Reinforcement learning in models of adaptive medical treatment strategies

Robert Durham Vincent

Doctor of Philosophy

School of Computer Science

McGill University

Montreal, Quebec

June 1, 2014

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Doctor of Philosophy

DEDICATION

This thesis is dedicated to my family, especially my wife, Debra, my daughters, Eleanor and Lucille, and my parents, Chester and Elizabeth.

ACKNOWLEDGMENTS

While a doctoral dissertation is often reckoned as an individual accomplishment, in reality it is a collective effort that would not be possible without the contributions of many people.

First and foremost, I acknowledge the unfailing support of my advisor, Joelle Pineau. She has been a source of creativity, perspective, and insight throughout my graduate studies.

My epilepsy research has benefited from working with two superb computer scientists, Arthur Guez and Keith Bush. Arthur's implementation of fitted Q iteration formed the basis of many of the experiments in this thesis, and Keith was an important source of ideas and consolation during my work on building a model of epilepsy. My modeling work in epilepsy was also inspired by discussions with Aaron Courville, who helped initiate this line of investigation. In addition, I gratefully acknowledge the assistance of Drs. Gabriella Panuccio and Massimo Avoli, who collected the *in vitro* electrophysiology data and provided valuable comments and suggestions on my epilepsy research.

Professors Aditya Mahajan and Issam El Naqa provided assistance and support for the radiation therapy portions of this project. Each has been generous in sharing their ideas and supportive of the project in general. I am especially indebted to Dr. El Naqa's lab for sharing their cell culture irradiation data.

I have also benefited from the scholarship support of both the Fonds de recherche du Québec - Nature et technologies (FRQNT) and Natural Sciences and Engineering Research Council of Canada (NSERC). Without this support, it is unlikely that I could have pursued graduate study. This research has also benefited from additional funding support of the Canadian Institutes of Health Research (CIHR) and NSERC.

I also acknowledge the contribution of the entire Reasoning and Learning Lab. I am indebted to the many fellow lab members who have contributed to making my graduate work more enjoyable and educational, especially Amin Atrash, Cosmin Paduraru, Jordan Frank, Pablo Castro, and Gheorghe Comanici.

ABSTRACT

This thesis investigates the application of reinforcement learning methods in the design and optimization of treatment strategies for several medical problems. We show that many of these problems can be posed as Markov decision processes, a general framework for modeling sequential decision making problems under uncertainty, give specific formulations, and explore possible solution methods.

We examine three problems from the medical treatment literature, first formulating them as computational models to assist in the development and evaluation of our reinforcement learning methods. For our first case study, we consider the problem of optimizing electrical stimulation patterns for the treatment of epilepsy, with a special focus on creating a computational model of epilepsy that exhibits characteristics similar to those observed in parahippocampal rodent brain slice experiments *in vitro*. Secondly, we examine the related problem of designing electrical stimulation strategies for the treatment of Parkinson's disease, using a coupled oscillator model as proposed by Tass (2003). Finally, we develop a differential model of cell population response to radiation therapy for cancer, and formulate the fractionation scheduling of radiation therapy as a Markov decision process and attempt to examine whether we can find an optimal non-uniform fractionation schedule.

Based on these and other results, we also provide a practical discussion of methods and techniques for the application of reinforcement learning in these kinds of medical domains, especially when a computational model is available.

ABRÉGÉ

Cette thèse étudie l'application de l'apprentissage par renforcement dans la conception et l'optimisation de stratégies de traitement pour plusieurs problèmes médicaux. Nous montrons que la plupart de ces problèmes peuvent être posés comme processus de décision de Markov, un cadre général pour la modélisation prise de décisions séquentielle des problèmes dans l'incertitude. Nous donnons quelques formulations spécifiques, et nous explorons des méthodes de résolution.

Nous étudions trois problèmes de la littérature de traitement médical. D'abord, nous les formulons comme des modèles informatiques pour aider à l'élaboration et l'évaluation de nos méthodes d'apprentissage par renforcement. Pour notre première étude de cas, nous considérons le problème de l'optimisation des modèles de stimulation électrique pour le traitement de l'épilepsie, avec un accent particulier sur la création d'un modèle informatique de l'épilepsie qui présente des caractéristiques similaires à celles observées dans les expériences avec la tranche parahippocampal des rongeurs *in vitro*. Deuxièmement, nous examinons le problème de la conception de stratégies pour la stimulation cérébrale profonde pour la maladie de Parkinson, en utilisant un modèle d'oscillateur couplé comme proposé par Tass (2003). Enfin, nous développons un modèle différentiel de réponse d'une population cellulaire à la radiothérapie pour le cancer, et nous formulons le fractionnement de la dose radiothérapie comme un processus de décision de Markov et nous examinons si nous pouvons trouver si la politique optimale est non-uniforme.

Sur la base de ces résultats et d'autres, nous fournissons également une discussion pratique des méthodes et techniques pour l'application de l'apprentissage par renforcement dans les autres domaines médicaux, en particulier quand un modéle informatique est disponible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure page

LIST OF ALGORITHMS

CHAPTER 1
Introduction

The design of medical treatment strategies has traditionally been dominated by the standardized application of clinical studies and experience, with a substantial reliance on trial-and-error to guide decisions for individual patients. Many disease treatments, for either chronic or acute conditions, require clinicians to make a series of sequential decisions, choosing whether to continue, change, or stop treatment. They do this under difficult, often changing situations, making decisions based on a patient's overall medical history and observed responses to therapies, as well as general clinical practice and intuition. Patient responses often differ in both degree and timing. Historically, the process of tailoring a treatment strategy to a patient is one of educated trial-and-error guided by the individual expertise of the clinician. This expertise in turn is based on both personal experience and information derived from conventional clinical studies, in which, because of cost and time constraints, data are often aggregated for a modest number of treatments and patients over some arbitrarily defined finite duration.

In the last decade or so there has been a growing interest in discovering more formally optimal *adaptive treatment strategies* for the management of diseases. The goal is the development of decision-making strategies that can predict which treatment, or sequences of treatments, will produce the optimal outcome for a patient, given their specific medical history and responses to prior treatment choices. These strategies would be derived from a mix of clinical and experimental data, and may be further refined and personalized as experience with a particular patient is accumulated. The inputs to such a strategy would be quantitative observations of a patient's condition and history, and the output would be a choice of treatment for

that time interval, such that the probability of a favorable outcome for the patient would be maximized over some time horizon (Murphy, 2005; Ernst et al., 2006).

Compared to past practice, recent evidence suggests that treatment outcomes can vary in a much more individualized manner, according to broad patient characteristics including sex, race, or even hair color (Liem et al., 2004). At a more detailed level, many specific biological markers have been identified that can affect responses to drugs and other medical treatments (Potti et al., 2010).

Many opportunities and challenges remain on the path towards the development of personalized, adaptive treatments. Even in these relatively early stages, there is tantalizing, if often preliminary, evidence that personalized or adaptive methods may be able to improve outcomes in many clinical settings (Sun et al., 2008; Rosin et al., 2011; Malof & Gaweda, 2011; Li et al., 2011). In addition to improved patient outcomes in a general sense, there is good reason to believe that adaptive strategies may facilitate a number of more specific goals:

- Greater specificity of treatments in time and space may reduce side effects (Sun et al., 2008).
- Increase the safety of IMRT treatments (Li et al., 2011).
- Greater convenience from the patient's perspective (McKay, 2009).

In addition to direct benefits such as these, the development of optimal adaptive treatment strategies holds promise of motivating a number of improvements in various enabling techniques (Adams et al., 2005). For these and other reasons, these methods clearly may lead to improvements in medical treatment on multiple fronts.

However, many challenges remain to the widespread adoption of adaptive strategies. First and foremost, these efforts impose novel requirements on both the total amount and the specific kinds of clinical data collected. Adaptive strategies benefit from access to data sets that are both sufficiently large and thoroughly randomized with respect to the range of possible treatment choices and patient types.

This requirement may, in turn, influence the design of clinical trials to better support these kinds of decision-making models (Murphy, 2005). Secondly, as we will discuss in this thesis, while many algorithms and methods exist within the literature on the control of dynamical systems, not all are well-suited for medical applications. As the field develops, refinements to the control algorithms will no doubt address some important practical limitations. There is also a need for guidelines suggesting the kinds of methods which may be appropriate for these applications. Thirdly, there is a lack of detailed models of the progression of many diseases. Ideally, we would like to have control methods that can function well in the absence of such models, yet make good use of the models where they do exist. Finally, validation of these strategies is an extremely important and difficult topic, both from the mathematical and the clinical perspective. Validation of adaptive methods is a large topic in and of itself, and we can only give a limited discussion of the issues in this thesis.

1.1 Problem domain

In this thesis we explore the possibility of applying *reinforcement learning* (Sutton & Barto, 1998) (RL) as a general approach for the design of adaptive clinical strategies for the treatment of chronic disease. Reinforcement learning is a paradigm which enables a computational agent to derive optimal decision-making strategies from observational data in application domains in which a model of the underlying process may be imperfect or absent. This is in contrast with other control strategies which often require that the agent have access to a complete, or nearly complete, model of the dynamics of the process to be controlled. In addition, RL approaches are intended to operate optimally in applications which exhibit uncertain responses to specific actions. These methods automatically take into account the fact that not every patient responds to the same treatment in exactly the same way.

A reinforcement learning problem is typically described as the problem of an *agent* interacting with some environment. The environment is described by a set of *states*. At each successive time step, the agent observes the state of the environment and may select from a number of *actions* which causes the environment to transition to some new state. After performing the action, the agent receives a *reward* which is a signal giving the agent some indication of the value of taking that action in that state. The agent's goal is to find a *policy* which guides its action selection such that it maximizes the total reward received over some time horizon (Sutton & Barto, 1998). The key idea is that the reward function need not encode every detail of the problem, it may instead give only a very general indication of the outcome. For example, in a robot navigation application, the agent may receive a positive reward when it reaches a specific goal, otherwise it may receive a zero reward for any intermediate move. The agent must therefore discover which actions will guide it to the positive reward in the smallest number of time steps.

The field of reinforcement learning has grown to encompass many algorithms and techniques which have been successfully applied to many problems traditionally arising in artificial intelligence and machine learning, including adversarial game play (Tesauro, 1995; Gelly & Silver, 2007), elevator scheduling (Crites & Barto, 1998), robot motion planning (Boada et al., 2002; Peters & Schaal, 2008), and many other physical or virtual systems (Sutton, 1996; Santamaria et al., 1997; Millán et al., 2002; Ernst et al., 2005b). There is now growing interest in applying RL methods to any number of medical treatment strategies (Ernst et al., 2006; Pineau et al., 2007; Pineau et al., 2009; Shortreed et al., 2011; Bothe et al., 2013). These studies have established that, within certain domains, the RL approach has promise. Many of these studies have necessarily looked at problems with either very short sequences of decisions, have been limited in the amount of data that they can utilize, or have lacked detailed explorations of the possible effects of parameter

or algorithmic variations. In this thesis we attempt to further this line of research and address some of these problems. We investigate several specific domains from a reinforcement learning perspective, applying RL algorithms to computational models which can assist the investigation of appropriate algorithms for each domain, and demonstrate results which suggest future areas for exploration and refinement. The specific disease types we have studied share a number of features, but differ in several important respects.

In medical domains, one of the most important features of reinforcement learning methods are their ability to overcome the lack of detailed, deterministic models of disease by "learning" a decision-making strategy directly from data. However, in so doing, many RL methods will require more, rather than less, data to support this optimization. Whether these data are derived from a clinical trial or a lab experiment performed *in vitro*[1] , there are often substantial costs in both time and money required to collect the data.

Therefore, in this thesis we argue for the utility of computational modeling of medical domains, and a major component of this thesis is the development and elaboration of specific computer models of three common medical scenarios: electrical stimulation for the control of both epilepsy and Parkinson's disease, and fractionated radiation therapy for the control of cancer tumors. The use of computational (or *in silico*) models of disease is not new, but recent years have brought increased interest in the application of such models, in part to assist with the development of new therapies, including adaptive control strategies in disease treatment (Adams et al., 2004; Ernst et al., 2006; Kovatchev et al., 2009; Valerio Jr., 2009; Kovatchev

[1] The term *in vitro* refers to experiments using biological material somehow extracted from a living entity, as opposed to *in vivo* experiments, which are performed on intact living entities.

et al., 2012). Within the RL framework, such models can be useful either as sources of inexpensive data for training and validation, or possibly as "black box" generative models necessary for certain RL solution methods.

1.2 Contributions

The specific contributions of this thesis include the following:

- **A general framework for casting sequential treatment design problems into the reinforcement learning paradigm.** We have outlined many of the major issues encountered in casting a medical problem as an RL environment, and provided guidance for navigating these decisions. Chapter 3 is intended to assist the development of similar approaches by researchers who may not be experts in computer science, machine learning, or reinforcement learning.
- **One of the first computational models of epilepsy to give an account of both seizure initiation and termination with realistic timing and response to external stimulation.** We develop and demonstrate this model with the specific goal of providing what could become an *in silico* model for the evaluation of adaptive control strategies for epilepsy.
- **A demonstration of the successful application of reinforcement learning to control a computational model of neural synchronization inspired by Parkinson's disease.** We give an implementation of an existing model as a reinforcement learning domain, and demonstrate that an RL agent can be trained to control the model.
- **A formulation of the scheduling of fractionation in radiation therapy as an RL problem.** We develop an extension to an existing computational model of the response of cell populations to irradiation and show preliminary results suggesting that such scheduling may produce improved results under some conditions.

1.3 Overview

The rest of this dissertation is organized as follows: Chapter 2 describes the technical background of Markov decision processes and reinforcement learning, describing both the mathematical theory and some specific methods and algorithms. Chapter 3 presents a general framework discussing the shared features and important design decisions which characterize sequential medical treatments such as the three specific case studies described in this thesis. Chapter 4 describes our first case study on the modeling and control of epilepsy, with special emphasis on the development of a computational model which mimics the behavior of existing *in vitro* models. Chapter 5 describes the application of reinforcement learning approaches to the case of a computational model of Parkinson's disease. Chapter 6 details our efforts to apply the framework to optimization of fractionation scheduling in adaptive radiation therapy. Finally, Chapter 7 gives some concluding thoughts and ideas for future research.

1.4 Publication status

Chapter 4 is based closely on a previously published paper, Vincent et al. (2011). The modeling research described was designed and implemented by the first author, who also wrote the bulk of the manuscript. The other authors, Drs. Courville and Pineau, guided the research and assisted with the manuscript. Drs. Massimo Avoli and Gabriella Panuccio provided the data and methods for the *in vitro* model. Chapters 5 and 6 are previously unpublished, but manuscripts on this research are in preparation for publication. As mentioned elsewhere, the experimental data referenced in Chapter 6 was provided by the lab of Dr. Issam El Naqa.

CHAPTER 2
Technical background

This chapter gives an overview of the general reinforcement learning framework, along with some specific challenges posed by medical treatment design problems. In particular, because many medical problems involve continuous, high-dimensional data, we discuss the computational problems encountered in these situations. We also provide the details of the algorithmic methods we used in approaching the specific case studies detailed in this thesis.

The reinforcement learning problem is a category of sequential decision making problem in which we attempt to train an *agent* to learn some optimal method for taking actions in a given *environment* (also called a *domain*). Each action taken by the agent is assumed to have a (possibly stochastic) effect on the environment. The agent typically defines optimality using some reward signal received for performing certain actions and/or reaching certain states.

2.1 Markov decision processes

A key property of many of these problems is the notion of the Markov property, in which the state of the environment at time t is a sufficient statistic to define the distribution of possible outcomes after taking an action in that state. In other words, the state of the system must be defined such that it fully describes the history of the environment up to the present, for example, by incorporating any crucial aspects of the patient's medical history into the current state.

Many discrete-time sequential decision-making problems can be formalized as a Markov decision process (MDP) (Bellman, 1957; Puterman, 2005). An MDP consists of four key components: a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, a transition function $T(s, a, s')$, and a reward function $R(s, a)$. On performing an action $a \in \mathcal{A}$

in state s at time step t, the agent receives a scalar reward $r = R(s, a)$ and the environment moves to a new state s' according to the transition function $T(s, a, s')$, mapping the current state and action onto a probability distribution over successor states in $\mathcal{S}$. Stochastic behavior in the environment is generally modeled primarily using the transition function. The state is assumed to be fully observable, that is, there is no noise or uncertainty in the state variables (although this assumption is often relaxed in practice). The reward may be modeled as either a deterministic function or a stochastic function.

Depending on the problem, the MDP may also specify a unique start state s_0 and a discount factor $\gamma \in (0, 1]$, which can be thought of as the agent's probability of living to see the next time step (Kaelbling et al., 1996). For $T = \infty$, γ must be less than one to preclude an infinite reward. For finite T it is acceptable for $\gamma = 1$.

Many of the MDPs used in traditional control problems are *finite* problems, where both $\mathcal{S}$ and $\mathcal{A}$ are finite sets. However, methods exist to extend MDPs over problems with continuous or countably infinite state or action sets.

The agent's goal is to find a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ that is a mapping from state to action such as to maximize the expected total reward over some time horizon:

$$R_T = E\left[\sum_{t=0}^{T} \gamma^t r_t\right]. \tag{2.1}$$

Given this structure and notation, one can write the value of any state s if the agent follows a fixed policy π as an expectation over the expected sequence of rewards:

$$V^{\pi}(s) = E_{\pi}\left[\sum_{t=0}^{T} \gamma^t r_t\right] \tag{2.2}$$

and the optimal value for a state (optimality is commonly denoted using a superscript $*$) is then:

$$V^{*}(s) = \max_{\pi} E_{\pi}\left[\sum_{t=0}^{T} \gamma^t r_t\right], \tag{2.3}$$

which can be expanded to the recursive equation:

$$V^*(s) = \max_{a \in \mathcal{A}} \left(R(s,a) + \gamma \sum_{s' \in \mathcal{S}} T(s,a,s') V^*(s') \right). \tag{2.4}$$

Therefore the value of a state is the maximum of the reward possible in this state plus the expected value of the successor states. The optimal policy $\pi^*(s)$ is then:

$$\pi^*(s) = \operatorname*{argmax}_{a \in \mathcal{A}} \left(R(s,a) + \gamma \sum_{s' \in \mathcal{S}} T(s,a,s') V^*(s') \right). \tag{2.5}$$

From Equation 2.5, it follows that if the transition and reward functions of the MDP are unknown, it is not possible to derive the optimal policy directly from the value function $V^*(s)$. To relax this limitation it is common to express the value using a state-action pair, which is formulated as:

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s' \in \mathcal{S}} T(s,a,s') \max_{a' \in \mathcal{A}} Q^*(s',a'). \tag{2.6}$$

.

Given an estimate of the Q-function, which can be learned from empirical data using a variety of methods, the optimal value function can be expressed as:

$$V^*(s) = \max_a Q^*(s,a) \tag{2.7}$$

and optimal (deterministic) policy:

$$\pi^*(s) = \operatorname*{argmax}_a Q^*(s,a) \tag{2.8}$$

without requiring knowledge of the reward and state transition functions (Sutton & Barto, 1998). Of course, depending on the details of the Q-function representation, finding the maximum with respect to the action may be difficult.

2.2 Dynamic programming solutions

If the state transition probabilities T and reward function R are known, and if the state and action sets $\mathcal{S}$ and $\mathcal{A}$ are finite, dynamic programming methods such

as value iteration can calculate an exact solution for the optimal value function $V^*(s)$ (Bellman, 1957). The two most basic approaches are commonly called *value iteration* and *policy iteration*. In their most basic form, these algorithms calculate near-exact solutions by iterating over the entire state and action space of the MDP. However, the total number of states $|\mathcal{S}|$ will grow exponentially with the number of dimensions in the state space, an observation which is often called the "curse of dimensionality". If the state space contains many dimensions, or some continuous dimensions, it can rapidly become impractical to evaluate the entire state space. In these cases, it may be possible to use other methods to find an approximate solution.

2.2.1 Value iteration

Value iteration is one of the most basic dynamic programming methods for solving finite MDPs (Sutton & Barto, 1998). Given a fully-specified MDP, we can apply Algorithm 2.1. Depending on the tolerance for error in the application, the algorithm can be iterated until some convergence criteria is satisfied, for example:

$$\max_{s \in \mathcal{S}} |\hat{V}_k(s) - \hat{V}_{k+1}(s)| < \epsilon \tag{2.9}$$

for an error tolerance $\epsilon > 0$. Throughout this chapter we use the notation $\hat{V}(s)$ and $\hat{Q}(s, a)$ to denote empirical estimates of the state value and state-action value function, respectively.

Once the approximately optimal value function has been found, one can derive the implied policy by applying Equation 2.5. Value iteration is quite flexible. It is possible to implement the state visits in any order that is convenient (Kaelbling et al., 1996). However, it is not immediately clear what, if anything, is a good choice of stopping criteria. Theoretical results show that, for a finite MDP, if Equation 2.9 is satisfied, then the difference between the optimal value function and the current estimate will obey:

$$\max_{s \in \mathcal{S}} |V^*(s) - \hat{V}_{k+1}(s)| < \frac{2\epsilon\gamma}{1-\gamma} \tag{2.10}$$

Require: A fully-specified, finite MDP: $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$.
Require: Some convergence criteria.
Initialize $\hat{V}_0$ arbitrarily.
repeat
for all $s \in \mathcal{S}$ **do**

$$\hat{V}_{k+1}(s) \leftarrow \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} T(s, a, s') \left[R(s, a) + \gamma \hat{V}_k(s') \right]$$

end for
until The convergence criteria is satisfied.

ALGORITHM 2.1: The standard value iteration algorithm. This is probably the simplest solution method for a fully-specified, finite MDP.

(Williams & Baird, 1993). In practice, it is often the case that the optimal policy is discovered well before the value function converges (Kaelbling et al., 1996).

2.2.2 Policy iteration

As an alternative to value iteration, *policy iteration*, as detailed in Algorithm 2.2, is a method which alternates two distinct phases of *policy evaluation* and *policy improvement*. It often converges in a surprisingly small number of iterations (Sutton & Barto, 1998).

One advantage of policy iteration over value iteration is the existence of a single clear stopping criteria for the overall algorithm - when the policy stops changing, the algorithm has converged. In addition, the policy evaluation step can be implemented as the solution of a set of linear equations (Kaelbling et al., 1996). However, these advantages come at the cost of some increase in the difficulty of implementation of each algorithm stage relative to value iteration. However, for a finite MDP there are at most $|\mathcal{A}|^{|\mathcal{S}|}$ possible policies, and the policy is guaranteed to improve at each iteration prior to convergence, so the algorithm is guaranteed to converge in a finite number of iterations and in most cases will converge in fewer iterations than value iteration, at the cost of greater complexity per iteration (Boutilier et al., 1999; Puterman, 2005).

