experiment could be treated as analogous to a multi-muscle, multi-electrode version of the current rat experiments, trying to find regions of the stimulus space which have large activations of particular muscles, or which produce a pattern of muscle activation matching a specified target. For this reason, these experiments might be a natural extension of the work of Chapter 4. However, the supine experiments are uncomfortable and boring for the participants; thus, only a limited amount of data from them has been collected, and relatively few opportunities to apply a machine learning algorithm to the supine experiments in closed-loop would be available.

Standing and stepping experiments present a substantially different sort of challenge than the supine experiments; in particular, the maintenance of a complex motor behavior is a much more challenging problem than the creation of a particular pattern of muscle activation, or the maximization of a particular muscle's activity. While it may be possible to describe stable standing as a particular, relatively constant pattern of muscle activation, it may also be that the body's muscular responses to perturbation are actually more important than those in equilibrium. Extracting this sort of information from EMG in an automated fashion may be very difficult. Stepping is a complex, cyclic activity, which, as in animals, is difficult to grade effectively, particularly at short timescales. Motion capture may be a useful means of assessing stepping performance, but motion capture is very time-consuming to manually process and so an automated motion capture analysis system may be necessary. As with standing, it may be that what is truly desirable in stepping is not simply the basic pattern of motor activity, but the ability to respond to perturbation while stepping. Fortunately, the basic stepping kinematic and motor pattern is fairly distinctive and responses to perturbations are relatively small, such that a performance measure could likely be created which responded to grossly correct or incorrect stepping, rather than needing to focus on the fine details of responses to rare and random events.

5.3 Pilot Applications of GP-BUCB to Human SCI Therapy: Introduction

Our pilot experiments focused on the use of GP-BUCB and variants of that algorithm in the context of stand training. While standing is a rhythmless motor behavior, with difficult-to-quantify success, human standing under epidural stimulation is somewhat understood and stand training is beneficial for the patients. As an additional point in favor of standing as an experimental target, stand training is conducted independently by the patients at home as an exercise, suggesting that monitoring and optimizing standing could fit into the daily routine of SCI patients undergoing EES therapy.

During an experimental session, the participant stands with assistance from the stimulator. For lateral stability and safety, the participant stands in the middle of a U-shaped apparatus, a stand frame, which can also be used by the participant to assist the sit-to-stand transition. One important use of the standing sessions in the clinic is the modification of stimulus parameters to improve standing performance. These parameters, including the set of active electrodes and their assigned polarities, the master stimulus voltage (the RestoreAdvanced either applies a voltage of $-v$ or v to each active electrode, where v is the master voltage), and the stimulus frequency, may be varied during the experiment. Since the effect of changing the stimulus is nearly immediate, observations of the patient's responses can potentially provide feedback on the resulting performance. Previously, this feedback process has involved human expert experimenters interacting with the patient, observing the patient visually, and monitoring the EMG activity in the patient's leg muscles. Using these observations, the clinicians and scientists carefully and gradually modify the stimulus to aid standing, while also maintaining safety and the efficacy of the session as exercise.

One major constraint during this optimization process, in part due to the limitations of the repurposed Medtronic hardware, is that the stimulus must be temporarily stopped when the pattern of active stimulus electrodes is changed. This means that during the interim the participant must either support him- or herself (typically with the arms alone) or must sit. This transition may also be disruptive to the neurological state of the spinal cord. Radical transitions may cause collapse from standing, and, even for more gradual transitions, the spinal cord requires on the order of seconds to one minute to acclimate to a new set of stimulus parameters. This potential for disruption is the reason the stimulus parameters are changed slowly. Given these constraints, the pilot experiments have so far studied the problem of exploring and exploiting over the space of master voltage and stimulus frequency, with a fixed combination of active stimulus electrodes during this process and only slow changes in voltage and frequency.

A description of the pilot experiments so far performed follows. In Section 5.4, the necessary mathematical infrastructure is described, including performance measures (Section 5.4.1), some required extensions to the GP-BUCB algorithm (Section 5.4.2), and the novel covariance functions which were created for this problem (Section 5.4.3). Preliminary results and a discussion of these are presented in Section 5.5.

5.4 Mathematical Methods

As in Chapter 4, it is necessary to make modifications to the existing theoretical and mathematical framework in order to produce an experimentally useful implementation. In particular, formulating a useful measure of standing performance is a very challenging problem; the methods used in the preliminary experiments are described in Section 5.4.1, but even these methods have not proven entirely satisfactory. Some alternatives are discussed both in Section 5.4.1 and in Section 5.6.2. The problem of making decisions in a rigorous fashion using these alternative performance measures is also not trivial; this problem is examined and solutions used in the preliminary experiments are described in Section 5.4.2. A further complication, described and addressed in Section 5.4.3, is the need to specify problem-specific covariance functions. All code implementing the algorithm was implemented in MATLAB (The MathWorks, Inc., Natick, MA).

5.4.1 Performance Measures

As schematically shown in Figure 1.3, one of the key components of the EES system is the choice of what sensor information and methods are used to quantify performance of the stimulus, both in the execution of the desired motor task and in terms of any other variables of interest, e.g., the patient's comfort. This decision is crucial. Clearly, if the reward function optimized by the algorithm is not reflective of actual performance, the algorithm cannot optimize performance; at best, it will optimize this putative performance measure. The difficulty lies in choosing an appropriate performance measure. In Chapter 4, it is assumed that an appropriate performance measure is the peak-to-peak amplitude of activation of the left tibialis anterior muscle, which has the advantage of being a scalar function. However, this function might be maximized by stimuli which are uncomfortable or are otherwise unacceptable in humans, or which are simply not useful therapeutically. A number of possible alternatives are discussed in the following sections.

5.4.1.1 Subjective Ratings

One reasonable way to optimize the performance of a stimulation system is to simply ask the user for his or her opinion; if the user's ratings are both repeatable and reflective of actual therapeutic utility, than this method requires relatively little infrastructure, as well as giving patients a way to control their own therapy. User feedback could perhaps be beneficial in terms of reducing the number of clinical visits required, reducing patient frustration, and increasing the patients' sense of agency in their own recovery. It is quite plausible that patients could be trained in the use of a grading scale or rubric, as simple as integers from zero to ten, which would yield repeatable results; similar methods have been used for gait analysis in animals (e.g., Basso et al., 1995), and a human user could much more effectively assess some important aspects of the stimulus response, such as discomfort, than could a fully autonomous system. Under such a system, the object being regressed upon as a function of stimulus $\boldsymbol{x}$ is $f(\boldsymbol{x})$, the user's likely rating for that stimulus. In the preliminary experiments described here, a very simple zero to ten system was used with both subjects.

Several difficulties with such ratings are apparent. First, a rating system would require the aforementioned patient training, and so would likely not be useful without the user acquiring a substantial degree of experience with the stimulator and their personal, subjective experience of the system's effects. This characteristic would render it quite difficult to use such a simple grading system during the initial training period, already the most challenging period for autonomous search of the stimulus parameter space. Secondly, it would likely be very difficult to develop a system under which the quantitative grades assigned by the user corresponded directly with utility. Certainly, it might quite plausibly be possible to develop a system which was ordinal, i.e., the nominal ordering of the categories in the rubric was correct in terms of utility, but it would likely be substantially more challenging to develop a system under which, for example, the difference in utility between an 8 and a 7 was the same as that between a 4 and a 3. Some discussion of the difficulties inherent in this non-equivalence of the rating and utility are discussed in Section 5.4.2.1. Further, if different aspects of the stimulus response are to be graded, e.g., performance and comfort, it might be difficult to determine how to balance these aspects against one another. A more precise and expressive rating system would also likely require a greater time for the user to assess and respond to any given stimulus, slowing the rate of testing. Without this detail, however, it might be very hard to use ratings to diagnose necessary changes to improve the stimulus, a major disadvantage as compared to muscle-activity-based systems. Additionally, if the stimulus applied was uncomfortable or dangerous, a system without its own sensing capabilities could harm the user without giving him or her the time or ability to respond appropriately. This would require careful design of safety systems. However, these difficulties aside, a performance measure based on subjective ratings is a reasonable approach for inexpensive and simple home use, particularly if it could be incorporated as part of a two-tier system for clinical and home adjustment of stimulus parameters. Medtronic's line of neurostimulators for chronic pain managment, for example, includes just such a two-tier system, though this system is for direct control of stimulus parameters, rather than user feedback; they manufacture both a complex and capable clinical control device, the N'Vision, and the MyStim, a simpler, less capable controller given to patients for home use. A simple, handheld unit like the MyStim would be an excellent interface device for a rating-based, closed-loop system.

5.4.1.2 Grading Vector-valued EMG

Performance must necessarily be quantified as a scalar in order for optimization of the performance to be a meaningful concept. Because the idea of standing performance itself is somewhat difficult to define, attempting to measure and work with performance directly is difficult, thus leading to the idea of using a holistic surrogate such as a user rating, discussed previously. Another alternative is to define performance as a known function of measurable physical quantities. Indeed, it is plausible that these measured variables, e.g., EMG signals, can be chosen in such a fashion as to have distinct patterns which result in useful, high-level behavior, such as naturalistic standing.

Mathematically, such a system must regress upon a vector-valued object, i.e., $\mathbf{f}(\boldsymbol{x}) \in \mathbb{R}^n$, such that there are n (possibly linked) observations which arise as the result of a single input $\boldsymbol{x} \in D$. In the case of interest, $\mathbf{f}(\boldsymbol{x})$ is vector of extracted features of several muscles' responses to a single EES stimulus $\boldsymbol{x}$. From such a vector-valued object, one fairly natural way define the reward is

$$r(\mathbf{f}(\boldsymbol{x})) = -\sqrt{(\mathbf{f}(\boldsymbol{x}) - \mathbf{t})^T W (\mathbf{f}(\boldsymbol{x}) - \mathbf{t})}, \tag{5.1}$$

where W is a symmetric, positive definite penalty matrix, such that $r(\boldsymbol{x})$ is the negation of a weighted 2-norm of $(\mathbf{f}(\boldsymbol{x}) - \mathbf{t})$, the difference between $\mathbf{f}(\boldsymbol{x})$ and a target response $\mathbf{t}$ in $\mathbb{R}^n$. Careful selection of the feature representation, W, and $\mathbf{t}$ should allow relatively precise tailoring of the reward function to reflect more or less acceptable deviations from a particular motor behavior. In order to formulate a GP-UCB-like algorithm which uses such a reward function, it is necessary to create a regression model using Gaussian processes that captures the variation of $\mathbf{f}(\boldsymbol{x})$ with respect to $\boldsymbol{x}$. If these features are chosen appropriately, it may be relatively easy to incorporate expert knowledge, e.g., by the selection of prior means and covariance functions.

One of the most important problems associated with such a model for $\mathbf{f}(\boldsymbol{x})$ is the problem of appropriately interlinking the individual GPs corresponding to the different entries in the vector $\mathbf{f}$. Anatomically, the nerves, interneurons, and motor pools responsible for activity in different muscles are located in close proximity to one another, such that nearby structures may be brought to threshold at roughly the same amplitude of stimulation. Thus, different variables of interest can be expected to co-vary due to their co-dependence on the same physical phenomena. Capturing this structure is important, and can lead to greater efficiency, particularly in the case in which sensors or channels of information are intermittently unavailable. One can think of the identity of the muscle or feature as a piece of side information supplied to the model. Recently, Krause and Ong (2011) approached this problem of using side information from the GP-bandit perspective. In their setting, in each round, the algorithm is presented with a *context* m_t from the set of possible contexts M and must then choose an action $\boldsymbol{x}_t$ to take in that round to maximize reward (where the reward depends on the context and the action). Krause and Ong formulate an algorithm, CGP-UCB, which uses covariance functions on the context space M to enable regression on a *contextual Gaussian process* over $D \times M$. Decisions in the SCI therapy setting yield a reward which is assumed in this section to be dependent on all of the contexts simultaneously, thus making employment of the complete CGP-UCB algorithm inappropriate, due to its assumption that a single context is in effect at any time. However, the core contextual GP model upon which CGP-UCB is based holds substantial promise for the multi-muscle or multi-characteristic problem. Using this model requires covariance functions which are capable of expressing the covariance of $f(\boldsymbol{x}, m)$ and $f(\boldsymbol{x}', m')$. Some ideas for anatomically appropriate covariance functions are discussed in Section 5.4.3.

In the preliminary experiments, some trials were also carried out using a reward function based on a GP model of an EMG feature vector over the allowed stimulus space. This necessitated formulating a decision rule which used the reward function and the GP model together to trade off exploration and exploitation. The chosen decision rule is discussed in Section 5.4.2.2.

5.4.2 Algorithmic Extensions

Some important extensions to the theory described by Srinivas et al. (2010) and Chapter 3 should be considered in light of the requirements for human application. As in the case of GP-BUCB, it is useful to consider alternate decision rules, as well as the circumstances under which probabilistic guarantees regarding the convergence of algorithms using a GP posterior and these decision rules can be demonstrated. Another important extension is the creation of an algorithm for selecting smoothly varying paths of stimuli, rather than the unconstrained transitions in the conventional GP-BUCB and GP-AUCB algorithms.

5.4.2.1 Divorcing Reward from the Function Regressed Upon

In GP-UCB, GP-BUCB, and GP-AUCB, the response function f that is actually measured and regressed upon is the same as the reward r; thus the GP-UCB decision rule

$$\boldsymbol{x}_t = \underset{\boldsymbol{x} \in D}{\operatorname{argmax}} [\mu_{t-1}(\boldsymbol{x}) + \alpha_t^{1/2} \sigma_{t-1}(\boldsymbol{x})] \tag{5.2}$$

is sensible, because it trades off the expected reward $\mathbb{E}[r(f(\boldsymbol{x}))] = \mathbb{E}[f(\boldsymbol{x})] = \mu_{t-1}(\boldsymbol{x})$ with a measure of the information to be gained by making the corresponding observation. It is not clear, however, that the reward is always the correct object upon which to regress; in particular, it may be that the reward is a quantity which is difficult to measure (e.g., standing performance) or even not directly observable, while there may be one or more objects which are relatively easy to measure (e.g., evoked potential amplitude, used in Chapter 4, and subjective ratings, proposed in Section 5.4.1.1), and which are strongly related to the reward. In addition, it may be that the reward function does not easily lend itself to being modeled as a GP, perhaps because covariance functions are very hard to specify or the reward is erratic, whereas the available surrogates are more easily modeled. As discussed in Sections 4.1 and 5.4.1.1, it is important that these easily measured and modeled surrogates are chosen such that when they are maximized, reward is also maximized. If more is known about the relationship between the surrogate and the true reward, perhaps an algorithm which understands this relationship can exploit this knowledge to perform better than if it simply considered the reward and surrogate as equal.

One particularly interesting case, and one for which theoretical results are likely obtainable with a modest degree of effort, involves cases in which f is a scalar function, modeled as drawn from a GP, and the mapping from $f(\boldsymbol{x})$ to the utility r is a scalar function g, such that $r(\boldsymbol{x}) = g(f(\boldsymbol{x}))$. If g is a finite, Lipschitz continuous, non-decreasing function of $f(\boldsymbol{x}) \in \mathbb{R}$, with Lipschitz constant k, it follows immediately that the set $\boldsymbol{X}^* \in D$ of maximizers of f also is a subset of the set of maximizers of $r = g(f)$ over D. Further, it follows from Theorem 1 that if the algorithm is simply run conventionally, ignoring $g(f)$ and treating f as the reward, the regret with respect to $r = g(f)$ is no more than k times the regret of the algorithm with respect to $r = f$, i.e., the regret increases by at most a factor of k. If $g(f)$ is given to the algorithm, it is likely that clever design could produce an algorithm which exploits knowledge of g to do considerably better than this. While g is unknown in the subjective rating setting arising from the human experiments, a number of intriguing forms for g exist. One potentially fruitful choice for theoretical analysis is

$$g(f) = \tanh(k(f - c)),$$

or similarly, the related logistic function,

$$g(f) = 1/(1 + \exp(4k(f - c))),$$

which both saturate to both the left and right, implementing a gentle classification between success and failure at a threshold c. Other reasonable choices include a hinge loss

$$g(f) = \min[0, k(f - c)]$$

or gain

$$g(f) = \max[0, k(f - c)],$$

or piecewise continuous functions of a variety of forms. There are many other interesting possibilities, but exploitation of a properly chosen g might give quite a bit of expressiveness and flexibility to the family of UCB-based algorithms.

5.4.2.2 Making Decisions Using Vector-Valued Functions

Section 5.4.1.2 introduces the idea of modeling many characteristics of the EMG activity of multiple muscles using a contextual GP, as well as the idea of using a reward function of the form of Equation (5.1), a weighted Euclidean norm. Making decisions using this combination of reward function and contextual GP model is examined in some detail in Appendix D.1. In summary, one plausible decision rule for batch or delay selection in this case is

$$\boldsymbol{x}_t = \operatorname*{argmax}_{\boldsymbol{x} \in D} \left[-\sqrt{(\mu_{\text{fb}[t]}(\boldsymbol{x}) - \mathbf{t})^T W (\mu_{\text{fb}[t]}(\boldsymbol{x}) - \mathbf{t})} + \beta_t^{1/2} \sqrt{\text{trace}(W \Sigma_{t-1}(\boldsymbol{x}))} \right]. \tag{5.3}$$

This decision rule is related to trading off the expected reward and a quantity similar to the standard deviation of a scalar function. Another useful characteristic is that the uncertainty term (the second term inside the brackets) is non-increasing as observations are added to the decision set, meaning it can be calculated lazily. Finally, the scalar case of this decision rule reduces to the GP-BUCB decision rule, Equation 3.7, for $t \to \infty$. A version of this decision rule has been implemented in the suite of code prepared for the pilot experiments.

5.4.2.3 Choosing Paths

One important observation from early feasibility testing in the human model is that it is important to choose stimuli which form smooth sample paths, i.e., to only command gradual changes in the stimulus. This constraint is necessary because the spinal cord's neurological and functional state has a memory and abrupt changes in the stimulus result in a disruption of behavior. This suggests that, particularly in terms of frequency and voltage, it is important to plan paths through the space of candidate stimuli. From an algorithmic perspective, a problem immediately presents itself; this sort of multi-step search is likely exponentially complex in the length B of the path, since such a path search becomes a search over leaves of a tree with depth B and a branching factor of greater than 1 (typically 5 in these pilot experiments). This exponential complexity in constructing batches is the very reason the GP-BUCB and GP-AUCB algorithms use a greedy decision rule to sequentially select individual stimuli and thereby construct a batch, rather than attempting to choose one out of all possible batches. One potential (though not entirely satisfactory) solution to this problem is to require that all valid paths must follow a particular set of construction rules, and further, to restrict the construction rules such that there is at most one valid path of length B or less to any $\boldsymbol{x} \in D$, given the current state, $\boldsymbol{x}_{t-1}$; let this set of legal paths be designated L. In this case, since $|L| \leq |D|$, there are again at most $|D|$ entities among which the algorithm must choose at any given time t. Following this formulation, two suggested decision rules, both with discounted and undiscounted versions, are presented in Appendix D.3. Among these four forms, the decision rule

$$\boldsymbol{X}_t = \operatorname*{argmax}_{\boldsymbol{X} \in L} \left[\sum_{\tau=t}^{t+B-1} [\lambda^{\tau-t}(\mu_{\mathrm{fb}[t]}(\boldsymbol{x}_\tau) + \beta_{\mathrm{fb}[t]}^{1/2}\sigma_{\tau-1}(\boldsymbol{x}_\tau))] \right]$$

has been implemented for the human experiments, where any path $\boldsymbol{X} \in L$ is the ordered sequence $\boldsymbol{X} = \{\boldsymbol{x}_t, \ldots, \boldsymbol{x}_{t+B-1}\}$, $\boldsymbol{X}_t \in L$ is the selected path, and λ is a discounting rate corresponding to the probability of a failure occurring at each step, i.e., the assumed likelihood that the path must be stopped due to that step's stimulus being unacceptable. This decision rule is intended to be used to search the space of voltage and frequency, given a fixed set of active electrodes, where subjective ratings quantify the reward. Alternatively, the decision rule

$$\boldsymbol{X}_t = \operatorname*{argmax}_{\boldsymbol{X} \in L} \left[\sum_{\tau=t}^{t+B-1} [\lambda^{\tau-t}(-\sqrt{(\mu_{\mathrm{fb}[t]}(\boldsymbol{x}) - \mathbf{t})^T W (\mu_{\mathrm{fb}[t]}(\boldsymbol{x}) - \mathbf{t})} + \beta_t^{1/2}\sqrt{\mathrm{trace}(W\Sigma_{t-1}(\boldsymbol{x}))})] \right]$$

has been implemented for path-based planning using the vector-valued EMG feature representation discussed in Sections 5.4.1.2 and 5.4.2.2. A second, more fundamental problem also occurs if decisions must constitute paths; it may be possible to construct a combination of D, the construction rules, and B, such that no paths or sequence of paths can ever connect particular parts of D. Even if this does not happen, it could occur that poorly-responding zones prevent the algorithm from moving between two regions of good performance, even if both high-performing zones have been previously explored. In such situations, there is a non-zero probability that the algorithm cannot find the optimum. This is, unfortunately, a consequence of requiring legal paths, rather than allowing the algorithm to "jump," as do GP-BUCB and GP-AUCB. One partial work-around might be to allow paths to transition through an "off" state, which neighbors many other states, but this makes decision-making very complex and further divorces the algorithm from rigorous theory.