Require: A fully-specified finite MDP: $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$.
Require: Some convergence criteria.
  Initialize V_0 and π_0 arbitrarily.
  repeat
    // Policy evaluation
    repeat
      for all $s \in \mathcal{S}$ **do**

$$V_{k+1}(s) \leftarrow \sum_{s' \in \mathcal{S}} T(s, \pi_k(s), s') \left[R(s, \pi_k(s)) + \gamma V_k(s') \right]$$

      end for
    until The convergence criteria is satisfied.
    // Policy improvement
    for all $s \in \mathcal{S}$ **do**

$$\pi_{k+1}(s) \leftarrow \operatorname*{argmax}_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} T(s, a, s') \left[R(s, a) + \gamma V_{k+1}(s') \right]$$

    end for
  until $\pi_{k+1}(s) = \pi_k, \forall s \in \mathcal{S}$.

Algorithm 2.2: The policy iteration algorithm.

2.2.3 Handling large MDPs

For MDPs with large finite state and/or action spaces, it may be possible to exploit the structure of the problem or use other methods to achieve approximate results.

On possible approach to handling large MDPs is *factoring* the state space. These methods exploit regularities in the structure of the MDP to restructure the state space and reduce the effective size of the MDP (Guestrin et al., 2003; Boutilier et al., 2000; Hoey et al., 1999; Boutilier et al., 1999). Factoring methods may lead to more efficient, if approximate, solutions for a wide variety of real-world problems having many millions of states (Hoey et al., 1999).

Alternative methods may find partial or approximate solutions for large MDPs using heuristic or sampling search methods (Bonet & Geffner, 2006). If the MDP

is not fully specified, a "black box" simulator may be substituted in some sampling search algorithms (Kearns & Singh, 2002).

2.3 Monte Carlo solutions

If the state transition dynamics T or the reward function R are unknown, but we have access to a large enough number of state and action trajectories derived either from the system under study or a generative model, then Monte Carlo methods may be used to estimate the value function (Sutton & Barto, 1998). Monte Carlo methods do not require a complete specification of the MDP in order to find a solution. Instead, these methods can compute optimal policies from observations of the dynamics of a system (Sutton & Barto, 1998). These methods require that we have access to a large (perhaps infinite) set of "trajectories", or series of state, action, and reward observations drawn from one or more policies executing within the environment. These methods are especially designed to exploit a "black box" implementation of the environment, in which we may not have direct knowledge of the reward function or state transition dynamics, but we do have the ability to query a generative model to determine the successor state for an arbitrary choice of initial state and action. Alternatively, it may be possible to derive a policy from a sufficiently large set of data previously drawn from empirical observations of an agent acting on the environment.

In the last several years there have been a number of very successful applications of methods that implement variants of Monte Carlo tree search. In these methods, the agent samples several trajectories originating at the current state with the aid of a generative model of the domain (Kocsis & Szepesvari, 2006; Silver & Veness, 2010). These methods have been very successful in several domains, especially some game-playing applications (Gelly & Silver, 2007).

As discussed previously, if we do not have direct access to the reward function R or the state transition function T, it is often useful to represent the value of state-action pairs using the Q-function rather than the value function. In this case, we can estimate the value of a particular state-action pair by simply taking the average of all returns from trajectories originating in that state-action pair. This means that we can perform policy evaluation by applying the equation:

$$Q^{\pi}(s,a) = \frac{\sum_{k=1}^{|R|} R_k(s,a)}{|R|} \tag{2.11}$$

where R is a vector of returns calculated from trajectories originating in state $s_t = s$ and $a_t = a$.

Given an estimate for Q^{π}, we can perform policy improvement by applying Equation 2.8. As in policy iteration, we terminate the process when the estimated policy stops changing.

2.4 Reinforcement learning

Many algorithms have been developed for solving MDPs using empirical observations of state, action, and reward. These are collectively known as *reinforcement learning* (RL) methods. Most of these methods are intended to "learn" from data, in which the agent has no prior knowledge of either the state transition function T or the reward function R. Some RL methods, which are termed *model-free* methods, make no attempt to explicitly represent these functions, while other methods do try to approximate these components of the MDP.

Many RL methods can be classified as temporal difference (TD) methods. Temporal difference methods share some characteristics of both dynamic programming and Monte Carlo methods. Like dynamic programming, they are bootstrap methods that iteratively refine an estimate of the V or Q function. Like Monte Carlo methods, they do not require an explicit model of the environment, but can optimize a policy entirely from empirical data. The most basic TD method is the

simple one-step TD update, also known as TD(0), which can be used to approximate the value function of an MDP from a series of trials by iteratively applying the update rule:

$$\hat{V}(s) \leftarrow \hat{V}(s) + \alpha \left[r + \gamma \hat{V}(s') - \hat{V}(s) \right], \tag{2.12}$$

where α is a "learning rate" parameter. In practice, the learning rate parameter is often a constant chosen such that $\alpha \in (0, 1)$. However, in many conditions, convergence will be improved if the algorithm gradually reduces the value of α during learning, i.e. to define $\alpha(t)$ such that both

$$\sum_{t=0}^{\infty} \alpha(t) = \infty$$

and

$$\sum_{t=0}^{\infty} \alpha(t)^2 < \infty$$

apply. These are sometimes called the Robbins-Monro conditions (Robbins & Monro, 1951).

A number of simple modifications to this update rule (Equation 2.12) have yielded practical algorithms for planning and control in MDPs. Examples of these include Q-learning (Watkins & Dayan, 1992), which is an *off-policy* algorithm, in that the agent is free to follow any *behavior policy* while learning the optimal *target policy*, and Sarsa (Rummery & Niranjan, 1994), which is an *on-policy* algorithm, in that the behavior and target policies are identical at each time step (Sutton & Barto, 1998).

Q-learning proceeds as follows. From an initial state s, the agent chooses an action a according to any policy, then observes the reward r and successor state s'. The agent then performs the following update:

$$\hat{Q}(s, a) \leftarrow \hat{Q}(s, a) + \alpha \left[r + \gamma \max_{a' \in \mathcal{A}} \hat{Q}(s', a') - \hat{Q}(s, a) \right], \tag{2.13}$$

where α is a "learning rate" parameter.

After the update, the agent starts the process over from the new state.

For Sarsa[1] , the agent simply visits a series of states, choosing the next action using a policy derived from Q. After recording the current state s and action a, the agent observes the reward r and successor state s' and chooses the subsequent action a'. Then the agent performs the update:

$$\hat{Q}(s,a) \leftarrow \hat{Q}(s,a) + \alpha\left[r + \gamma\hat{Q}(s',a') - \hat{Q}(s,a)\right]. \tag{2.14}$$

The key difference is the use of the max operator in the Q-learning case. This allows the agent to improve the estimate of the Q-function even when the policy is sub-optimal. However, this comes at an increase in complexity for each time step, if, for example, the action space is large and therefore the max operator may be computationally expensive.

A distinct characteristic of TD methods is that they can refine their value function estimates *on-line*, that is, they can actually "learn by doing" by making choices in a real environment, whereas dynamic programming and Monte Carlo methods typically learn *off-line* using either a full model or previously collected data.

2.4.1 Exploitation and exploration

A fundamental issue in most RL applications is the need to trade off between *exploration*, the need to thoroughly sample the state-action space, and *exploitation*, the agent's imperative to maximize expected reward given the value function it has learned so far. Most RL agents approach this by implementing a stochastic policy which gives extra weight to the best known action. The most simple of these is commonly called an ε-*greedy* policy (Sutton & Barto, 1998). In this case, the agent

[1] The name is derived from the conventional symbols for the values used in the update: $\langle s, a, r, s', a' \rangle$ (Sutton & Barto, 1998).

chooses the action associated with the highest Q-value with probability $1 - \varepsilon$, alternatively choosing an action uniformly randomly with probability ε, for some small ε.

An alternative stochastic policy is the *softmax* policy which chooses actions according to a modified Boltzmann distribution:

$$\pi(s) = \frac{e^{\frac{Q(s,a)}{\tau}}}{\sum_{a \in \mathcal{A}} e^{\frac{Q(s,a)}{\tau}}}, \tag{2.15}$$

where τ is a positive real number referred to as the *temperature* (Sutton & Barto, 1998). Large values of τ cause the policy to approximate a uniformly random policy, whereas as τ approaches zero the policy approaches the simple greedy policy.

In these and other cases, the policy may be annealed over time to decrease the amount of exploration by reducing the value of ε for ε-greedy policies or reducing the temperature τ for softmax policies. Reducing the amount of exploration may improve the empirical return in stationary systems, but this comes at the cost of reducing the agent's ability to adapt to subsequent changes in the environment.

Exploration and exploitation can be more explicitly modeled using a number of more recent algorithms. These include algorithms for which rigorous proofs exist that show they will compute near-optimal control policies after a polynomial number of time steps, with high probability. Such algorithms are commonly referred to as *probably approximately correct* (PAC). For example, the R-MAX (Brafman & Tennenholtz, 2003) and E^3 algorithms (for Explicit Explore and Exploit) (Kearns & Singh, 1998) rely on initially optimistic estimates of the value of unknown or rarely-visited states in order to promote exploration of those states. Another mechanism, Model-Based Interval Estimation (Wiering & Schmidhuber, 1998; Strehl & Littman, 2005) selects actions using estimates of the upper confidence bounds of state values. All three of these algorithms rely on creating explicit models of the environment using the observed statistics of state transition frequencies.

Another relatively recent algorithm is UCT (Kocsis & Szepesvari, 2006), in which the problem of action selection in a particular state is modeled as an n-armed bandit problem, combined with a fairly conventional Monte Carlo tree search approach to look ahead and evaluate potential future states. While it does not construct a model of the MDP, the UCT algorithm relies on a "black box" implementation of the domain, which is used to explore the action choices as the agent considers possible future sequences of states, actions, and rewards. These sequences are used to generate estimates of the relative action values in the current state of the real controlled environment. UCT has been successfully applied to difficult problems such as computer Go (Gelly & Silver, 2007).

2.4.2 Delayed reward

The reward for a series of actions may be *delayed*, in that the agent may receive a non-zero reward well after some important prior action has been taken. Because of the relative ease of design and implementation, it is fairly common practice to define a non-zero reward function only in terminal states, such as at the end of a game or achievement of some concrete goal. This gives rise to another fundamental issue in TD methods, the *credit assignment* problem. The agent may require a great deal of experience before it discovers the relative importance of certain intermediate states and actions that might nevertheless be critical to reaching successful terminal states. In these cases, the agent may speed learning by assigning partial credit for rewards received in the present state to states encountered earlier in the episode.

The TD update rules for TD(0), Q-learning, and Sarsa all rely on the observation that we can write the discounted return for a state as an expansion of Equation 2.1, such that the R_t, the discounted return beginning at state s_t, is:

$$R_t = \sum_{k=1}^{T} \gamma^{k-1} r_{t+k} = r_{t+1} + \gamma V(s_{t+1}). \tag{2.16}$$

There is no reason we cannot consider longer expansions of this equation:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 V(s_{t+2}), \tag{2.17}$$

and so on. In practice, such intermediate calculations can improve the convergence rate of TD methods, especially when the reward signal is only assessed in some (e.g. terminal) states (Singh & Sutton, 1996; Sutton & Barto, 1998).

One common method for approximating this calculation is the use of *eligibility traces* (Klopf, 1972), which maintain a scalar activation $e(s, a)$ over all states (or state features) (Sutton & Barto, 1998). The activation typically decays exponentially over time according to a parameter λ, and is increased or incremented whenever the state is visited. While there are several variants, one common update rule for $e(s, a)$ is:

$$e(s,a) \leftarrow \begin{cases} 1 & \text{if } s = s_t \text{ and } a = a_t \\ \lambda\gamma e(s,a) & \text{otherwise.} \end{cases} \tag{2.18}$$

The update rule for Sarsa with eligibility traces (often called Sarsa(λ)) then becomes:

$$\hat{Q}(s,a) \leftarrow \hat{Q}(s,a) + \alpha \left[r + \gamma \hat{Q}(s',a') - \hat{Q}(s,a) \right] e(s,a). \tag{2.19}$$

2.4.3 Continuous and infinite spaces

The reinforcement learning methods we have discussed this far apply primarily to discrete, finite state and action spaces. In these domains it is possible, at least in theory, for the agent to exhaustively sample the state and action space. In an environment with either countably infinite or continuous states and/or actions, it is no longer possible to sample all possible state-action pairs. Instead we must rely on a function approximation method to represent either a mapping from states to actions ($\pi : \mathcal{S} \rightarrow \mathcal{A}$) or states and actions to values ($Q : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$), which must

somehow enable the agent to generalize from prior experience to newly encountered states.

The problem of continuous state spaces has received considerable attention, with many methods showing promise in some applications (Santamaria et al., 1997; Smart & Kaelbling, 2000; Ormoneit & Sen, 2002; Lagoudakis & Parr, 2003; Engel et al., 2005; Ernst et al., 2005a; Peters & Schaal, 2008). Less well developed, however, are approaches for handling environments with continuous actions, although several interesting approaches have been proposed (Smart & Kaelbling, 2000; Millán et al., 2002; Lazaric et al., 2007; Busoniu, 2009; Pazis & Lagoudakis, 2009; Mansley et al., 2011). However, most RL methods for continuous spaces do not offer the same theoretical guarantees of convergence and optimality that are possible with discrete spaces.

In general, function approximation methods seek to minimize some error metric, such as the mean-squared error (MSE) of the approximate value function, $\hat{V}$, over the expected distribution of states, with respect to the "true" value function, V^π:

$$MSE = \sum_{s \in \mathcal{S}} P(s) \left[V^\pi(s) - \hat{V}(s) \right]^2 . \tag{2.20}$$

Many general forms of function approximation have been applied to RL. Nearly any regression method can be used to learn some mapping from state features to the value or Q-function. Some common examples are as follows:

- Coarse coding (i.e. linear combinations of discretized state features) (Sutton, 1996; Santamaria et al., 1997; Smith, 2002; Martín H. & de Lope, 2009).
- Regression trees (Ernst et al., 2005a).
- Nearest-neighbor regression (Smart & Kaelbling, 2000).
- Other parameterized functions (Baird & Klopf, 1993).

We will give more details of some examples of these methods in Section 2.5.

A common feature of many of these methods is the desirability of some smoothness constraint on the Q-function. That is, these methods tend to assume that the distance between Q-values should be small for states and/or actions that are neighbors in the state space. Many therefore have difficulties handling spaces with discontinuities. This also may have implications for learning policies with the need to rapidly switch among different therapies; it can be difficult to represent such fast switching behavior with insufficiently rich function approximators (Ernst et al., 2006).

2.4.4 Actor-critic methods

Actor-critic methods are family of methods which are of special interest in the case of continuous state and action spaces (Sutton & Barto, 1998). In these methods, an *actor*, which is a representation of the current policy, is maintained separately from the *critic*, which is the current value function estimate. In this sense they are similar in conceptual structure to general policy iteration, Algorithm 2.2. The critic guides the improvement of the policy. While actor-critic methods are constrained to on-policy learning, they also have the advantage that they can learn effectively in domains that require a stochastic policy, which may be especially useful in multi-agent settings, for example (Sutton & Barto, 1998).

The output of the critic is often the "temporal difference" error:

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t), \tag{2.21}$$

while the actor implements a parameterized policy, for example the stochastic policy:

$$\pi_t(s, a, \theta) = Pr\{a_t = a | s_t = s, \theta\} = \frac{e^{\theta(s,a)}}{\sum_{b \in \mathcal{A}} e^{\theta(s,b)}}, \tag{2.22}$$

where the parameter vector $\theta(s, a)$ are weights assigned to each state-action pair. An update rule for this scheme might then be:

$$\theta(s_t, a_t) \leftarrow \theta(s_t, a_t) + \beta\delta_t, \tag{2.23}$$

where β is a positive learning rate (Sutton & Barto, 1998).

Given this separation of the two functions, it is often possible to represent action selection using some smoothly parameterized mapping from the state to the action spaces. This property makes these methods especially interesting in continuous domains. If the actor implements a policy $\pi(a, s, \theta) = Pr\{a = a_t | s = s_t, \theta\}$, and the policy is differentiable with respect to the parameter vector θ, we can follow the gradient in terms of the expected future reward, $J(\theta)$, $\Delta\theta = \beta\nabla_\theta J(\theta)$ where β is a learning rate parameter. With appropriate annealing of the learning rate, these methods often can be proved to converge to a local maximum in the policy space (Amari, 1998). In contrast, the policy derived from the state-action value function Q, as in Equation 2.8, can be sensitive to arbitrarily small changes in the current estimate of Q (Peters & Schaal, 2008).

2.4.5 Batch vs. on-line learning

We have placed a great deal of emphasis on RL methods that learn on-line, in which the agent observes the state, chooses an action, observes the reward and successor state, and then immediately updates its behavior policy incrementally after each time step (Sutton & Barto, 1998). In many common formulations, this has several interesting advantages. Incorporating a single new state transition and reward observation is usually computationally cheap. In some cases, an agent may be able to adapt to a non-stationary environment, because it is continually refining its Q-function estimate. Most importantly, the agent can learn from a real environment, for which no prior observations or black-box simulator may be available. However,

this approach has some drawbacks. In practice, the agent will require many observations to learn a good policy, especially if we have no way to incorporate some prior knowledge of the environment. This is not generally possible or desirable in a clinical setting, where the efficacy of a therapy has to be characterized well before it is actually used on a patient.

In many medical domains, it is not possible to train an agent entirely on-line. Commonly, data may be collected in a fixed series of experimental trials and the potentially disruptive effects of a randomized, untrained policy may impose an unacceptable burden on the patient.

In these cases it may be preferable to apply a *batch* or *off-line* reinforcement learning method, in which the agent is trained using a series of previously recorded trajectories containing state, action, and reward information.

Thankfully a set of alternative methods exist which can help address this need. There are a number of algorithms for *batch* reinforcement learning in which the agent learns from an entire series of prior experience data, typically taking the form of several complete histories of observations taken in the target environment. In some cases, it may be possible to use data collected under a policy that is not in any way optimal. It may even be feasible to use a completely random policy, as long as the coverage of states and actions is good enough.

These methods often resemble the Monte Carlo approach described in Section 2.3. However, a number of interesting refinements are possible.

One approach that is useful in batch settings is Least Squares Policy Iteration, or LSPI (Lagoudakis & Parr, 2003). LSPI operates by estimating the state-action value function Q using samples generated under an arbitrary policy. The algorithm mimics the general policy iteration algorithm (Algorithm 2.2), but substitutes a

function approximation approach to either the policy representation or the state-action value function. This allows LSPI to be used in continuous domains, or in domains which are simply too large to be solved with standard policy iteration.

Another related method is kernel-based reinforcement learning (Ormoneit & Sen, 2002), in which the Q-function is approximated using an iterative series of regressions using a kernel function. This approach has been applied to medical domains, in the context of the optimization of drug treatment for depression (Pineau et al., 2007).

2.4.6 Instance-based methods

While we have not applied them in this thesis, instance-based methods are an interesting option to consider for real-world medical applications of RL. Such methods attempt to learn a function by memorizing a set of data points, which are stored unmodified and used to guide future predictions. Examples include k-nearest-neighbor methods, optionally including locally-weighted regression (Moore & Atkeson, 1993; Atkeson et al., 1997; Smart & Kaelbling, 2000). These methods tend to suffer from fairly high costs in both memory and processing time. For example, searching a kd-tree for the nearest neighbor in an d-dimensional space of n examples has complexity $O(d \log n)$ (Bentley, 1975).

One interesting method is the HEDGER algorithm (Smart & Kaelbling, 2000), which uses locally-weighted regression (LWR) within a constrained hull of memorized points to approximate the state space of a continuous state-action domain. This method has a number of properties that might make it appropriate for medical domains. In particular, its use of the hull calculation provides a clear delineation between cases where the algorithm is acting on good information versus where it is just guessing.

2.4.7 Partial observability

So far we have assumed that the agent is able to perceive the complete state of the system in all cases. In practice, many real-world problems are better characterized by incomplete or partial knowledge, in that the agent may have some inexact estimate of state or receive only some stochastic signal which depends on the true state.

Problems with this character are commonly referred to as *partially observable* Markov decision processes, or POMDPs (Sondik, 1971). These can be formalized by extending the MDP concept to include a set of observations and some stochastic distribution over states, actions, and observations. The formal definition of a POMDP therefore consists of six components (Kaelbling et al., 1998):

- The elements $\mathcal{S}$, $\mathcal{A}$, T, and R which define a standard MDP (Section 2.1).
- The set Ω of possible *observations* the agent can perceive.
- A probability distribution $O(s, a, o)$ which defines the probability of making observation $o \in \Omega$ when selecting action $a \in \mathcal{A}$ in a given state $s \in \mathcal{S}$

Most solution methods require that the agent maintain a *belief state* which is a distribution over all possible states, and therefore has dimension $|\mathcal{S}| - 1$ (Sondik, 1971). This belief state is a sufficient statistic to define the probabilities of subsequent observations and rewards. The POMDP and its resulting belief state therefore can be thought of as defining a new Markov process where the state is the continuous belief state and the reward and state transition functions are now defined in terms of both the belief state and observation functions (Kaelbling et al., 1998).

Because of this clear increase in complexity, planning in a partially observable domain is much more difficult than in the equivalent fully observable MDP (Sutton & Barto, 1998). However, a number of approximate methods have been developed (Ng & Jordan, 2000; Pineau et al., 2003; Silver & Veness, 2010; Ross et al.,

2011), some of which can support efficient planning in POMDPs of practically useful sizes.

2.5 Algorithmic approaches

In this section we discuss in greater detail some algorithms which may be useful for adaptive treatment problems such as those we consider in this thesis. This represents only the handful of algorithms we have been able to implement and test for this thesis, as well as suggestions of another family of algorithms that may be useful in similar research.

2.5.1 Sarsa(λ) with tile coding

Sarsa(λ) with tile coding is one of the first effective methods discovered for representing arbitrary reinforcement learning problems with continuous state and action spaces (Rummery & Niranjan, 1994; Sutton, 1996). Tile coding is a form of *coarse-coding*, function approximation methods that model function as linearly weighted combination of basis functions, often with either binary or Gaussian activation. These methods involve imposing one or more overlapping grids of basis functions over the state and action space, and evaluating a query point by performing a weighted summation over the basis functions in the neighborhood of the query point. The approximator may be trained by adjusting the weights and, optionally, the distribution of the centers of the basis functions themselves.

Tile coding (Sutton, 1996; Santamaria et al., 1997) is derived from the CMAC model of Albus (1971). In this method, the state and possibly the action space are overlaid with a set of several overlapping "tilings". Each tiling corresponds to a particular discretization of the space, and a given point in the continuous space is said to "activate" one tile in each tiling. This is illustrated in Figure 2–1. The Q-function is calculated from the set of N binary basis functions:

$$Q(s,a) = \sum_{i=1}^{N} w_j \phi_j. \tag{2.24}$$

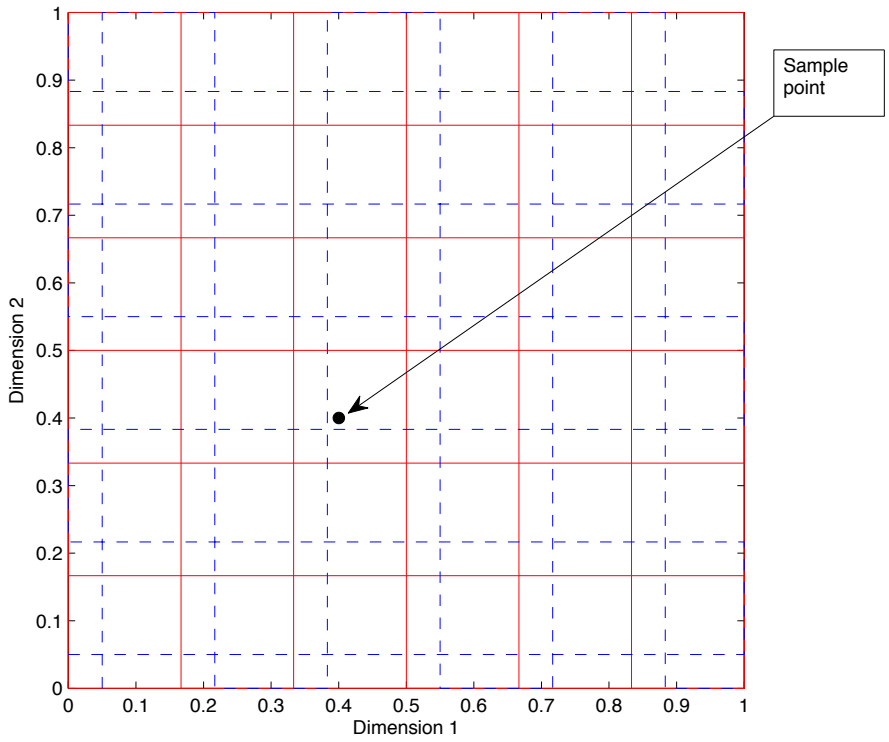

FIGURE 2–1: Illustration of tile coding with two overlapping tilings in two dimensions. One tiling is illustrated using solid lines, a second using dashed lines. The sample point will activate exactly two features, one from each tiling.

An element or "tile" ϕ_j is 1 when the state value is contained within its "receptive field" and zero otherwise. The weights w_j can be learned using a temporal difference algorithm that is a simple extension of Q-learning or Sarsa. For example, Sarsa uses the following rule to update the estimated Q-function at each time step t:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha\delta_t, \tag{2.25}$$

where α is a learning rate and:

$$\delta_t = r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t). \tag{2.26}$$

In the case of Sarsa(λ), this rule is extended by the addition of eligibility traces (Sutton & Barto, 1998):

$$Q_{t+1}(s, a) = Q_t(s, a) + \alpha\delta_t e_t(s, a), \forall s, a. \tag{2.27}$$

Note that when the algorithm is extended to include eligibility traces, this update must be performed for all state and action pairs, or at least those with a non-zero eligibility trace $e_t(s, a)$.

The eligibility traces themselves are updated at each time step according to one of several possible rules. In all of our experiments we use replacing traces (Singh & Sutton, 1996; Sutton & Barto, 1998):

$$e(s, a) = \begin{cases} 1 & \text{if } s = s_t \text{ and } a = a_t \\ 0 & \text{if } s = s_t \text{ and } a \neq a_t \\ \lambda\gamma e(s, a) & \text{otherwise.} \end{cases} \tag{2.28}$$

For the extension to tile coding, we define a set of binary state features ϕ_j with some mapping from a state s to a group of ϕ_j features. This mapping is commonly achieved by discretizing the state variables, and then mapping the discretized values

onto the set of features using a deterministic hashing function. Hashing in this manner can facilitate a significant reduction in memory usage, especially in problems with many dimensions (Sutton & Barto, 1998). The update rules are then modified to associate the eligibility traces with the individual features rather than states or actions, and to apply the update rule the feature weights w_j:

$$w_j = w_j + \frac{\alpha}{N}\delta_t, \forall j \in h(s_t), \tag{2.29}$$

where the function $h(s)$ is a hashing function which returns a set of N feature indices according to a deterministic function of the state variables and tiling parameters. While in theory the size of the set N could take on arbitrary values, and need not reflect the dimensionality of the state space, in practice some commonly implemented methods constrain N in some way, for example, restricting it to the powers of two.

One potential advantage of coarse-coding methods in general is their ability to incorporate a priori knowledge of the state space, by, for example, using a non-uniform discretization to concentrate tiles in areas or along dimensions where higher precision is desired (Sutton & Barto, 1998).

While these methods can be effective in some cases, they are still affected by the "curse of dimensionality" in that the number of basis functions needed will tend to increase exponentially with the number of state or action dimensions. This increases the cost of the approach in both storage and computation speed. While methods such as hashing (Sutton & Barto, 1998) can reduce this problem, they do not eliminate it entirely.

Despite its limitations, Sarsa(λ) remains a useful, easy-to-implement baseline method for reinforcement learning in continuous domains.

2.5.2 kNN-TD(λ)

A somewhat less venerable pair of algorithms, kNN-TD(λ) and Ex$\langle a \rangle$ have been demonstrated to give excellent empirical results on a range of domains with

continuous state or state-action spaces (Martín H. & de Lope, 2009; Martín H. et al., 2011).

The algorithms are closely related, differing primarily in how actions are chosen. Both discretize the state space by imposing an arbitrary grid of points over the entire space. In most implementations, this grid is chosen based on prior knowledge rather than being created by memorizing actual observed data points (as in the more traditional view of kNN methods).

The Q-function is approximated by selecting the k points closest to a query point. The algorithm treats normalized Euclidean distances between the query points and the k-neighbors as a normalized weight vector over the neighbors. This is a key difference between these algorithms and Sarsa(λ), because in tile coding, both state values and value updates are calculated using uniform weighting.

During learning, the algorithm uses an ε-greedy policy to promote exploration. Q-function updates use standard on/off policy TD rules. The eligibility traces are defined over each sample point in the state (or state-action) space.

In kNN-TD(λ), action selection is performed using a simple maximization over a discrete action space. Ex$\langle a \rangle$ extends the algorithm to continuous action spaces by choosing an action using a weighted average over a discretized set of actions.

More formally, the algorithm defines a set D of m discrete sample points on $\mathcal{S}$ For a given state $s \in \mathcal{S}$, choose the k points in D with the smallest Euclidean distance from s. Call this set K_s. We represent the mean of the estimated Q-function $\bar{Q}(s, a)$ as a weighted sum over all sample points and discrete actions.

For each of the k nearest neighbors of a state vector s, we calculate weights:

$$w_i = \frac{1}{1 + d_i^2}, \forall i \in K_s, \tag{2.30}$$

where d_i is the Euclidean distance from point i to s. These weights are then normalized:

$$p_i = \frac{w_i}{\sum_{j \in K_s} w_j}, \forall i \in K_s. \tag{2.31}$$

We calculate the Q-function value for a particular action a by performing a weighted summation over the neighbors:

$$\bar{Q}(s, a) = \sum_{i \in K_s} \hat{Q}(i, a) p_i, \tag{2.32}$$

where $\hat{Q}(i, a)$ is a weight associated with a particular sample point $(i, a) \in D$. Greedy action selection is then:

$$a = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, \bar{Q}(s, a). \tag{2.33}$$

We can calculate TD-error δ either on-policy:

$$\delta = r + \gamma \bar{Q}(s', a') - \bar{Q}(s, a), \tag{2.34}$$

or off-policy:

$$\delta = r + \gamma \max_{a'} \bar{Q}(s', a') - \bar{Q}(s, a). \tag{2.35}$$

The update rule is then:

$$\hat{Q}(i, a)_{t+1} \leftarrow \hat{Q}(i, a)_t + \alpha \delta p_i, \forall i \in K_s. \tag{2.36}$$

The algorithm typically uses a slightly modified form of replacing eligibility traces:

$$e(i, j) \leftarrow \begin{cases} p_i & j = a \\ 0 & j \neq a. \end{cases} \tag{2.37}$$

The update rule becomes:

$$\hat{Q}_{t+1}(i, a) \leftarrow \hat{Q}_t(i, a) + \alpha \delta e_t(i, a) \tag{2.38}$$

$$e(i, a)_{t+1} \leftarrow \gamma \lambda e_t(i, a). \tag{2.39}$$

2.5.3 Fitted Q iteration with extremely randomized trees

The fitted Q iteration algorithm (Ernst et al., 2005a), which builds on previous approaches to fitted value iteration (Gordon, 1999; Ormoneit & Sen, 2002), takes as input a set $\mathcal{F}$ of 4-tuples of the form $\langle s_t, a_t, r_t, s_{t+1} \rangle$, where each tuple is an instance of the single-step transition dynamics of the system. Unlike earlier formulations of batch RL, the fitted Q iteration algorithm is well suited for problems with continuous state and action spaces. Also, the algorithm has been shown to make efficient use of training data (Kalyanakrishnan & Stone, 2007), which is especially important in medical applications, where data may be sparse and expensive to collect.

The algorithm uses the recurrence relation:

$$Q_N(s_t, a_t) = R(s, a) + \gamma \max_{a' \in \mathcal{A}} Q_{N-1}(s', a'), \forall N > 1 \tag{2.40}$$

with $Q_1(s, a) \equiv R(s, a)$. For each successive round (as $N \to \infty$), this sequence converges to the true Q function (Equation 2.6) in the infinity norm.

Even if we do not know the transition dynamics or reward function $R(s, a)$ of the MDP, we can nonetheless approximate Equation 2.40 using the fitted Q iteration algorithm. At each iteration k of the algorithm, we form an estimate $\hat{Q}_k$ of the true Q_N function by iteratively learning the mapping:

$$\hat{Q}_k(s_t, a_t) = r_t + \gamma \max_{a' \in \mathcal{A}} \hat{Q}_{k-1}(s_{t+1}, a'). \tag{2.41}$$

By using the empirical reward r_t in this formulation, the reinforcement learning problem can be cast as a batch supervised learning problem. Any regression algorithm can be used to learn the mapping $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$.

Convergence of the fitted Q iteration algorithm depends in part on the properties of the regression function X. For example, with most tree regression algorithms there is no guarantee of convergence if the tree structure varies at each round. However, if the tree structure is fixed after a number of rounds, the algorithm should

Require: A set $\mathcal{D}$ of 4-tuples sampled from the environment.
Require: Regression function X, such as Algorithm 2.4.
  $\hat{Q}_0 = 0$
  $N = 0$
  repeat
    $N = N + 1$
    // Build training set $\mathcal{T}$:
    for $\ell = 1 \rightarrow |\mathcal{D}|$ **do**
      $i_\ell = (s_\ell, a_\ell)$
      $o_\ell = r_\ell + \gamma \max_{a \in \mathcal{A}} \hat{Q}_{N-1}(s'_\ell, a)$
      $T_\ell = (i_\ell, o_\ell)$
    end for
    // Use regression method (e.g. Algorithm 2.4) to compute $\hat{Q}_N$:
    $\hat{Q}_N = X(\mathcal{T})$
  until Stopping criteria are satisfied

ALGORITHM 2.3: Fitted Q iteration.

subsequently converge (Ernst et al., 2005a). Other regression functions may be used instead of tree-based methods, for example, there are successful applications of FQI using various neural-network architectures for regression (Kalyanakrishnan & Stone, 2007; Malof & Gaweda, 2011).

Extremely-randomized trees

Any regression tree algorithm could be an appropriate choice of supervised regression algorithm, given their representational power, conceptual simplicity, efficiency of evaluation, and their excellent performance in the presence of noisy or irrelevant features.

Following Ernst et al. (2005a), Ernst et al. (2006), Guez et al. (2008) and Panuccio et al. (2013), we use the Extremely Randomized ("Extra") tree regression algorithm of Geurts et al. (2006) as the regression algorithm for FQI. Like many similar "random forest" algorithms (Breiman, 2001), the Extra tree algorithm has a demonstrated ability to represent arbitrary functions while ignoring outliers and irrelevant variables. In particular, we choose the Extra trees algorithm because of its

proven empirical performance in experiments with several reinforcement learning domains (Ernst et al., 2005a; Ernst et al., 2006).

Like most random forest algorithms, the Extra tree algorithm builds an ensemble of trees, and the final output returned by the overall regressor or classifier is the result of some combination of the outputs of the individual trees. The algorithm has three key parameters: M, the number of trees to create; K, the number of candidate tests at each node; and n_{min}, the minimum number of nodes which may be split.

The algorithm is fairly straightforward when operating on data with numeric state features. Its input is a training set $\mathcal{T}$ consisting of a set of 2-tuples, each containing a vector of numeric input features and a scalar output. The algorithm grows M binary decision trees independently, each using the entire training set. Because each tree is independent, each tree may be generated in parallel. Starting at the root of the tree, a set of K tests is generated by first randomly selecting one of the input features (without replacement), then sampling a cut point from the current range of that input feature. If K is greater than the dimensionality of the input space, we generate additional tests by sampling with replacement. For each candidate test, a score is calculated based on the relative variance reduction (or information gain) each test achieves. From the set of K tests, the test which produces the best score is retained and used to split the data set into left and right children.

The procedure is repeated for each of the children until either all of the inputs or outputs are constant at a given node, or the number of tuples remaining at the node is less than the parameter n_{min}.

For a specific regression tree, the output at a given leaf node is the mean of the output values for each of the tuples at that node.

Evaluation of a query point is straightforward: For each tree in the ensemble, beginning at the root we use each node's test to select the left or right branch according to the feature values in the test point. When a leaf node is encountered,

Require: Parameters K, and n_{min}.
Require: A training set $\mathcal{T}$ where the jth element is a tuple of the form $T_j = (\mathbf{i}_j, o_j)$, where $\mathbf{i}_j \in \mathbb{R}^N$ is an input vector of N features $i_j^k, \forall 1 \leq k \leq N$ and $o_j \in \mathbb{R}$ is the scalar output associated with $\mathbf{i}_j$.
function EXTRA($\mathcal{T}$)
  if $|\mathcal{T}| < n_{min}$ OR $\mathbf{i}_j = \mathbf{i}_k, \forall j \neq k$ OR $o_j = o_k, \forall j \neq k$ **then**
    return LEAF$\left(\frac{\sum_{j=1}^{|\mathcal{T}|} o_j}{|\mathcal{T}|}\right)$ // Use the mean of the outputs.
  end if
  $\mathcal{X} \leftarrow \varnothing$ // $\mathcal{X}$ will be the set of K tests.
  while $|\mathcal{X}| < K$ **do**
    if $|\mathcal{X}| < N$ **then** // Choose a feature.
      $f_k \sim DU[1, N]$ without replacement
    else
      $f_k \sim DU[1, N]$ with replacement
    end if
    $c_k \sim CU[\min_{\mathcal{T}}(i^{f_k}), \max_{\mathcal{T}}(i^{f_k})]$ // Choose a cut-point.
    $\mathcal{X} \leftarrow \mathcal{X} \cup (f_k, c_k)$
  end while
  $(f_{max}, c_{max}) \leftarrow \text{argmax}_{(f_i, c_i) \in \mathcal{X}}$ SCORE$(f_i, c_i, \mathcal{T})$
  $(\mathcal{L}, \mathcal{R}) \leftarrow$ SPLIT$(f_{max}, c_{max}, \mathcal{T})$
  return TREE$(f_{max}, c_{max},$ EXTRA$(\mathcal{L}),$ EXTRA$(\mathcal{R}))$
end function
function SCORE$(f, c, \mathcal{T})$
  $(\mathcal{L}, \mathcal{R}) \leftarrow$ SPLIT$(f_{max}, c_{max}, \mathcal{T})$
  return $\left(\text{VAR}(\mathcal{T}) - \text{VAR}(\mathcal{R})\frac{|\mathcal{R}|}{|\mathcal{T}|} - \text{VAR}(\mathcal{L})\frac{|\mathcal{L}|}{|\mathcal{T}|}\right)$ / (VAR$(\mathcal{T})$ + 0.0001)
end function
function SPLIT$(f, c, \mathcal{T})$ // Split a set using the given feature and cut-point.
  $\mathcal{L} \leftarrow \left\{(\mathbf{i}_j, o_j) : (\mathbf{i}_j, o_j) \in \mathcal{T}, i_j^f < c\right\}$
  $\mathcal{R} \leftarrow \mathcal{T} \setminus \mathcal{L}$
  return $(\mathcal{L}, \mathcal{R})$
end function

ALGORITHM 2.4: The algorithm to construct a single extremely randomized regression tree (Geurts et al., 2006). The forest of M trees is constructed by running the EXTRA() function on the full training set M independent times. $DU[a, b]$ and $CU[a, b]$ denote, respectively, the discrete and continuous uniform distributions on the interval $[a, b]$. The function VAR() calculates the variance of the output values of the training set. The functions TREE() and LEAF() construct and return components of the tree.

that value is used as the predicted output for that point. The final prediction is the arithmetic mean of the predictions of each individual tree.

Implementation details

In practice, the random forest approximator can be structured in one of two different fashions. First, it is possible to incorporate the action as a feature in the regression, yielding a single tree with the dimensionality of the combined state and action spaces. We then evaluate a point by querying the tree with a vector composed of both the state and proposed action. Alternatively, for problems with a finite action space, it is possible to create a set of forests, one for each action. Each forest is trained on those tuples which contain the action associated with the forest. Either approach can be chosen for many problems, but the first approach is to be preferred for problems with continuous or unbounded action spaces (Guez, 2010).

Off-line methods such as FQI can be considered off-policy methods, since they can theoretically learn the optimal structure from data collected under even a completely random behavior policy. In practice, there are advantages to structuring the learning process in a way that encourages exploration around the likely optimal target policy. In our experiments, we mimicked the approach described in Ernst et al. (2006), in which data are generated in a series of several rounds, with the procedural structure outlined in Algorithm 2.5. During round zero, the actions will be chosen arbitrarily because the initial value of Q_N is also arbitrary. Depending on the nature of the domain, it may be reasonable for each episode to vary in length and/or initial state.

Require: A black box implementation of the environment.
Require: Constants N_I (number of iterations) N_R (number of rounds) and N_e (number of episodes)
Initialize Q_N arbitrarily.
R = 0
repeat
Generate N_e new episodes with an ε-greedy policy using the current Q_N.
Append the new episodes to the data set $\mathcal{D}$.
$Q_N = FQI(\mathcal{D}, X, N_I)$
$R = R + 1$
until $R = N_R$

Algorithm 2.5: The practical implementation of a multi-round approach to fitted Q iteration.

CHAPTER 3
A general framework for sequential decision-making in medical domains

In this chapter we propose a general framework for designing and testing a reinforcement learning approach for adaptive treatment strategies. While these strategies share many common features with other, more common, RL applications, there are a number of domain-specific features and concerns that deserve special attention. We will describe some key design decisions, the solutions that we have used in this work, and some possible alternative methods that might be appropriate in other settings.

The principal contributions of this chapter are the practical guidelines observations to assist researchers designing similar approaches to medical problems.

3.1 Problem definition

The general problem that is the focus of this thesis is the design and implementation of reinforcement learning approaches to the creation of dynamic treatment strategies for medical domains. In dynamic strategies, the treatment type, dosage, or schedule may be continually modified in response to changes in the patient's condition (Murphy, 2005). Such strategies have been proposed for the management of a number of chronic conditions including emotional and behavioral disorders (Pineau et al., 2007; Zafra-Cabeza et al., 2011), drug and alcohol abuse (Murphy et al., 2007; McKay, 2009), radiation therapy for tumors (Ghilezan et al., 2010), anemia (Gaweda et al., 2007; Malof & Gaweda, 2011), diabetes (Bothe et al., 2013), and HIV/AIDS (Adams et al., 2004; Ernst et al., 2005a).

The intention is that these strategies should improve outcome measures, but also that they should exhibit the following key properties:

- Personalization - in that each decision is made with consideration of the relevant medical history of the patient up to the present time.
- Optimality - in that they reflect an effort to maximize some objective measure of reward over a time horizon relevant to patient outcomes.
- Robustness - in that they perform well in the presence of noisy or uncertain measurements.
- Safety - in that the treatment should not increase risks of side effects or other undesirable outcomes.

A goal of the reinforcement learning approach in particular is to discover these strategies directly from data. The power of the method lies in its ability to do this optimization without relying on any model of the underlying disease processes, while still permitting the incorporation of insights learned from such a model, if available.

The framework incorporates the following key phases:

1. Data collection
2. Computational modeling
3. Formulation as a reinforcement learning problem:
    - State and action modeling
    - Time scales
    - Reward function
    - Function approximator and algorithm
4. Validation

It is important to note that this is often an inherently iterative process. For example, it is common to reevaluate the specific RL formulation (i.e., step 3 above) after a phase of validation.

3.2 Data collection

In sequential problems such as those we treat here, we are generally interested in modeling time-series data, in which each data sample depends in some way on the prior state of the system. This is in contrast to many conventional classification tasks in machine learning, often collectively referred to as "supervised learning" methods, which typically assume that each data item is drawn independently from the same underlying distribution over the feature space (Dietterich, 2002).

3.2.1 Preliminary considerations

There are a number of possible approaches to data collection for later use in reinforcement learning methods. The precise experimental method chosen will vary according to any number of practical and theoretical factors.

As we have discussed in Section 2.4.1, all reinforcement learning methods must balance the trade-off between exploration and exploitation. In practice, this means that the agent will choose actions which are likely to produce the best overall outcome, while retaining enough randomization to guarantee sufficient exploration of the state-action space.

As discussed in Section 2.4, some RL methods are able to discover an optimal policy entirely from *off-policy* data, that is, data collected using either an arbitrary or even completely random policy, as long as the policy provides adequate coverage of the state space. Fitted Q iteration (Ernst et al., 2005a) and Q-learning (Watkins & Dayan, 1992) are two examples of off-policy RL methods. In these methods, the *behavior policy* followed by the agent need not be identical to the *target policy* it is learning.