5.4.3 Novel Covariance Functions

The multipolar array configurations needed for the current and future human experiments require new, carefully structured covariance functions. The parameterization of the input space used in Chapter 4 for bipolar electrode configurations is based upon the four spatial coordinates of the single cathode and single anode used. For any two different bipolar pairs $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^4$, the covariance function $k(\boldsymbol{x}, \boldsymbol{x}')$ compares the (rescaled) Euclidean proximity of the cathodes and anodes and computes the covariance between the muscle responses to these stimuli on the basis of this proximity. The logical extension of such a covariance function to $n \geq 2$ active electrodes in a given stimulus would be to again describe the spatial locations of the active electrodes in $\mathbb{R}^{2n}$; however, this approach has many deficiencies. With several electrodes of the same polarity, exchangeability issues arise, i.e., the set of active electrodes (A1+, A2+, C9-) is actually the same as (A2+, A1+, C9-), though these two descriptions might naturally correspond to different points in $\mathbb{R}^6$. Thus, the covariance function would have to recognize the equivalence of different descriptions of the same configuration, a problem not present in the bipolar case. Worse, if the algorithm is allowed to use multi-electrode configurations with different numbers of active electrodes, it must be able to compare these objects. Indeed, a configuration with two cathodes and one anode might be functionally very similar to the bipolar configuration in which one of these two cathodes is off (i.e., is neither a cathode nor an anode). In such a case, the covariance function certainly should express the commonality between their respective responses, even though these two objects would exist in $\mathbb{R}^6$ and $\mathbb{R}^4$ respectively. Clearly, a covariance function which operates directly on vector quantities of arbitrary size is not appropriate. The covariance function should therefore act on some representation of the electrode configuration which is independent of the number of cathodes or anodes.

One natural way to create the required unified representation would be to calculate the electric field or voltage distribution created by each stimulus configuration, and then compare key characteristics of the fields generated any pair of stimuli. This is reasonable because these fields can be calculated from first principles (given good measurements of the electrical properties of the tissue) and the effects of epidural electrostimulation arise via these fields. Particularly if computational speed is an issue, very simple (i.e., closed-form algebraic) calculations may be appropriate. Fortunately, electricity is well-understood and follows simple physical laws. As an example, given a spherical electrode of radius ρ with a fixed surface voltage V, the voltage $V(r)$ at distance $r \geq \rho$ from the center of the electrode in a homogeneous medium is

$$V(r) = V\frac{\rho}{r}, \tag{5.4}$$

assuming that the infinite distance boundary condition is ground, i.e., $V(r) \to 0$ as $r \to \infty$. If several electrodes of different positions and voltages are desired (i.e., the chosen stimulus $\boldsymbol{x}$ includes more than one active electrode), an approximate solution can be obtained by summing the corresponding voltage functions, yielding a voltage value corresponding to the influence of all of the electrodes together. Given a set of target locations believed to be representative of the voltage distribution's influence on the spinal cord, the vector of voltage values $\mathbf{V}(\boldsymbol{x})$ corresponding to a stimulus $\boldsymbol{x}$ can be calculated. Since these locations are fixed, $\mathbf{V}(\boldsymbol{x})$ has the same size and meaning, regardless of how many electrodes are active in stimulus $\boldsymbol{x}$. To calculate a covariance function for $\boldsymbol{x}$ and $\boldsymbol{x}'$, it then remains to compare the vectors $\mathbf{V}(\boldsymbol{x})$ and $\mathbf{V}(\boldsymbol{x}')$ in terms of their similarity with respect to the responses of interest. One plausible way to do this is to weight some elements of $\mathbf{V}$ more heavily than others, using anatomical intuition. Using information from Sharrard (1955), as well as Sharrard (1964) and Harkema et al. (2011), it is possible to localize any of several motor pools of interest within the human spinal cord with respect to the electrode. These localizations can also be verified with respect to the array by use of the data from the supine stimulation experiments described in Section 5.2.

If the model must consider EMG from several muscles, the set of which is designated M, it is necessary to construct a covariance function which enables predicting these responses, treating the individual muscle or feature as a context. One way to do this is to create a diagonal weight matrix W_m for each muscle $m \in M$. Then, if the ith entry in $\mathbf{V}$ is at a location where electrical stimulus might plausibly exert substantial effects on the spinal inputs and circuits associated with muscle m, e.g., near the corresponding motor pool, $[W_m]_{i,i}$ should be large; conversely, if the ith entry in $\mathbf{V}$ can be expected to not be particularly influential on muscle m, the weight $[W_m]_{i,i}$ should be small. One way to do this simply is to choose the weights as the height values of a Gaussian bump centered on the location of the motor pool associated with muscle m. The re-weighted vectors $W_m\mathbf{V}(\boldsymbol{x})$ and

$W_{m'}\mathbf{V}(\boldsymbol{x}')$ can then be fed into a covariance function

$$k(W_m\mathbf{V}(\boldsymbol{x}), W_{m'}\mathbf{V}(\boldsymbol{x}'))$$

which computes the covariance of the responses to $\boldsymbol{x}$ in muscle m and the responses to $\boldsymbol{x}'$ in muscle m'. Appropriate covariance functions might be linear,

$$k_{lin}(W_m\mathbf{V}(\boldsymbol{x}), W_{m'}\mathbf{V}(\boldsymbol{x}')) = (W_m\mathbf{V}(\boldsymbol{x})) \cdot (W_{m'}\mathbf{V}(\boldsymbol{x}')), \tag{5.5}$$

or perhaps could take the form

$$k_{lin,0} = \max[0, (W_m\mathbf{V}(\boldsymbol{x})) \cdot (W_{m'}\mathbf{V}(\boldsymbol{x}'))],$$

where this second form would avoid having large, negative covariances between polarity-flipped configurations. Alternatively, rather than focusing on individual muscles, as in the vector-valued EMG case, it might be more appropriate to consider a holistic view of activity in general, as in the subjective rating case; it would then be reasonable to attempt to correspondingly use knowledge of motor pool locations to weight elements of $\mathbf{V}$, via the choice of a combined weight matrix $W_{combined}$, constructed as a linear combination of individual matrices W_m. The covariance function has been employed using both the subjective rating and vector-valued EMG modes, and results are described in the next section.

5.5 Preliminary Results and Discussion

Some fairly simple preliminary experiments have been conducted with two different human patients. In the first patient, designated ARI (who has no voluntary motor control or sensation below the level of his injury), data from two sessions were used. In the first, the human experts performed a standard procedure, a voltage and frequency sweep, using a configuration of active electrodes known to produce good standing for ARI (3+ 4+ 8+ 14+ 15+ // 9- 10-, in which cathodes form a horseshoe shape around two anodes at the caudal end of the array). This experimental session consisted of a gradual (0.2 V increment after reaching threshold) increase in voltage from 0 V to 4.6 V, with the stimulus frequency fixed at 25 Hz. After this voltage sweep, the voltage was fixed at 3.8 V, and the frequency of stimulation was increased in 5 Hz steps from 10 Hz to 40 Hz. During this process, the participant was instructed to provide a rating from zero to ten of how well he was standing, one such rating at each set of stimulus parameters. Subjective ratings from these two sweeps are presented in Figure 5.1(a). On a subsequent experimental day, the algorithm was given the data from the previous session and given a quadratic mean function chosen such that the maximum subjective

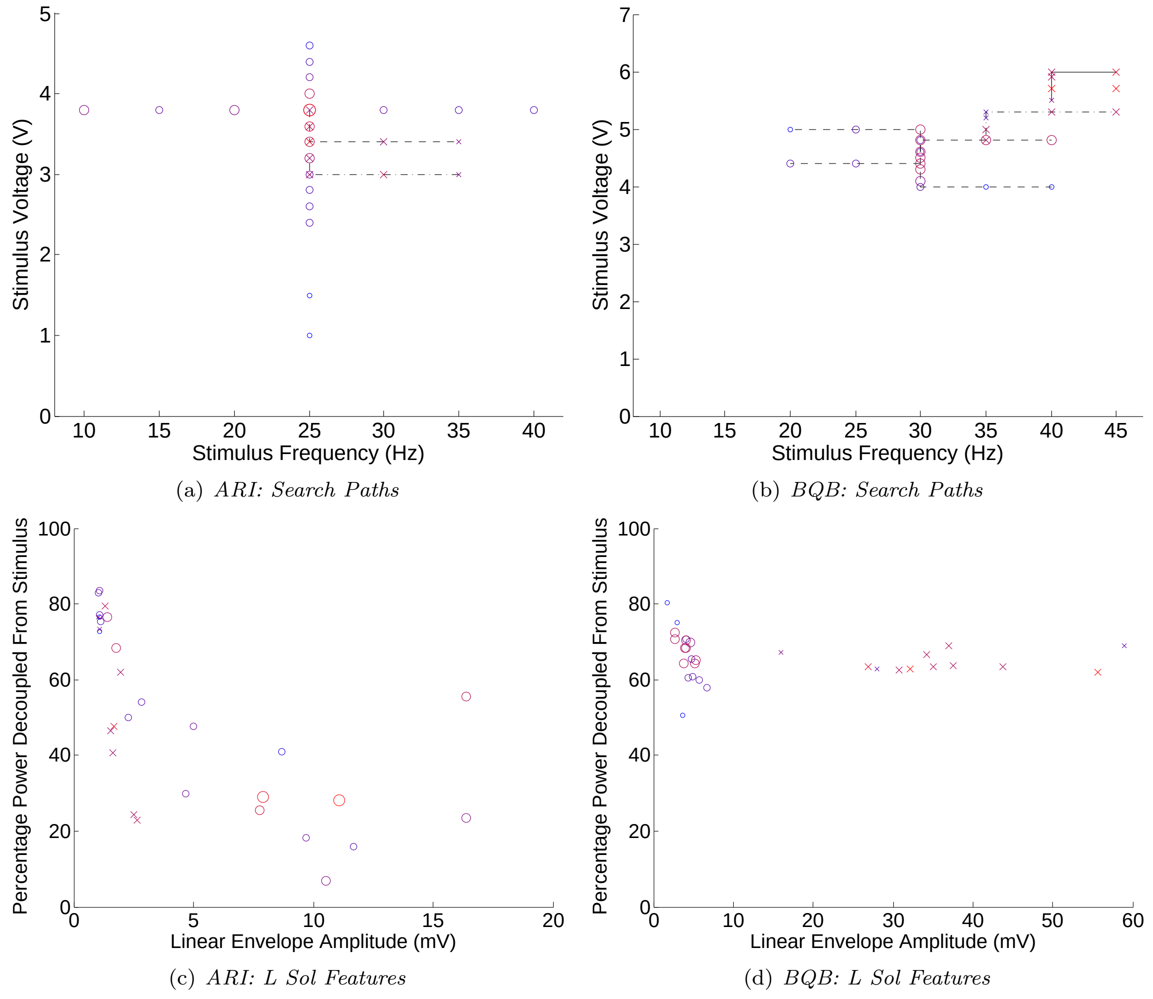


Figure 5.1: Search paths, ratings, and EMG features from experiments with participants ARI and BQB. (a): Voltage and frequency sweeps executed by human experts (circle) and the algorithm using subjective ratings (dashed, x) with participant ARI. Subjective ratings given by the participant are shown as color and size of the marker, where red and large markers are high subjective ratings and blue and small markers are low subjective ratings. Participant ARI was instructed to give ratings from 0 (least favored) to 10 (most favored) and gave ratings covering this whole range. (b): Voltage and frequency sweeps conducted by the algorithm with participant BQB, where the first session was executed using subjective ratings (circle, dashed) and the second was executed using EMG-based grading (x, with individual paths respectively shown as dash-dot and solid). The starting state for the first session lies at the rough center of the four separate, diverging paths executed on that day; these paths give good coverage of the region around that state. Participant BQB was instructed to give ratings from 0 to 10, with 5 as a "baseline," and gave no rating lower than 3 and no rating higher than 6. (c): Features calculated from participant ARI's EMG activity in a representative muscle, the left soleus, in response to each stimulus. The same scheme for labeling trials and subjective ratings is maintained from (a). The crescent shape shown in this feature space seems to correspond with a continuum from quiescence (upper left) to activation driven by the stimulus (lower right). (d): Features calculated from BQB's EMG activity under the stimuli shown in (b), again maintaining the labeling of trials and subjective ratings used in that figure. Qualitiatively different behavior appears to have occurred in the left soleus during these two sessions; in the first session (circles), the muscle was quiescent, while in the second (x) it was active, quite strongly in some cases.

rating was at 3.7 V, 25 Hz, near the previous day's maximum. The algorithm then continued this search over voltage and frequency, making two individual five-step paths. The first path started from 3.8 V and 25 Hz. The second started from 3.4 V and 25 Hz, an intermediate point on the first path, a value selected by the human experts overseeing the experiment. These experimental paths were successfully completed, yielding subjective standing quality ratings which were generally similar to those given in the previous session.

Two sessions were also conducted with the second patient, designated BQB (who has some sensation, but no motor control below the level of his lesion). Again, a known effective combination of active stimulus electrodes (2+ 3+ 4+ 9+ 13+ 14+ 15+ // 7- 8- 10-, consisting of two outer rows of cathodes and one midline cathode, combined with midline anodes, and all positioned at the caudal end of the array) was selected, with voltage and frequency values (4.5 V, 30 Hz) known to produce effective standing chosen as the starting point for the first session. The algorithm then was asked to choose paths. The first path started from this known point. After the completion of the first path, the stimulus was set to 4.4 V and 30Hz, which was used as the starting state for the remaining three paths of the first session. All four paths in this first session were selected using the subjective rating method. During the second session, the algorithm selected actions based on hand-selected EMG targets and weights. These targets and weights were chosen such that the highest-rated stimuli also tended to have the highest EMG-based grades when the reward was calculated using Equation (5.1), substituting the observations actually made for the vector $\mathbf{f}$. The algorithm was then given the EMG observations from an earlier session involving quiet standing using the same combination of active electrodes and one combination of voltage and frequency, as well as the EMG data from the first session, which had been guided by subjective ratings. Since it was making decisions using only EMG, the algorithm was now unable to directly use the subjective ratings given during the rating-based session. Within this second, EMG-based session, the algorithm selected two paths, the first beginning from the best stimuli found during the fourth path of the previous session. The second path executed during this session used the EMG observed during the first path to make decisions about what stimuli to apply during the second. This session thus provided an opportunity to test the efficiency of the data handling and processing necessary to select a subsequent path based on EMG observations acquired earlier in the same session. Using the current manual, two computer process, it may be possible to execute three paths during one hour-long session, but more paths will likely require an integrated data acquisition and processing system.

While no rigorous assertions regarding the search strategy's effectiveness can be made on the basis of the current experiments, several conclusions may be drawn from this experience. First, the algorithm and software implementation are capable of collaboratively planning experiments with human experts. In this procedure, given parameters set by the human experts, the algorithm proposed experiments, which were then approved and executed. This collaborative system allows the combination of the rigorous, quantitative search and optimization capabilities of GP-BUCB and its derivatives with the human experts' ability to assess a variety of important diagnostic information unavailable to the algorithm via the subjective ratings or EMG. Second, wide variation is present in the subjective grades given by different patients, perhaps in part because the instructions given may not have been consistent, but also perhaps due to differences in the perceived effects of EES therapy for individual patients. This difference in perceptions may be related to the differences in individual injuries as well, particularly the degree of sensation preserved. From an algorithmic or procedural standpoint, this implies that the rating scale must be both carefully specified and also customized to each patient. Careful specification is important because ratings must be repeatable and consistent to be useful. Patient-specific customization is also important because patient perception is highly individual, yet each patient's ratings must have sufficient resolution to identify improvement if and when it occurs, even if this improvement is small. Third, the Euclidean distance metric used for reward under the EMG-based selection procedure may require modification or replacement. It is clear that in some features (e.g., linear envelope amplitude, which roughly translates to contractile activity), there are qualitative differences between regions of the feature space (e.g., a muscle being "on" vs. "off") which are not effectively captured by simple Euclidean distance from a target; Equation (5.1) instead implies linear and symmetrical degradation of reward as distance from the target increases. From the perspective of standing performance, it may be that it is very important that a particular muscle is indeed "on", but not terribly important how strongly it is contracting during the period recorded, making the distance metric a poor fit. Another type of reward function may be more suitable to these features than Equation (5.1), e.g., the soft classification reward functions suggested in Section 5.4.2.1. Third, as a simple matter of logistics, the experimental cycle must eventually be made faster than it currently is, such that more than two or three paths of EMG-based stimuli selections can be made; without greater throughput, it will be very hard for the algorithm to effectively search while independent of expert assistance.

Another important point concerns the EMG features selected. Ten muscles were used in the EMG-based selection of actions for patient BQB: each of the soleus, tibialis anterior, medial gastrocnemius, medial hamstring, and vastus lateralis, in each of the left and right lower limbs. For each muscle, two features were computed: the average amplitude of the linear enveloped signal (using a second-order Butterworth filter, 4 Hz cutoff, applied forward and backward using the MATLAB filtfilt command) and the percentage power decoupled from the stimulus frequency and its multiples. This second feature was computed by performing a Fast Fourier Transform of the EMG signal and then computing the ratio of the power in frequency ranges other than $\bigcup_{n=1}^{\infty}[(n-0.1)f,(n+0.1)f]$, where f is the stimulus frequency, to the power of the original signal; since the frequency content of a repeating signal with period f should lie in the suppressed bands, this ratio should give the proportion of the signal which is not driven by the stimulus. The suppressed bands are 20% of the frequency spectrum, so this ratio's value is expected to be approximately 80% when the signal is white noise (which has a flat frequency spectrum and should be present when the muscle is quiescent) and substantially lower when the EMG is driven in synchrony with the stimulus. It was hypothesized that these features would give the ability to detect the level of activity of the muscle, as well as the degree to which this activity was being controlled by the spinal cord, rather than being driven by the stimulus, and that high performance would be observable as moderate-to-high linear envelope amplitudes and large amounts of stimulus decoupled power. Unfortunately, the experiments conducted so far have not borne out this hypothesis. In patient ARI, most muscles, as typified by Figure 5.1(c), did not show a co-occurrence of both high activity and high proportion of decoupled power, even under those stimuli which received very high subjective ratings. On the other hand, patient BQB did achieve EMG which had large values for both features, but this may be an artifact associated with a persistent and large amplitude tremor which occurs in both lower limbs during most of his standing bouts, rather than emergent, spinal-controlled activity. Neither of these observations implies that EMG with the hypothesized feature combination is infeasible for either patient, nor do these observations indicate that good standing would not appear in this region of the feature space; however, it does appear that (subjectively) good standing can occur outside of this region and that poor standing may in some cases produce EMG signals which do lie in this region. Both of these facts suggest that different (or at least additional) features may be necessary for low-error EMG-based recognition of good standing.

5.6 Extensions

The methods discussed in Section 5.4 have been used for the pilot experiments, but several substantial opportunities for improvement, alternatives to current approaches, or extensions to the theory suggest themselves.

5.6.1 Time Series Information and Coordination of Muscles

While it is reasonable to suggest using contextual GP models for modeling the activity of different muscles under the same stimulus, representing the aggregate activity of each muscle as a scalar or set of scalars may miss physiologically and behaviorally important information. In particular, properties such as coordination of muscles in response to disturbances require examining multiple EMG channels and how they interact with one another over time; muscle groups should activate in the proper anatomical pattern for coordination, e.g., flexors in the same leg activating together or extensors in the same leg activating together. These patterns are not describable by the time-averaged activity of each muscle; the crucial diagnostic information is instead carried in the relative times at which the muscles are active. This is not clearly something to which GPs are applicable; a (contextual) GP predicts a (set of) scalar(s), rather than a time trajectory. A set of scalars could be used to parameterize these interactions over time, but it is not clear how this representation should be constructed, or how expert knowledge should be used to construct covariance functions which respect the structure of these functions over the stimulus space. Further, it is not clear how a representation of muscle activity in terms of these features should be used to determine the quality or performance of the high-level motor activity. These issues remain intriguing, but open.