Many RL methods are only useful with *on-policy* data, in which the data are collected according to the current best known target policy, which may be optimized after every time step, or after a group of time steps, possibly by searching

within the "policy space". Methods which are derived from policy iteration (Algorithm 2.2), including actor-critic methods, tend to fall into this category. In these methods it is generally necessary to alternate phases of policy evaluation and policy improvement, which may pose practical issues in some experimental settings.

However, as we will discuss in our results, even algorithms that are technically off-policy may find a good final policy more quickly when the data are not collected under a purely random policy. As a result, the distinction between the two classes of algorithms can sometimes become blurred. In Chapter 5, we will give an example in the case of fitted Q iteration that clearly illustrates that the method gives better results when the data are collected using a series of rounds in which a randomized version of the intermediate policies is used to gather the data for the next round.

Related to the question of on-policy and off-policy data collection is the issue of *on-line* versus *off-line* learning, as defined in 2.4. On-line methods "learn by doing" in a very literal way, applying a policy, observing the results, and updating the policy after each decision. In some instances, it may be practical to optimize a policy on-line by allowing an RL agent to interact directly with the system to be controlled. However, it is likely that most medical applications will benefit from, or even require, off-line learning, in which the agent will attempt to find an optimal policy by operating on a larger set or "batch" of data collected under some preliminary policy, thus yielding an improved final policy. Therefore we strongly suspect that off-line, off-policy methods will be the most useful for the majority of adaptive treatment strategies.

3.2.2 Clinical trials

One possible source of data for possible application to optimization via reinforcement learning is data derived from randomized clinical trials. A number of recent studies have demonstrated that, with proper trial design, it may be possible to learn such strategies from clinical trial data (Murphy, 2005; Pineau et al.,

2007; Shortreed et al., 2011). One design methodology which seeks to enhance the value of clinical trial data for RL methods has been named the "Sequential Multiple Assignment Randomized Trial" (SMART) (Murphy, 2005). The innovation of this method is the pairing of treatment decisions (actions) with clinical observations (states) over multiple treatment stages, where treatment decisions at each stage are randomized for each study participant.

An observation that motivates these study designs is that the time horizon over which we evaluate the effects of the treatment has a significant effect on the choice of action, even in early stages of the treatment (Murphy, 2005). This is a consequence of the fact that the optimum long-term outcome may not correspond to the best short-term outcome after any particular treatment decision. This corresponds to the idea of delayed reward in the reinforcement learning paradigm, in that the value of a state-action pair is determined not only by the immediate reward, but also on the expected reward over subsequent states and actions.

A key idea that follows from the paradigm is that we need to perform multiple randomizations on each individual for each treatment decision over the course of a clinical trial (Murphy, 2005). By randomizing each individual multiple times over the course of a trial, it is possible to observe the final outcomes for all possible patterns of treatment (given a small enough decision space). For example, in a simple trial with only three stages (pre-treatment: S_0, mid-treatment: S_1, and post-treatment: S_2), and two treatment actions (a_1 and a_2), we may choose treatment a_1 for some patients and a_2 for others after the initial evaluation (stage S_0). After observing the patients' response to the first treatment choice during S_1, we would then assign some patients to receive treatment a_1 and others to a_2 such that all four possible patterns of treatments were observed. In the presence of delayed effects, it is possible that some portion of the patients will show a good final post-treatment response in S_2, even if they did not respond in S_1. Therefore, we must randomize

a study participant's treatment over each treatment phase in order to fully capture the dynamics of the treatment in the presence of delayed reward (Murphy, 2005). However, the SMART design does permit the use of information learned in earlier phases to modify action selection in later phases. These design considerations can be thought of as another instance of the exploration/exploitation trade-off.

3.2.3 Other experimental sources

While multiple-randomization (SMART) trial methods have been described primarily for applications in clinical trials with humans, there is good reason to adopt similar methodologies in other classes of experiment. For example, the design of both *in vitro* and *in vivo* animal experiments could be modified to better capture the full range of responses to sequential treatment decisions.

For example, to better model the evolution of the response of specific tissue types to some therapy, it may be useful to perform *in vitro* experiments in which the treatment is sampled from some probability distribution at each decision point. As with SMART data, the distribution may be permitted to evolve over time as data are collected, but this is not strictly required. Using such a data collection strategy may impose greater costs on the study, but it helps avoid flawed conclusions resulting from hidden factors (Murphy, 2005).

Data from non-randomized observational clinical studies is subject to the same weaknesses that limit the usefulness of such data in general clinical decision making. Non-randomized studies are susceptible to biases and to the influence of hidden variables, which will presumably limit the use of such data in the design of adaptive treatment strategies. In some cases, data from observational studies might nevertheless be useful for helping guide general design considerations.

3.3 Computational modeling

In addition to data from clinical trials and other biological sources, it may be useful to incorporate data derived from other models of disease treatment, especially laboratory or computational models. Compared to clinical data, purely computational models can provide effectively unlimited quantities of data over a wide range of simulated conditions. These models also therefore provide a completely reproducible environment which may permit manipulations of conditions that might be impossible, or merely very difficult, in the best laboratory conditions. Unlike most experimental sources, this data can be created without noise or missing data. Finally, some promising reinforcement learning algorithms require a black box simulator of the system (Kocsis & Szepesvari, 2006).

Most computational models involve creating one or more differential equations which hopefully capture the important details of the evolution of the system over time. Once these differential equations are chosen (along with appropriate parameters), it is possible to integrate them over time using any of a number of standard techniques (Press et al., 2007).

3.3.1 Advantages of computational modeling

There are many good arguments in favor of computational modeling in a variety of systems, especially in biology and neuroscience.

First, the amount of data that can be collected is essentially unlimited. This is immensely important in domains such as reinforcement learning, where many of the historically important methods require a great deal of data to learn an optimal policy. In many cases the data can be created faster than real time, which improves the speed of training and comparison of control algorithms.

An additional argument for computational models is their reproducibility. We can design a model such that even if it has stochastic elements, the random number sequences used in the model may be deterministic and this can be exploited to create

experiments in which we can be certain that a particular manipulation is the only change which will have a measurable effect on the system's behavior.

Finally, the models may allow us to manipulate variables systematically, at a level of detail that simply is not possible in real systems.

3.3.2 Limitations of the modeling approach

The problems with computational models are many, however. First, limitations of both scientific knowledge and computational power typically require that these models be greatly simplified, sometimes to the point of being high-level or phenomenological descriptions of the behavior of the modeled system, rather than being based on "first principles". For example, there are many highly-reduced models of neurons (Abbott, 1999; Izhikevich, 2003) which avoid explicit modeling of ion channels and other mechanisms. As models increase in realism, the second challenge that arises is the problem of choosing parameters for each model mechanism. While there may be good sources in the experimental literature to guide these choices, there are inevitably many parameters which must be chosen arbitrarily, or nearly so. For example, the computational model of epileptiform activity which is presented in Chapter 4 of this thesis requires dozens of parameters. For most of these, many plausible values have been reported in the relevant literature (Jahr & Stevens, 1990; Huguenard & Prince, 1992). In some cases, few if any good estimates of these parameters are available. Our computational model of cellular survival in radiation fractionation (Chapter 6) is much simpler, but still requires some difficult parameter choices. One of the many barriers to effective models of biological systems is the lack of well-curated estimates for the relevant parameters.

The computational complexity of these models may also pose a problem. As the model grows in complexity, or if the integration time step needs to be very small to insure stability, the model may take considerable computing power to evaluate. In the worst case, some computational models of complex neural circuits can take

many hours of computer time to simulation a few seconds of real time, even in massively parallel implementations. This can make detailed models nearly useless for some applications.

3.3.3 Model validation and evaluation

Empirical evaluation of computational models, in the sense of deciding how well a particular model captures the dynamics of the modeled phenomenon, can still be somewhat poorly defined. While it is theoretically possible to compare the model's output to the real systems using standard statistical tests, there are a number of reasons why this is not done. Often, models are completely deterministic, so no meaningful statistics can be defined for them. More significantly, because the models are at best very imperfect simulations of the real systems, they rarely copy the behavior in a detailed enough way to permit other detailed quantitative comparisons. In particular, the model may intentionally not express purely observational issues such as measurement noise.

We argue for a specific approach to evaluation in our model of epilepsy (Chapter 4) which involves choosing key statistics based on timing characteristics of ictal and interictal events. However, this approach may not be applicable to other models.

3.3.4 Numerical integration

Among the practical considerations for any modeling problem is the method of numerical integration to use, and the relevant time step to consider.

In many cases we would argue that the simplest numerical integration method, commonly referred to as Euler or Eulerian integration (Press et al., 2007), is sufficient. In this case, for a state variable y, we simply write:

$$y_{t+1} = y_t + f(t, y_t)\Delta t \tag{3.1}$$

where the function $f(t, y)$ is the derivative of y with respect to t, evaluated at t, y:

$$\dot{y} = f(t, y). \tag{3.2}$$

For functions with sufficiently "smooth" derivatives, this will often give satisfactory results for small values of Δt.

In many cases our choice of integration methods and parameters was based on empirical observations of the systems of equations. For the complex model such as that presented in Chapter 4, we found that the model's behavior was unacceptably sensitive to the choice of Δt when using Euler integration, at least for any computationally feasible value of Δt. We instead use another standard choice, 4th-order Runge-Kutta integration (Press et al., 2007), which updates the value using the more complex rule:

$$y_{t+1} = y_t + \frac{1}{6}\Delta t(k_1 + 2k_2 + 2k_3 + k_4) \tag{3.3}$$

where:

$$k_1 = f(t, y_t) \tag{3.4}$$

$$k_2 = f(t + \frac{1}{2}\Delta t, y_t + \frac{1}{2}\Delta t k_1) \tag{3.5}$$

$$k_3 = f(t + \frac{1}{2}\Delta t, y_t + \frac{1}{2}\Delta t k_2) \tag{3.6}$$

$$k_4 = f(t + \Delta t, y_t + \Delta t k_3). \tag{3.7}$$

4th-order Runge-Kutta methods are quite common in the computational neuroscience literature, being used in several of the sources used to develop our own model (Bazhenov et al., 2004; Golomb et al., 2006). This method typically gives a good trade-off between computational efficiency, implementation complexity, and numerical accuracy in these models.

As can be seen in the case of the model of epilepsy, the differential equations themselves are highly nonlinear and change rapidly for small changes in the input variables, necessitating a more stable integration method. In comparison, we found Euler integration was sufficient for the models presented in Chapters 5 and 6.

3.4 Formulation as a reinforcement learning problem

The field of reinforcement learning has generated a large body of research, although the most successful methods and applications are those with either finite state and action spaces, or where prior knowledge of an application's structure allows us to select a parameterized policy that can be optimized effectively. Even in these cases, reinforcement learning methods often have poor "data efficiency", requiring large (potentially infinite) amounts of data to learn an optimal policy (Kaelbling et al., 1996). In a more complex environment with either countably infinite or continuous states and/or actions, it is no longer feasible to sample all possible state-action pairs. Instead we must use a function approximation method that estimates the value of unvisited states and untried actions, generally imposing some strong assumptions about the "shape" of the state-action space. Given this approximation, the agent can learn a functional mapping from states to actions, which should allow the agent to generalize from its prior experience to newly encountered states.

Additionally, there are often many potentially relevant dimensions in a medical problem, given the amount of data that may be collected per patient. These data may include both objective measures of the patient's status (from biological assays, e.g.) or more subjective measures derived from psychological tests or expert assessments. For example, a single study on schizophrenia treatment recorded 50 different variables for each patient, 30 of which were re-assessed at each stage of the trial (Swartz et al., 2003; Shortreed et al., 2011). Many of these data are aggregate measures themselves, and therefore could be further subdivided into individual components of a larger assessment.

3.4.1 State space representations

Many medical outcome measures are most naturally expressed using a mix of discrete categorical and continuous variables, which may vary over a wide or unknown range. In addition, as we have discussed, state spaces in medical problems

are typically high-dimensional, with highly heterogeneous dimensions. This gives rise to several problems in state space representation, including:

1. The need for a state that is a sufficient statistic for the problem domain.
2. The issue of irrelevant state variables.
3. The problem of high-dimensional state spaces (the "curse of dimensionality").
4. The need to approximate continuous state variables.

A state representation is *sufficient* in a statistical sense if it contains enough information to completely specify the relevant parameters of the associated distributions (Wasserman, 2004). In the context of RL, a state should be sufficient to specify the distribution of future rewards and state transitions. Since we express the RL policy π as a mapping from state to action, if our state is insufficiently informative, our policy may lack crucial information. In medical domains we will rarely know which state variables are sufficient for our MDP, and so we must rely on a certain amount of intuition to guide the choice of state variables.

Therefore, we will often make the conservative choice to include state variables which are in fact irrelevant. These irrelevant state variables may become a source of error or reduced data efficiency in an RL problem. Some methods are known to be relatively robust to the presence of uninformative state variables, so when in doubt, such methods are of tremendous value (Ernst et al., 2005a).

As the number of state variables grows, the problem is affected by what is known as the "curse of dimensionality", the fundamental insight that the number of states in a system increases exponentially with the number of dimensions. This effect manifests itself in a number of ways, for example, by increasing the amount of data required to guarantee a particular confidence bound (Wasserman, 2004). Again, a number of RL methods attempt to address this problem using various approximations to reduce the effective dimensionality of the state.

We have already discussed the complications introduced by continuous state variables. In many cases, it may be possible to discretize the state variable, that is, to simply map a continuous value onto a set of discrete points. While many continuous quantities can be discretized in some meaningful way, it is not always clear how best to map an imprecise but nominally continuous value into a discrete set. This requires insight into the specific limitations and behavior of the relevant data sources. In some cases, clinical data may have well-established standards for reducing a complex set of measurements into a discrete or categorical variable, as with the common mapping of blood pressure into categories (hypotension, prehypertension, etc.) according to widely-known cutoff values. If available, these categories can be simply represented by mapping the continuous measurements onto a set of distinct integral values.

In many cases, however, this sort of simplification may not be so straightforward, either because the guidelines for categorization may be unclear, or because there may be reason to believe that a specific application will require fine-grained readings in order to achieve efficient control.

Any discretization will tend to introduce bias into the estimate of the value function. Simply mapping a continuous value into a set of discrete values may introduce greater errors by making it more difficult or impossible to properly minimize the mean-square error (Equation 2.20).

3.4.2 Action choices

Just as the state space of many medical domains may be represented most naturally as a high-dimensional, continuous state space, likewise the action choices available for medical therapies are often naturally continuous or countably infinite. As with the state spaces, increasing the size and dimensionality of the action space poses a substantial problem for optimization, because of the combinatorial explosion in the size of the potential solution space. As discussed in Section 2.4, RL

algorithms such as Q-learning require the agent to find the maximum over the action space on every update. If computing this maximization is computationally expensive, the performance of an RL agent will be adversely affected.

In theory, clinicians can choose from a wide range of drugs or therapies at varying dosages, as well as other alternative or complementary therapies. However, in many clinical settings it may be natural to restrict the range of possible actions into a small set of discrete choices. This may be the result of many causes, for example, the impracticality of exploring every possible value of action (e.g. every possible dosage of a drug), or the practical limitation in the precision of relevant devices or measurements. In practice, these choices will change the nature of the possible solutions space, but they may not preclude a good solution. However, such restrictions, where they can be exploited, may prove very useful for RL applications. For example, in the HIV model of Adams et al. (2004), the choice of drug dosages is specified using a continuous variable, and an optimal solution can be given in terms of those continuous values (Adams et al., 2005). In practice, it would probably not be practical to implement a treatment schedule that adjusted the dosage in small increments on a time scale of hours. Using an RL framework, however, it is possible to find solutions that eventually achieve drug-free control of the infection after a therapeutic period, in a structured treatment interruption (STI) methodology that restricts the dosages of the therapeutic drugs to either zero or a specific dosage with a 5-day time step (Ernst et al., 2006).

In general, action spaces with discrete variables and fewer dimensions will tend to lend themselves to more rapid and confident development of RL solutions.

3.4.3 Time scale and discount factors

All RL problems are fundamentally time-based, and therefore require that decisions be made about how the problem will evolve through time.

Time step

Since RL problems typically assume discrete time, the first and most obvious choice is the length of the fundamental time step of the system. This choice will depend on the nature of the system under study. For chronic medical treatments in a clinical setting, a time step of days or even weeks may be appropriate (Ernst et al., 2006; Shortreed et al., 2011). However, as in our work with implantable medical devices, it makes sense to use a much shorter time scale, perhaps seconds or even milliseconds, in order to guarantee that the system can respond to rapid changes in the patient's condition. In general, the time step should be chosen to reflect the presumed time scale of the patient's response to treatment. A drug may take days or weeks to show any effect, whereas electrical stimulation may take effect within milliseconds of application.

Time horizon

Another fundamental detail of any RL domain is the time horizon of the system. RL problems can be divided into "finite" and "infinite" time horizon cases. In the first case, the decision-making task must terminate after some finite number of time steps, where typically the exact number of steps, or at least an upper bound, is known in advance. Most therapies that seek to either cure a condition or achieve remission can probably be thought of as finite time horizon problems. In these cases, we may think of the treatment as seeking to move the patient from a "bad" state to a stable "good" state.

However, for some chronic conditions, or those with very short time steps, it may be more natural to assume an infinite time horizon. In this case, the assumption is that the decision-making RL agent will need to operate indefinitely. For example, the patient's response to the treatment may be unstable, and therefore continual maintenance is required to avoid a relapse.

TABLE 3–1: Illustration of the fractional decay after 10 and 100 time steps for various common values of the discount factor γ.

γ	γ^{10}	γ^{100}
0.95	0.60	0.006
0.98	0.82	0.13
0.99	0.90	0.37
0.995	0.95	0.61

In addition, it is possible to formulate MDP and RL tasks in a manner that attempts to maximize average reward over an indefinite time horizon (Mahadevan, 1996; Tsitsiklis & Van Roy, 1999; Gosavi, 2004). While these methods might be applicable in some medical domains, we have not investigated them here. In general we feel that any medical treatment strategy will naturally be weighted towards maintaining a long-term quality of life, while placing some emphasis on immediate relief, and the discounted reward case provides a good mechanism for representing this idea.

Discount factor

A key parameter affecting the timing characteristics of the system is the discount factor applied to future rewards in the infinite-horizon case, which is commonly denoted by the symbol γ in reinforcement learning literature (e.g. see Equation 2.1). The discount factor can be thought of as the probability that the agent will survive until the following time step (Kaelbling et al., 1996). In practice, the choice of γ gives the designer a means to trade off between the priority given to immediate and delayed reward; decreasing the value for γ will make the agent increasingly "myopic", with the limiting case of $\gamma = 0$ requiring that the agent be entirely concerned with maximizing its immediate reward, probably at the expense of future rewards (Sutton & Barto, 1998). In contrast, if γ is close to one, the agent will focus on longer term outcomes. Common values for γ lie in the interval $[0.95, 0.99]$. It is useful to consider the decay rate for common values of γ, which we have summarized in Table 3–1 for 10 and 100 time steps. The key point is that

the compounding effect of the exponential decay may have large effects on system behavior for small changes in the nominal value of γ. The best choice for γ will depend on the characteristic timing of the system. If we have very few, long time steps, as in some clinical studies, it may make sense to choose a smaller value for γ. Where the dynamics of the system are very fine-grained (on the order of seconds or less), such that an agent may need to consider rewards relatively far into the future, it may be more reasonable to use a value very close to one.

On the other hand, if a problem has a finite time horizon, a discount factor of one gives great weight to the final outcome of some fixed-length intervention. However, in an infinite horizon setting, we must choose $\gamma < 1$ in order to preclude an infinite total reward (Sutton & Barto, 1998).

3.4.4 Reward functions

Much of the value in reinforcement learning methods derives from the method's ability to form a model for optimal decision making using a relatively simple reward function. The reward function should demonstrate to the agent the costs or rewards intrinsic to the environment, but the function need not be complicated. The classic example is game playing, where RL methods may avoid specifying any details about game tactics or strategy, but simply define a reward function in terms of whether the agent wins or loses the game (Tesauro, 1992; Gelly & Silver, 2007). Stated mathematically, the function might simply be:

$$R(s,a) = \begin{cases} 1 & \text{if the agent wins the game at state } s \\ 0 & \text{otherwise.} \end{cases} \tag{3.8}$$

A simple choice of reward function gives the agent freedom to find good solutions even in problems that are otherwise poorly-modeled. While results can vary, this simplification can ease system design while leading to surprisingly good results,

as the agent may discover decision rules that are superior to those of even expert practitioners (Tesauro, 1995).

There is considerable latitude in the definition of reward function. The only clear requirement is that the function must be bounded over the entire state (and action) space. For most RL algorithms, the reward function can be chosen to be whatever is conceptually convenient. It may be linear, nonlinear, and/or discontinuous. However, it has been shown that the precise choice of reward function can have a significant impact on learning rates in RL problems (Ng et al., 1999; Matarić, 2001).

A poorly chosen reward function can lead to unexpected and undesirable results. For example, in Chapter 5, a case study on a model of Parkinsonian hypersynchrony (Tass, 2003), some of our initial experiments used a reward function which considered both the number of simultaneously firing neurons and the amount of stimulation applied to the system. Early RL experiments using this reward function showed promising results which initially suggested that the learning agent was achieving the desired goal of decreasing synchrony. However, detailed examination of the results showed that the agent was learning to stimulate the network such that most of the oscillators in the model were never reaching the firing state – a situation which is certainly unrealistic and probably undesirable. Modifying the reward function to include a small reward for each firing oscillator helped the system find a policy which was more moderate in its approach.

Another example arose in the case of the examination of the optimization of adaptive radiation therapy in Chapter 6. In this case, the goal of the system is to preserve normal tissue, while destroying most or all of the cancerous tissue. While this will be covered in more technical detail in Chapter 6, it was found that some intuitively appealing forms of reward function (e.g. the ratio of live normal cells to live cancer cells) did not yield useful results, and we show that other formulations

can meet the numerical requirements imposed by RL while permitting the discovery of good treatment policies.