5.6.2 Dynamical Systems Approaches: Cost Functions and LQG

The cost functions proposed so far may fail to strike the right balance of abstraction versus concreteness; the subjective rating system is high-level and could yield meaningful results, but is somewhat poorly defined conceptually, while the vector-valued EMG approach relies on the specification of low-level targets which may not be easy to choose so as to actually produce good standing. Another alternative, which offers both greater rigor and an intermediate level of abstraction, is to treat the human participant as a dynamical system, for which some states (motion capture, center of pressure, etc.) are observed directly and for which some CNS control outputs (EMG) may also be observed. In this case, a physical model of the human could be used to infer a simple parameterization of the composite controller (the spinal cord under the influence of the EES system); this becomes a system identification problem, classically treated in the literature of controls and dynamical systems (for an introduction to system identification, see the text by Ljung, 1999). If the EES system is assumed to modulate the parameters of the controller, it might be possible to model these parameters as functions of the stimulus. The controller parameters would then be the objects of interest for learning and regression, perhaps modeled as draws from a Gaussian process. Given a regression model for the controller parameters, a closed-loop therapy system requires the ability to predict the reward corresponding to any stimulus; depending on the controller model used, well-known methods like linear quadratic Gaussian control (LQG, see Athans, 1971, for an early and thorough review) could be used to assign a cost to any given set of controller parameters. In the LQG case, for any draw from the posterior over controller parameters, there is an analytical expectation for the expected controller cost with respect to disturbances; this cost is the time integral of the variance of a Gaussian (accounting for deviations of the state from the desired trajectory and control costs) summed with other Gaussian variances (accounting for terminal costs). With relatively few samples, it might be possible to build a useful notion of the distribution over the composite controller parameters of the cost of running such a system, such that stimulus decisions could be made by comparing these cost distributions. This quantification would require a stochastic procedure, such that the algorithm would no longer be as computationally efficient. However, such a perspective might be more reflective of useful performance than direct functions of the EMG or other sensors and more tractable than attempting to work with user ratings directly. Given access to clinical sensors and off-board computing, such an approach might be reasonable.

5.6.3 Alternative Covariance Functions

A reasonable criticism of the covariance functions proposed in Section 5.4.3 is that, while they have the virtue of being very efficient computationally, this efficiency has been obtained at the price of both accuracy and precision in terms of the stimulating voltage distribution in and around the spinal cord. Certainly, the voltage function used is quite crude; the electrodes are not spheres, nor point sources, and the medium is not a single, homogeneous material. Additionally, capacitative and electrochemical effects may be significant, and the chosen weights and target regions of the spinal cord may not be appropriate. It is important to remember that the algorithmic purpose of the covariance function is to obtain a useful notion of how the responses to different electrode configurations co-vary, and how large the uncertainty about these responses may plausibly be, but this is only a means to an end. The covariance function does not need to perfectly capture the response function's shape; rather, it only needs to provide enough of a guide to enable efficient and intelligent experimental choices, the resulting data from which will drive the GP model.

Nevertheless, it remains reasonable to refine the proposed covariance function. Sophisticated finite element models of the spinal cord, epidural electrode array, and the surrounding volume have been created, which might be suitable for these purposes (see, e.g., Minassian et al., 2007). These simulations are typically quite computationally expensive. For this reason, creating a large, precomputed set of such simulations (corresponding to large sets of observations $\boldsymbol{X}_{t-1}$ or decision sets D) is likely infeasible. It might be possible to execute some subset of these simulations ahead of time, however, particularly if the decision set is small, perhaps growing slowly with time. Additional simulations could be performed on an online, as-needed basis, in a scheme somewhat similar to the lazy variance calculations discussed in Section 3.5. A hybrid method might also be reasonable; a sufficiently large "library" of full finite element calculations could be used to calculate corrections to the predictions of fast but crude models (the simple, homogeneous medium models discussed above, or perhaps linear combinations of very simple finite element simulations), giving the advantages speed, accuracy, and large decision or observation sets. It may also be reasonable to use prior knowledge of the relevant structures in the spinal cord and the mechanisms of neuronal excitation (see Minassian et al., 2007) or neuronal modeling software like NEURON (Hines and Carnevale, 2001) to choose appropriate functional weightings on the resulting simulation outputs to describe actual activation (and hopefully the degree of assistance toward the desired motor behavior).

One interesting note follows from the possible studies with NEURON; since the finite element and NEURON simulations are computationally expensive, a limited budget of these experiments is available, making this problem itself an appropriate application for the GP-UCB, GP-BUCB, and GP-AUCB algorithms. The algorithms would choose which simulations should be run, attempting to find stimuli which yield favorable patterns of neural activation, as determined by finite element simulations and NEURON. The results of these experiments could be used (perhaps with smoothing or abstraction) to build priors on the responses in actual human patients, thus avoiding the need to relearn the anatomy of the human spinal cord for every individual patient, consequently freeing the algorithm's clinical action selections to learn patient-specific variations.

5.6.4 Expansions of the Decision Set

Somewhat related to the question of the decision set "moving" with respect to time, discussed in Section 4.4.1, it is reasonable to consider decision sets which expand with respect to time. Certainly, the proof of the cases of Theorem 1 which allow the decision set D to be continuous rest on notions of how to grow the decision set appropriately as the algorithm acquires more data, discussed by Srinivas et al. (2009). However, these methods of growing the finite decision set D_t over which the algorithm must make a choice in each round constitute increasingly fine discretizations of a continuous, true D. An interesting question is how, and when, to expand the decision set into new and unexplored territory. One reason to do this is clear; if a set of stimuli are known to be safe and productive, it would be reasonable to allow the algorithm to explore in a limited "sandbox" including this set and its immediate surroundings within the stimulus space. Subsequently, the sandbox should be allowed to grow in some fashion which respects notions of sensitivity and safety. In the case of experimental stimuli, some reasonable notions of neighborship may be created, e.g., moving an active cathode or anode by one interval on the array, inactivating a cathode or anode, activating a cathode or anode next to an already existing cathode or anode, etc. Again, it is not clear precisely how these notions of neighborship correspond to functional similarity or sensitivity, nor is it clear how to choose when to expand the decision set in a rigorous, algorithmic fashion. Further, while this may be necessary for practical reasons, this sort of expansion of the decision set in a fashion which is contiguous with respect to the domain may preclude finding the optimum in some cases. It may also (or instead) be appropriate to track which regions of the decision set are known to be safe and which are known to be dangerous; dangerous regions should be avoided, even if they would otherwise be worth exploring. This could perhaps be captured in parallel by employing a Gaussian process classifier (see Rasmussen and Williams, 2006, Chapter 3), using a similarly structured covariance function and operating over the same domain. An open question concerns how to formulate the decision rule in this case; certainly, it is important and useful to learn more about this classification of safe versus dangerous, but should this be incorporated into the decision rule, or should this information be acquired as made available by a more conventional decision rule? Additionally, if it is desirable to incorporate this model of safety and confidence therein into the decision rule, how should this be done? These questions will have to be answered if it is desirable to incorporate explicit models of safety into GP-BUCB-like algorithms.

Chapter 6

Summary Conclusions

6.1 Conclusions

This dissertation develops two novel algorithms, GP-BUCB and GP-AUCB, provides new theoretical guarantees about their performance and that of a class of similar algorithms, and applies a variant of GP-BUCB to successfully manage an SCI therapy problem in real animals. This sort of autonomous, online direction of real EES-enabled SCI therapy has not previously been successfully executed by an algorithm. These results represent a substantial step toward both autonomously adaptive neurostimulators for SCI and lower-cost EES-based therapies for SCI patients. These techniques may also be applicable to other multi-electrode neurostimulation problems, such as deep brain stimulation and retinal prostheses.

Section 1.4 lays out the major goal of this dissertation, the creation, implementation, and testing of an algorithm capable of directing epidural electrostimulation-based SCI therapy. Specifically, it was necessary to develop an algorithm which could:

- Incorporate and exploit structural assumptions about the problem;
- Learn the responses of the spinal cord and muscles in a query-efficient manner; and
- Perform effective therapy, as measured by on-line metrics, even if observations must be made in batches or with a delay.

This dissertation satisfies each one of these requirements directly.

In Chapter 3, this dissertation develops the simple and efficient GP-BUCB and GP-AUCB algorithms. It also develops theoretical, high-probability bounds on the regret (i.e., sub-optimality in performance) of a general class of algorithms which includes GP-BUCB and GP-AUCB. This result, Theorem 1, is the main theorem of the work. Further, this dissertation presents an improved bound, Theorem 4, which is asymptotically independent of batch size or delay length, if the algorithm is initialized with an easily constructed set of observations of a finite, batch-size-dependent size. These bounds also provide high-probability convergence guarantees.

In simulation studies in several problems, including both synthetic and real data, the GP-BUCB and GP-AUCB algorithms attained similar performance to other, state-of-the-art batch algorithms for Bayesian optimization, which do not have theoretical bounds. Further, GP-BUCB and GP-AUCB performed comparably to the sequential GP-UCB algorithm, indicating that they overcome the disadvantages inherent in batching their actions and observations.

In Chapter 4, a variant of GP-BUCB successfully controls experiments in four SCI rats, with the goal of choosing appropriate bipolar stimuli on a multi-electrode array so as to maximize the average reward obtained. The reward is measured as the amplitudes of evoked potentials elicited in the rat's left tibialis anterior muscle. These experiments are selected in batches of five. In all four of these animals, the algorithm finds high-performing stimuli (i.e., stimuli which resulted in large evoked potentials, of putative therapeutic benefit), indicating that the algorithm effectively and efficiently searches the space of stimuli. As the experiments progress, the algorithm gradually builds up a human-interpretable set of predictions about the responsiveness of the spinal cord. This posterior over the evoked potential amplitudes enables the selection of appropriate experiments, both for the purpose of exploring the responses of the spinal cord and exploiting those stimuli which respond strongly. The algorithm competes directly with an expert human researcher in three of these animals. In these trials, the algorithm's performance under two different metrics is respectively superior or comparable to that of the expert human, indicating that the algorithm manages the exploration and exploitation of the evoked potentials appropriately. This finding indicates that, with appropriate reward metrics, variants of GP-BUCB or GP-AUCB would be effective for autonomously controlling more complex SCI therapy problems. This capability potentially represents a substantial savings in cost and great improvement in availability of EES-based SCI therapy.

Finally, Chapter 5 sets out a roadmap to human experiments, lays substantial practical and theoretical foundations, and describes pilot experiments conducted with two patients.

6.2 Future Work

The most important extension of the work presented in this dissertation will be the execution of experiments using GP-BUCB, GP-AUCB, or their derivatives to control EES for SCI therapy in humans. Chapter 5 describes the steps taken, in progress, and to be taken in the future toward human experiments. Also described in Chapter 5 are a number of important algorithmic and theoretical challenges which are associated with the needs of these experiments, but which also would have wider applications. Another important component of the SCI therapy program and the evaluation of the GP-BUCB and GP-AUCB algorithms is continued experimentation in animals, discussed below.

The first and most vital extension to the current animal experiments is to perform a parylene array experiment which lasts longer than the 3-4 experimental sessions and approximately 10 batches which were achieved in the first two parylene array animals. Laying aside the practical difficulties associated with the ongoing development of this technology, an experiment of substantial duration in a parylene array should be pursued and would allow improved evaluation of several components of the current animal experiments; most importantly, how the algorithm copes with a comparatively large decision set.

Another important extension of the animal experiments would be multi-muscle simultaneous optimization, i.e., experiments in which the reward function measures characteristics of several muscles. In this setting, the suboptimality of individual muscles must be balanced so as to obtain the highest possible aggregate reward. In animal 5, several experimental days (P34, 36, and 37) were devoted to two-muscle testing. Some promising results were obtained, but, due a coding error on P34 and to the slow failure of the array (which began on P36), consistent muscle responses were not obtained from day to day. Repetition of these experiments in a future animal, or perhaps on the archived data obtained from the single-muscle experiments, would be of great interest.

In future experiments using the wired array preparation, it would also be quite interesting to examine multipolar configurations, i.e., those for which there are a variable number cathodes and anodes, but at least one of each. Disallowing monopolar configurations or the configuration in which all electrodes are off, and restricting all electrodes to have the same master voltage, this would give 328 viable configurations on a 7 electrode array, substantially more than the 42 bipolar configurations. This experiment would have the advantages of both a simple and durable experimental preparation and a relatively large space of electrode configurations over which to search. This stimulus set would also mimic devices, like the Medtronic RestoreAdvanced neurostimulator used by Harkema et al. (2011), which have a master voltage control and can have arbitrarily many active electrodes, an important step toward using those devices.

Another significant advance in the animal experiments would be to make the entire system fully autonomous; all experiments described in this thesis were performed using a human researcher to control the stimulator and the recording system, while the algorithm performed an executive or directing role. This architecture has a number of advantages, among them that the human experimenter provides a fail-safe with respect to data acquisition (e.g., if an element of the data processing fails, the observations of the human experimenter can often be used to reconstruct the missing information) as well as with respect to animal safety (if an unexpected condition arises, the human experimenter can terminate stimulation, or, if a stimulus known to be painful is requested, the human experimenter can refuse to perform the requested experiment). This latter failsafe must somehow be maintained in an automatic system. Creating an integrated system in which the algorithm is controlling the data acquisition in (nearly) real time would allow a substantial acceleration of the testing process, and would constitute a very substantial step toward the goal of autonomy, a crucial requirement for a home-use or implantable device. Fortunately, the parylene arrays are controlled through a software interface, to which the GP-BUCB algorithm could be connected for both data acquisition and control. A safety system would have to be created, allowing the user to veto stimuli and terminate the delivery of stimuli which were distressing to the animal.

Clearly, future animal experiments should switch away from evoked potentials to a more complex motor response, in particular, standing or stepping. Unfortunately, these activities are quite complex, and quantifying their performance is both difficult and imprecise. Many attempts at doing so have been made (e.g. Basso et al., 1995; Antri et al., 2002), but none of them are entirely satisfactory for these purposes. For gait in particular, these techniques generally require extensive manual annotation of video recordings, which would necessitate a long feedback loop. If a rough, but sufficient analysis could be automated (e.g., pattern recognition on the raw motion capture trajectories, or on the EMG activity), however, these activities could be used in real time. This, however, requires a substantial effort in terms of building an effective automated grading system, possibly on the basis of a classifier, and for which careful feature selection will be necessary.

Bibliography

J. Abernethy, E. Hazan, and A. Rakhlin. Competing in the dark: An efficient algorithm for bandit linear optimization. In *COLT*, 2008.

Kim D. Anderson. Targeting recovery: Priorities of the spinal cord-injured population. *Journal of Neurotrauma*, 21(10):1371–1383, 2004.

M. Antri, D. Orsal, and J.-Y. Barthe. Locomotor recovery in the chronic spinal rat: Effects of long-term treatment with a 5-HT2 agonist. *European Journal of Neuroscience*, 16(3):467–476, 2002.

N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404, May 1950.

Michael Athans. The role and use of the stochastic linear-quadratic-Gaussian problem in control system design. *IEEE Transactions on Automatic Control*, 16(6):529–552, 1971.

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

Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *JMLR*, 3, 2002.

J. Azimi, A. Fern, and X.Fern. Batch Bayesian optimization via simulation matching. In *NIPS*, 2010.

Javad Azimi, Alan Fern, Xiaoli Fern, Glencora Borradale, and Brent Heeringa. Batch active learning via coordinated matching. In *Proceedings of the 29th International Conference on Machine Learning*, 2012a.

Javad Azimi, Ali Jalali, and Xiaoli Fern. Hybrid batch Bayesian optimization. In *ICML*, 2012b.

Florence M. Bareyre, Martin Kerschensteiner, Oliver Raineteau, Thomas C. Mettenleiter, Oliver Weinmann, and Martin E. Schwab. The injured spinal cord spontaneously forms a new intraspinal circuit in adult rats. *Nature Neuroscience*, 7(3):269–277, 2004.

D. Michele Basso, Michael S. Beattie, and Jacqueline C. Bresnahan. A sensitive and reliable locomotor rating scale for open field testing in rats. *Journal of Neurotrauma*, 12(1):1–21, 1995.

Elizabeth J. Bradbury and Stephen B. McMahon. Spinal cord repair strategies: Why do they work? *Nature Reviews Neuroscience*, 7:644–653, August 2006.

E. Brochu, M. Cora, and N. de Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. In *TR-2009-23, UBC*, 2009.

S. Bubeck and N. Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122, 2012.

S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvári. Online optimization in X-armed bandits. In *NIPS*, 2008.

S. Bubeck, R. Munos, and G. Stoltz. Pure exploration in multi-armed bandits problems. In *ALT*, 2009.

Lance L. Cai, Andy J. Fong, Chad K. Otoshi, Yongqiang Liang, Joel W. Burdick, Roland R. Roy, and V. Reggie Edgerton. Implications of assist-as-needed robotic step training after a complete spinal cord injury on intrinsic strategies of motor learning. *Journal of Neuroscience*, 26(41): 10564–10568, 2006.

Shayok Chakraborty, Vineeth Balasubramanian, and Sethuraman Panchanathan. Dynamic batch mode active learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

Christopher and Dana Reeve Foundation. One degree of separation: Paralysis and spinal cord injury in the United States. 636 Morris Turnpike, Suite 3A, Short Hills, NJ 07078, 2009.

Jean V. Coumans, Ted Tai-Sen Lin, Hai Ning Dai, Linda MacArthur, Marietta McAtee, Carmen Nash, and Barbara S. Bregman. Axonal regeneration and functional recovery after complete spinal cord transection in rats by delayed treatment with transplants and neurotrophins. *The Journal of Neuroscience*, 21(23):9334–9344, 2001.

Grégoire Courtine, Roland R. Roy, Joseph Raven, John Hodgson, Heather Mckay, Hong Yang, Hui Zhong, Mark H. Tuszynski, and V. Reggie Edgerton. Performance of locomotion and foot grasping following a unilateral thoracic corticospinal tract lesion in monkeys (*Macaca mulatta*). *Brain*, 128 (10):2338–2358, 2005.

Grégoire Courtine, Bingbing Song, Roland R. Roy, Hui Zhong, Julia E. Herrmann, Yan Ao, Jingwei Qi, V. Reggie Edgerton, and Michael V. Sofroniew. Recovery of supraspinal control of stepping via indirect propriospinal relay connections after spinal cord injury. *Nature Medicine*, 14(1):69 – 74, 2008.

Grégoire Courtine, Yury Gerasimenko, Rubia van den Brand, Aileen Yew, Pavel Musienko, Hui Zhong, Bingbing Song, Yan Ao, Ronaldo M Ichiyama, Igor Lavrov, et al. Transformation of non-functional spinal circuits into functional states after the loss of brain input. *Nature neuroscience*, 12(10):1333–1342, 2009.

Grégoire Courtine, Rubia van den Brand, and Pavel Musienko. Spinal cord injury: Time to move. *The Lancet*, 377:1896–1898, June 2011.

T. M. Cover and J. A. Thomas. *Elements of Information Theory.* Wiley Interscience, 1991.

D. D. Cox and S. John. SDO: A statistical method for global optimization. *Multidisciplinary Design Optimization: State of the Art*, 1997.

Eric D. Crown and James W. Grau. Preserving and restoring behavioral potential within the spinal cord using an instrumental training paradigm. *Journal of Neurophysiology*, 86(2):845–855, 2001.

V. Dani, T. P. Hayes, and S. M. Kakade. Stochastic linear optimization under bandit feedback. In *COLT*, 2008.

Glen M. Davis, Nur A. Hamzaid, and Ché Fornusek. Cardiorespiratory, metabolic, and biomechanical responses during functional electrical stimulation leg exercise: Health and fitness benefits. *Artificial organs*, 32(8):625–629, 2008.

Thomas Desautels, Andreas Krause, and Joel Burdick. Parallelizing exploration–exploitation tradeoffs with Gaussian process bandit optimization. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.

Milan R. Dimitrijevic, Yury Gerasimenko, and M. M. Pinter. Evidence for a spinal central pattern generator in humans. *Annals of the New York Academy of Sciences*, 860:360, 1998.

M. Dudik, D. Hsu, S. Kale, N. Karampatziakis, J. Langford, L. Reyzin, and T. Zhang. Efficient optimal learning for contextual bandits. In *UAI*, 2011.

V. Reggie Edgerton, Soo J. Kim, Ronaldo M. Ichiyama, Yury P. Gerasimenko, and Roland R. Roy. Rehabilitative therapies after spinal cord injury. *Journal of Neurotrauma*, 23(3-4):560–570, 2006.

Jeremy L. Emken, Susan J. Harkema, Janell A. Beres-Jones, Christie K. Ferreira, and David J. Reinkensmeyer. Feasibility of manual teach-and-replay and continuous impedence shaping for robotic locomotor training following spinal cord injury. *IEEE Transactions on Biomedical Engineering*, 55(1):322–334, January 2008.

Christie Engesser-Cesar, Ronaldo M. Ichiyama, Amber L. Nefas, Mary Ann Hill, V. Reggie Edgerton, Carl W. Cotman, and Aileen J. Anderson. Wheel running following spinal cord injury improves locomotor recovery and stimulates serotonergic fiber growth. *European Journal of Neuroscience*, 25(7):1931–1939, 2007.

J. W. Fawcett, A. Curt, J. D. Steeves, W. P. Coleman, M. H. Tuszynski, D. Lammertse, P. F. Bartlett, A. R. Blight, V. Dietz, J. Ditunno, B. H. Dobkin, L. A. Havton, P. H. Ellaway, M. G. Fehlings, A. Privat, R. Grossman, J. D. Guest, N. Kleitman, M. Nakamura, M. Gaviria, and D. Short. Guidelines for the conduct of clinical trials for spinal cord injury as developed by the ICCP panel: Spontaneous recovery after spinal cord injury and statistical power needed for therapeutic trials. *Spinal Cord*, 45:190–205, 2007.