While it can be difficult to define exactly what constitutes a "good" reward function, it is probably reasonable to suggest that a simple function that clearly reflects the desired goal is a safe initial choice. In Shortreed et al. (2011), the reward function for the treatment of schizophrenia is chosen to be the negative of the area under the curve for a global measure of symptoms. While minimizing symptoms may be the highest priority in medical problems, it may also be useful to define the reward function in terms of a trade-off between the costs associated with treatments and the costs associated with symptoms. For example, in the HIV model of Adams et al. (2004), the reward function is:

$$R(s,a) = c_1 a_1^2 + c_2 a_2^2 + c_3 s_V + c_4 s_E \tag{3.9}$$

where variables a_1 and a_2 are real-valued actions representing to drug dosage levels, and variables s_V and s_E are state variables corresponding to the viral load and immune response, respectively. There are other state variables that do not participate in the reward function. The coefficients c_1 through c_4 are constants which are chosen to adjust the priorities of the agent, as well as to correct for differences in the range of the state and action variables. Naturally, c_1, c_2, and c_3 must be negative to direct the agent to minimize treatment and viral load, whereas c_4 is positive to encourage the development of a strong immune response.

3.4.5 Function approximator and algorithm

Given a state and action space, appropriate time scale choices, and a reward function, we will finally have to select from one of several RL algorithms, and the specific method used to represent the state (or joint state-action) space. In the simplest case, it may be possible to represent a discrete, finite state-action space with a simple tabular Q-function. In this case, no function approximation is necessary and

the problem can often be solved by straightforward methods such as Q-learning or Sarsa.

Function approximation

If the problem is of high dimensionality, or contains continuous or infinite values in either the state or action space, we need to apply a function approximator. Function approximation methods, as first described here in Section 2.4.3, provide a solution to the problem of continuous state or action values by allowing an RL agent to directly work with an approximate representation of a continuous space. While approximation methods may introduce bias, they allow for the use of nominally continuous values in the state space, while making some assumptions about the structure of the space in order to make reinforcement learning practical.

The use of a function approximator may also constrain which optimization algorithms are available for our problem. Many algorithms exist which operate on various approximate representations of the Q-function (Ormoneit & Sen, 2002; Peters & Schaal, 2008). These methods permit the discovery of a solution for the optimal policy, given the constraints imposed by the choice of approximator. Most choices of approximator will require that we make some assumptions about the structure of the state-action space. For example, one common assumption is the smoothness of the value function over the state space, which implies that the optimal policy for an unvisited state will be similar to that of a nearby visited state. In other words, states which lie close to one another according to some metric should choose the same (or similar) actions. One specialization of this assumption is the choice of approximating the Q-function as a linear combination of basis functions (e.g. tile coding (Sutton, 1996)).

While strong continuity assumptions may be reasonable in many domains, there is evidence that they may not be adequate in some medical domains. For example, in structured drug therapies for HIV, there is a need to "turn on" and "turn

off" therapies in response to small changes in the patient's state (Adams et al., 2004; Ernst et al., 2006). Mathematically, these requirements manifest themselves as discontinuities in the mapping of state to action. That is, minor changes in the state may lead to sudden changes in optimal action. Approximation methods that do not allow for large changes in action at critical points in the state space will perform poorly on these tasks (Ernst et al., 2006).

One promising family of methods for representing these poorly structured state spaces are regression trees, especially stochastic algorithms such as random forests (Breiman, 2001) and extremely randomized ("Extra") trees (Geurts et al., 2006) (as discussed previously in Section 2.5.3). These approximators have several important features: they are able to represent functions with discontinuities, they show a good rejection of spurious features, and they are reasonably simple and efficient both in terms of both ease of implementation and computational complexity. Their main drawback is that they are inherently a batch, supervised learning method, which forces most RL methods that rely on them to perform off-line learning using batch methods (Ernst et al., 2005a). This limits the ability of the system to respond to non-stationary environments. Until recently, no satisfactory method has been proposed for using randomized trees to represent a Q-function in an on-line setting, but there have been some attempts to design random forest algorithms for streaming or on-line settings (Abdulsalam et al., 2007). A recent algorithm by Barreto (2014) appears to at least partially address this issue.

Clearly in the case of high dimensional data, it will be useful to apply methods that can efficiently use such a high-dimensional, heterogeneous data set, or to use prior knowledge to select which state variables will be included. Again, the regression tree methods are especially effective here, as they have a demonstrated ability to find good approximations of complex function in the presence of irrelevant state features and outliers in the data.

There has been some recent work on methods for learning a good state representation directly from data, often termed *feature selection* (Kolter & Ng, 2009; Geramifard et al., 2011; Fard et al., 2013). These methods attempt to learn an approximate representation of the state space by projecting it onto a set of basis functions, but they use more sophisticated algorithms than tile coding, for example, to reduce the dimensionality of a problem while ensuring irrelevant state features will not affect the results.

Continuous action spaces

While there are many shared issues that arise in both the continuous state and continuous action cases, in actual practice, the problem of continuous action spaces has been less well studied.

There are several reasons that continuous actions might be needed in an RL domain. One-shot tasks such as throwing a projectile cannot easily be optimized using discrete actions. Many other control processes involve elements that are most naturally expressed as continuous variables. For example, the force on a robotic arm (Peters & Schaal, 2008), the angle of a control surface (Gaskett et al., 1999; Sterne, 2004), or the current on a stimulating electrode, are all naturally continuous values.

Any approach to modeling a domain using a continuous action space has to solve a more difficult problem than in the discrete action case. Methods for continuous spaces typically must assume that it is appropriate to choose "similar" actions in "similar" states. This generally imposes strong smoothness constraints on the joint state/action space. However, while it is often reasonable to assume that expected future rewards will vary smoothly with the action choice, this is not guaranteed in general. For example, using interpolation to generate actions may yield unexpected consequences. The mean of two good actions may not always be equally good: if a

car is driving straight towards a wall, it should not take the mean of a 90° right turn and a 90° left turn (Sterne, 2004).

While the action approximation problem is less well-studied than the problem of state approximation, the problem of generalization in continuous action spaces has been approached using several classes of algorithms (Martín H. & de Lope, 2009; Busoniu, 2009; Pazis & Lagoudakis, 2009; Antos et al., 2007; Baird & Klopf, 1993; Ackley & Littman, 1990). Most of these begin with the selection of a general function approximation method that is used to model the continuous action space, either separately or jointly with the state space. While it is often convenient to use the same approximator, or class of approximator, for both the state and action spaces, this is not a requirement. For practical purposes, it is important that the approximator supports the efficient calculation of the action with the highest expected value for a given state, as in Equation 2.8. This is often difficult, especially in action spaces with many dimensions.

3.4.6 Data efficiency

While not always a high priority in some applications, RL algorithms vary considerably in their data efficiency. Many classic methods for temporal difference learning, for example, require a great deal of training data (Sutton & Barto, 1998). This may not be an issue if a good simulator is available for the environment, as may be the case with domains such as standard board games. For example, the backgammon agent developed by Tesauro (1995) learned via self-play in hundreds of thousands of simulated games.

However, in many medical domains data collection is expensive and difficult. It is unlikely that clinical trials can be conducted over tens of thousands of subjects, therefore a much greater emphasis is likely to be placed on data efficiency. Some algorithms for RL can learn a good policy with relatively little data (Ernst

et al., 2005a; Peters & Schaal, 2008), generally at the cost of greater computational complexity, a more restricted policy space, or both.

One general framework that addresses data efficiency is the idea of *regularization*, which is a general method for avoiding overfitting, often imposed as part of least-squares or kernel-based regression (Ormoneit & Sen, 2002; Lagoudakis & Parr, 2003). Regularization imposes some constraints on the solution, which may be expressed as a restriction on some error norm (Kolter & Ng, 2009), or on the Bayesian prior of the distributions (Strens, 2000). In many cases, regularization is naturally combined with feature selection within the general function approximation framework, as discussed in Section 3.4.5 (Kolter & Ng, 2009; Fard et al., 2013).

3.5 Validation

In most real-world environments, we do not have access to a known optimal policy for an adaptive treatment strategy. However, it can be important to know how "good" our policy is, relative to some baseline, and over some breadth of possible variations in the environment. This is a surprisingly complex problem, and remains an area of active research (Fonteneau et al., 2009; Păduraru et al., 2012; Păduraru, 2013).

Evaluating medical problems is especially challenging. Of course, a variety of ethical and practical issues arise that are beyond the scope of this thesis. But at a minimum, we would expect to be asked to demonstrate that adaptive treatment methods will produce better outcomes for some classes of patient according to some metric. More ideally, we would like to demonstrate that the expected results for all patients are improved by this strategy, or at least that no patients will be harmed by the strategy.

Given that we cannot expect to perfectly approximate the state space, a necessary precaution might include the imposition of rigid limits on the action space,

such that no action proposed by the agent is permitted to lie outside some well-established clinical guidelines. This may help guarantee some margin of safety, but it still leaves the question of efficacy unanswered.

The problem is complicated by the need to evaluate the likely value of a policy using off-policy data. In most clinical trials, we would expect that we would collect data under some predetermined policy π^0, which might be uniformly random in some cases. We would use this data to perform an optimization and derive a new, "better" policy π^1 that we would now like to use and evaluate. However, the data was collected under π^0, and so the distribution of states and rewards over the original data will not correspond to the distribution expected under the new policy.

Various methods have been proposed to address this problem. A simple method was proposed in Guez et al. (2008). In this study, data was collected from an *in vitro* model of epilepsy using fixed, constant policies. The data was then used to train an RL agent, which implemented an adaptive policy. The researchers were then faced with the need to estimate the performance of the adaptive policy without applying the adaptive policy to identical *in vitro* setting. Their proposed solution involves rejection sampling of states and actions according to the derived policy π^1. In this case, we consider only those data which are compatible with the new policy, in other words, those states where $a_t = \pi^1(s_t)$, and ignore states where the action is not consistent with the adaptive policy. In this way, we can compute a measure of the likely performance of the derived policy using data collected under the original, experimental policy. This method was later applied in Malof and Gaweda (2011).

It is also possible to address this issue by use of the *bootstrap*, which refers to a general class of methods in which a data set is resampled several times to generate alternate copies which differ in detail but which are assumed to share the same distribution (Wasserman, 2004). Effectively, these methods simulate the creation of alternative data sets by assuming that the "true" distribution of the process which

generated the data is reflected in the empirical distribution of the data itself. Bootstrap methods allow calculation of confidence intervals and other statistical measures, and can be readily applied in some reinforcement learning contexts (Shortreed et al., 2011).

More sophisticated mathematical methods have been proposed for evaluation of learned policies (Fonteneau et al., 2009; Păduraru et al., 2012; Păduraru, 2013). However, many of these approaches are either constrained to deterministic MDPs or impose fairly strong continuity requirements on components of the MDP.

Another important form of validation is sensitivity analysis, or exploring how the model is affected by changes in parameters. In general, it is difficult to find an analytic expression for the confidence bounds for the expected reward or policy in any non-trivial RL setting. However, one can perform an empirical analysis by generating results for a range of possible conditions, and examining the distribution of the results.

3.5.1 Empirical evaluation of RL methods

Analogous methods can be used to evaluate and compare RL algorithms. We demonstrate in Chapters 5 that, as with the computational models themselves, it is often useful to vary the parameters of the learning algorithms. The behavior of the algorithm under these changes of parameters provide a useful empirical estimate of the stability of the overall solution, and gives us a rough estimate of the likelihood of finding a solution of similar quality given a different data set. While it is generally possible to perform some amount of optimization over the RL algorithm parameters, it is not always practical to cover the entire range of possible parameter settings, especially when stochastic behavior is present. Superior RL algorithms will either provide some guidance for making good parameter choices, or they will be relatively insensitive to those choices, at least for large classes of interesting problems.

Also, since many RL algorithms have a stochastic component, either as part of the policy or the function approximator, it is again possible to vary the initialization of the algorithm to examine the distribution of possible outcomes, even under identical parameters and data sets. This strategy can be applied even if we do not have access to a computational model, or when the computational model is completely deterministic. We have applied this approach extensively in Chapter 6.

3.5.2 Policy evaluation with computational models

When we do have a computational model available, a number of empirical evaluation methods become practical. There are several specific approaches that can be useful. First, when an environment is fundamentally stochastic, one can evaluate the model over several instances simply by changing the seed of the random number generator and compute the empirical distribution of the outcome measures. We demonstrate this approach in Chapter 5, where most of our confidence intervals are determined empirically.

A related approach can be simulated by varying the initial conditions of the model, where appropriate. This effectively randomizes the model, and with it, the behavior of the RL algorithm. However, some models may have very specific initial conditions that do not lend themselves to obvious variation. However, we illustrate an application of this method in Chapter 6, in which a small perturbation in the initial conditions of the model has major effects on the learned policy.

Finally, when we have identified key model parameters, we attempt to validate whether changes in the model parameter may have large effects on the quality of the learned policy. This approach is used heavily in both Chapters 5 and 6.

3.5.3 Qualitative aspects of validation

Another form of validation encompasses a more qualitative examination of the derived policies, to explore the possibility of pathological or dangerous behavior. For example, in Section 6.4 describing our attempts to model radiation therapy in a

RL context, we give an example of a model in which the system is taking on states which would be either impossible or highly undesirable (Figure 6–5. This case demonstrates a strength of the modeling approach, which is the ability to rapidly prototype a new RL formulation when a deficiency is found. This means that, in practice, we often repeat parts of the overall design process, for example, testing a domain using a different reward function. In this case, we found other formulations of the problem which enforce a more realistic outcome.

Another strength of the modeling approach adopted by this thesis is the ability to test a policy's behavior in a variety of artificial conditions. For example, we could examine whether a policy would ever choose an otherwise illegal action value, or see how a policy will respond to an "impossible" state (or series of states). While strictly empirical, this approach would help to bound the possible worst-case behavior of the policy, increasing confidence that it will not make absurd or dangerous treatment decisions.

3.6 Common issues and limitations

Medical domains present special challenges with respect to data collection. This especially includes noisy, biased data, or missing data. Each of these problems poses issues for any statistical method in machine learning, but perhaps more so in the case of reinforcement learning where the algorithms tend to have relatively low data efficiency.

3.6.1 Noise and other sources of uncertainty

Medical data are often inherently noisy. By "noise" we generally mean signals which we believe to be derived from an unbiased source of measurement error or other stochastic processes. However, there are many situations in which what we call noise may derive from deterministic but unknown state variables and other intrinsic properties of the system. This is especially true of medical data, because of the complexity of the biological systems (including animal models), the errors and

uncertainties of treatments, and the high rate of errors introduced by measurement and record-keeping problems.

Therefore, successful methods must have the capability to reject noise and irrelevant or uninformative state variables. It is often difficult to know *a priori* which state features may be relevant, but some methods (e.g. randomized trees) have a proven ability to develop models which ignore irrelevant features, whereas other methods (e.g. coarse coding) do not inherently protect against these features (Breiman, 2001; Geurts et al., 2006).

3.6.2 Sample biases

Because medical data are often derived from clinical treatments, there may be large biases in the data collection process. For example, it is commonly observed that data from medical studies often exhibit measurable biases and clustering effects which presumably reflect systematic differences in data collection among different physicians, measurement devices, or medical centers. These effects can be quite large, often comparable in size to inter-patient variability (Localio et al., 2001; Wynants et al., 2013). This suggests that data collection practices may vary persistently and substantially at different organizations, and this effect appears irrespective of the underlying technology or methodology.

Many diseases are themselves very complex, and clinicians may face limits on exactly what treatments can be applied or what data can be collected in a given case. For example, much of the subcortical electrophysiological data collected from epilepsy patients is acquired from "pre-surgical" patients, who are being evaluated for surgical removal of the epileptic focus. Clinically, it makes sense to perform recordings over a narrowly-focused area near the intended surgical target. However, this means that each patient has a different number of electrodes, implanted at different locations, and recorded a brief period in an atypical environment (Klatt

et al., 2012). It is not clear how well such data will generalize to other patients, especially those who are not themselves candidates for surgical interventions.

These kinds of biases can significantly distort results in any machine learning paradigm, and reinforcement learning is no exception. Most traditional learning algorithms either require "unbiased" data or data which approximates a known good policy. In general, not all reinforcement learning methods can successfully learn from "off-policy" data. That is, to learn a good policy, they must be able to collect data from a policy which approximates the best known policy at any given time. While methods do exist to account for these biases in conventional statistical tests (Wynants et al., 2013), it has not been well studied in the reinforcement learning context.

3.6.3 Missing data

In addition to the previously mentioned issues, medical datasets are notorious for missing or incomplete data. Many studies are conducted with patients who fail to complete part of the study, or, because of the finite duration of most studies, there is often no information about outcomes after some fixed end time. For example, a drug study may fail to account fully for side effects that appear only after the study has concluded. This missing or truncated data will tend to increase the variance of estimates of the value function and the policy in an RL setting.

There are a number of well-studied methods for choosing good estimates for missing data values, and some of these have been discussed in the context of reinforcement learning. Two examples that are especially relevant in the context of medical domains are found in Shortreed et al. (2011) and Lizotte et al. (2008). In these studies, standard methods for missing data "imputation" have been adapted to RL problems. These methods typically involve creating a model of the data that is used to fill in missing values, often sampling several possible values from the estimated distribution. The exact choice of model will to some extent depend on

the judgment of the researchers as to the structural causes of the missing data, if any (Rubin, 1976). The missing data problem is in some ways analogous to the problem of partial observation described in Section 2.4.7, however, given the computational challenges posed by high-dimensional POMDPs, imputing missing data is arguably a more practical approach, where feasible (Shortreed et al., 2011).

For a fuller treatment of the topic, we refer the reader to a comprehensive treatment of the subject such as Molenberghs and Kenward (2007).

3.7 Summary and conclusion

Reinforcement learning methods show considerable promise as the basis for the development of adaptive treatment strategies for disease. Many of these strategies can be readily cast as RL problems. The key design decisions are the choice of reward function, state space representation, and action space representation. Good methods for batch-mode, off-policy reinforcement learning exist which can use data collected using a fixed or random policy to construct an optimized policy. Validation of such a policy can be accomplished using a variety of methods, depending on the nature of the data, the availability of a model (computational or biological), and the choice of RL algorithm. We will illustrate more of the specifics in the case studies profiled in Chapters 4, 5, and 6.

A number of challenges remain to be studied, however. Increasing the data efficiency of RL methods, especially in the case of missing or biased data, is clearly of great importance. There is a need for more practical methods for evaluation of policies derived from RL models, especially in the presence of continuous states and actions.

CHAPTER 4
Electrical stimulation for epilepsy

Epilepsy is an often devastating condition of the central nervous system. It is defined by the presence of seizures that are thought to be episodes of abnormal synchronization of neural activity. Anticonvulsant drug therapies tend to be the most common and least invasive means of treating epilepsy. However, roughly a quarter to a third of patients do not respond to drug therapies (Kwan & Brodie, 2000). For these patients, other, more radical surgical options are often attempted, such as either removal of seizure-promoting parts of the brain or separation of the brain hemispheres (Shorvon et al., 2013).

As an alternative to these therapies, slightly less invasive methods involving implantation of an electrical stimulation device have emerged in the last two decades (Rutecki, 1990). These devices, which resemble a cardiac pacemaker, are implanted in the chest with electrodes extending to either the vagus nerve or, more recently, using deep brain stimulation (DBS). These implanted devices can reduce seizure frequency or severity in many patients who do not respond to drug therapies (Uthman et al., 2004). While these devices are well tolerated in many cases, there is reason for concern that long-term electrical stimulation of the nervous system may have serious side effects.

The first generation of these devices has no means of measuring any aspect of the patient's condition, and so they operate in a fixed mode, with operating parameters adjusted based on clinical experience (Koo et al., 2001). Newer devices are able to detect the electrical activity of the patient's brain in real time, and adjust the stimulation accordingly. This is commonly referred to as "responsive stimulation" (Sun et al., 2008). To date, these devices implement a simple fixed policy

rather than an adaptive strategy, typically an algorithm which detects the onset of a seizure and immediately applies stimulation in an attempt to abort or truncate the seizure (Echauz et al., 2009; Shoeb et al., 2009; Sun et al., 2008). Devices using these algorithms are currently undergoing clinical trials (Sun et al., 2008). While it is hoped that these devices may both improve efficacy and reduce side effects, this remains an area of active research (Smith et al., 2010).

In this chapter, we discuss the use of computational modeling and reinforcement learning to optimize the stimulation patterns in electrical stimulation therapy for epilepsy. This work will build on prior work using off-line learning with *in vitro* models of epilepsy (Pineau et al., 2009; Bush & Pineau, 2009; Guez et al., 2008). The main goal of this research is to enable the development of a superior class of algorithm that can significantly improve the long-term effectiveness of electrical stimulation. This chapter is adapted from previously published work (Vincent et al., 2011).

4.1 Problem description

Many of the methods being considered for the design of adaptive strategies require a large quantity of data to support both the design and validation of potential responsive stimulation algorithms (Guez et al., 2008; Bush & Pineau, 2009). Models of epilepsy, whether biological or computational, are a potential source of data for either designing and evaluating these algorithms.

In vitro models of epilepsy using rodent brain slice preparations provide a means to study the disease in a relatively consistent manner. They involve a significant reduction in the size and complexity of the neural circuits compared to the intact brain, while maintaining most of the key electrographic features of epilepsy seen *in vivo*. One common acute *in vitro* model of epilepsy is the 4-aminopyridine (4-AP) model (Perreault & Avoli, 1991). Brain slices treated with micromolar concentrations of 4-AP exhibit epileptiform activity consisting of events analogous

to both ictal discharges and interictal spikes. An *ictal discharge* or *ictal event* is an electrographic anomaly lasting several seconds or more (and is often called a *seizure*), whereas an *interictal discharge* or *interictal spike* is typically a brief event lasting less than a second. A seizure, as defined in a clinical setting, is generally considered to be the overt manifestation of the symptoms of an ictal event. There are no symptoms generally associated with an interictal event. In the 4-AP model, ictal events may last for several tens of seconds, with interictal periods of several minutes, repeating over a period of several hours.

While *in vitro* models of epilepsy are useful for research, they nonetheless require large investments in training, equipment, and animal care. In contrast, computational models can provide large amounts of data for little cost. Experiments on computational models are by their nature perfectly reproducible, and may permit precise manipulations which may be difficult or impossible in a biological model.

To generate useful synthetic data, a computational model of epilepsy must mimic key characteristics of the corresponding biological system. One important characteristic is the timing and duration of the epileptiform events, because this likely reflects fundamental properties of the epileptic network. However, prior computational models do not exhibit spontaneous transitions between epileptiform and normal states (Netoff et al., 2004; Bazhenov et al., 2004; Cressman et al., 2009; Ullah et al., 2009; Fröhlich et al., 2010), or they exhibit only brief events that resemble interictal spikes (Traub et al., 1995; Traub et al., 2001; Franaszczuk et al., 2003). Alternatively, many existing computational models are either too abstract (Biswal & Dasgupta, 2002a; Ohayon et al., 2004; Suffczynski et al., 2004) or too computationally demanding (Traub et al., 2005) to permit the kinds of investigations needed for responsive stimulation algorithms.