Andy J. Fong, Lance L. Cai, Chad K. Otoshi, David J. Reinkensmeyer, Joel W. Burdick, Roland R. Roy, and V. Reggie Edgerton. Spinal cord-transected mice learn to step in response to quipazine treatment and robotic training. *The Journal of neuroscience*, 25(50):11738–11747, 2005.

Joan Fruitet, Alexandra Carpentier, Remi Munos, and Maureen Clerc. Bandit algorithms boost brain computer interfaces for motor-task selection of a brain-controlled button. In *Advances in Neural Information Processing Systems 25*, pages 458–466, 2012.

Joan Fruitet, Alexandra Carpentier, Rémi Munos, and Maureen Clerc. Automatic motor task selection via a bandit algorithm for a brain-controlled button. *Journal of neural engineering*, 10 (1):016012, 2013.

Parag Gad, Jaehoon Choe, Mandheerej Singh Nandra, Hui Zhong, Roland R. Roy, Yu-Chong Tai, and V. Reggie Edgerton. Development of a multi-electrode array for spinal cord epidural stimulation to facilitate stepping and standing after a complete spinal cord injury in adult rats. *Journal of neuroengineering and rehabilitation*, 10(1):2, 2013.

Yury Gerasimenko, Roland R. Roy, and V. Reggie Edgerton. Epidural stimulation: Comparison of the spinal circuits that generate and control locomotion in rats, cats and humans. *Experimental neurology*, 209(2):417–425, 2008.

D. Ginsbourger, R. Riche, and L. Carraro. Kriging is well-suited to parallelize optimization. In Yoel Tenne and Chi-Keong Goh, editors, *Computational Intelligence in Expensive Optimization Problems*, volume 2 of *Adaptation, Learning, and Optimization*, pages 131–162. Springer Berlin Heidelberg, 2010.

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

Tayfun Gürel and Carsten Mehring. Unsupervised adaptation of brain-machine interface decoders. *Frontiers in neuroscience*, 6, 2012.

Susan Harkema, Yury Gerasimenko, Jonathan Hodes, Joel Burdick, Claudia Angeli, Yangsheng Chen, Christie Ferreira, Andrea Willhite, Enrico Rejc, Robert G. Grossman, and V. Reggie Edgerton. Effect of epidural stimulation of the lumbosacral spinal cord on voluntary movement, standing, and assisted stepping after motor complete paraplegia: A case study. *The Lancet*, 377 (9781):1938–1947, 2011.

Elad Hazan and Satyen Kale. Better algorithms for benign bandits. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 38–47. Society for Industrial and Applied Mathematics, 2009.

Philipp Hennig and Christian J. Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research (JMLR)*, 13:1809–1837, June 2012.

R. Herman, J. He, S. D'Luzansky, W. Willis, and S. Dilli. Spinal cord stimulation facilitates functional walking in a chronic, incomplete spinal cord injured. *Spinal Cord*, 40:65–68, 2002.

M.L. Hines and N.T. Carnevale. NEURON: A tool for neuroscientists. *The Neuroscientist*, 7: 123–135, 2001.

Jr. John F. Ditunno and Christopher S. Formal. Chronic spinal cord injury. *New England Journal of Medicine*, 330(8):550–556, February 1994.

D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *J Glob. Opti.*, 13:455–492, 1998.

Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

Eric R. Kandel, James H. Schwartz, Thomas M. Jessell, Steven A. Siegelbaum, and A. J. Hudspeth. *Principles of Neural Science*. McGraw-Hill, 5th ed. edition, 2012.

Soheila Karimi-Abdolrezaee, Desiree Schut, Jian Wang, and Michael G. Fehlings. Chondroitinase and growth factors enhance activation and oligodendrocyte differentiation of endogenous neural precursor cells after spinal cord injury. *PloS one*, 7(5):e37589, 2012.

R. Kleinberg, A. Slivkins, and E. Upfal. Multi-armed bandits in metric spaces. In *STOC*, pages 681–690, 2008.

K. John Klose, Patrick L. Jacobs, James G. Broton, Rosalind S. Guest, Belinda M. Needham-Shropshire, Nathan Lebwohl, Mark S. Nash, and Barth A. Green. Evaluation of a training program for persons with SCI paraplegia using the Parastep® 1 ambulation system: Part 1. Ambulation performance and anthropometric measures. *Archives of physical medicine and rehabilitation*, 78 (8):789–793, 1997.

L. Kocsis and C. Szepesvári. Bandit based Monte-Carlo planning. In *ECML*, 2006.

Alojz Kralj and Slobodan Grobelnik. Functional electrical stimulation—A new hope for paraplegic patients? *Bulletin of prosthetics research*, 20:75–102, 1973.

A. Krause and C. Guestrin. Near-optimal nonmyopic value of information in graphical models. In *UAI*, 2005.

A. Krause and C. S. Ong. Contextual Gaussian process bandit optimization. In *NIPS*, 2011.

A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research (JMLR)*, 9:235–284, February 2008.

W. T. Liberson, H. J. Holmquest, David Scot, and Margot Dow. Functional electrotherapy: Stimulation of the peroneal nerve synchronized with the swing phase of the gait of hemiplegic patients. *Archives of Physical Medicine and Rehabilitation*, 42:101–105, 1961.

D. Lizotte, T. Wang, M. Bowling, and D. Schuurmans. Automatic gait optimization with Gaussian process regression. In *IJCAI*, pages 944–949, 2007.

Lennart Ljung. *System Identification: Theory for the User.* Prentice Hall PTR, 1999.

S. Mangold, T. Keller, A. Curt, and V. Dietz. Transcutaneous functional electrical stimulation for grasping in subjects with cervical spinal cord injury. *Spinal Cord*, 43(1):1–13, 2004.

Anitha Manohar, Robert D. Flint III, Eric Knudsen, and Karen A. Moxon. Role of neuronal plasticity after spinal cord injury for neurorobotic control. In *Proceedings of the 5th International IEEE EMBS Conference on Neural Engineering*, 2011.

K. Minassian, I. Persy, F. Rattay, M. M. Pinter, H. Kern, and Milan R. Dimitrijevic. Human lumbar cord circuitried can be activated by extrinsic tonic input to generate locomotor-like activity. *Human Movement Science*, 26:275–295, 2007.

M. Minoux. Accelerated greedy algorithms for maximizing submodular set functions. *Optimization Techniques, LNCS*, pages 234–243, 1978.

J. Mockus. *Bayesian Approach to Global Optimization.* Kluwer Academic Publishers, 1989.

J. Mockus, V. Tiesis, and A. Zilinskas. The application of Bayesian methods for seeking the extremum. In *Toward Global Optimization*, volume 2, pages 117–129. Elsevier, 1978.

Mandheerej S. Nandra, Igor A. Lavrov, V. Reggie Edgerton, and Yu-Chong Tai. A parylene-based microelectrode array implant for spinal cord stimulation in rats. In *Micro Electro Mechanical Systems (MEMS), 2011 IEEE 24th International Conference on*, pages 1007–1010. IEEE, 2011.

Ana Paula Pêgo, Sarka Kubinova, Dasa Cizkova, Ivo Vanicky, Fernando Milhazes Mar, Mónica Mendes Sousa, and Eva Sykova. Regenerative medicine for the treatment of spinal cord injury: More than just promises? *Journal of Cellular and Molecular Medicine*, 16(11):2564–2582, 2012.

Thomas E. Prieto, J.B. Myklebust, R.G. Hoffmann, E.G. Lovett, and B.M. Myklebust. Measures of postural steadiness: Differences between healthy young and elderly adults. *Biomedical Engineering, IEEE Transactions on*, 43(9):956–966, 1996.

Arthur Prochazka, Vivian K. Mushahwar, and Douglas B. McCreery. Neural prostheses. *The Journal of physiology*, 533(1):99–109, 2001.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Carl Edward Rasmussen and Hannes Nickisch. Gaussian processes for machine learning (GPML) toolbox. *The Journal of Machine Learning Research*, 11:3011–3015, 2010.

H. Robbins. Some aspects of the sequential design of experiments. *Bul. Am. Math. Soc.*, 55, 1952.

Roland R. Roy, John A. Hodgson, Sharlene D. Lauretz, David J. Pierotti, Richard J. Gayek, and V. Reggie Edgerton. Chronic spinal cord-injured cats: Surgical procedures and management. *Laboratory Animal Science*, 42(4):335– 343, August 1992.

Ilya O. Ryzhov, Warren B. Powell, and Peter I. Frazier. The knowledge gradient algorithm for a general class of online learning problems. *Operations Research*, 60(1):180–195, 2012.

Sabato Santaniello, Giovanni Fiengo, Luigi Glielmo, and Warren M. Grill. Closed-loop control of deep brain stimulation: A simulation study. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 19(1):15–24, 2011.

Brenda R. Santos, Alain Delisle, Christian Larivière, André Plamondon, and Daniel Imbeau. Reliability of centre of pressure summary measures of postural steadiness in healthy young adults. *Gait & posture*, 27(3):408–415, 2008.

Burr Settles. *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan and Claypool, 2012.

W.J.W. Sharrard. The distribution of the permanent paralysis in the lower limb in poliomyelitis: A clinical and pathological study. *Journal of Bone & Joint Surgery, British Volume*, 37(4):540–558, 1955.

W.J.W. Sharrard. The segmental innervation of the lower limb muscles in man: Arris and Gale lecture delivered at the Royal College of Surgeons of England on 2nd January 1964. *Annals of the Royal College of Surgeons of England*, 35(2):106, 1964.

C. Norman Shealy, J. Thomas Mortimer, and James B. Reswick. Electrical inhibition of pain by stimulation of the dorsal columns: Preliminary clinical report. *Anesthesia and Analgesia*, 46: 489–491, July-August 1967a.

C. Norman Shealy, Normal Taslitz, J. Thomas Mortimer, and Donald P. Becker. Electrical inhibition of pain: Experimental evaluation. *Anesthesia and Analgesia*, 46(3):299–305, May-June 1967b.

Aleksandrs Slivkins. Contextual bandits with similarity information. Technical report, arXiv, 2011.

N. Srinivas, A. Krause, S. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. Technical report, arXiv, 2009.

N. Srinivas, A. Krause, S. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *ICML*, 2010.

Adam Thrasher, Geoffrey M. Graham, and Milos R. Popovic. Reducing muscle fatigue due to functional electrical stimulation using random modulation of stimulation parameters. *Artificial Organs*, 29(6):453–458, 2005.

Sandrine Thuret, Lawrence D.F. Moon, and Fred H Gage. Therapeutic interventions after spinal cord injury. *Nature Reviews Neuroscience*, 7(8):628–643, 2006.

G. E. Uhlenbeck and L. S. Ornstein. On the theory of Brownian motion. *Physical Review*, 36: 823–841, 1930.

Rubia van den Brand, Janine Heutschi, Quentin Barraud, Jack DiGiovanna, Kay Bartholdi, Michèle Huerlimann, Lucia Friedli, Isabel Vollenweider, Eduardo Martin Moraud, Simone Duis, Nadia Dominici, Silvestro Micera, Pavel Musienko, and Grégoire Courtine. Restoring voluntary control of locomotion after paralyzing spinal cord injury. *Science*, 336(6085):1182–1185, 2012.

Carmen Vidaurre, Claudia Sannelli, Klaus-Robert Müller, and Benjamin Blankertz. Co-adaptive calibration to improve BCI efficiency. *Journal of neural engineering*, 8(2):025009, 2011.

Grace Wahba. *Spline models for observational data*, volume 59. Society for industrial and applied mathematics, 1990.

Joseph M. Waltz. Spinal cord stimulation: A quarter century of development and investigation. *Stereotactic and functional neurosurgery*, 69(1-4):288–299, 1997.

Charles Watson, George Paxinos, and Gulgun Kayalioglu. *The Spinal Cord*. Academic Press, 2008.

L. C. Weaver and C. Polosa, editors. *Autonomic Dysfunction After Spinal Cord Injury*, 2006. Elsevier.

Anton Wernig and S. Müller. Laufband locomotion with body weight support improved walking in persons with severe spinal cord injuries. *Spinal Cord*, 30(4):229–238, 1992.

C. Widmer, N. Toussaint, Y. Altun, and G. Rätsch. Inferring latent task structure for multitask learning by multiple kernel learning. *BMC Bioinformatics*, 11(Suppl 8:S5), 2010.

Jun Wu, Tiansheng Sun, Chaoqun Ye, Jianhua Yao, Bing Zhu, and Hongying He. Clinical observation of fetal olfactory ensheathing glia transplantation (OEGT) in patients with complete chronic spinal cord injury. *Cell Transplantation*, 21(Supplement 1):S33–S37, 2012.

Shihao Zhang, Rishi Wadhwa, Justin Haydel, Jamie Toms, Kendrick Johnson, and Bharat Guthikonda. Spine and spinal cord trauma. *Neurologic Clinics*, 31:183–206, 2013.

Appendix A

Theoretical Results: Proofs

The proofs of a number of theoretical results from Chapter 3 are presented in this Appendix.

A.1 Theorem 1

In order to prove Theorem 1, this section first establishes a series of supporting lemmas.

Lemma 6 shows that a bound on the local information hallucinated during the batch implies a relationship between batch and sequential confidence intervals:

Lemma 6. *Suppose that at some round t, there exists a bound, C, uniform over D, on the conditional mutual information with respect to $f(\boldsymbol{x})$ which will be acquired by the set of actions initiated since the last observation at round $fb[t]$, where this bound is of form*

$$I(f(\boldsymbol{x}); \boldsymbol{y}_{fb[t]+1:t-1} \mid \boldsymbol{y}_{1:fb[t]}) \leq C, \ \forall \boldsymbol{x} \in D, \tag{A.1}$$

for some constant $C > 0$. Choose

$$\beta_t = \exp(2C)\alpha_{fb[t]} \tag{A.2}$$

where Equation (3.6) relates sequential confidence intervals $C^{seq}_{fb[t]+1}(\boldsymbol{x})$ with the parameter $\alpha_{fb[t]+1}$ and Equation (3.8) relates batch confidence intervals $C^{batch}_t(\boldsymbol{x})$ with the parameter β_t. Then, conditional on the event that for all $\boldsymbol{x} \in D$, $f(\boldsymbol{x}) \in C^{seq}_{fb[t]+1}(\boldsymbol{x})$, it holds that $f(\boldsymbol{x}) \in C^{batch}_{t'}(\boldsymbol{x})$ for all $\boldsymbol{x} \in D$ and all t' such that $fb[t] + 1 \leq t' \leq t$.

Proof. Noting that the confidence intervals $C^{\text{seq}}_{\text{fb}[t]+1}(\boldsymbol{x})$ and $C^{\text{batch}}_t(\boldsymbol{x})$ are both centered on $\mu_{\text{fb}[t]}(\boldsymbol{x})$,

$$C^{\text{seq}}_{\text{fb}[t]+1}(\boldsymbol{x}) \subseteq C^{\text{batch}}_t(\boldsymbol{x}) \ \forall \boldsymbol{x} \in D \iff \alpha^{1/2}_{\text{fb}[t]}\sigma_{\text{fb}[t]}(\boldsymbol{x}) \leq \beta^{1/2}_t \sigma_{t-1}(\boldsymbol{x}) \ \forall \boldsymbol{x} \in D.$$

By the definition of the conditional mutual information with respect to $f(\boldsymbol{x})$, and by employing

Equation (A.1), Equation (3.14) follows. Choosing β_t as in Equation (A.2), it follows that

$$\alpha_{\mathrm{fb}[t]}^{1/2}\sigma_{\mathrm{fb}[t]}(\boldsymbol{x}) = \beta_t^{1/2}\exp(-C)\cdot\sigma_{\mathrm{fb}[t]}(\boldsymbol{x}) \leq \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x}),$$

where the inequality is based on Equation (3.14), thus implying $C_{\mathrm{fb}[t]+1}^{\mathrm{seq}}(\boldsymbol{x}) \subseteq C_t^{\mathrm{batch}}(\boldsymbol{x})\,\forall \boldsymbol{x} \in D$. In turn, if $f(\boldsymbol{x}) \in C_{\mathrm{fb}[t]+1}^{\mathrm{seq}}(\boldsymbol{x})$, then $f(\boldsymbol{x}) \in C_t^{\mathrm{batch}}(\boldsymbol{x})$. Further, since Equation (A.2) relates β_t to $\alpha_{\mathrm{fb}[t]}$, then $\beta_{t'} = \beta_t$ for all $t' \in \{\mathrm{fb}[t]+1, \ldots, t\}$. Since $\sigma_{t'}$ is non-increasing, $C_{t'}^{\mathrm{batch}}(\boldsymbol{x}) \supseteq C_t^{\mathrm{batch}}(\boldsymbol{x})$ for all such t', completing the proof. □

With a bound C on the conditional mutual information gain with respect to $f(\boldsymbol{x})$ for any $\boldsymbol{x} \in D$, as in Equation (A.1), Lemma 6 links the confidence intervals and GP-BUCB decision rule at time t with the GP posterior after observation fb$[t]$. Lemma 7 extends this link to all $t \geq 1$ and all $\boldsymbol{x} \in D$, given a high-probability guarantee of confidence interval correctness at the beginning of all batches. This step is required for the regret bound of Theorem 1.

Lemma 7. *Suppose there exist a constant $C > 0$, a sequence of actions $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-1}\}$, and a feedback mapping fb$[t]$ such that for all $\boldsymbol{x} \in D$*

$$C \geq I(f(\boldsymbol{x}); \boldsymbol{y}_{fb[t]+1:t-1} \mid \boldsymbol{y}_{1:fb[t]}),\ \forall t \geq 1.$$

Then, if $\beta_t = \exp(2C)\alpha_{fb[t]}$, $\forall t \geq 1$,

$$\begin{aligned} &P(f(\boldsymbol{x}) \in C_{fb[t]+1}^{seq}(\boldsymbol{x})\,\forall \boldsymbol{x} \in D,\,\forall t \geq 1) \geq 1-\delta \\ &\implies P(f(x) \in C_t^{batch}(\boldsymbol{x})\,\forall \boldsymbol{x} \in D,\,\forall t \geq 1) \geq 1-\delta. \end{aligned}$$

Proof. If β_t is chosen as specified, then for any t and τ such that $\tau = \mathrm{fb}[t]$ and $f(x) \in C_{\tau+1}^{\mathrm{seq}}(\boldsymbol{x})$, Lemma 6 implies that $f(x) \in C_t^{\mathrm{batch}}(\boldsymbol{x})$. If there exists a set $\mathcal{S} = \{\tau_1, \tau_2, \ldots\}$ such that fb$[t] \in \mathcal{S}$ for all $t \geq 1$, and additionally $f(x) \in C_{\tau+1}^{\mathrm{seq}}(\boldsymbol{x})$ for all $\boldsymbol{x} \in D$ and $\tau \in \mathcal{S}$, then $f(\boldsymbol{x}) \in C_t^{\mathrm{batch}}(\boldsymbol{x})$ for all $\boldsymbol{x} \in D$ and all $t \geq 1$. The event $f(\boldsymbol{x}) \in C_{\mathrm{fb}[t]+1}^{\mathrm{seq}}(\boldsymbol{x})$, $\forall \boldsymbol{x} \in D$, $\forall t \geq 1$ satisfies these conditions directly. The lemma follows because for logical propositions A and B, $[A \implies B] \implies [P(A) \leq P(B)]$ and thus if $P(A) \geq 1-\delta \implies P(B) \geq 1-\delta$. □

The high-probabilty confidence intervals are next be related to the instantaneous regret and thence to the cumulative regret.

Lemma 8. *Conditional on the event $f(\boldsymbol{x}) \in C_t^{batch}(\boldsymbol{x})$, $\forall \boldsymbol{x} \in D$, $\forall t \geq 1$, and given that actions $\boldsymbol{x}_t$, $\forall t \geq 1$ are selected using Equation (3.7), it holds that*

$$R_T \leq \sqrt{TC_1\beta_T\gamma_T},$$

where $C_1 = 8/\log(1+\sigma_n^{-2})$, γ_T *is defined in Equation (3.3), and* β_t *is defined in Equation (A.2).*

Proof. Given $f(\boldsymbol{x}) \in C_t^{\text{batch}}(\boldsymbol{x})$, $\forall \boldsymbol{x} \in D$, $\forall t \geq 1$, using the GP-BUCB decision rule, Equation (3.7), Lemma 5.2 in Srinivas et al. (2010), and our assumptions about $C_t^{\text{batch}}(\boldsymbol{x})$, it follows that the instantaneous regret r_t is bounded as $r_t \leq 2\beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x}_t)$, $\forall t \geq 1$. From Lemma 5.3 and 5.4 in Srinivas et al. (2010) it follows that $\sum_{t=1}^{T} r_t^2 \leq C_1\beta_T\gamma_T$. The claim then follows as a consequence of the Cauchy Schwarz inequality, since $R_T^2 \leq T\sum_{t=1}^{T} r_t^2$. □

Proof of Theorem 1. Taken together, Lemmas 6 through 8, a bound C satisfying Equation (A.1), and a high-probability guarantee that some set of sequential confidence intervals always contain the values of f allow us to construct a batch algorithm with high-probability regret bounds. Srinivas et al. (2010) develop choices of the exploration-exploitation tradeoff parameter α_t such that guarantees of the form $P(f(\boldsymbol{x}) \in C_t^{\text{seq}}(\boldsymbol{x}) \, \forall \boldsymbol{x} \in D, \forall t \geq 1) \geq 1-\delta$ are realizable for arbitrarily small δ. Employing Lemma 8, as well as Lemmas 5.1 and 5.8 of Srinivas et al. (2010) (for assumptions 1 and 2) and Theorem 6 of Srinivas et al. (2010) (for assumption 3), Theorem 1 follows as an immediate corollary. □

A.2 Theorem 4: Initialization Set Size Bounds

Thorough initialization of GP-BUCB can drive down the constant C, which bounds the information which can be hallucinated in the course of post-initialization batches and also governs the asymptotic scaling of the regret bound with batch size B. First, we connect the information which can be gained in post-initialization batches with the amount of information being gained in the initialization, through Lemma 3, the formal statement of which is in Section 3.3.5, and the proof of which is presented here.