The primary contribution of this chapter is a computational model of epileptiform behavior in brain slices treated with 4-AP *in vitro* (Avoli et al., 2002). We

seek to create of a computational model of epilepsy that generates data which can be used to develop and test dynamic electrical stimulation algorithms for seizure suppression. This model builds on prior work on bursting behavior in low-magnesium preparations (Golomb et al., 2006), using a simple single-compartment model for cortical pyramidal cells and fast-spiking inhibitory interneurons. Instead of using a mean-field approximation for modeling synaptic currents, we use an explicit model of individual synapses (Destexhe et al., 1998).

We further extend this prior work by incorporating a model of changes in extracellular ion (Na^+, K^+) concentration before and during seizures (Heinemann et al., 1977; Kager et al., 2000). These ion concentration changes may be an important contributor to the initiation and maintenance of ictal behavior (Fröhlich et al., 2008).

Less well understood, however, are the mechanisms which account for seizure termination (Löscher & Köhling, 2010). Our model incorporates a slow, activity-dependent reduction in synaptic efficiency, which accounts for the transition from seizure to non-seizure states. While this mechanism is too simple to enable a direct comparison, this slow depression mechanism follows a time course similar to the observed relationship between seizure termination and intracellular or extracellular acidosis (Xiong et al., 2000; Ziemann et al., 2008).

Given the goal of providing simulated data for adaptive stimulation experiments, the computational model must respond to electrical stimulation realistically. We are particularly concerned with the effects of fixed-frequency stimulation or "periodic pacing" in the 0.5-1 Hz range. Stimulation in this frequency range has been shown consistently to reduce or suppress ictal events *in vitro* (D'Arcangelo et al., 2005; Durand & Bikson, 2001). We demonstrate that our model responds to a simulated electrode input from a stimulation device with a reduction in duration and frequency of seizure-like events in a manner that resembles an *in vitro* model.

4.2 Data and modeling methods

A set of six examples of biological data were acquired using rat brain slices *in vitro*, using the protocol detailed in this section. These data were then compared with the results observed in the computational model, which combines elements from several previously published models, as described below.

4.2.1 *In vitro* data collection

Male, adult Sprague-Dawley rats (250-300 g) were decapitated under deep isoflurane anesthesia. The brain was quickly removed and placed in cold ($0 - 2^{\circ}$ C) artificial cerebro-spinal fluid (aCSF), having the following composition (mM): 124 NaCl, 2 KCl, 2 $MgSO_4$, 2 $CaCl_2$, 1.25 KH_2-PO_4, 26 $NaHCO_3$ and 10 D-glucose, continuously bubbled with gas mixture (O_2 95% and CO_2 5%) to equilibrate at pH~7.40. Combined hippocampus-entorhinal cortex (EC) slices 450 μm thick were cut as previously described (D'Arcangelo et al., 2005) using a VT1000S vibratome (Leica, Germany). Slices were then transferred to an interface recording chamber, lying between warm ($\sim 32^{\circ}$ C) aCSF and humidified gas (O_2 95% and CO_2 5%), where they were allowed to recover for at least one hour before beginning continuous bath application (1 ml/min) of 4-aminopyridine (4-AP). 4-AP is a potassium (K^+) channel blocker which increases neuronal excitability (Perreault & Avoli, 1991). Consequently, continuous perfusion of brain slices with micromolar concentrations of 4-AP leads to the generation of spontaneous epileptiform discharges resembling electrographic seizures and interictal spikes. We analyzed six slices obtained separately from six animals. All efforts were made to minimize the number of animals used and their suffering. All the procedures were carried out in accordance with the CCAC (Canadian Council for Animal Care) and McGill University guidelines.

Field potential recording was performed with aCSF-filled pipettes (tip diameter < 10 μm; resistance= 5-10 MΩ) pulled from borosilicate capillary tubing (World

Precision Instruments Inc., Sarasota, FL, USA) using a P-97 puller (Sutter Instrument, Novato, CA, USA). Extracellular signals were fed to a Cyberamp 380 amplifier (Molecular Devices, Palo Alto, CA) connected to a digital interface device (Digidata 1322A, Molecular Devices). Data were acquired at a sampling rate of 5 kHz, using the software Clampex 8.2 (Molecular Devices), stored on the hard drive and analyzed off-line. Recording electrodes were placed in the deep layers of the medial EC. Extracellular current pulses (100-250 μA, pulse width 100 μsec) were delivered in the subiculum through a bipolar concentric Pt-Ir electrode (FHC, Bowdoin, ME, USA) plugged into a high voltage stimulus isolator unit (A360, WPI Inc., Sarasota, Florida, USA) connected to the pulse generator Pulsemaster A300 (WPI Inc., Sarasota, Florida, USA).

An input/output curve was generated to adjust the stimulus intensity to reliably obtain an interictal-like event in the EC. The apparatus parameters were then fixed and the periodic pacing protocols at 0.5 Hz and 1.0 Hz were implemented. Each stimulation phase proceeded until at least 4 consecutive ictal-like discharges were observed (control) or until at least three times the previously observed interval between consecutive ictal-like discharges (stimulation) had elapsed. Each stimulation phase was immediately preceded by a control period and followed by a post-stimulation recovery period, which served as the control recording for the following stimulation protocol. Ictal discharge onset and termination were labeled in-house via visual inspection.

4.2.2 Computational model

To investigate the effects of extracellular ion concentrations on neuronal network dynamics, we used model neurons which extend standard Hodgkin-Huxley kinetics (Hodgkin & Huxley, 1952). The membrane current I_x associated with an arbitrary ion channel x is determined by equations of the general form:

$$I_x = \bar{g}_x r(V_m - E_x), \tag{4.1}$$

where $\bar{g}_x$ is the maximal conductance of the ion channel, V_m is the membrane potential, E_x is the equilibrium potential of the ion, and r is a gating variable which describes the fraction of gates which are in the open state.

A typical gating variable r follows a differential equation of the form:

$$\frac{dr}{dt} = \phi \frac{r_\infty - r}{\tau_r}, \tag{4.2}$$

where r_∞ and τ_r are the equilibrium value and time constants of the gate, respectively. Both of these are often functions of the membrane potential, V_m. The rate is modulated by the temperature correction factor, ϕ.

In some cases, the values of r_∞ and τ_r may be expressed as a function of a forward rate constant α_r and backward rate constant β_r:

$$r_\infty = \frac{\alpha_r}{\alpha_r + \beta_r} \tag{4.3}$$

and

$$\tau_r = \frac{1}{\alpha_r + \beta_r}. \tag{4.4}$$

In this case, Equation 4.2 may be rewritten as:

$$\frac{dr}{dt} = \phi(\alpha_r(1 - r) - \beta_r r). \tag{4.5}$$

The forward rate constant α_r represents the rate at which the gate is transitioning to an open state, whereas β_r represents the rate of conversion to a closed state. Both values are often functions of the membrane potential V_m, although they may also depend on other factors, such as extracellular ion concentrations.

Excitatory cell model

The excitatory cell model is a single-compartment pyramidal cell model that is relatively simple but which can be tuned to either a regular-spiking (RS) mode or to an intrinsically-bursting (IB) mode with only a small parameter change (Golomb & Amitai, 1997; Golomb et al., 2006). This property, as illustrated in Figure 4–1,

enables the computational model to reflect the observed tendency for some neurons to transition from RS to IB mode in the presence of elevated K^+ (Jensen et al., 1994). This effect manifests itself as a change in both the pattern and the average rate of firing. The effect on the mean rate is illustrated in Figure 4–2 B. These effects should tend to increase the probability of the firing of postsynaptic neurons in response to the firing of a presynaptic neuron.

The membrane potential V_m is governed by the current balance equation

$$C\frac{dV_m}{dt} = -I_{\mathrm{Na}} - I_{\mathrm{NaP}} - I_{\mathrm{Kdr}} - I_{\mathrm{Kslow}} - I_{leak} - I_{syn} + I_{app}, \tag{4.6}$$

where $C = 1\ \mu\mathrm{F/cm}^2$ is the specific capacitance of the membrane. In addition to the applied (electrode) current I_{app} and synaptic current I_{syn}, the model includes five ion currents: the fast sodium current I_{Na}, the persistent sodium current I_{NaP}, the delayed-rectifier potassium current I_{Kdr}, the slow potassium current I_{Kslow}, and a leak current I_{leak} (Golomb & Amitai, 1997; Golomb et al., 2006).

For the fast Na^+ current I_{Na}, the activation is assumed to be instantaneous, so the current is given by:

$$I_{\mathrm{Na}} = \bar{g}_{\mathrm{Na}} m_\infty^3 h (V_m - E_{\mathrm{Na}}), \tag{4.7}$$

where

$$m_\infty = \frac{1}{1 + \exp\left(\frac{-(V_m - \theta_m)}{\sigma_m}\right)}, \tag{4.8}$$

$$h_\infty = \frac{1}{1 + \exp\left(\frac{-(V_m - \theta_h)}{\sigma_h}\right)} \tag{4.9}$$

$$\tau_h = 1 + \frac{7.5}{1 + \exp\left(\frac{-(V_m - \theta_{th})}{\sigma_{th}}\right)}, \tag{4.10}$$

where E_{Na} is the calculated reversal potential of Na^+, and m and h are the activation and inactivation variables (i.e. fraction of open channels). The constants are given in Table 4–1. The relationship between the inactivation variable h, τ_h, and h_∞ is given

by Equation 4.2, with the temperature correction factor $\phi = 10$ at the simulation temperature of 37° (Golomb et al., 2006).

The persistent Na^+ current, I_{NaP}, is a non-inactivating current. It plays a critical role in the generation of intrinsic bursting behavior (Golomb et al., 2006), so it is of particular interest in our model. Again, the activation of this current is assumed to be instantaneous, and so it follows the equations:

$$I_{NaP} = \bar{g}_{NaP} p_\infty (V_m - E_{Na}), \tag{4.11}$$

where the activation variable p is given by

$$p_\infty = \frac{1}{1 + \exp\left(\frac{-(V_m - \theta_p)}{\sigma_p}\right)}, \tag{4.12}$$

where p is the activation of the channel. All other constants are given in Table 4–1.

The delayed-rectifier K^+ current, I_{Kdr}, is also a non-inactivating current, obeying the following equations:

$$I_{Kdr} = \bar{g}_{Kdr} n^4 (V_m - E_K), \tag{4.13}$$

where

$$n_\infty = \frac{1}{1 + \exp\left(\frac{-(V_m - \theta_n)}{\sigma_n}\right)} \tag{4.14}$$

$$\tau_n = 1 + \frac{5}{1 + \exp\left(\frac{-(V - \theta_{tn})}{\sigma_{tn}}\right)}, \tag{4.15}$$

where n is the activation of the channel and E_K is the calculated reversal potential of K^+. Again, all constants are given in Table 4–1.

The slow K^+ current, I_{Kslow}, is also a non-inactivating current. It represents the collective effects of potassium currents with time constants in the tens or hundreds of milliseconds, those which typically are responsible for spike rate adaptation in

regular spiking cells (Golomb et al., 2006). The kinetics are as follows:

$$I_{\text{Kslow}} = \bar{g}_{\text{Kslow}} z (V_m - E_{\text{K}}), \tag{4.16}$$

where

$$z_\infty = \frac{1}{1 + \exp\left(\frac{-(V_m - \theta_z)}{\sigma_z}\right)}, \tag{4.17}$$

$$\tau_z = 75, \tag{4.18}$$

where z is the activation of the channel; constants are specified in Table 4–1.

Finally, the leak current I_{leak} is simply:

$$I_{leak} = g_{leak}(V_m - E_{leak}), \tag{4.19}$$

where E_{leak}, the reversal potential of the leak current, is calculated from the ion concentrations.

The synaptic current I_{syn} is the sum of three individual synaptic currents:

$$I_{syn} = I_{\text{NMDA}} + I_{\text{AMPA}} + I_{\text{GABA}_\text{A}}, \tag{4.20}$$

these are further described in Section 4.2.2.

Inhibitory cell model

We model inhibitory cells using the Wang-Buzsáki model of fast-spiking (FS) interneurons (Wang & Buzsáki, 1996). The membrane potential V_m is given by the current balance equation:

$$C\frac{dV_m}{dt} = -I_{\text{Na}(i)} - I_{\text{Kdr}(i)} - I_{leak(i)} - I_{syn} + I_{app}. \tag{4.21}$$

The inhibitory cell includes only three ion currents, the fast sodium $I_{\text{Na}(i)}$, the delayed-rectifier potassium $I_{\text{Kdr}(i)}$, and a leak current $I_{leak(i)}$.

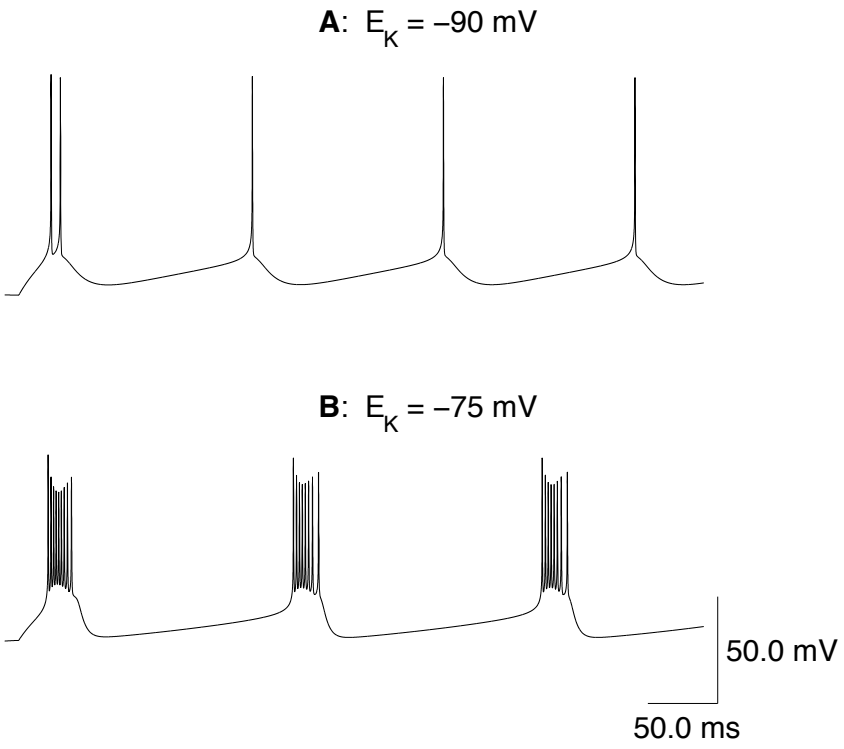

FIGURE 4–1: The pyramidal cell model proposed by Golomb and Amitai undergoes a mode change as $[K^+]_o$ increases and the potassium reversal potential depolarizes. Parameters are $\bar{g}_{\mathrm{NaP}} = 0.06$ mS/cm^2, $I_{app} = 1$ nA/cm^2. **A**: Regular spiking mode exhibited when the potassium reversal potential $E_{\mathrm{K}} = -90$ mV, the resting value. **B**: The model has transitioned to intrinsic bursting mode when $E_{\mathrm{K}} = -75$ mV.

TABLE 4–1: Constants for the model excitatory cell (Golomb et al., 2006).

Symbol	Value	Units
θ_m	-30	mV
σ_m	9.5	mV
θ_h	-45	mV
σ_h	-7	mV
θ_{th}	-40.5	mV
σ_{th}	-6	mV
θ_p	-47	mV
σ_p	3	mV
θ_n	-33	mV
σ_n	10	mV
θ_{tn}	-27	mV
σ_{tn}	-15	mV
θ_z	-39	mV
σ_z	5	mV
$\bar{g}_{\mathrm{Na}}$	35	mS/cm^2
$\bar{g}_{\mathrm{NaP}}$	0.06	mS/cm^2
$\bar{g}_{\mathrm{Kdr}}$	6	mS/cm^2
$\bar{g}_{\mathrm{Kslow}}$	1.8	mS/cm^2
g_{leak}	0.05	mS/cm^2

As in the excitatory cell model, the activation variable M of the fast sodium current has instantaneous dynamics:

$$I_{\mathrm{Na}(i)} = \bar{g}_{\mathrm{Na}(i)} M_\infty^3 H(V_m - E_{\mathrm{Na}}), \tag{4.22}$$

where

$$\alpha_M = \frac{0.1(V_m - \theta_{\alpha M})}{1 - \exp\left(\frac{-(V_m - \theta_{\alpha M})}{\sigma_{\alpha M}}\right)}, \tag{4.23}$$

$$\beta_M = 4\exp\left(\frac{-(V_m - \theta_{\beta M})}{\sigma_{\beta M}}\right), \tag{4.24}$$

$$\alpha_H = 0.07\exp\left(\frac{-(V_m - \theta_{\alpha H})}{\sigma_{\alpha H}}\right), \tag{4.25}$$

$$\beta_H = \frac{1}{1 + \exp\left(\frac{-(V_m - \theta_{\beta H})}{\sigma_{\beta H}}\right)}, \tag{4.26}$$

where all constants are given in Table 4–2.

TABLE 4–2: Constants for the model inhibitory cell (Wang & Buzsáki, 1996).

Symbol	Value	Units
$\theta_{\alpha M}$	-35	mV
$\sigma_{\alpha M}$	10	mV
$\theta_{\beta M}$	-60	mV
$\sigma_{\beta M}$	18	mV
$\theta_{\alpha H}$	-58	mV
$\sigma_{\alpha H}$	20	mV
$\theta_{\beta H}$	-28	mV
$\sigma_{\beta H}$	10	mV
$\theta_{\alpha N}$	-34	mV
$\sigma_{\alpha N}$	10	mV
$\theta_{\beta N}$	-44	mV
$\sigma_{\beta N}$	80	mV
$\bar{g}_{\mathrm{Na}(i)}$	35	mS/cm^2
$\bar{g}_{\mathrm{Kdr}(i)}$	9	mS/cm^2
$g_{leak(i)}$	0.1	mS/cm^2

The value of M_∞ is derived from α_M and β_M using Equation 4.3. The relationship among the variables H, α_H, and β_H is given by Equation 4.5, with a temperature correction factor $\phi = 5$ at the simulation temperature of 37° C.

The delayed-rectifier potassium current is also similar to that of the excitatory cell:

$$I_{\mathrm{Kdr}(i)} = \bar{g}_{\mathrm{Kdr}(i)} N^4 (V_m - E_{\mathrm{K}}), \tag{4.27}$$

where

$$\alpha_N = \frac{0.01(V_m - \theta_{\alpha N})}{\left(1 - \exp\left(\frac{-(V_m - \theta_{\alpha N})}{\sigma_{\alpha N}}\right)\right)}, \tag{4.28}$$

$$\beta_N = 0.125 \exp\left(\frac{-(V_m - \theta_{\beta N})}{\sigma_{\beta N}}\right), \tag{4.29}$$

where all constants are given in Table 4–2. The reversal potentials E_{K} and E_{Na} are calculated from the ion concentrations as described in Section 4.2.2.

While the inhibitory model neuron does not exhibit a fundamental change in behavior similar to the excitatory cell's transition from RS to IB behavior, increasing the extracellular K^+ concentration does increase the spiking rate of the neuron (See

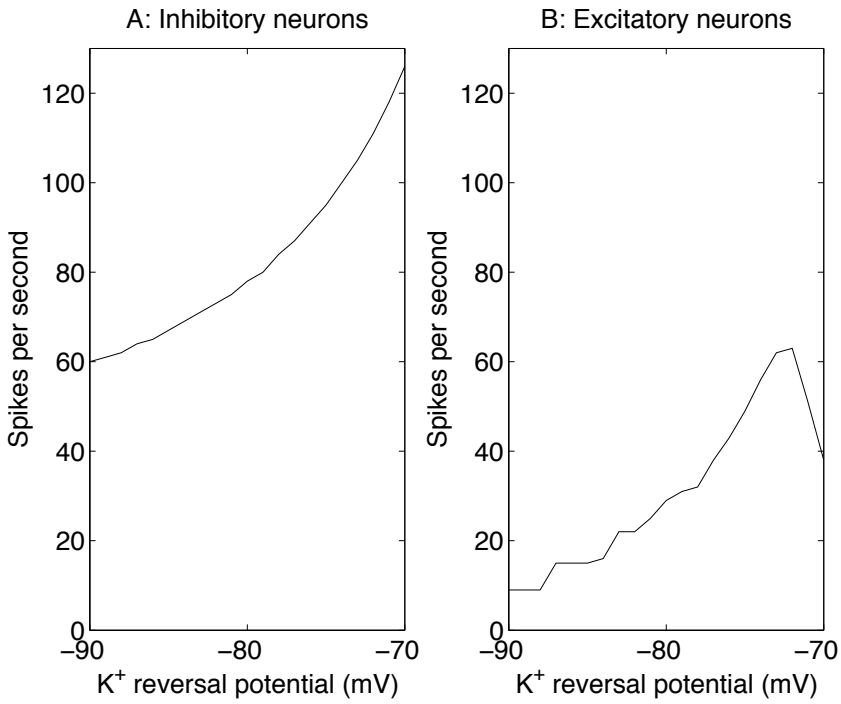

FIGURE 4–2: Spiking rates of the model neurons as a function of K^+ reversal potential. Applied current is 1 nA.

Figure 4–2 A). This effect should provide negative feedback against the greater excitability of the pyramidal cells at high values of $[K^+]_o$.

Synaptic currents

All synaptic currents are calculated according to the simplified scheme described by Destexhe et al. (1998). Each current is controlled by a gating variable which represents the fraction of open channels in the synapse. As an example, the synaptic current I_{AMPA} has the form:

$$I_{\mathrm{AMPA}} = \bar{g}_{\mathrm{AMPA}} r (V_m - E_{\mathrm{AMPA}}), \tag{4.30}$$

where $\bar{g}_{\mathrm{AMPA}}$ is the maximal conductance of the synapse, E_{AMPA} is the reversal potential of the synapse (usually 0 mV), and r is the gating variable representing the fraction of open channels. In some cases, the gating variable may be further multiplied by additional factors which influence synaptic transmission. These factors

TABLE 4–3: Rate constants for synaptic currents (Destexhe et al., 1998).