Proof of Lemma 3. Since the initialization procedure is greedy and information gain is submodular (See Section 3.2.3), the information gain from adding the last element of the initialization set, $I(f; \boldsymbol{x}_{T^{\text{init}}}^{\text{init}} \mid D_{T^{\text{init}}-1}^{\text{init}})$, must be the smallest marginal information gain in the initialization process, and thus is no greater than the mean of the marginal gains, i.e,

$$I(f; \boldsymbol{x}_{T^{\text{init}}}^{\text{init}} \mid D_{T^{\text{init}}-1}^{\text{init}}) \leq I(f; D_{T^{\text{init}}}^{\text{init}})/T^{\text{init}}.$$

Further, again because information gain is submodular and the initializaiton set was constructed greedily, no subsequent decision can yield information gain greater than $I(f; \boldsymbol{x}_{T^{\text{init}}}^{\text{init}} \mid D_{T^{\text{init}}-1}^{\text{init}})$, and thus $\gamma_{B-1}^{\text{init}} \leq (B-1)\cdot I(f; \boldsymbol{x}_{T^{\text{init}}}^{\text{init}} \mid D_{T^{\text{init}}-1}^{\text{init}})$. Combining these two inequalities with the definition of $\gamma_{T^{\text{init}}}$ yields the result. □

A.2.1 Initialization Set Size: Linear Kernel

For the linear kernel, there exists a logarithmic bound on the maximum information gain of a set of queries, precisely, $\exists\, \eta \geq 0 : \gamma_t \leq \eta d \log(t+1)$ (Srinivas et al., 2010). We attempt to initialize GP-BUCB with a set D^{init} of size T^{init}, where, motivated by this bound and the form of Inequality (3.16), we assume T^{init} is of the form

$$T^{\text{init}} = k\eta d(B-1)\log B. \tag{A.3}$$

We must show that there exists a k of finite size for which an initialization set of size T^{init} as in Equation (A.3) implies that any subsequent set $\mathcal{S}$, $|\mathcal{S}| = B-1$, produces a conditional information gain with respect to $\mathbf{f}$ of no more than C. This requires showing that the inequality $\frac{B-1}{T^{\text{init}}}\gamma_{T^{\text{init}}} \leq C$ holds for this choice of k and thus T^{init}. Since we consider non-trivial batches, i.e., $B-1 \geq 1$, if k is sufficiently large such that $k\eta d(B-1) \geq 1$,

$$\log(\log(B) + 1/(k\eta d(B-1))) \leq \log(\log(B)+1) \leq \log B.$$

Using Equation (A.3) and the bound for $\gamma_{T^{\text{init}}}$, and following algebraic rearrangement, this inequality implies that if $k\eta d(B-1) \geq 1$,

$$\frac{B-1}{T^{\text{init}}}\gamma_{T^{\text{init}}} \leq C \Longleftarrow \frac{\log k}{k \log B} + \frac{\log\eta + \log d}{k\log B} + \frac{2}{k} \leq C.$$

By noting that the maximum of $\frac{\log k}{k}$ over $k \in (0,\infty)$ is $1/e$ and choosing for convenience $C = 2/e$, we obtain for $k \geq 1/(\eta d(B-1))$:

$$\frac{B-1}{T^{\text{init}}}\gamma_{T^{\text{init}}} \leq \frac{2}{e} \Longleftarrow \frac{1}{e\log B} + \frac{1}{k}\left(\frac{\log\eta + \log d + 2\log B}{\log B}\right) \leq \frac{2}{e},$$

or equivalently, choosing k to satisfy both constraints simultaneously,

$$\frac{B-1}{T^{\text{init}}}\gamma_{T^{\text{init}}} \leq \frac{2}{e} \Longleftarrow k \geq \max\left[\frac{1}{\eta d(B-1)}, \frac{e(\log\eta + \log d + 2\log B)}{2\log(B) - 1}\right].$$

Thus, for a linear kernel and such a k, an initialization set D^{init} of size T^{init}, where $T^{\text{init}} \geq k\eta d(B-1)\log(B)$, ensures that the hallucinated conditional information in any future batch of size B is $\leq \frac{2}{e}$.

A.2.2 Initialization Set Size: Matérn Kernel

For the Matérn kernel, $\gamma_t \leq \nu t^{\epsilon}$, $\epsilon \in (0,1)$ for some $\nu > 0$ (Srinivas et al., 2010). Hence:

$$\begin{aligned}\frac{(B-1)}{T^{\text{init}}}\gamma_{T^{\text{init}}} \leq C \Longleftarrow \frac{\nu(B-1)(T^{\text{init}})^{\epsilon}}{T^{\text{init}}} &\leq C \\ \Longleftrightarrow \nu(B-1)(T^{\text{init}})^{\epsilon-1} &\leq C \\ \Longleftrightarrow T^{\text{init}} &\geq \left(\frac{\nu(B-1)}{C}\right)^{1/(1-\epsilon)}.\end{aligned}$$

Thus, for a Matérn kernel, an initialization set D^{init} of size $T^{\text{init}} \geq \left(\frac{\nu(B-1)}{C}\right)^{1/(1-\epsilon)}$ implies that the conditional information gain of any future batch is $\leq C$. Choosing $C = 1$, we obtain the results presented in the corresponding row of Table 3.1.

A.2.3 Initialization Set Size: Squared Exponential (RBF) Kernel

For the RBF kernel, the information gain is bounded by an expression similar to that of the linear kernel, $\gamma_t \leq \eta(\log(t+1))^{d+1}$ (Srinivas et al., 2010). Again, motivated by Inequality (3.16), one reasonable choice for an initialization set size is $T^{\text{init}} = k\eta(B-1)(\log B)^{d+1}$. It is necessary to show that there exists a finite k such that the conditional information gain of any post-initialization batch is $\leq C$. By a similar parallel argument to that for the linear kernel (Appendix A.2.1), and assuming that $B \geq 2$ and $k\eta(B-1) \geq 1$, it follows that

$$\begin{aligned}&\frac{B-1}{T^{\text{init}}}\gamma_{T^{\text{init}}} \leq C \\ &\Longleftarrow \frac{\log k + \log\eta + \log(B-1)}{k^{1/(d+1)}(\log B)}\frac{\log[(\log B)^{d+1}+1]}{k^{1/(d+1)}(\log B)} \leq C^{1/(d+1)} \\ &\Longleftarrow \frac{\log k}{k^{1/(d+1)}(\log B)} + \frac{\log\eta}{k^{1/(d+1)}(\log B)} + \frac{(d+2)}{k^{1/(d+1)}} \leq C^{1/(d+1)},\end{aligned}$$

where the last implication follows because for $a \geq 0, b \geq 1, (a^b+1) \leq (a+1)^b$.

By noting that the maximum of $k^{-1/(d+1)}\log k$ over $k \in (0,\infty)$ is $(d+1)/e$ and choosing $C = (2(d+1)/e)^{d+1}$, we obtain for $k \geq 1/(\eta(B-1))$:

$$\frac{B-1}{T^{\text{init}}}\gamma_{T^{\text{init}}} \leq \left(\frac{2d+2}{e}\right)^{d+1} \Longleftarrow \frac{d+1}{e\log B} + \frac{1}{k^{1/(d+1)}}\left(\frac{\log\eta + (d+2)\log B}{\log B}\right) \leq \frac{2d+2}{e},$$

or equivalently, incorporating the constraint $k \geq 1/(\eta(B-1))$ explicitly,

$$\frac{B-1}{T^{\text{init}}}\gamma_{T^{\text{init}}} \leq \left(\frac{2d+2}{e}\right)^{d+1} \Longleftarrow k \geq \max\left[\frac{1}{\eta(B-1)}, \left(\frac{e(\log\eta + (d+2)\log B)}{(d+1)(2\log(B)-1)}\right)^{d+1}\right].$$

Thus, for a Squared Exponential kernel and such a k, an initialization set D^{init} of size T^{init}, where $T^{\text{init}} \geq k\eta(B-1)(\log(B))^{d+1}$, ensures that the hallucinated conditional information in any future batch of size B is no more than $\left(\frac{2d+2}{e}\right)^{d+1}$.

A.3 GP-AUCB: Finite Batch Size

In the absence of an explicitly specified maximum batch size, it is interesting to consider the scaling of batche sizes produced by GP-AUCB for large T. We are concerned with the case where actions are chosen when much is known about the structure of the reward function: many actions could be selected with little "danger" of choosing poorly, but also little information gain. In such a case, a great deal of regret could be accumulated between observations if the posterior mean fails to correctly order the available actions in D with respect to their reward.

The set of size T which gains the least information with respect to f, conditioned on observations $\boldsymbol{y}(\mathcal{S})$, is one which queries $x_* = \operatorname{argmin}_{x\in D} \sigma^2(x|\boldsymbol{y}(\mathcal{S}))$ T times. These samples gain information $1/2 \log(1 + T\sigma_n^{-2}\sigma^2_{\mathcal{S}}(x_*))$, where $\sigma^2(x|\boldsymbol{y}_{\mathcal{S}})) = \sigma^2_{\mathcal{S}}(x)$ is the posterior variance, conditioned on the observations $\boldsymbol{y}_{\mathcal{S}}$. Using this observation, if a batch is terminated when a threshold C for hallucinated conditional information with respect to f is exceeded, as in the GP-AUCB algorithm, the maximum possible length of a batch, B_{max}, can be bounded as follows:

$$\begin{aligned} C \geq & 1/2 \log(1 + (B_{max}-1)\sigma_n^{-2}\sigma^2_{\mathbf{S}}(x_*)) \\ \implies & \left[\frac{\sigma_n^2}{\sigma^2_{\mathbf{S}}(x_*)}[\exp(2C)-1]\right] + 1 \geq B_{max}. \end{aligned} \tag{A.4}$$

Thus, if there does not exist any $x \in D$ such that $\sigma^2(x) = 0$, which is the case under the GP model for any finite noise, this upper limit on B_{max} is finite for any finite C and any previous sampling history; the batch sizes of GP-AUCB do not diverge to infinity in a finite number of rounds.

Bounding the rate at which the batch length B_{max} can grow is of interest, however. Consider cases where time is indexed by action number t or by batch number N. In the case of iteration number, by rearrangement of Equation (3.14) and using Inequalities (3.11) and (3.13), we have

$$\sigma_t^2(\boldsymbol{x}) \geq \sigma_0^2(\boldsymbol{x}) \exp(-2I(f;\boldsymbol{y}_{1:t-1})) \geq \sigma_0^2(\boldsymbol{x}) \exp(-2\gamma_{t-1}) \ \forall t \in \mathbb{N}.$$

At time t, using this result and Inequality (A.4), the maximum length of the batch which can be constructed under GP-AUCB (or any sampling procedure such that the batch terminates when the information gain threshold C is exceeded) is bounded as

$$B_{max} \leq \left[\frac{\sigma_n^2}{\min_{x\in D}(\sigma_0^2(x))}[\exp(2C)-1][\exp(2\gamma_{t-1})]\right] + 1.$$

This bound is $O(\exp(tC))$, since γ_t is no more than linear in t.

A similar bounding result may be obtained for the Nth batch. After $N-1$ batches, the posterior variance of $f(\boldsymbol{x})$, $\sigma^2_{N-1}(\boldsymbol{x})$, may be bounded as follows, for any $\boldsymbol{x} \in D$, via Equation (3.14) and Inequalities (3.11) and (3.13):

$$\sigma^2_{N-1}(\boldsymbol{x}) \geq \sigma^2_0(\boldsymbol{x}) \exp\left(-2(N-1)C_B\right) \forall t \in \mathbb{N}.$$

Here, C_B is an upper bound on the information which is obtained when the observations corresponding to the batch are made. C_B is greater than C, since the batch terminates only when the information which would be hallucinated in order to select the next action exceeds the threshold C. One useful bound is $C_B \leq C + 1/2 \log\left(1 + \sigma_n^{-2} \max_{\boldsymbol{x} \in D} \sigma^2_0(\boldsymbol{x})\right)$, which follows because the termination condition is checked every round and mutual information is submodular. Using Equation (A.4), the length of the Nth batch is thus bounded as

$$B_{max} \leq \left[\frac{\sigma_n^2}{\min_{x \in D}(\sigma^2_0(x))}[\exp(2C) - 1][\exp\left(2(N-1)C_B\right)]\right] + 1,$$

which is $O(\exp(NC))$, but is bounded for finite batch number.

Appendix B

Tabulated Computational Results

B.1 Tables of Results from Experiments

These experiments are described in detail in Section 3.6. Tables of numerical results are presented here; these include the regret (or elapsed time) with the standard error. Each table presents the results of each data set and algorithm combination tested for a particular experimental setting, averaged over 200 runs. Minimum regret of zero indicates that the optimal set was visited by every run.

Data Set	Algorithm	AR, Query 100	MR, Query 100	AR, Query 200	MR, Query 200
Matérn GP	GP-UCB	0.1268 ± 0.0076	0.0285 ± 0.0075	0.0845 ± 0.0073	0.0243 ± 0.0069
	GP-BUCB	0.1434 ± 0.0040	0.0113 ± 0.0032	0.0855 ± 0.0035	0.0107 ± 0.0032
	SM-UCB	0.1479 ± 0.0055	0.0089 ± 0.0052	0.0849 ± 0.0037	0.0035 ± 0.0011
	SM-MEI	0.1549 ± 0.0048	0.0147 ± 0.0031	0.0937 ± 0.0036	0.0099 ± 0.0026
SE GP	GP-UCB	0.0513 ± 0.0038	0.0054 ± 0.0033	0.0322 ± 0.0033	0.0021 ± 0.0012
	GP-BUCB	0.0577 ± 0.0014	0.0005 ± 0.0002	0.0329 ± 0.0008	0.0003 ± 0.0001
	SM-UCB	0.0612 ± 0.0018	0.0016 ± 0.0011	0.0349 ± 0.0011	0.0004 ± 0.0002
	SM-MEI	0.0593 ± 0.0017	0.0016 ± 0.0007	0.0338 ± 0.0011	0.0006 ± 0.0002
Rosenbrock	GP-UCB	0.0571 ± 0.0005	0.0000 ± 0.0000	0.0353 ± 0.0003	0.0000 ± 0.0000
	GP-BUCB	0.0579 ± 0.0005	0.0000 ± 0.0000	0.0359 ± 0.0003	0.0000 ± 0.0000
	SM-UCB	0.0598 ± 0.0004	0.0000 ± 0.0000	0.0366 ± 0.0003	0.0000 ± 0.0000
	SM-MEI	0.0560 ± 0.0005	0.0000 ± 0.0000	0.0340 ± 0.0003	0.0000 ± 0.0000
Cosines	GP-UCB	0.2109 ± 0.0013	0.0009 ± 0.0002	0.1152 ± 0.0007	0.0001 ± 0.0000
	GP-BUCB	0.2110 ± 0.0013	0.0010 ± 0.0002	0.1158 ± 0.0008	0.0002 ± 0.0001
	SM-UCB	0.2195 ± 0.0012	0.0010 ± 0.0002	0.1213 ± 0.0008	0.0003 ± 0.0001
	SM-MEI	0.2092 ± 0.0013	0.0019 ± 0.0004	0.1173 ± 0.0011	0.0010 ± 0.0003
Vaccine Design	GP-UCB	0.8147 ± 0.0402	0.3465 ± 0.0346	0.6009 ± 0.0354	0.2987 ± 0.0304
	GP-BUCB	0.8605 ± 0.0374	0.2834 ± 0.0291	0.6013 ± 0.0314	0.2326 ± 0.0269
	SM-UCB	0.8149 ± 0.0321	0.1521 ± 0.0212	0.5261 ± 0.0264	0.1446 ± 0.0207
	SM-MEI	0.7750 ± 0.0337	0.1525 ± 0.0214	0.5125 ± 0.0266	0.1066 ± 0.0171
SCI	GP-UCB	0.3099 ± 0.0142	0.1540 ± 0.0129	0.2329 ± 0.0127	0.1345 ± 0.0121
	GP-BUCB	0.2965 ± 0.0102	0.0666 ± 0.0085	0.1920 ± 0.0085	0.0544 ± 0.0076
	SM-UCB	0.3016 ± 0.0092	0.0398 ± 0.0061	0.1813 ± 0.0069	0.0303 ± 0.0052
	SM-MEI	0.3622 ± 0.0085	0.0146 ± 0.0019	0.2280 ± 0.0049	0.0096 ± 0.0008

Table B.1: Average (AR) and Minimum regret (MR) for fixed batch size B = 5.

Data Set	Algorithm	AR, Round 100	MR, Round 100	AR, Round 200	MR, Round 200
Matérn GP	GP-UCB	0.1307 ± 0.0054	0.0167 ± 0.0052	0.0811 ± 0.0050	0.0095 ± 0.0030
	GP-BUCB	0.1601 ± 0.0046	0.0096 ± 0.0039	0.0925 ± 0.0041	0.0079 ± 0.0038
	GP-AUCB	0.1527 ± 0.0052	0.0105 ± 0.0041	0.0898 ± 0.0047	0.0101 ± 0.0041
SE GP	GP-UCB	0.0482 ± 0.0012	0.0013 ± 0.0005	0.0290 ± 0.0010	0.0009 ± 0.0005
	GP-BUCB	0.0656 ± 0.0014	0.0003 ± 0.0001	0.0369 ± 0.0008	0.0002 ± 0.0001
	GP-AUCB	0.0597 ± 0.0015	0.0013 ± 0.0005	0.0343 ± 0.0010	0.0009 ± 0.0005
Rosenbrock	GP-UCB	0.0598 ± 0.0004	0.0000 ± 0.0000	0.0369 ± 0.0003	0.0000 ± 0.0000
	GP-BUCB	0.0601 ± 0.0004	0.0000 ± 0.0000	0.0376 ± 0.0003	0.0000 ± 0.0000
	GP-AUCB	0.0635 ± 0.0005	0.0000 ± 0.0000	0.0382 ± 0.0003	0.0000 ± 0.0000
Cosines	GP-UCB	0.2224 ± 0.0013	0.0013 ± 0.0003	0.1224 ± 0.0008	0.0004 ± 0.0002
	GP-BUCB	0.2199 ± 0.0013	0.0012 ± 0.0003	0.1214 ± 0.0009	0.0001 ± 0.0000
	GP-AUCB	0.2693 ± 0.0013	0.0024 ± 0.0005	0.1352 ± 0.0010	0.0002 ± 0.0000
Vaccine Design	GP-UCB	0.8217 ± 0.0371	0.3058 ± 0.0317	0.5834 ± 0.0316	0.2555 ± 0.0286
	GP-BUCB	0.9650 ± 0.0337	0.2501 ± 0.0279	0.6453 ± 0.0277	0.2100 ± 0.0248
	GP-AUCB	0.9653 ± 0.0355	0.2031 ± 0.0252	0.6153 ± 0.0281	0.1783 ± 0.0243
SCI	GP-UCB	0.3092 ± 0.0131	0.0920 ± 0.0100	0.2108 ± 0.0106	0.0718 ± 0.0091
	GP-BUCB	0.3558 ± 0.0095	0.0339 ± 0.0060	0.2068 ± 0.0067	0.0237 ± 0.0050
	GP-AUCB	0.3155 ± 0.0122	0.0455 ± 0.0072	0.1880 ± 0.0084	0.0317 ± 0.0058

Table B.2: Average (AR) and Minimum regret (MR) for fixed delay length B = 5.