	α ($M^{-1}sec^{-1}$)	β (sec^{-1})
AMPA	1.1×10^6	190
NMDA	7.2×10^4	6.6
$GABA_A$	5×10^6	180

may include, for example, either activity-dependent depression effects or phenomena such as the magnesium block of NMDA receptors (Jahr & Stevens, 1990).

The gating variable, r, of a synaptic current is controlled by the differential equation:

$$\frac{dr}{dt} = \alpha[T](1 - r) - \beta r, \tag{4.31}$$

where $[T]$ is the concentration of neurotransmitter in the synaptic cleft and α and β are rate constant parameters chosen to fit observed synaptic activation time courses. For simplicity, the neurotransmitter concentration is modeled as a single 1 millisecond pulse with concentration 1 mM (Destexhe et al., 1998). The parameter values are given in Table 4–3. For computational efficiency, the synaptic state variables are associated with the presynaptic neuron. This choice is possible because the model neurons have a single compartment, therefore all synapses are triggered simultaneously.

The AMPA and $GABA_A$ receptor models are straightforward implementations of the simplified first-order model presented in Destexhe et al. (1998). There is evidence that increasing $[K^+]_o$ will decrease the driving force of the $GABA_A$ inhibitory effect by decreasing the efficiency of combined K^+/Cl^- transport (Jensen et al., 1993), and strong evidence that $GABA_A$ plays an important role in seizure initiation (Avoli et al., 2002). The details of these interactions remain an area of ongoing research (Rivera et al., 2004). However, modeling the complex interactions of this mechanism was not attempted in the current study, as few sufficiently detailed quantitative descriptions of these effects exist, and modeling concentrations

of $Cl^-$and Ca^{2+}would significantly increase the computational requirements of the overall model.

The model NMDA receptor is modified slightly from prior work to account for the effects of $[K^+]_o$ on the synaptic efficacy. The NMDA receptor is a slow-acting glutamatergic receptor, which has a well-known dependence on voltage and extracellular Mg^{2+} concentration (Jahr & Stevens, 1990). More relevant to the 4-AP model, there is evidence that elevated $[K^+]_o$ increases both glutamate release (Crowder et al., 1987; Fujikawa et al., 1996) and NMDA receptor activation (Poolos & Kocsis, 1990).

We use the following equation to calculate the synaptic current due to NMDA:

$$g_{\text{NMDA}} = \bar{g}_{\text{NMDA}} B(V_m, [Mg^{2+}]_o) G([K^+]_o) s \tag{4.32}$$

$$I_{\text{NMDA}} = g_{\text{NMDA}}(V_m - E_{\text{NMDA}}), \tag{4.33}$$

where $\bar{g}_{\text{NMDA}}$ is the maximal conductance of the NMDA receptor, s is the gating variable for NMDA, and $E_{\text{NMDA}} = 0$ mV. The function B accounts for the voltage-dependent magnesium block associated with the NMDA receptor (Jahr & Stevens, 1990; Dayan & Abbott, 2001):

$$B(V_m, [Mg^{2+}]_o) = \frac{1}{1 + \frac{[Mg^{2+}]_o}{3.57} \exp\left(\frac{-V_m}{16.13}\right)}. \tag{4.34}$$

The enhancement of NMDA receptor activity by $[K^+]_o$ is given by the equation:

$$G([K^+]_o) = \frac{100.0}{1.0 + \exp(-\frac{[K^+]_o - 11.75}{1.7974})}. \tag{4.35}$$

This equation and constants were chosen to approximate the behavior modeled by Kager et al. (2000).

In addition, the excitatory-excitatory synaptic currents (whether AMPA or NMDA) are moderated by two different forms of synaptic depression, one fast (D) and another slow (Q).

The fast depression variable is a phenomenological reflection of the decrease in the availability of "synaptic resources" after firing (Abbott et al., 1997; Bazhenov et al., 2004). When a neuron i fires, the value of D is instantaneously replaced according to the equation:

$$D^i_{t+1} = D^i_t(1 - \Delta_D), \tag{4.36}$$

and recovers according to the differential equation:

$$\frac{dD^i}{dt} = \frac{(1 - D^i)}{\tau_D}, \tag{4.37}$$

where $\Delta_D = 0.07$ is the fraction of synaptic resources used per firing, D^i_t is the depression factor of the ith neuron at time t, and $\tau_D = 0.7$ sec is the recovery time.

The slow depression follows the same algorithm as fast depression, but on a much longer time scale and with a slower recovery rate. It is meant to model slower phenomena that provide negative feedback and seizure termination. It has an identical form as that for the fast depression D:

$$Q^i_{t+1} = Q^i_t(1 - \Delta_Q), \tag{4.38}$$

and

$$\frac{dQ^i}{dt} = \frac{(1 - Q^i)}{\tau_Q}, \tag{4.39}$$

where Δ_Q is the fractional decrease in Q per firing, and τ_Q is the recovery time constant. This slower effect is intended to be consistent with possible mechanisms of seizure termination, such as activity-dependent acidosis (Xiong et al., 2000; Ziemann et al., 2008), for example.

The factors D and Q act to reduce the synaptic conductance in excitatory-excitatory synapses (either AMPA or NMDA). For example, for an AMPA synapse from presynaptic pyramidal cell i to postsynaptic pyramidal cell j, the full equation

for the synaptic current is:

$$I^j_{\text{AMPA}} = \bar{g}_{\text{AMPA}} r^i D^i Q^i (V^j_m - E_{\text{AMPA}}). \tag{4.40}$$

Ion concentrations

We model the changes in the extracellular ion concentrations of both Na^+ and K^+ following the formalism introduced by Kager et al. (2000).

Changes in intracellular and extracellular ion concentrations are modeled by integrating the appropriate currents over time and converting to concentration units. Therefore the change in intracellular concentration for an ion is given by:

$$\frac{d[\text{Ion}]_i}{dt} = \frac{A \sum_{\text{Ion}} I_{Ion}}{F V_i}, \tag{4.41}$$

where A is the surface area of the membrane, F is the Faraday constant, and V_i is the intracellular volume. The corresponding change in extracellular concentration is given by:

$$\frac{d[\text{Ion}]_o}{dt} = -\frac{A \sum_{\text{Ion}} I_{Ion}}{F V_o}, \tag{4.42}$$

where V_o is the volume of the extracellular space around the cell, here estimated to be $0.15 V_i$ (Kager et al., 2000).

Ion concentrations are restored primarily through the action of an active "ion pump" model which responds to elevated extracellular K^+ and intracellular Na^+ (Läuger, 1991; Kager et al., 2000).

The pump activation is given by:

$$A_{pump} = \left(1 + \frac{[K^+]_{\text{o(eq)}}}{[K^+]_{\text{o}}}\right)^{-2} \left(1 + \frac{[Na^+]_{\text{i(eq)}}}{[Na^+]_i}\right)^{-3}, \tag{4.43}$$

where $[K^+]_{\text{o(eq)}} = 3.5$ mM and $[Na^+]_{\text{i(eq)}} = 10$ mM are the equilibrium values for extracellular potassium and intracellular sodium (Kager et al., 2000).

Because the pump exchanges 2 K^+ for 3 Na^+, the current from the pump is given by:

$$I_{Kpump} = -2I_{max}A_{pump} \tag{4.44}$$

$$I_{Napump} = 3I_{max}A_{pump}, \tag{4.45}$$

where the value of I_{max} is chosen to balance the ion flux at equilibrium.

The model of the extracellular space includes a model of glial uptake (Kager et al., 2000) and diffusion between compartments (Bazhenov et al., 2004). The glial buffering system assumes a fixed reverse rate constant $k_1 = 0.015$[1] and a forward rate constant k_2 that depends on $[K^+]_o$ (Kager et al., 2000):

$$k_2 = \frac{k_1}{1 + \exp\left(\frac{[K^+]_o - 15}{-1.09}\right)}. \tag{4.46}$$

Diffusion is assumed to occur only between adjacent extracellular compartments. The resulting differential equation for $[K^+]_o$ is:

$$\frac{d[K^+]_o}{dt} = \frac{kI_K^\Sigma}{Fd} + G + \frac{\mathcal{D}}{\Delta x^2}([K^+]_{o(+)} + [K^+]_{o(-)} - 2[K^+]_o), \tag{4.47}$$

where F is the Faraday constant, $k = 10$ is a unit conversion constant (needed to convert the result into units of mM/sec), $d = 0.15\,\mu$m is the ratio of the extracellular volume to the surface area, and I_K^Σ is the sum of all K^+ currents, including the inward pump. The constant $\mathcal{D} = 2\times10^{-6}$ cm^2/s is the diffusion coefficient and $\Delta x = 100\,\mu$m is the distance between adjacent neurons. $[K^+]_{o(+)}$ and $[K^+]_{o(-)}$ are the extracellular K^+ concentrations of adjacent cells in the one-dimensional array (Bazhenov et al., 2004).

[1] This value gives substantially faster rates than used in Kager et al. (2000).

The variable G represents the kinetics of the glial buffering model. It is described by the equations:

$$G = k_1 \frac{([\mathrm{B}]_{\max} - [\mathrm{B}])}{k_{1N}} - k_2[\mathrm{K}^+]_\mathrm{o}[\mathrm{B}] \tag{4.48}$$

$$\frac{d[\mathrm{B}]}{dt} = k_1([\mathrm{B}]_{max} - [\mathrm{B}]) - k_2[\mathrm{K}^+]_\mathrm{o}[\mathrm{B}], \tag{4.49}$$

where [B] is the free buffer concentration, $[\mathrm{B}]_{\max} = 500$ mM is the maximum buffering capacity of the glial cells, and $k_{1N} = 1.1$ (Bazhenov et al., 2004).

While there is debate about the importance of these ion concentration effects, it seems reasonable to conclude that they may be a source of positive feedback that either helps initiate or sustain ictal events (Kager et al., 2000; Bazhenov et al., 2004; Fröhlich et al., 2008).

To account for other unknown mechanisms of homeostasis of the relevant ion concentrations, and to maintain stability over the long simulation periods used in this study, we introduced an additional exponential decay towards the equilibrium values for both intracellular and extracellular K^+ and Na^+ concentrations. This decay is intentionally fairly slow, with a time constant of two seconds.

Reversal potentials

In biological networks, the Na^+ and K^+ reversal potentials, E_{Na} and E_{K} depend on the relative concentration of relevant ions in the intracellular and extracellular space. This dependency is most simply expressed using the Nernst equation:

$$E = \frac{RT}{zF} \log \frac{[ion]_\mathrm{o}}{[ion]_\mathrm{i}}, \tag{4.50}$$

where R is the ideal gas constant, T is the temperature in Kelvins, F is the Faraday constant, z is the charge of the ion, and $[ion]_\mathrm{o}$ and $[ion]_\mathrm{i}$ are, respectively, the extracellular and intracellular concentrations of the ion.

Many prior computational models assume a constant value for the reversal potential. While a reasonable approximation for models of normal activity, the assumption of constant reversal potentials is not realistic in the case of epileptiform activity, given the evidence for large changes in concentration of, for example, extracellular K^+ ions observed during ictal events *in vitro* (Heinemann et al., 1977; Avoli et al., 2002).

Electrical stimulation

Electrical stimulation I_{app} is modeled as a direct input to two of the excitatory cells in the simulation. The current applied in most experiments was 0.4 nA, each pulse having a duration of 40 msec.

Network structure

Evidence suggests that epileptiform activity in the *in vitro* slice model may initiate within layer V of the entorhinal cortex (Avoli et al., 2002), therefore we attempt to model a single layer of cortical cells without additional sources of complexity or feedback.

The network is arranged in a simple linear structure of 32 units. Larger networks (on the order of 100 units) were used in some simulations, with comparable results. However, the computation time required for larger networks renders them impractical for long simulations.

Each neuron synapses onto its ten nearest neighbors. Every fourth neuron is inhibitory, giving an overall 3:1 ratio between excitatory and inhibitory neurons.

Background activity

In order to simulate the intrinsic background level of activity in the network, each model neuron is driven by a Poisson-distributed sequence of random firing events. That is, each model neuron has a small probability of firing spontaneously during each time step. In most of the simulations presented below, this firing rate was set such that each neuron fires on average once every 50 seconds. This choice

is somewhat arbitrary, but in early experiments the model was found to give similar results over a fairly wide range of values for the average firing rate.

Implementation

The software is implemented in the C programming language. Numeric integration is accomplished using a 4th-order Runge-Kutta method with a time step of 0.01 milliseconds, except for the synaptic conductances, which are integrated using the method proposed in Destexhe et al. (1998). On a 3.0 GHz Intel Xeon processor, the code requires about ten hours of real time to calculate the state of the model over 1200 seconds of simulated time. Source code for the model can be downloaded at the URL: `http://www.cs.mcgill.ca/~rvince3/ivmodel.tar.gz`.

4.3 Experiments and results

We now describe the observations made using the computational model, and compare these observations with the results of experiments performed *in vitro*. It is worth emphasizing that we use the data from the biological experiments only for comparison, not to directly fit the parameters of the model.

4.3.1 Typical behavior of the computational model

We examined the behavior of the computational model and visually compared it with typical behavior of the *in vitro* slice model.

Figure 4–3, panel B shows the behavior of the computational model compared to epileptiform activity generated by a brain slice treated with 4-AP (panel A). Note that the y axis for the biological model in Figure 4–3 represents the measured extracellular field potential, whereas for the computational model, the y axis is the simulated field potential, calculated as described in Tateno et al. (1998).

The duration of seizure-like states and the inter-seizure intervals are qualitatively similar in this example. In addition, both models show some interictal activity, although the interictal noise level in the computational model is lower. The computational model also shows a fairly clear preictal phase, with increasing brief

bursts of activity which precede the sustained ictal-like event. The distribution of interictal bursts in the biological model is somewhat less orderly, but shows a similar increase in frequency leading up to the ictal phase.

Figure 4–4 shows additional detail of the first ictal-like event generated by the computational model (Figure 4–3 B). The raster plot (Figure 4–4 A) shows the detailed firing pattern of all the neurons in the network, including the detailed firing pattern during five seconds of ictal-like activity. The activity is characterized by hypersynchronous firing at approximately 3 Hz.

4.3.2 Comparison of the distribution of state durations

To quantitatively validate the model, we compared the distribution of simulated seizure durations and inter-seizure intervals (ISI) to that observed *in vitro*. The inter-seizure interval is defined as the time elapsed from the onset of a particular seizure to the onset of the next seizure.

Data obtained from *in vitro* recordings were manually labeled, whereas for the computational model, the event durations were determined from a trace representing one hour of simulated time with $\Delta_Q = 0.00042$ and $\tau_Q = 160$ seconds. These parameter values were chosen from a range of values observed to give event timings similar to those observed *in vitro*. Ictal phases were automatically identified using an algorithmic criterion. The onset of an ictal event is declared when more than 20 units have fired within 8 consecutive 10-msec windows. The ictal event is deemed to have terminated when fewer than 10 units fire for 40 consecutive 10-msec windows.

As shown in Table 4–4, simulated seizure-like events have a mean duration similar to those observed *in vitro*. The computational model clearly exhibits less variance than is present in the biological data, but the agreement is nevertheless quite good. The estimates for the biological data are derived from the mean seizure duration and inter-seizure interval for six experiments using different animals, so the difference in variance is unsurprising.

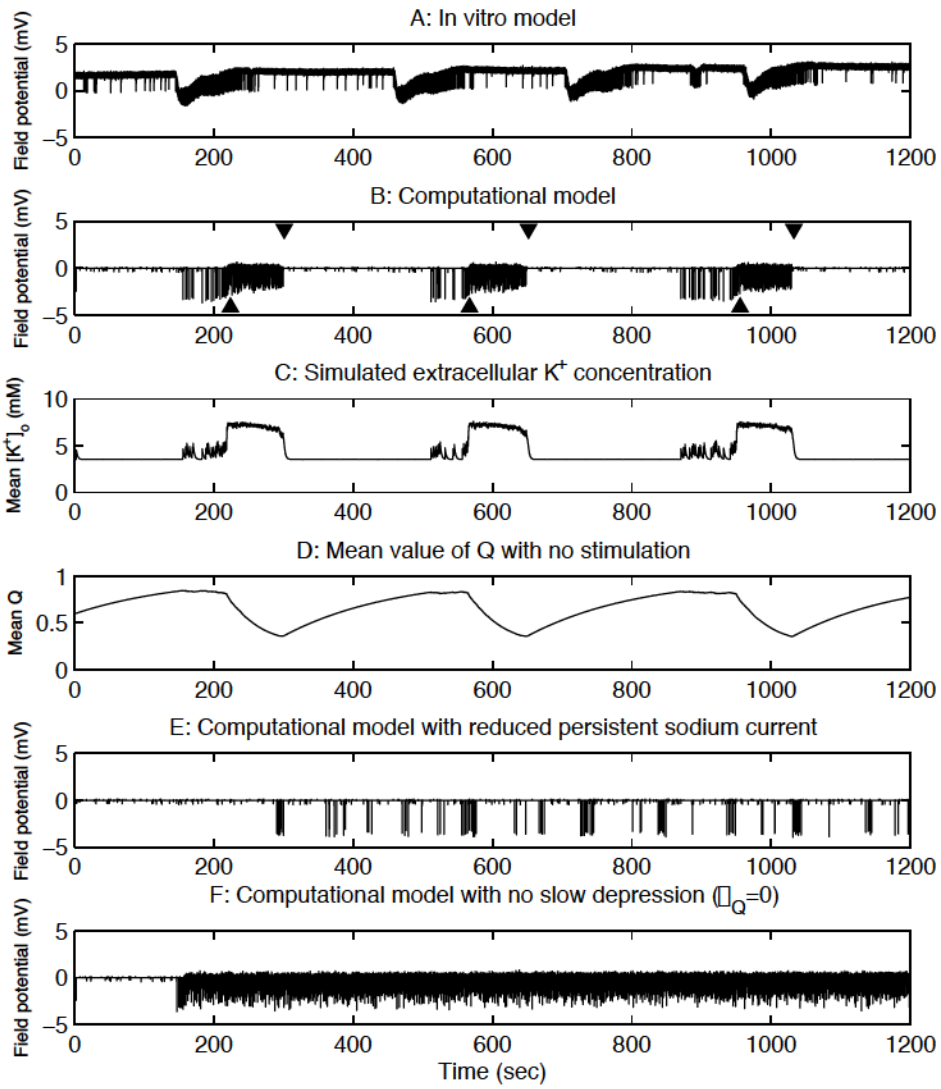

FIGURE 4–3: A: Example traces generated by the *in vitro* 4-AP biological model; B: The computational model (upward-pointing arrowheads indicate automatically calculated points of seizure onset, downward-pointing arrows indicate seizure termination); C: The mean extracellular potassium concentration of the computational model; D: The mean value of the slow depression variable Q over the same simulation period, with no stimulation and Δ_Q = 0.00042, τ_Q = 160. E: The computational model with reduced $\bar{g}_{\mathrm{NaP}}$; F: The computational model with $\Delta_Q = 0$, i.e. no slow depression;

TABLE 4–4: Comparison of mean seizure duration (SD) and inter-seizure interval (ISI) for the computational and *in vitro* models (Δ_Q = 0.00042 and τ_Q = 160).

	Mean SD	Mean ISI
In vitro model	81±18 sec	369±105 sec
Computational model	80±5 sec	357±38 sec

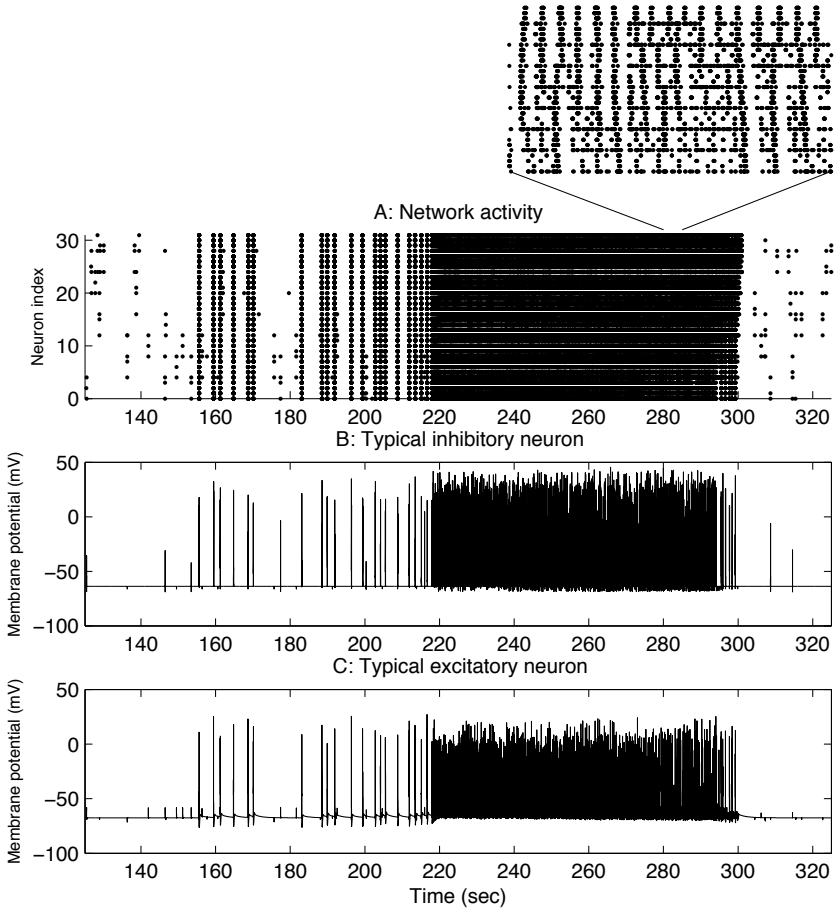

FIGURE 4–4: Detail of activity in the computational model during the first ictal-like event depicted in Figure 4–3 B. A: The raster plot shows the firing times of all units in the network (the inset shows additional detail for a typical five second interval of ictal-like activity); B: Membrane potential for one simulated inhibitory neuron; C: Membrane potential for one simulated excitatory neuron.

4.3.3 Extracellular potassium concentration

It has long been established that extracellular potassium concentration increases substantially, by as much as 400%, during seizure-like events *in vitro* (Heinemann et al., 1977; Avoli et al., 2002). This fact has contributed to speculation that extracellular potassium plays an important role in either initiating or sustaining seizure-like events.

The behavior of extracellular potassium in the computational model is illustrated in Figure 4–3, panel C. This trace corresponds to the mean $[K^+]_o$ value of the network during the same simulation period depicted in panel B. During seizure-like events, the mean value of $[K^+]_o$ increases to slightly more than twice the baseline (maximum value typically ~7.6 mM).