Data Set	Algorithm	AR, Query 100	MR, Query 100	AR, Query 200	MR, Query 200
Matérn GP	GP-BUCB, $B = 5$	0.1405 ± 0.0033	0.0080 ± 0.0024	0.0827 ± 0.0028	0.0076 ± 0.0024
	GP-BUCB, $B = 10$	0.1751 ± 0.0029	0.0068 ± 0.0016	0.0980 ± 0.0020	0.0060 ± 0.0016
	GP-BUCB, $B = 20$	0.2843 ± 0.0047	0.0038 ± 0.0009	0.1513 ± 0.0024	0.0029 ± 0.0008
	SM-UCB, $B = 5$	0.1509 ± 0.0048	0.0117 ± 0.0043	0.0889 ± 0.0045	0.0110 ± 0.0043
	SM-UCB, $B = 10$	0.1891 ± 0.0028	0.0029 ± 0.0009	0.1036 ± 0.0017	0.0025 ± 0.0009
	SM-UCB, $B = 20$	0.3022 ± 0.0051	0.0025 ± 0.0008	0.1597 ± 0.0026	0.0005 ± 0.0002
	SM-MEI, $B = 5$	0.1524 ± 0.0047	0.0141 ± 0.0041	0.0905 ± 0.0040	0.0099 ± 0.0036
	SM-MEI, $B = 10$	0.1897 ± 0.0037	0.0076 ± 0.0025	0.1064 ± 0.0028	0.0068 ± 0.0023
	SM-MEI, $B = 20$	0.2978 ± 0.0047	0.0081 ± 0.0019	0.1609 ± 0.0029	0.0063 ± 0.0015
SE GP	GP-BUCB, $B = 5$	0.0600 ± 0.0014	0.0005 ± 0.0001	0.0344 ± 0.0008	0.0002 ± 0.0001
	GP-BUCB, $B = 10$	0.0937 ± 0.0024	0.0005 ± 0.0001	0.0515 ± 0.0014	0.0004 ± 0.0001
	GP-BUCB, $B = 20$	0.1653 ± 0.0045	0.0010 ± 0.0002	0.0864 ± 0.0023	0.0004 ± 0.0002
	SM-UCB, $B = 5$	0.0607 ± 0.0016	0.0006 ± 0.0002	0.0349 ± 0.0011	0.0004 ± 0.0002
	SM-UCB, $B = 10$	0.0920 ± 0.0024	0.0004 ± 0.0002	0.0501 ± 0.0013	0.0001 ± 0.0000
	SM-UCB, $B = 20$	0.1660 ± 0.0048	0.0006 ± 0.0001	0.0869 ± 0.0024	0.0003 ± 0.0001
	SM-MEI, $B = 5$	0.0606 ± 0.0017	0.0014 ± 0.0004	0.0349 ± 0.0011	0.0011 ± 0.0003
	SM-MEI, $B = 10$	0.0920 ± 0.0025	0.0019 ± 0.0006	0.0501 ± 0.0014	0.0013 ± 0.0005
	SM-MEI, $B = 20$	0.1639 ± 0.0049	0.0013 ± 0.0002	0.0853 ± 0.0024	0.0009 ± 0.0002
Rosenbrock	GP-BUCB, $B = 5$	0.0576 ± 0.0004	0.0000 ± 0.0000	0.0356 ± 0.0003	0.0000 ± 0.0000
	GP-BUCB, $B = 10$	0.0573 ± 0.0004	0.0000 ± 0.0000	0.0353 ± 0.0003	0.0000 ± 0.0000
	GP-BUCB, $B = 20$	0.0771 ± 0.0004	0.0000 ± 0.0000	0.0453 ± 0.0003	0.0000 ± 0.0000
	SM-UCB, $B = 5$	0.0590 ± 0.0005	0.0000 ± 0.0000	0.0368 ± 0.0003	0.0000 ± 0.0000
	SM-UCB, $B = 10$	0.0639 ± 0.0005	0.0000 ± 0.0000	0.0386 ± 0.0003	0.0000 ± 0.0000
	SM-UCB, $B = 20$	0.0828 ± 0.0008	0.0000 ± 0.0000	0.0483 ± 0.0004	0.0000 ± 0.0000
	SM-MEI, $B = 5$	0.0558 ± 0.0005	0.0000 ± 0.0000	0.0340 ± 0.0003	0.0000 ± 0.0000
	SM-MEI, $B = 10$	0.0626 ± 0.0006	0.0000 ± 0.0000	0.0374 ± 0.0004	0.0000 ± 0.0000
	SM-MEI, $B = 20$	0.0806 ± 0.0007	0.0000 ± 0.0000	0.0464 ± 0.0004	0.0000 ± 0.0000
Cosines	GP-BUCB, $B = 5$	0.2107 ± 0.0013	0.0014 ± 0.0003	0.1158 ± 0.0008	0.0003 ± 0.0001
	GP-BUCB, $B = 10$	0.2066 ± 0.0013	0.0009 ± 0.0002	0.1131 ± 0.0009	0.0002 ± 0.0001
	GP-BUCB, $B = 20$	0.2136 ± 0.0017	0.0023 ± 0.0007	0.1186 ± 0.0011	0.0006 ± 0.0004
	SM-UCB, $B = 5$	0.2211 ± 0.0014	0.0012 ± 0.0003	0.1210 ± 0.0009	0.0003 ± 0.0001
	SM-UCB, $B = 10$	0.2330 ± 0.0014	0.0013 ± 0.0004	0.1278 ± 0.0008	0.0003 ± 0.0001
	SM-UCB, $B = 20$	0.2729 ± 0.0015	0.0019 ± 0.0004	0.1505 ± 0.0011	0.0001 ± 0.0000
	SM-MEI, $B = 5$	0.2106 ± 0.0016	0.0033 ± 0.0006	0.1184 ± 0.0012	0.0015 ± 0.0005
	SM-MEI, $B = 10$	0.2253 ± 0.0016	0.0027 ± 0.0005	0.1257 ± 0.0010	0.0011 ± 0.0002
	SM-MEI, $B = 20$	0.2631 ± 0.0016	0.0041 ± 0.0006	0.1454 ± 0.0010	0.0011 ± 0.0003
Vaccine Design	GP-BUCB, $B = 5$	0.9413 ± 0.0406	0.3302 ± 0.0340	0.6615 ± 0.0348	0.2775 ± 0.0293
	GP-BUCB, $B = 10$	1.0379 ± 0.0349	0.1839 ± 0.0278	0.6540 ± 0.0299	0.1711 ± 0.0265
	GP-BUCB, $B = 20$	1.4637 ± 0.0323	0.1024 ± 0.0176	0.8327 ± 0.0247	0.0951 ± 0.0169
	SM-UCB, $B = 5$	0.8531 ± 0.0366	0.1790 ± 0.0245	0.5428 ± 0.0279	0.1444 ± 0.0215
	SM-UCB, $B = 10$	1.0513 ± 0.0275	0.0906 ± 0.0174	0.6170 ± 0.0219	0.0866 ± 0.0168
	SM-UCB, $B = 20$	1.5212 ± 0.0278	0.0349 ± 0.0113	0.8341 ± 0.0190	0.0349 ± 0.0113
	SM-MEI, $B = 5$	0.8239 ± 0.0325	0.1667 ± 0.0229	0.5418 ± 0.0256	0.1383 ± 0.0214
	SM-MEI, $B = 10$	1.0751 ± 0.0330	0.1202 ± 0.0231	0.6557 ± 0.0249	0.0801 ± 0.0158
	SM-MEI, $B = 20$	1.5440 ± 0.0270	0.0277 ± 0.0098	0.8590 ± 0.0195	0.0271 ± 0.0098
SCI	GP-BUCB, $B = 5$	0.2748 ± 0.0103	0.0492 ± 0.0076	0.1728 ± 0.0082	0.0433 ± 0.0071
	GP-BUCB, $B = 10$	0.3884 ± 0.0091	0.0440 ± 0.0065	0.2275 ± 0.0069	0.0391 ± 0.0060
	GP-BUCB, $B = 20$	0.6031 ± 0.0093	0.0427 ± 0.0063	0.3349 ± 0.0070	0.0298 ± 0.0052
	SM-UCB, $B = 5$	0.3075 ± 0.0094	0.0445 ± 0.0066	0.1894 ± 0.0072	0.0392 ± 0.0063
	SM-UCB, $B = 10$	0.4162 ± 0.0078	0.0190 ± 0.0030	0.2290 ± 0.0049	0.0152 ± 0.0025
	SM-UCB, $B = 20$	0.6608 ± 0.0120	0.0236 ± 0.0045	0.3571 ± 0.0072	0.0213 ± 0.0043
	SM-MEI, $B = 5$	0.3734 ± 0.0089	0.0170 ± 0.0026	0.2379 ± 0.0050	0.0109 ± 0.0013
	SM-MEI, $B = 10$	0.4838 ± 0.0078	0.0132 ± 0.0022	0.2981 ± 0.0048	0.0087 ± 0.0015
	SM-MEI, $B = 20$	0.7177 ± 0.0099	0.0124 ± 0.0022	0.4086 ± 0.0060	0.0072 ± 0.0013

Table B.3: Average (AR) and Minimum regret (MR) for batch sizes $B = 5$, 10, and 20, non-adaptive algorithms.

Data Set	Algorithm	AR, Query 100	MR, Query 100	AR, Query 200	MR, Query 200
Matérn GP	GP-AUCB, $B_{max} = 5$	0.1303 ± 0.0057	0.0182 ± 0.0053	0.0819 ± 0.0053	0.0166 ± 0.0052
	GP-AUCB, $B_{max} = 10$	0.1293 ± 0.0060	0.0185 ± 0.0058	0.0816 ± 0.0057	0.0138 ± 0.0042
	GP-AUCB, $B_{max} = 20$	0.1326 ± 0.0061	0.0197 ± 0.0059	0.0835 ± 0.0059	0.0187 ± 0.0059
	GP-AUCB Local, $B_{max} = 5$	0.1290 ± 0.0042	0.0112 ± 0.0036	0.0774 ± 0.0038	0.0108 ± 0.0036
	GP-AUCB Local, $B_{max} = 10$	0.1667 ± 0.0071	0.0238 ± 0.0069	0.1020 ± 0.0068	0.0190 ± 0.0057
	GP-AUCB Local, $B_{max} = 20$	0.1952 ± 0.0081	0.0173 ± 0.0055	0.1148 ± 0.0062	0.0153 ± 0.0055
	HBBO UCB, $B_{max} = 5$	0.1455 ± 0.0070	0.0179 ± 0.0064	0.0895 ± 0.0065	0.0163 ± 0.0063
	HBBO UCB, $B_{max} = 10$	0.1459 ± 0.0047	0.0136 ± 0.0038	0.0873 ± 0.0040	0.0121 ± 0.0037
	HBBO UCB, $B_{max} = 20$	0.1587 ± 0.0055	0.0150 ± 0.0044	0.0950 ± 0.0047	0.0137 ± 0.0044
	HBBO MEI, $B_{max} = 5$	0.1515 ± 0.0059	0.0197 ± 0.0051	0.0935 ± 0.0052	0.0169 ± 0.0050
	HBBO MEI, $B_{max} = 10$	0.1644 ± 0.0063	0.0224 ± 0.0053	0.1017 ± 0.0054	0.0162 ± 0.0047
	HBBO MEI, $B_{max} = 20$	0.1771 ± 0.0069	0.0131 ± 0.0040	0.1049 ± 0.0050	0.0111 ± 0.0040
SE GP	GP-AUCB, $B_{max} = 5$	0.0489 ± 0.0015	0.0005 ± 0.0002	0.0286 ± 0.0009	0.0004 ± 0.0002
	GP-AUCB, $B_{max} = 10$	0.0505 ± 0.0016	0.0017 ± 0.0006	0.0296 ± 0.0011	0.0010 ± 0.0003
	GP-AUCB, $B_{max} = 20$	0.0590 ± 0.0041	0.0061 ± 0.0035	0.0362 ± 0.0035	0.0026 ± 0.0012
	GP-AUCB Local, $B_{max} = 5$	0.0540 ± 0.0035	0.0044 ± 0.0033	0.0323 ± 0.0026	0.0007 ± 0.0005
	GP-AUCB Local, $B_{max} = 10$	0.0591 ± 0.0021	0.0022 ± 0.0012	0.0340 ± 0.0015	0.0020 ± 0.0012
	GP-AUCB Local, $B_{max} = 20$	0.0683 ± 0.0027	0.0012 ± 0.0005	0.0382 ± 0.0015	0.0008 ± 0.0005
	HBBO UCB, $B_{max} = 5$	0.0547 ± 0.0020	0.0040 ± 0.0016	0.0331 ± 0.0017	0.0021 ± 0.0013
	HBBO UCB, $B_{max} = 10$	0.0554 ± 0.0017	0.0010 ± 0.0004	0.0326 ± 0.0011	0.0003 ± 0.0001
	HBBO UCB, $B_{max} = 20$	0.0610 ± 0.0023	0.0015 ± 0.0005	0.0343 ± 0.0013	0.0006 ± 0.0003
	HBBO MEI, $B_{max} = 5$	0.0533 ± 0.0022	0.0017 ± 0.0006	0.0315 ± 0.0014	0.0013 ± 0.0005
	HBBO MEI, $B_{max} = 10$	0.0601 ± 0.0023	0.0021 ± 0.0006	0.0346 ± 0.0014	0.0014 ± 0.0005
	HBBO MEI, $B_{max} = 20$	0.0640 ± 0.0036	0.0032 ± 0.0017	0.0361 ± 0.0021	0.0010 ± 0.0002
Rosenbrock	GP-AUCB, $B_{max} = 5$	0.0572 ± 0.0005	0.0000 ± 0.0000	0.0353 ± 0.0003	0.0000 ± 0.0000
	GP-AUCB, $B_{max} = 10$	0.0580 ± 0.0005	0.0000 ± 0.0000	0.0359 ± 0.0004	0.0000 ± 0.0000
	GP-AUCB, $B_{max} = 20$	0.0577 ± 0.0005	0.0000 ± 0.0000	0.0359 ± 0.0003	0.0000 ± 0.0000
	GP-AUCB Local, $B_{max} = 5$	0.0574 ± 0.0005	0.0000 ± 0.0000	0.0356 ± 0.0003	0.0000 ± 0.0000
	GP-AUCB Local, $B_{max} = 10$	0.0579 ± 0.0005	0.0000 ± 0.0000	0.0360 ± 0.0003	0.0000 ± 0.0000
	GP-AUCB Local, $B_{max} = 20$	0.0602 ± 0.0006	0.0000 ± 0.0000	0.0368 ± 0.0004	0.0000 ± 0.0000
	HBBO UCB, $B_{max} = 5$	0.0579 ± 0.0005	0.0000 ± 0.0000	0.0362 ± 0.0003	0.0000 ± 0.0000
	HBBO UCB, $B_{max} = 10$	0.0578 ± 0.0005	0.0000 ± 0.0000	0.0356 ± 0.0003	0.0000 ± 0.0000
	HBBO UCB, $B_{max} = 20$	0.0580 ± 0.0006	0.0000 ± 0.0000	0.0362 ± 0.0004	0.0000 ± 0.0000
	HBBO MEI, $B_{max} = 5$	0.0540 ± 0.0005	0.0000 ± 0.0000	0.0332 ± 0.0003	0.0000 ± 0.0000
	HBBO MEI, $B_{max} = 10$	0.0545 ± 0.0005	0.0000 ± 0.0000	0.0334 ± 0.0004	0.0000 ± 0.0000
	HBBO MEI, $B_{max} = 20$	0.0550 ± 0.0005	0.0000 ± 0.0000	0.0343 ± 0.0004	0.0000 ± 0.0000
Cosines	GP-AUCB, $B_{max} = 5$	0.2168 ± 0.0012	0.0009 ± 0.0002	0.1191 ± 0.0009	0.0002 ± 0.0001
	GP-AUCB, $B_{max} = 10$	0.2183 ± 0.0015	0.0014 ± 0.0003	0.1182 ± 0.0009	0.0002 ± 0.0001
	GP-AUCB, $B_{max} = 20$	0.2156 ± 0.0014	0.0020 ± 0.0004	0.1186 ± 0.0010	0.0005 ± 0.0001
	GP-AUCB Local, $B_{max} = 5$	0.2118 ± 0.0014	0.0009 ± 0.0002	0.1162 ± 0.0008	0.0002 ± 0.0000
	GP-AUCB Local, $B_{max} = 10$	0.2108 ± 0.0015	0.0016 ± 0.0004	0.1160 ± 0.0011	0.0005 ± 0.0002
	GP-AUCB Local, $B_{max} = 20$	0.2187 ± 0.0015	0.0016 ± 0.0003	0.1218 ± 0.0011	0.0005 ± 0.0002
	HBBO UCB, $B_{max} = 5$	0.2122 ± 0.0012	0.0010 ± 0.0002	0.1160 ± 0.0009	0.0002 ± 0.0001
	HBBO UCB, $B_{max} = 10$	0.2129 ± 0.0013	0.0009 ± 0.0003	0.1155 ± 0.0008	0.0001 ± 0.0000
	HBBO UCB, $B_{max} = 20$	0.2168 ± 0.0017	0.0019 ± 0.0004	0.1204 ± 0.0012	0.0004 ± 0.0001
	HBBO MEI, $B_{max} = 5$	0.2018 ± 0.0014	0.0031 ± 0.0008	0.1126 ± 0.0010	0.0017 ± 0.0006
	HBBO MEI, $B_{max} = 10$	0.2031 ± 0.0014	0.0030 ± 0.0006	0.1138 ± 0.0010	0.0008 ± 0.0002
	HBBO MEI, $B_{max} = 20$	0.2074 ± 0.0016	0.0033 ± 0.0005	0.1177 ± 0.0012	0.0009 ± 0.0002
Vaccine Design	GP-AUCB, $B_{max} = 5$	0.8811 ± 0.0451	0.3048 ± 0.0341	0.6217 ± 0.0373	0.2714 ± 0.0316
	GP-AUCB, $B_{max} = 10$	0.8410 ± 0.0402	0.3016 ± 0.0325	0.6048 ± 0.0345	0.2710 ± 0.0305
	GP-AUCB, $B_{max} = 20$	0.8455 ± 0.0387	0.2963 ± 0.0315	0.6136 ± 0.0353	0.2725 ± 0.0301
	GP-AUCB Local, $B_{max} = 5$	0.9044 ± 0.0367	0.2729 ± 0.0293	0.6174 ± 0.0314	0.2580 ± 0.0287
	GP-AUCB Local, $B_{max} = 10$	0.9564 ± 0.0375	0.2197 ± 0.0265	0.6202 ± 0.0301	0.1911 ± 0.0248
	GP-AUCB Local, $B_{max} = 20$	1.0741 ± 0.0367	0.1562 ± 0.0228	0.6656 ± 0.0277	0.1436 ± 0.0211
	HBBO UCB, $B_{max} = 5$	0.8472 ± 0.0413	0.3141 ± 0.0326	0.6021 ± 0.0346	0.2767 ± 0.0315
	HBBO UCB, $B_{max} = 10$	0.8628 ± 0.0381	0.3456 ± 0.0324	0.6343 ± 0.0326	0.3153 ± 0.0305
	HBBO UCB, $B_{max} = 20$	0.8606 ± 0.0420	0.3188 ± 0.0344	0.6154 ± 0.0368	0.2862 ± 0.0327
	HBBO MEI, $B_{max} = 5$	0.8574 ± 0.0403	0.3030 ± 0.0310	0.6020 ± 0.0316	0.2134 ± 0.0248
	HBBO MEI, $B_{max} = 10$	0.8712 ± 0.0378	0.3135 ± 0.0324	0.6299 ± 0.0325	0.2366 ± 0.0290
	HBBO MEI, $B_{max} = 20$	0.8675 ± 0.0370	0.2934 ± 0.0316	0.6105 ± 0.0308	0.2168 ± 0.0275
SCI	GP-AUCB, $B_{max} = 5$	0.3152 ± 0.0132	0.1444 ± 0.0121	0.2339 ± 0.0120	0.1287 ± 0.0115
	GP-AUCB, $B_{max} = 10$	0.3242 ± 0.0138	0.1252 ± 0.0110	0.2315 ± 0.0118	0.1124 ± 0.0107
	GP-AUCB, $B_{max} = 20$	0.3226 ± 0.0125	0.1403 ± 0.0111	0.2354 ± 0.0112	0.1326 ± 0.0111
	GP-AUCB Local, $B_{max} = 5$	0.3114 ± 0.0134	0.0961 ± 0.0106	0.2128 ± 0.0113	0.0810 ± 0.0100
	GP-AUCB Local, $B_{max} = 10$	0.3084 ± 0.0128	0.0727 ± 0.0088	0.1962 ± 0.0099	0.0606 ± 0.0080
	GP-AUCB Local, $B_{max} = 20$	0.3766 ± 0.0128	0.0739 ± 0.0089	0.2306 ± 0.0095	0.0627 ± 0.0081
	HBBO UCB, $B_{max} = 5$	0.2751 ± 0.0100	0.0557 ± 0.0081	0.1745 ± 0.0087	0.0491 ± 0.0078
	HBBO UCB, $B_{max} = 10$	0.3060 ± 0.0093	0.0516 ± 0.0078	0.1862 ± 0.0081	0.0448 ± 0.0073
	HBBO UCB, $B_{max} = 20$	0.3186 ± 0.0106	0.0613 ± 0.0090	0.1966 ± 0.0094	0.0559 ± 0.0087
	HBBO MEI, $B_{max} = 5$	0.3206 ± 0.0099	0.0292 ± 0.0055	0.1989 ± 0.0058	0.0104 ± 0.0008
	HBBO MEI, $B_{max} = 10$	0.3362 ± 0.0086	0.0201 ± 0.0038	0.2080 ± 0.0053	0.0087 ± 0.0008
	HBBO MEI, $B_{max} = 20$	0.3527 ± 0.0087	0.0274 ± 0.0049	0.2221 ± 0.0057	0.0093 ± 0.0008

Table B.4: Average (AR) and Minimum regret (MR) for maximum adaptive batch sizes $B_{max} = 5$, 10, and 20.

Data Set	Algorithm	AR, Round 100	MR, Round 100	AR, Round 200	MR, Round 200
Matérn GP	GP-BUCB, $B = 5$	0.1530 ± 0.0029	0.0037 ± 0.0013	0.0857 ± 0.0020	0.0033 ± 0.0013
	GP-BUCB, $B = 10$	0.2089 ± 0.0032	0.0036 ± 0.0014	0.1138 ± 0.0019	0.0033 ± 0.0014
	GP-BUCB, $B = 20$	0.3314 ± 0.0053	0.0033 ± 0.0012	0.1758 ± 0.0028	0.0022 ± 0.0012
	GP-AUCB, $B_{max} = 5$	0.1501 ± 0.0056	0.0130 ± 0.0052	0.0883 ± 0.0053	0.0120 ± 0.0052
	GP-AUCB, $B_{max} = 10$	0.1742 ± 0.0038	0.0062 ± 0.0018	0.0904 ± 0.0022	0.0029 ± 0.0013
	GP-AUCB, $B_{max} = 20$	0.3144 ± 0.0095	0.0217 ± 0.0040	0.1220 ± 0.0042	0.0087 ± 0.0029
	GP-AUCB Local, $B_{max} = 5$	0.1578 ± 0.0028	0.0057 ± 0.0017	0.0891 ± 0.0020	0.0050 ± 0.0016
	GP-AUCB Local, $B_{max} = 10$	0.2089 ± 0.0035	0.0022 ± 0.0007	0.1138 ± 0.0021	0.0014 ± 0.0007
	GP-AUCB Local, $B_{max} = 20$	0.3287 ± 0.0052	0.0017 ± 0.0007	0.1746 ± 0.0027	0.0013 ± 0.0007
SE GP	GP-BUCB, $B = 5$	0.0663 ± 0.0015	0.0007 ± 0.0002	0.0369 ± 0.0008	0.0004 ± 0.0002
	GP-BUCB, $B = 10$	0.1027 ± 0.0024	0.0005 ± 0.0002	0.0553 ± 0.0012	0.0004 ± 0.0002
	GP-BUCB, $B = 20$	0.1784 ± 0.0047	0.0008 ± 0.0005	0.0931 ± 0.0024	0.0002 ± 0.0001
	GP-AUCB, $B_{max} = 5$	0.0591 ± 0.0015	0.0015 ± 0.0009	0.0338 ± 0.0011	0.0013 ± 0.0009
	GP-AUCB, $B_{max} = 10$	0.0673 ± 0.0027	0.0009 ± 0.0004	0.0361 ± 0.0015	0.0003 ± 0.0001
	GP-AUCB, $B_{max} = 20$	0.0885 ± 0.0025	0.0024 ± 0.0012	0.0422 ± 0.0013	0.0010 ± 0.0007
	GP-AUCB Local, $B_{max} = 5$	0.0683 ± 0.0015	0.0011 ± 0.0003	0.0387 ± 0.0010	0.0006 ± 0.0003
	GP-AUCB Local, $B_{max} = 10$	0.0941 ± 0.0022	0.0004 ± 0.0001	0.0506 ± 0.0011	0.0002 ± 0.0001
	GP-AUCB Local, $B_{max} = 20$	0.1074 ± 0.0028	0.0004 ± 0.0001	0.0549 ± 0.0014	0.0001 ± 0.0001
Rosenbrock	GP-BUCB, $B = 5$	0.0594 ± 0.0005	0.0000 ± 0.0000	0.0371 ± 0.0003	0.0000 ± 0.0000
	GP-BUCB, $B = 10$	0.0602 ± 0.0005	0.0000 ± 0.0000	0.0375 ± 0.0003	0.0000 ± 0.0000
	GP-BUCB, $B = 20$	0.0794 ± 0.0005	0.0000 ± 0.0000	0.0468 ± 0.0003	0.0000 ± 0.0000
	GP-AUCB, $B_{max} = 5$	0.0638 ± 0.0005	0.0000 ± 0.0000	0.0379 ± 0.0003	0.0000 ± 0.0000
	GP-AUCB, $B_{max} = 10$	0.0809 ± 0.0007	0.0000 ± 0.0000	0.0423 ± 0.0003	0.0000 ± 0.0000
	GP-AUCB, $B_{max} = 20$	0.2001 ± 0.0019	0.0004 ± 0.0001	0.0570 ± 0.0005	0.0000 ± 0.0000
	GP-AUCB Local, $B_{max} = 5$	0.0598 ± 0.0004	0.0000 ± 0.0000	0.0369 ± 0.0003	0.0000 ± 0.0000
	GP-AUCB Local, $B_{max} = 10$	0.0602 ± 0.0004	0.0000 ± 0.0000	0.0373 ± 0.0003	0.0000 ± 0.0000
	GP-AUCB Local, $B_{max} = 20$	0.0806 ± 0.0004	0.0000 ± 0.0000	0.0474 ± 0.0003	0.0000 ± 0.0000
Cosines	GP-BUCB, $B = 5$	0.2199 ± 0.0012	0.0010 ± 0.0003	0.1216 ± 0.0008	0.0002 ± 0.0001
	GP-BUCB, $B = 10$	0.2265 ± 0.0015	0.0019 ± 0.0005	0.1255 ± 0.0010	0.0003 ± 0.0001
	GP-BUCB, $B = 20$	0.2401 ± 0.0015	0.0030 ± 0.0005	0.1358 ± 0.0012	0.0003 ± 0.0001
	GP-AUCB, $B_{max} = 5$	0.2719 ± 0.0014	0.0023 ± 0.0004	0.1356 ± 0.0010	0.0003 ± 0.0001
	GP-AUCB, $B_{max} = 10$	0.3930 ± 0.0007	0.0696 ± 0.0032	0.2034 ± 0.0014	0.0006 ± 0.0002
	GP-AUCB, $B_{max} = 20$	0.4205 ± 0.0015	0.1032 ± 0.0037	0.3933 ± 0.0008	0.0726 ± 0.0032
	GP-AUCB Local, $B_{max} = 5$	0.2220 ± 0.0014	0.0013 ± 0.0004	0.1217 ± 0.0010	0.0003 ± 0.0001
	GP-AUCB Local, $B_{max} = 10$	0.2251 ± 0.0015	0.0020 ± 0.0007	0.1247 ± 0.0010	0.0004 ± 0.0002
	GP-AUCB Local, $B_{max} = 20$	0.2383 ± 0.0014	0.0020 ± 0.0004	0.1340 ± 0.0010	0.0002 ± 0.0001
Vaccine Design	GP-BUCB, $B = 5$	0.9260 ± 0.0380	0.1995 ± 0.0235	0.5953 ± 0.0289	0.1796 ± 0.0230
	GP-BUCB, $B = 10$	1.2659 ± 0.0345	0.1252 ± 0.0215	0.7446 ± 0.0266	0.1200 ± 0.0214
	GP-BUCB, $B = 20$	1.8475 ± 0.0307	0.0490 ± 0.0124	1.0281 ± 0.0208	0.0345 ± 0.0097
	GP-AUCB, $B_{max} = 5$	0.9702 ± 0.0394	0.2391 ± 0.0292	0.6358 ± 0.0325	0.2149 ± 0.0280
	GP-AUCB, $B_{max} = 10$	1.1655 ± 0.0446	0.2131 ± 0.0288	0.6466 ± 0.0293	0.1515 ± 0.0222
	GP-AUCB, $B_{max} = 20$	2.1901 ± 0.0789	0.6204 ± 0.0627	1.0715 ± 0.0521	0.1958 ± 0.0258
	GP-AUCB Local, $B_{max} = 5$	1.0020 ± 0.0342	0.2335 ± 0.0287	0.6645 ± 0.0301	0.2074 ± 0.0280
	GP-AUCB Local, $B_{max} = 10$	1.1721 ± 0.0278	0.1076 ± 0.0163	0.6812 ± 0.0201	0.0825 ± 0.0138
	GP-AUCB Local, $B_{max} = 20$	1.8291 ± 0.0306	0.0226 ± 0.0059	1.0094 ± 0.0201	0.0189 ± 0.0050
SCI	GP-BUCB, $B = 5$	0.3614 ± 0.0100	0.0386 ± 0.0065	0.2140 ± 0.0071	0.0201 ± 0.0040
	GP-BUCB, $B = 10$	0.5019 ± 0.0086	0.0200 ± 0.0037	0.2757 ± 0.0052	0.0094 ± 0.0008
	GP-BUCB, $B = 20$	0.7114 ± 0.0075	0.0045 ± 0.0013	0.3775 ± 0.0041	0.0033 ± 0.0006
	GP-AUCB, $B_{max} = 5$	0.3641 ± 0.0136	0.0648 ± 0.0091	0.2203 ± 0.0100	0.0455 ± 0.0076
	GP-AUCB, $B_{max} = 10$	0.4735 ± 0.0215	0.0747 ± 0.0093	0.2548 ± 0.0114	0.0434 ± 0.0073
	GP-AUCB, $B_{max} = 20$	0.7793 ± 0.0353	0.1353 ± 0.0173	0.3831 ± 0.0197	0.0543 ± 0.0073
	GP-AUCB Local, $B_{max} = 5$	0.3701 ± 0.0109	0.0434 ± 0.0069	0.2192 ± 0.0076	0.0235 ± 0.0049
	GP-AUCB Local, $B_{max} = 10$	0.4893 ± 0.0083	0.0199 ± 0.0040	0.2674 ± 0.0050	0.0103 ± 0.0021
	GP-AUCB Local, $B_{max} = 20$	0.7197 ± 0.0070	0.0032 ± 0.0010	0.3849 ± 0.0042	0.0023 ± 0.0005

Table B.5: Average (AR) and Minimum regret (MR) for delay lengths $B = 5$, 10, and 20.

Data Set	Algorithm	Query 40	Query 100	Query 200
Matern GP	GP-UCB	0.5992 ± 0.0010	1.9532 ± 0.0037	6.5840 ± 0.0040
	GP-UCB Lazy	0.1764 ± 0.0026	0.2357 ± 0.0033	0.3824 ± 0.0044
	GP-BUCB	0.5947 ± 0.0006	1.9363 ± 0.0027	6.5302 ± 0.0053
	GP-BUCB Lazy	0.2957 ± 0.0018	0.3481 ± 0.0026	0.4592 ± 0.0035
	SM-UCB	2.9618 ± 0.0078	10.8404 ± 0.0123	44.5631 ± 0.0137
	SM-UCB Lazy	6.3990 ± 0.0186	16.0585 ± 0.0370	37.0588 ± 0.0703
	SM-MEI	3.0716 ± 0.0010	11.1101 ± 0.0035	45.1232 ± 0.0072
	SM-MEI Lazy	15.8325 ± 0.0273	38.2692 ± 0.0358	80.2965 ± 0.0446
	HBBO UCB	0.5594 ± 0.0016	1.8075 ± 0.0044	6.2063 ± 0.0064
	HBBO MEI	0.5658 ± 0.0007	1.8250 ± 0.0008	6.2486 ± 0.0009
	GP-AUCB	0.6699 ± 0.0001	1.8984 ± 0.0003	6.2417 ± 0.0020
	GP-AUCB Lazy	0.3527 ± 0.0021	0.4040 ± 0.0029	0.5125 ± 0.0039
	GP-AUCB Local	0.4708 ± 0.0008	1.5885 ± 0.0030	5.6950 ± 0.0071
	GP-AUCB Lazy Local	0.2823 ± 0.0022	0.3338 ± 0.0029	0.4420 ± 0.0040
SE GP	GP-UCB	0.5993 ± 0.0001	1.9520 ± 0.0004	6.5182 ± 0.0011
	GP-UCB Lazy	0.2891 ± 0.0061	0.4265 ± 0.0109	0.6406 ± 0.0170
	GP-BUCB	0.6011 ± 0.0001	1.9462 ± 0.0003	6.5005 ± 0.0008
	GP-BUCB Lazy	0.3703 ± 0.0053	0.4982 ± 0.0100	0.7039 ± 0.0165
	SM-UCB	2.9105 ± 0.0009	10.6819 ± 0.0028	44.0896 ± 0.0062
	SM-UCB Lazy	7.3562 ± 0.0410	17.7201 ± 0.0773	39.3720 ± 0.1229
	SM-MEI	3.0133 ± 0.0010	10.9425 ± 0.0028	44.6246 ± 0.0056
	SM-MEI Lazy	17.6639 ± 0.0782	41.0165 ± 0.1256	83.6221 ± 0.1674
	HBBO UCB	0.5549 ± 0.0006	1.7936 ± 0.0008	6.1503 ± 0.0013
	HBBO MEI	0.5630 ± 0.0006	1.8161 ± 0.0007	6.1973 ± 0.0011
	GP-AUCB	0.6749 ± 0.0001	1.8999 ± 0.0003	6.2147 ± 0.0007
	GP-AUCB Lazy	0.4348 ± 0.0057	0.5588 ± 0.0103	0.7555 ± 0.0162
	GP-AUCB Local	0.4684 ± 0.0001	1.5820 ± 0.0003	5.6564 ± 0.0010
	GP-AUCB Lazy Local	0.3206 ± 0.0050	0.4442 ± 0.0095	0.6404 ± 0.0155
Rosenbrock	GP-UCB	0.5916 ± 0.0030	1.9470 ± 0.0091	6.4193 ± 0.0172
	GP-UCB Lazy	0.3356 ± 0.0013	0.4521 ± 0.0020	0.6437 ± 0.0034
	GP-BUCB	0.5841 ± 0.0009	1.9074 ± 0.0032	6.3437 ± 0.0071
	GP-BUCB Lazy	0.3787 ± 0.0012	0.4900 ± 0.0020	0.6732 ± 0.0034
	SM-UCB	2.8155 ± 0.0010	10.4979 ± 0.0023	43.1365 ± 0.0113
	SM-UCB Lazy	6.4663 ± 0.0075	15.9830 ± 0.0109	36.1740 ± 0.0158
	SM-MEI	2.9311 ± 0.0077	10.7848 ± 0.0181	43.6449 ± 0.0199
	SM-MEI Lazy	15.5865 ± 0.0180	36.8226 ± 0.0204	76.0677 ± 0.0253
	HBBO UCB	0.6033 ± 0.0012	1.8202 ± 0.0017	6.0460 ± 0.0019
	HBBO MEI	0.6076 ± 0.0010	1.8366 ± 0.0011	6.0840 ± 0.0014
	GP-AUCB	0.6510 ± 0.0001	1.8495 ± 0.0004	6.0354 ± 0.0008
	GP-AUCB Lazy	0.4372 ± 0.0010	0.5463 ± 0.0018	0.7317 ± 0.0031
	GP-AUCB Local	0.4582 ± 0.0001	1.5449 ± 0.0002	5.4983 ± 0.0039
	GP-AUCB Lazy Local	0.3569 ± 0.0011	0.4609 ± 0.0016	0.6349 ± 0.0029
Cosines	GP-UCB	0.5829 ± 0.0001	1.9113 ± 0.0004	6.3865 ± 0.0013
	GP-UCB Lazy	0.2750 ± 0.0007	0.3654 ± 0.0010	0.4913 ± 0.0015
	GP-BUCB	0.5810 ± 0.0001	1.9094 ± 0.0004	6.3497 ± 0.0011
	GP-BUCB Lazy	0.3452 ± 0.0009	0.4267 ± 0.0012	0.5242 ± 0.0013
	SM-UCB	2.8365 ± 0.0014	10.6211 ± 0.0046	43.3563 ± 0.0134
	SM-UCB Lazy	5.9717 ± 0.0038	15.0289 ± 0.0059	34.3330 ± 0.0123
	SM-MEI	2.9458 ± 0.0010	10.9126 ± 0.0049	43.9570 ± 0.0134
	SM-MEI Lazy	14.8944 ± 0.0084	35.7111 ± 0.0094	74.2792 ± 0.0243
	HBBO UCB	0.6536 ± 0.0012	1.8741 ± 0.0013	6.1179 ± 0.0015
	HBBO MEI	0.6654 ± 0.0014	1.9006 ± 0.0015	6.1691 ± 0.0017
	GP-AUCB	0.6493 ± 0.0002	1.8570 ± 0.0003	6.0636 ± 0.0008
	GP-AUCB Lazy	0.3962 ± 0.0006	0.4772 ± 0.0010	0.5727 ± 0.0011
	GP-AUCB Local	0.4602 ± 0.0001	1.5595 ± 0.0004	5.5394 ± 0.0008
	GP-AUCB Lazy Local	0.4940 ± 0.0009	0.5946 ± 0.0022	0.6863 ± 0.0023
Vaccine Design	GP-UCB	1.8238 ± 0.0005	6.0469 ± 0.0019	20.3408 ± 0.0047
	GP-UCB Lazy	0.6347 ± 0.0094	0.7021 ± 0.0094	0.9267 ± 0.0094
	GP-BUCB	1.8252 ± 0.0004	5.9950 ± 0.0016	20.1145 ± 0.0039
	GP-BUCB Lazy	1.1121 ± 0.0024	1.1472 ± 0.0024	1.2584 ± 0.0024
	SM-UCB	8.3346 ± 0.0032	32.0995 ± 0.0167	134.3169 ± 0.0458
	SM-UCB Lazy	22.5270 ± 0.2438	46.4903 ± 0.7422	99.7192 ± 1.6901
	SM-MEI	8.5207 ± 0.0023	32.6054 ± 0.0135	135.4947 ± 0.0391
	SM-MEI Lazy	49.0936 ± 0.2575	115.2695 ± 0.7228	243.0290 ± 1.2733
	HBBO UCB	2.3149 ± 0.0128	6.2659 ± 0.0234	19.8061 ± 0.0377
	HBBO MEI	2.2665 ± 0.0121	6.2584 ± 0.0211	19.9313 ± 0.0376
	GP-AUCB	2.2982 ± 0.0003	6.1147 ± 0.0014	19.5122 ± 0.0041
	GP-AUCB Lazy	1.2533 ± 0.0049	1.2877 ± 0.0050	1.3968 ± 0.0049
	GP-AUCB Local	1.4302 ± 0.0003	4.8850 ± 0.0013	17.4855 ± 0.0039
	GP-AUCB Lazy Local	1.0196 ± 0.0083	1.0676 ± 0.0090	1.1843 ± 0.0095
Spinal Cord Therapy	GP-UCB	0.0721 ± 0.0001	0.2404 ± 0.0006	0.8161 ± 0.0021
	GP-UCB Lazy	0.0138 ± 0.0001	0.0323 ± 0.0002	0.0917 ± 0.0003
	GP-BUCB	0.0721 ± 0.0000	0.2395 ± 0.0003	0.8134 ± 0.0014
	GP-BUCB Lazy	0.0236 ± 0.0001	0.0421 ± 0.0001	0.1042 ± 0.0004
	SM-UCB	0.5920 ± 0.0005	1.9495 ± 0.0020	6.7668 ± 0.0024
	SM-UCB Lazy	0.7619 ± 0.0009	2.0412 ± 0.0039	4.8634 ± 0.0116
	SM-MEI	0.6547 ± 0.0002	2.1107 ± 0.0019	7.0953 ± 0.0037
	SM-MEI Lazy	1.8187 ± 0.0018	4.4822 ± 0.0096	9.3589 ± 0.0268
	HBBO UCB	0.0702 ± 0.0003	0.2333 ± 0.0008	0.8007 ± 0.0019
	HBBO MEI	0.0759 ± 0.0003	0.2482 ± 0.0009	0.8317 ± 0.0021
	GP-AUCB	0.0742 ± 0.0001	0.2268 ± 0.0004	0.7722 ± 0.0019
	GP-AUCB Lazy	0.0266 ± 0.0001	0.0442 ± 0.0002	0.1050 ± 0.0004
	GP-AUCB Local	0.0585 ± 0.0003	0.1996 ± 0.0007	0.7152 ± 0.0018
	GP-AUCB Lazy Local	0.0342 ± 0.0004	0.0533 ± 0.0006	0.1186 ± 0.0012

Table B.6: Mean wall-clock execution times and standard deviations of estimate (S).

Appendix C

Action-matched Animal Plots

During two of the animal experiment runs (animal 5, run 1 and animal 7) a substantial number of actions performed by the human experimenter were missed or dropped. An alternate view of these experiments is presented here, where “pass” actions are inserted for those missed by either the algorithm or the human experimenter. Without inserted passes, the same action indices for the human and algorithm do not correspond to the same point in time, and visual interpretation of the regret plots is difficult; after inserted passes, this synchrony is restored.

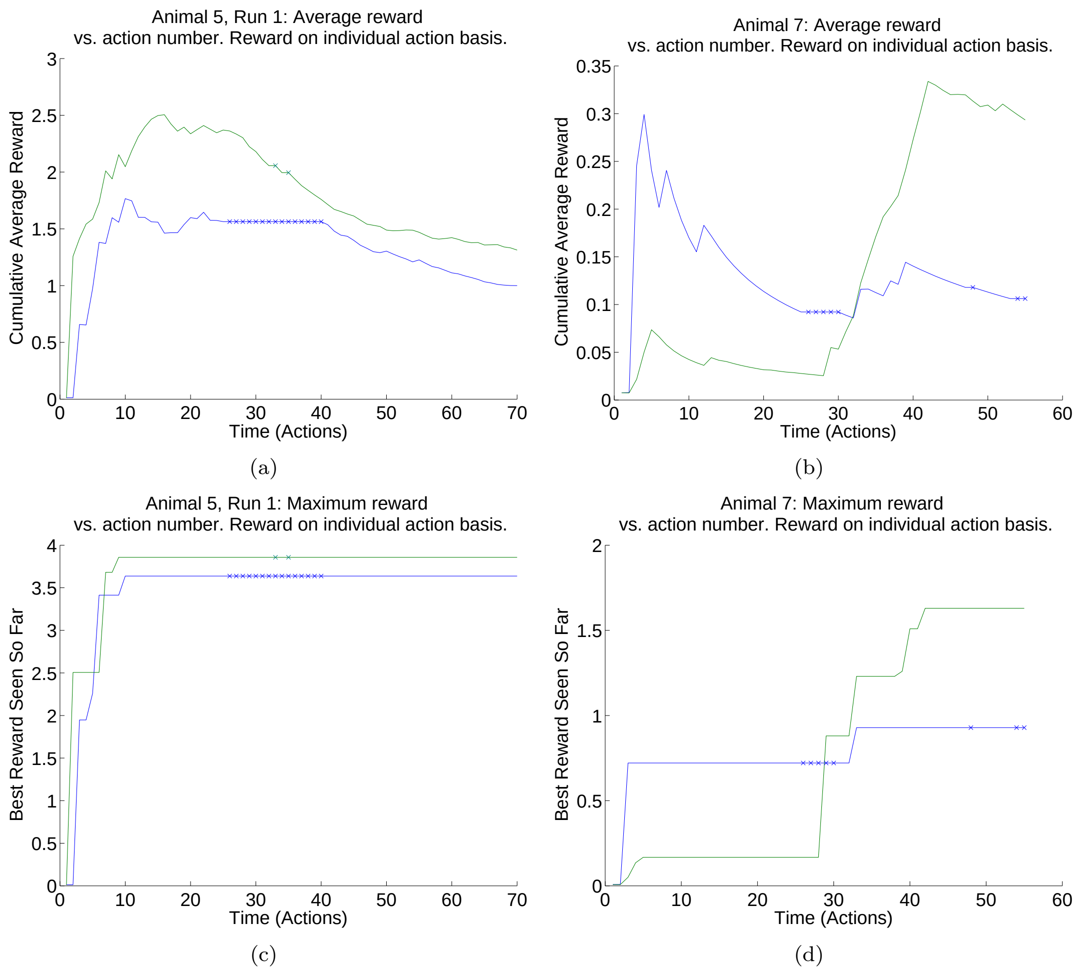


Figure C.1: Action-matched plots for animal 5, run 1 and animal 7. A missed action is treated as a "pass" to restore action synchrony between the human experimenter and the algorithm. These actions are shown with an "x" in the plots above. C.1(a) & (c): Animal 5, run 1. During this experimental run, the human experimenter missed a full day of experiments (P15). Compare these plots to Figures 4.8(b) and (d), which do not have the passes corresponding to the three missed batches on P15. (b) & (d): Animal 7. The human experimenter did not conduct a fourth and final batch on the second testing day (P12). Several actions were also missing from the third testing day, P13. These plots are action compensated versions of Figures 4.11(b) and (d).

Appendix D

Toward Human Studies: Mathematical Results

D.1 Decision-making with an Aggregated Objective

When trying to use several, possibly related GP models $f_1, \ldots, f_n$ to make a decision about a known function r of those individual GPs, it is natural to attempt to apply a UCB-like approach to the problem. Unfortunately, unless r is a linear combination of these individual GPs, $r(\mathbf{f})$ is not itself a Gaussian process, nor is the posterior over $r(\mathbf{f}(\boldsymbol{x}))$ Gaussian. This problem even arises if one common and natural formulation of a reward function, that of penalized deviation from a target $\mathbf{t}$ via a weighted norm term, is used, e.g.,

$$r(\mathbf{f}(\boldsymbol{x})) = -\sqrt{(\mathbf{f}(\boldsymbol{x}) - \mathbf{t})^T W (\mathbf{f}(\boldsymbol{x}) - \mathbf{t})}, \tag{D.1}$$

where W is a symmetric, positive definite penalty matrix, such that r is -1 times a weighted 2-norm in $\mathbb{R}^n$. Such an objective function makes a great deal of sense in terms of convex optimization, and has a unique global maximum at $\mathbf{f}(\boldsymbol{x}) = \mathbf{t}$. Further, for any $\boldsymbol{x}$, the posterior over $\mathbf{f}(\boldsymbol{x})$ is $\mathbf{f}(\boldsymbol{x}) \sim \mathcal{N}(\mu(\boldsymbol{x}), \Sigma(\boldsymbol{x}))$, and as $\mu - \mathbf{t}$ becomes very large, the distribution of the weighted squared norm begins to look like the corresponding marginalization of the posterior onto the unit vector in the direction $\mu - \mathbf{t}$; this marginal distribution is a Gaussian. However, of the most interest in terms of active learning is the region near the optimum, where such an $f(\mathbf{f})$ is most strongly non-Gaussian.

Inspired by GP-UCB and GP-BUCB, it seems reasonable that it would be desirable to create a decision function of form

$$\boldsymbol{x}_t = \underset{\boldsymbol{x} \in D}{\operatorname{argmax}} \left[\mathbb{E}[r(\mathbf{f}(\boldsymbol{x})) | \boldsymbol{y}_{1:\text{fb}[t]}] + \beta_t^{1/2} \sqrt{\mathbf{Var}\left(r(\mathbf{f}(\boldsymbol{x})) | \boldsymbol{y}_{1:\text{fb}[t]} \right)} \right], \tag{D.2}$$

which once again trades off exploitation, captured by the mean reward term on the left, with exploration, captured by the standard deviation term on the right. It is possible to calculate

$\mathbb{E}[r(\mathbf{f}(\boldsymbol{x}))^2|\boldsymbol{y}_{1:\mathrm{fb}[t]}] = \mathbb{E}[(\mathbf{f}(\boldsymbol{x})-\mathbf{t})^T W(\mathbf{f}(\boldsymbol{x})-\mathbf{t})|\boldsymbol{y}_{1:\mathrm{fb}[t]}] = (\mu_{\mathrm{fb}[t]}(\boldsymbol{x})-\mathbf{t})^T W(\mu_{\mathrm{fb}[t]}(\boldsymbol{x})-\mathbf{t})+\mathrm{trace}(W\Sigma_{\mathrm{fb}[t]})$. This leaves calculating the expected reward, $\mathbb{E}[r(\mathbf{f}(\boldsymbol{x}))|\boldsymbol{y}_{1:\mathrm{fb}[t]}]$. Unfortunately, despite its relation to the χ distribution, I was unable to obtain a general expression for this expectation, and so I made recourse to bounding arguments. By use of Jensen's inequality, which states that for a convex function $h(\boldsymbol{x})$,

$$\mathbb{E}[h(\boldsymbol{x})] \geq h(\mathbb{E}[\boldsymbol{x}]), \tag{D.3}$$

it is possible to derive an upper bound

$$\mathbb{E}[r(\mathbf{f}(\boldsymbol{x}))|\boldsymbol{y}_{1:\mathrm{fb}[t]}] \leq -\sqrt{(\mu_{\mathrm{fb}[t]}(\boldsymbol{x})-\mathbf{t})^T W(\mu_{\mathrm{fb}[t]}(\boldsymbol{x})-\mathbf{t})}$$

via the concavity of r with respect to $\mathbf{f}$, as well as a lower bound

$$\mathbb{E}[r(\mathbf{f}(\boldsymbol{x}))|\boldsymbol{y}_{1:\mathrm{fb}[t]}] \geq \sqrt{(\mu_{\mathrm{fb}[t]}(\boldsymbol{x})-\mathbf{t})^T W(\mu_{\mathrm{fb}[t]}(\boldsymbol{x})-\mathbf{t}) + \mathrm{trace}(W\Sigma_{t-1})}$$

by noting that $-\sqrt{r}$ is convex with respect to r over $r \in \mathbb{R}^+$. Using the definition of the variance in terms of the expectation of the square and the square of the expectation, and substituting in the upper bound on $\mathbb{E}[r(\mathbf{f}(\boldsymbol{x}))|\boldsymbol{y}_{1:\mathrm{fb}[t]}]$, it can be shown that

$$\mathbf{Var}\left(r(\mathbf{f}(\boldsymbol{x}))|\boldsymbol{y}_{1:\mathrm{fb}[t]}\right) = \mathbb{E}[r(\mathbf{f}(\boldsymbol{x}))^2|\boldsymbol{y}_{1:\mathrm{fb}[t]}] - \mathbb{E}[r(\mathbf{f}(\boldsymbol{x}))|\boldsymbol{y}_{1:\mathrm{fb}[t]}]^2 \leq \mathrm{trace}(W\Sigma_{t-1}). \tag{D.4}$$

By analogy to the GP-UCB and GP-BUCB decision rules (Equations 3.5 and 3.7), and using the upper bounds above to create a term capturing the reward and another term capturing the uncertainty, this suggests a decision rule of form

$$\boldsymbol{x}_t = \operatorname*{argmax}_{\boldsymbol{x}\in D}\left[-\sqrt{(\mu_{\mathrm{fb}[t]}(\boldsymbol{x})-\mathbf{t})^T W(\mu_{\mathrm{fb}[t]}(\boldsymbol{x})-\mathbf{t})} + \beta_t^{1/2}\sqrt{\mathrm{trace}(W\Sigma_{t-1}(\boldsymbol{x}))}\right] \tag{D.5}$$

should have useful characteristics. Note that for the scalar case, with a very large target t, the decision rule reduces to that of GP-BUCB. Further, as we learn more and more about the function near the optima, $\Sigma_{t-1}(\boldsymbol{x})$ should decrease for these decisions, making the upper bound on $\mathbb{E}[r(\mathbf{f}(\boldsymbol{x}))|\boldsymbol{y}_{1:\mathrm{fb}[t]}]$ tighter. Additionally, for $\mu_{t-1}(\boldsymbol{x}) - \mathbf{t}$ large, the decision rule can also be expected to closely bound the actual form in Equation D.2, allowing us to disregard these decisions as poor-performing. This leaves the poor cases as those in which $\Sigma_{t-1}(\boldsymbol{x})$ is very large and $\mu_{t-1}(\boldsymbol{x}) - \mathbf{t}$ is small; in this case, overestimating either the mean or variance should result in allocation of observations to these actions, driving down $\Sigma_{t-1}(\boldsymbol{x})$ and resolving the issue through observation. This decision rule is also practical because it is very easy to calculate; if the posterior over the values $\mathbf{f}(\boldsymbol{x})$ is available, this decision rule simply requires some linear algebraic calculations. Further, it can be shown that, under the assumption that observations can only be added to the observation set, but none can leave, the term $\sqrt{\text{trace}(W\Sigma_{t-1}(\boldsymbol{x}))}$ is non-increasing (see Appendix D.2). Because this term is non-increasing as observations are added to the observation set, the calculation of $\sqrt{\text{trace}(W\Sigma_{t-1}(\boldsymbol{x}))}$ can be done lazily, as can be done for the standard deviation in Section 3.5, enabling substantial computational savings.

D.2 Proof Multi-Muscle Uncertainty Term is Non-Increasing

A quite useful characteristic of the GP-BUCB decision rule, Equation (3.7), is that the uncertainty term (i.e., the standard deviation) cannot increase as observations are added. For computational reasons, it is important to demonstrate that Equation (D.5) also has the same property. As a first step, we define the matrix root of the weight matrix W as a symmetric, positive definite matrix $W^{1/2} = (W^{1/2})^T > 0$, such that $(W^{1/2})^2 = W$. Such a matrix can be constructed by noting that $W = VDV^T$, where $V \in \mathbb{R}^{n\times n}$ is a matrix whose columns are the set of orthonormal right eigenvectors of W and D is the diagonal matrix of the corresponding eigenvalues of W; choosing $W^{1/2} = VD^{1/2}V^T$ produces the desired properties. We then consider the uncertainty of the algorithm's estimate of $\mathbf{f}(\boldsymbol{x})$ after steps t and t' of the algorithm, where $t' > t$, the corresponding observation sets $\mathbf{y}_t$ and $\mathbf{y}_{t'}$, and the sets of past actions $\boldsymbol{X}_t$ and $\boldsymbol{X}_{t'}$. We may describe the posterior covariance function between stimuli $\boldsymbol{x}$ and $\boldsymbol{x}'$ and muscle indices i and j as $k_t((\boldsymbol{x}, i), (\boldsymbol{x}', j)) = k((\boldsymbol{x}, i), (\boldsymbol{x}', j))|\mathbf{y}_t$ and write the posterior covariance matrix at time t' for $\mathbf{f}(\boldsymbol{x})$ as

$$\Sigma_{t'}(\boldsymbol{x}) = \Sigma_t(\boldsymbol{x}) - \mathbf{k}_t(K + \sigma_n^2 I)^{-1}\mathbf{k}_t^T,$$

where $\boldsymbol{X}_{t+1;t'} = \boldsymbol{X}t' \setminus \boldsymbol{X}t$ is the set of observations occurring between times t and t', $\mathbf{k}_t = \mathbf{k}_t(\boldsymbol{x}, \boldsymbol{X}_{t+1:t'}) \in \mathbb{R}^{n\times[(t'-t)\times n]}$ is the covariance between the observations associated with $\boldsymbol{X}_{t+1;t'}$, including all n channels, and $K_t = K_t(\boldsymbol{X}_{\mathbf{t+1:t'}}, \boldsymbol{X}_{\mathbf{t+1:t'}}) \in \mathbb{R}^{[(t'-t)\times n]\times[(t'-t)\times n]}$ is the posterior covariance at time t between the noisy observations $\mathbf{y}_{t+1:t'}$ of $\mathbf{f}(\boldsymbol{X}_{t+1:t'})$ in $\boldsymbol{X}_{t+1:t'}$. Multiplying left and right by $W^{1/2}$, and then using the linearity of the trace and its invariance to circular permutations of symmetric matrices, we obtain

$$\text{trace}(W\Sigma_{t'}(\boldsymbol{x})) = \text{trace}(W\Sigma_t(\boldsymbol{x})) - \text{trace}(W^{1/2}\mathbf{k}_t(K_t + \sigma_n^2 I)^{-1}\mathbf{k}_t^T W^{1/2}). \tag{D.6}$$

Noting that $\mathbf{h} = W^{1/2}\mathbf{k}_t\mathbf{y}_{t+1:t'} \in \mathbb{R}^{n\times 1}$ is a linear combination of the $n(t'-t)$ multivariate Gaussian observations, $\mathbf{h}$ also has a multivariate Gaussian distribution, such that its covariance matrix, $W^{1/2}\mathbf{k}_t(K_t + \sigma_n^2 I)^{-1}\mathbf{k}_t^T W^{1/2}$, is positive semi-definite. Since the trace of this matrix is therefore non-negative, it follows from Equation (D.6) that $\text{trace}(W\Sigma_{t'}(\boldsymbol{x})) \leq \text{trace}(W\Sigma_t(\boldsymbol{x}))$ for $t' > t$.

D.3 Path-Based Decision Rules

As discussed in Section 5.4.2.3, it may be desirable to plan for smooth paths of length no more than B which travel from the present stimulus state through the decision set. This set of possible paths may be denoted L. While there are potentially exponentially many paths through the graph of possible stimuli, if there is a set of restrictions on path construction such that each end-point (i.e., $\boldsymbol{x} \in D$) may be reached by at most one path, these restrictions imply $|L| \leq |D|$. One reasonable idea for selecting paths from L is to extend the decision rule used by the GP-BUCB algorithm, resulting in the following equation:

$$\boldsymbol{X}_t = \underset{\boldsymbol{X} \in L}{\operatorname{argmax}} \left[\sum_{\tau=t}^{t+B-1} \left(\mu_{\mathrm{fb}[t]}(\boldsymbol{x}_\tau) + \beta_{\mathrm{fb}[t]}^{1/2} \sigma_{\tau-1}(\boldsymbol{x}_\tau) \right) \right], \tag{D.7}$$

where $\boldsymbol{X} = \{\boldsymbol{x}_t, \ldots \boldsymbol{x}_{t+B-1}\}$. This construction follows the form of the GP-BUCB decision rule, and might even be amenable to the same confidence interval analysis, at least locally. However, this decision rule may philosophically differ from the GP-BUCB rule in that the quantity which corresponds to information gained no longer maps easily to the actual information gain $I(f; \boldsymbol{y}(\boldsymbol{X}) | \boldsymbol{y}(\boldsymbol{X}_{\mathrm{fb}[t]}))$. Motivated by the transformation between $\sigma_{t-1}(\boldsymbol{x}_t)$ and $I(f; \boldsymbol{y}(\boldsymbol{x}_t) \mid \boldsymbol{y}(\boldsymbol{X}_{\mathrm{fb}[t]}))$, i.e.,

$$\sigma_{t-1}(\boldsymbol{x}_t) = \sigma_n \sqrt{-1 + \exp(I(f; \boldsymbol{y}(\boldsymbol{x}_t) \mid \boldsymbol{y}(\boldsymbol{X}_{\mathrm{fb}[t]})))}, \tag{D.8}$$

it may be reasonable to apply the same transformation to $I(f; \boldsymbol{y}(\boldsymbol{X}) | \boldsymbol{y}(\boldsymbol{X}_{\mathrm{fb}[t]}))$ to obtain a quantity $e(\boldsymbol{X})$ which corresponds to the information gain from the group of observations as follows:

$$\begin{aligned} e(\boldsymbol{X}) &= \sigma_n \sqrt{-1 + \exp(I(f; \boldsymbol{y}(\boldsymbol{X}) | \boldsymbol{y}(\boldsymbol{X}_{\mathrm{fb}[t]})))} \\ &= \sigma_n \sqrt{-1 + \prod_{\tau=t}^{t+B-1} (1 + \sigma_n^{-2} \sigma_{\tau-1}(\boldsymbol{x}_\tau))} \end{aligned} \tag{D.9}$$

where $\boldsymbol{X}$ is again $\boldsymbol{X} = \{\boldsymbol{x}_t, \ldots \boldsymbol{x}_{t+B-1}\}$. This yields a decision rule of the form

$$\boldsymbol{X}_t = \underset{\boldsymbol{X} \in L}{\operatorname{argmax}} \left[\sum_{\tau=t}^{t+B-1} [\mu_{\mathrm{fb}[t]}(\boldsymbol{x}_\tau)] + \beta_{\mathrm{fb}[t]}^{1/2} e(\boldsymbol{X}) \right]. \tag{D.10}$$

In either or both of these cases, it might be appropriate to consider the possibility that the experiment might have to be stopped during the traversal of $\boldsymbol{X}$ with some probability. Letting the uniform probability of failure of each transition be $1 - \lambda$, and assuming the reward and observation are obtained even if the individual action is a failure, the discounted version of the first decision rule is

$$\boldsymbol{X}_t = \underset{\boldsymbol{X}\in L}{\operatorname{argmax}} \left[\sum_{\tau=t}^{t+B-1} [\lambda^{\tau-t}(\mu_{\mathrm{fb}[t]}(\boldsymbol{x}_\tau) + \beta_{\mathrm{fb}[t]}^{1/2}\sigma_{\tau-1}(\boldsymbol{x}_\tau))] \right]. \tag{D.11}$$

This decision rule has been implemented for a version of the human experimental code which is intended to search over the space of voltage and frequency parameters corresponding to a fixed set of active electrodes. Similarly, the discounted version of Equation (D.9), designated $e_\lambda(\boldsymbol{X})$, is

$$e_\lambda(\boldsymbol{X}) = \sigma_n \sqrt{-1 + \prod_{\tau=t}^{t+B-1} [(1+\sigma_n^{-2}\sigma_{\tau-1}(\boldsymbol{x}_\tau))^{\lambda^{\tau-t}}]} \tag{D.12}$$

and the corresponding decision rule becomes

$$\boldsymbol{X}_t = \underset{\boldsymbol{X}\in L}{\operatorname{argmax}} \left[\sum_{\tau=t}^{t+B-1} [\lambda^{\tau-t}\mu_{\mathrm{fb}[t]}(\boldsymbol{x}_\tau)] + \beta_{\mathrm{fb}[t]}^{1/2} e_\lambda(\boldsymbol{X}) \right]. \tag{D.13}$$

Either of these frameworks may make sense as a method of selecting paths through the stimulus space.

Appendix E

Code Availability

Code implementing the algorithms discussed in Chapter 3 is available at `www.its.caltech.edu/~tadesaut/`.