The observed increase in $[K^+]_o$ from 3.5 mM to 7.6 mM corresponds to a change in E_K, the K^+reversal potential, from -89 mV to -68 mV. This change is more than enough to cause the model pyramidal cells to transition from RS to IB modes, as illustrated in Figure 4–1.

The observed changes in extracellular potassium concentration in the computational model are consistent with the timing and magnitude of those previously reported in relevant biological models (Heinemann et al., 1977; Avoli et al., 2002). These values are also consistent with those reported in related computational models (Kager et al., 2000; Bazhenov et al., 2004). In all of these cases, the value of $[K^+]_o$ is observed to increase rapidly during the beginning of the ictal event, then to level off or even to decline before the network returns to the interictal state.

4.3.4 Effect of persistent sodium current

While research suggests that some cortical cells tend to transition from regular-spiking to intrinsic bursting in the presence of elevated $[K^+]_o$ (Jensen et al., 1994),

this phenomenon has not been widely examined computationally. It has been suggested (Su et al., 2001) that the persistent sodium current I_{NaP} may play an important role in the generation of intrinsic bursting. It is also known that the maximal conductance of this current mediates bursting behavior in the model pyramidal neuron (Golomb et al., 2006). To verify the importance of this mode change to the initiation and maintenance of ictal-like events, we recorded the behavior of the computational model with a reduced value for the maximal conductance, $\bar{g}_{NaP}$.

Reducing the maximal conductance of the persistent sodium current from 0.06 to 0.025 mS/cm^2 inhibits the transition from RS to IB. This completely eliminates the generation of ictal-like events in the model. Instead, the model exhibits only brief bursts of activity (Figure 4–3, panel E).

In the model pyramidal neuron, the persistent sodium current is an important enabler of the transition to bursting behavior, as illustrated in Figure 4–1. When this bursting behavior is blocked, the model is robbed of an important source of positive feedback, and epileptiform events are no longer possible.

4.3.5 Effect of slow depression on seizure termination

To verify the contribution of the slow, activity-dependent depression state variable Q to the termination of seizure-like events, we run experiments in which the decay parameter Δ_Q is set to zero. As can be seen in Figure 4–3, panel F, this causes the seizure-like events to continue indefinitely. The slow depression mechanism is clearly responsible for seizure termination in the computational model. When the mechanism is disabled, the model cannot transition from the seizure-like state to the non-seizure state.

Figure 4–3, panel D, shows the average value of the state variable Q throughout the same simulation depicted in panels B and C. The mean value of Q clearly tracks the state changes of the network. As the value increases towards a threshold (~0.8), the network becomes susceptible to an ictal event. During the ictal event,

Q quickly decays. A minimum (~0.4) is reached as the event terminates, and the value begins to recover. While this mechanism does not specifically model the effects of acidification, this general pattern is quite consistent with the time course of intracellular acidification reported by Xiong et al. (2000)

The choice of the parameters Δ_Q and τ_Q affect the mean duration of seizure-like events and the inter-seizure interval. For example, choosing $\Delta_Q = 0.00024$ and $\tau_Q = 240$ produces a model with a mean seizure duration of 165 ± 7 seconds and a mean inter-seizure interval of 500 ± 21 seconds. These values are large, but not implausible, as the maximum seizure length recorded in the control condition in our biological data was 182 seconds.

4.3.6 Response to stimulation

We further validated the computational model by comparing the effects of electrical stimulation in the computational model to the effects observed in the *in vitro* data described in Section 4.2.1.

Electrical stimulation was applied to the computational model by setting I_{app} to 0.4 nA for 40 milliseconds, simulating a square monopolar pulse applied directly to the cell membrane. The pulses are applied at 0.5 or 1.0 Hz, and the effect on the slice's ability to transition to the ictal state is quantified by calculating the fraction of time spent in seizure for each stimulation frequency.

The seizure suppression observed in each model for each stimulation frequency is similar over the reported frequencies. These results are summarized in Figure 4–5, comparing the fraction of time spent in the seizure state at a given stimulation frequency. The computational model exhibits 58% reduction of seizure-like events at 0.5 Hz, achieving complete suppression at 1.0 Hz. The biological model exhibits a mean 70% reduction in seizure-like events at 0.5 Hz, with a complete elimination of seizure-like events at 1.0 Hz.

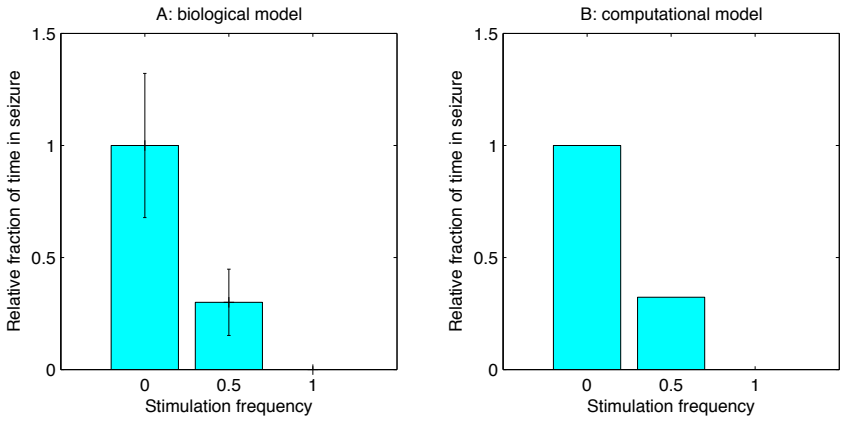

FIGURE 4–5: Effect of electrical stimulation on the relative percentage of time spent in seizure-like activity in A: the biological model and B: the computational model (Δ_Q = 0.00024, τ_Q = 240). Error bars are not shown for the computational model because the means are calculated using a single one-hour trial under each condition.

Figure 4–6 illustrates the effect of 1.0 Hz stimulation on the model. The stimulation leads to elevated $[K^+]_o$(Figure 4–6 B), which would ordinarily lead to an increase in excitability in the network. However, as seen in Figure 4–6 C, the stimulation also forces the activity-dependent slow depression variable into a near steady-state value well below that which typically precedes ictal events. The resulting reduction in synaptic efficiency is apparently sufficient to counterbalance the effects of slightly increased $[K^+]_o$.

4.4 Discussion

The computational model presented above shows many similarities to the 4-AP *in vitro* animal model. Most important is the model's bistability, which is exhibited as spontaneous transitions from the interictal to ictal states, and subsequently returning to the interictal state. As with the biological model, these transitions occur with a fairly regular temporal pattern, and with fairly consistent durations of ictal and interictal states. We can control the timing of these phases by adjusting the parameters Δ_Q and τ_Q. The reduced variance in the timing of these phases is likely due to the relative homogeneity of the computational model, with identical parameters used for all model neurons and synapses.

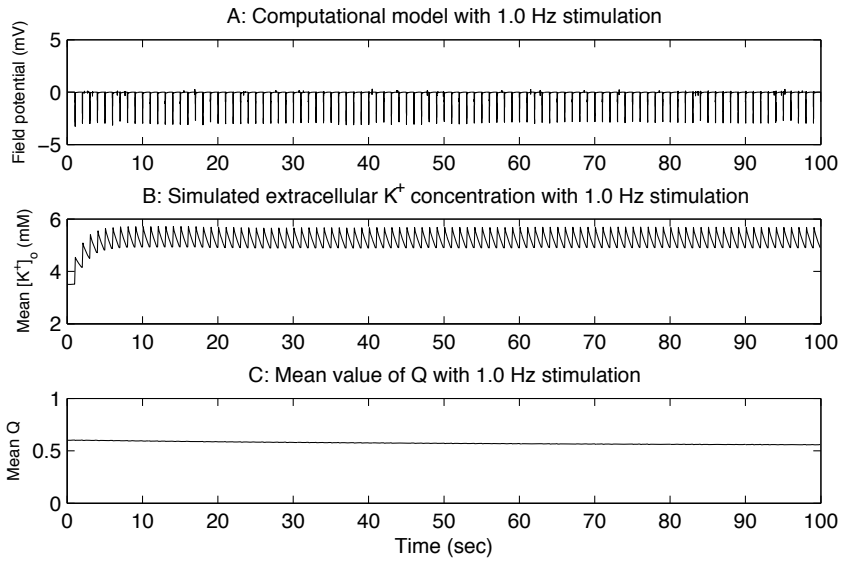

FIGURE 4–6: The computational model under the effects of stimulation at 1.0 Hz. A: The field potential trace of the model; B: The mean extracellular potassium concentration for the model under stimulation; C: The mean value of the slow depression variable Q. Only the first hundred seconds of the simulation is shown to illustrate the details of stimulation events; the model continues in this steady-state for the entire subsequent simulation.

Our model is able to mimic these statistics over realistic periods of simulated time, on the order of several tens of minutes, which allows us to measure the durations of several ictal and interictal states. However, longer simulation times impose severe restrictions on the complexity of some aspects of the model. This limits our ability to state with confidence that the computational model will reflect the response of the 4-AP model to novel stimulation patterns. Confirming this would require extensive electrophysiological work, as well as possibly mandating a more complete model of seizure initiation that includes the contributions of GABAergic receptors and the KCC2 K^+-Cl^- cotransporter (Avoli et al., 2002; Rivera et al., 2004).

Our computational model further demonstrates that a simple mechanism consisting of a slow activity-dependent decrease in synaptic transmission can account for the timing of seizure-like activity seen in the 4-AP model *in vitro*. This mechanism also provides an account for the effects of periodic pacing, as the continual

excitation of the network serves to depress the efficiency of synaptic transmission. As a result, the synaptic efficiency cannot rise above the critical level required for the onset of ictal events.

It is not clear how closely our proposed seizure termination mechanism reflects the underlying biology. There is good qualitative agreement between the biological and computational data observed. However, this remains an understudied area. A reduction in synaptic efficiency is just one of several possible mechanisms which could reduce the runaway propagation of activity in a network. It is likely that many alternative mechanisms (e.g. hyperpolarization of the cell membrane, relative changes in ion channel conductivity, recruitment of slow inhibitory factors, etc.) would have similar effects of reduced efficiency in synaptic transmission.

Our model suggests a clear role for extracellular K^+ concentration as a mechanism for the initiation, but not the termination, of ictal events. This is consistent with observations in biological models which show that $[K^+]_o$ tends to increase rapidly at ictal onset, reaching a steady-state value early in the ictal event, even tapering off slightly well before the ictal event terminates (Heinemann et al., 1977; Avoli et al., 2002). This observation implies that it is unlikely that K^+ accumulation alone could account for both seizure onset and termination, e.g. from depolarization blocking of Na^+ channels (Fröhlich et al., 2008).

Conversely, evidence from intracellular recordings of pH have shown increasing acidification during ictal events (Xiong et al., 2000), reaching a minimum pH as the seizure terminates. This argues for the plausibility of acidosis as a correlate, if not the direct cause, of seizure termination. Our slow depression mechanism, while it does not attempt to model the physiological effects of acidosis, is consistent with the observed time course of changes in pH during ictal events.

We are not aware of any work which measures extracellular K^+ concentration during electrical stimulation *in vitro*. As described in Section 4.3.6, our model

predicts that $[K^+]_o$ should be somewhat elevated during electrical stimulation, it would be interesting to investigate whether this is observed empirically.

A key problem in this area of research is fitting the many parameters used in the model. In this work, the data from the *in vitro* experiments was not directly used to tune any parameters, but only to perform the empirical comparisons. Alternatively, it may be possible to perform automatic parameter fitting in computational models such as this. However, given that these models typically include numerous of free parameters, there are a number of potential pitfalls. Most obviously, the amount of both data and computation time required to perform such a fit may be prohibitively large. There is a tremendous risk of overfitting, which may lead to poor generalization in novel experimental conditions (e.g. predicting suppression in new stimulation frequencies). There is also a real possibility of the existence of numerous local minima, which would make the resulting optimization problem especially challenging.

Ultimately, the goal of this model is to provide an alternative source of data for the evaluation of responsive stimulation algorithms. Both the computational model and the *in vitro* biological model exhibit much greater regularity in seizure timing than is typically observed *in vivo*. However, the computational model does show substantial similarities in the timing of ictal events and the response to periodic pacing. The somewhat greater prevalence of preictal spiking in the computational model is of some concern, as this may provide an artificially large signal warning of the onset of an ictal event.

4.5 Related Work

While many models of epileptiform activity have been proposed (Traub & Wong, 1981; Traub & Wong, 1982; Traub et al., 2001; Biswal & Dasgupta, 2002a; Wendling et al., 2005; Lytton & Omurtag, 2007; Wendling, 2008; Cressman et al., 2009; Ullah et al., 2009), few if any have attempted to simulate activity over the

typical duration of common *in vitro* experiments, which is measured in many minutes to hours (D'Arcangelo et al., 2005). Understandably, much of the attention has been focused on very specific issues, such as the sources of observed spectral properties of the EEG (Traub et al., 1993; Traub et al., 2001; Bazhenov et al., 2004) or the dynamics of the interictal-to-ictal transition (da Silva et al., 2003; Wendling et al., 2005; Fröhlich et al., 2010).e Relatively little attention has been paid to accounts of the important ictal-to-interictal transition, that is, the process of seizure termination (During & Spencer, 1992; Schindler et al., 2007; Ziemann et al., 2008; Krishnan & Bazhenov, 2011), whether in computational models or other settings.

Many other studies have explored the specific network characteristics which account for the difference between epileptic and non-epileptic networks, but these generally do not give an account of how the system transitions from one state to the other (Netoff et al., 2004). They also tend to give accounts of events lasting on the order of tens or hundreds of milliseconds (Lytton & Omurtag, 2007), which is not consistent with the dynamics of most observed epileptiform events either *in vivo* or *in vitro*.

One model which does capture the full behavior of a bistable network in absence epilepsy was developed by Suffczynski et al. (2004). However, this model is very different from ours in its use of a continuum (i.e. not a neural network) approach, and the very different timing characteristics of absence seizures.

A number of authors explicitly studied seizure control in computational models (Biswal & Dasgupta, 2002b; Chiu & Bardakjian, 2004; Tsakalis & Iasemidis, 2006; Schiff & Sauer, 2008; Echauz et al., 2009). However, none of the models used has been designed explicitly to mimic realistic timing of epileptiform events in biological models, something that we feel is a key issue for realistic modeling.

In other work by our lab, we have conducted experiments in reinforcement learning with real data taken from *in vitro* experiments (Guez et al., 2008; Pineau

et al., 2009; Guez, 2010; Panuccio et al., 2013). In all of these experiments, learning was performed off-line using fitted Q iteration with extremely randomized trees (Ernst et al., 2005a). Training data were recorded during experiments with rat brain tissue perfused with a convulsant drug bath. To provide training data, some phases of these experiments used a fixed stimulation pattern at various frequencies (0.2, 0.5, or 1.0 Hz, e.g.), with known anticonvulsant effects (D'Arcangelo et al., 2005). The data was subsequently hand-labeled by experts to determine whether a given time window overlapped a seizure and/or a stimulation event. These experiments generally used a state vector with 114 continuous dimensions derived from discrete Fourier transformations of the fixed-width time windows.

The action space was represented as a choice of four options, either no stimulation, or stimulation at one of the fixed frequencies used in data collection. The reward function used was of the form:

$$r_t = s_t + x_t \tag{4.51}$$

where s_t and x_t are components assessed to independently penalize both seizure or stimulation events, respectively, as follows:

$$s_t = \begin{cases} -1.0 & \text{if this window overlaps a seizure event} \\ 0.01 & \text{otherwise,} \end{cases} \tag{4.52}$$

and:

$$x_t = \begin{cases} -0.04 & \text{if this window contains a stimulation event} \\ 0.0 & \text{otherwise.} \end{cases} \tag{4.53}$$

In preliminary work (Guez et al., 2008; Pineau et al., 2009), validation was performed entirely off-line by evaluating the learned policy on a held-out data set, using rejection sampling to eliminate data windows which were inconsistent with the learned policy. These preliminary experiments showed promising results for

the policies derived from FQI when compared to fixed-frequency stimulation, in reducing both seizure activity and mean stimulation frequency.

Subsequent work using *in vitro* models to perform validation (Panuccio et al., 2013) showed generally similar results: while fixed-frequency stimulation can prevent epileptiform activity, policy derived from FQI could also prevent epileptiform activity with a reduced amount of stimulation.

4.6 Contributions

The primary contribution of this chapter is a bistable network model that roughly matches the observed statistical distribution of ictal/interictal phases in the 4-AP *in vitro* model. We show that our computational model can spontaneously transition between the ictal and interictal states, over simulations covering several tens of minutes. The model is computationally efficient (relative to other, similar models) and can generate data for many hundreds of seconds of simulated time. We also make the model available for the use of other researchers[2] .

4.7 Future Work

While our model is substantially more efficient than many other similar models, allowing simulation times of many minutes, it still requires many minutes of computer time to simulate a single minute of network activity. This makes it quite difficult to perform detailed experiments with reinforcement learning on this model. We have previously performed similar work on a much less detailed model (Vincent, 2008), but found that performing full reinforcement learning experiments with the enhanced model would be prohibitive. It would therefore be useful to develop a model with similar dynamics that could run in real time or better.

[2] Source code may be downloaded at `http://www.cs.mcgill.ca/~rvince3/ivmodel.tar.gz`

This issue could be addressed in several ways. First, it may be possible to limit the model's complexity by reducing the number of neurons. While there seems to be a general temptation to increase the size of networks in computational neuroscience experiments towards 10^3 units and beyond (Izhikevich, 2004), in our experience there seems to be good reason to believe that relatively small networks can still capture most of the relevant dynamics.

Alternatively to reducing the network size, it would be possible to use a reduced model of a neuron (Izhikevich, 2003; Rulkov, 2008) to create a much more efficient implementation of a similar model. The limitation of these "map-based" models is that they do not allow for explicit modeling of effects such as extracellular ion concentrations which likely play a role in epileptic behavior. However, it may be possible to simulate these effects through some phenomenological extension of the reduced model.

Finally, practical implementation details could be modified to make simulation more time-efficient, such as adapting the code to a parallel-processing environment. There is ample precedent for such an implementation (Traub et al., 2001; Lytton & Omurtag, 2007), but it is unclear how much of an improvement would be gained given the increase in complexity and infrastructure requirements such an implementation would require.

After the development of this model, other mechanisms of seizure termination have been proposed (Krishnan & Bazhenov, 2011; Igelström, 2013). It would be worthwhile to explore whether insights from these mechanisms could be incorporated into our model.

CHAPTER 5
Electrical stimulation for Parkinson's disease

This chapter discusses modeling and experiments using a model of synchronization of neural populations developed by Tass (2003). This model is of interest as a general model of phase resetting via electrical stimulation, which is a possible mechanism for reducing abnormally synchronized neural firing (hypersynchrony) in several diseases of the nervous system. Although this is a greatly simplified model of coupled oscillators, it presents a difficult control problem to which reinforcement learning has not been applied previously. Our contribution in exploring this model is to cast it as a reinforcement learning problem and to investigate which state features, policy classes, and learning algorithms might prove useful for achieving desynchronization with minimal cost.

5.1 Problem description

A number of neurological diseases are characterized by abnormally synchronized patterns of firing in various subpopulations of neurons. While epilepsy is one clear example, another is Parkinson's disease (PD), in which abnormal synchronization in the thalamus and basal ganglia is associated with the resting motor tremors that are a characteristic symptom of the disease (Tass, 2003).

While in early stages of the disease drug therapies remain the preferred treatment, deep brain stimulation of structures such as the subthalamic nucleus can provide relief for the symptoms of advanced PD (Tass, 2003). As is the case for epilepsy, most existing devices used for this purpose are open-loop devices which cannot implement any kind of responsive algorithm, yet there is reason to believe that a responsive device with a closed-loop stimulation approach could produce superior results with reduced side effects (Rosin et al., 2011; Beuter & Modolo, 2012).

Tass (2003) developed a simple model of synchronization of neural populations which is intended to permit detailed investigations into the application of electrical stimulation to the reduction of synchronization in a network of coupled neurons. This model treats the individual neurons as simple phase oscillators, rather than attempting to model detailed aspects of the neuron as described in Chapter 4. The desynchronization is accomplished by *phase resetting*, by using the stimulus signal to force an oscillator, or a set of oscillators, to a known phase. It has been argued that this model is also appropriate as a model for epileptiform synchronization (Echauz et al., 2009).

While a number of different control mechanisms previously have been applied to this model (Echauz et al., 2009; Tass & Majtanik, 2006; Popovych et al., 2005; Tass, 2003), none of these studies has considered reinforcement learning methods, and most have assumed that the optimization agent has an unrealistically complete amount of information about the domain. In this chapter, we investigate the possibility of applying a model-free reinforcement learning paradigm to this problem. RL methods could provide several important advantages - for example, both a principled method for adapting to a dynamically changing environment and an automatic means to discover superior control patterns.

5.2 Data and modeling

The model implements a network of N interacting phase oscillators described by the equation:

$$\dot{\psi}_j = \Omega - \frac{K}{N} \sum_{k=1}^{N} \sin\left(\psi_j - \psi_k\right) + X_j(t) S_j(\psi_j) + F_j(t), \tag{5.1}$$

where ψ_j is the phase of oscillator with index j.

The first term represents the intrinsic frequency of the oscillator, where Ω is the common eigenfrequency of all the oscillators in radians per second, i.e. $\Omega = 2\pi F$ where F is the eigenfrequency in hertz (Hz).

The second term encompasses the coupling among oscillators, where K is the global coupling strength.

The third term represents the effect of stimulation, where an external, time-varying stimulation signal $X_j(t)$ interacts with a phase-dependent sensitivity $S_J(\psi_j)$. In practice the stimulation X_j is usually implemented as a discrete signal in the set $\{0, 1, -1\}$, and the function $S_j(\psi_j)$ is of the form $I\cos\left(\psi_j\right)$ where I is the globally constant stimulation intensity.

The final term F_j represents the perturbing effects of noise, implemented as Gaussian white noise with mean zero and variance D.

The phase of each oscillator is typically initialized to uniformly random values between 0 and 2π. In the absence of stimulation ($X_j = 0 \forall j$), the oscillators will become nearly synchronized after about three seconds, as illustrated in Figure 5–1. The time to synchronization is dependent on the initialization, as well as the values of the noise variance D and the coupling strength K. This noise term and initial state are the only stochastic elements of the otherwise deterministic model.

Obviously, both the intrinsic frequency Ω and the coupling strength K could be represented as non-uniform values, or varied among some reasonable physiological continuum. While this idea is mentioned in Tass (2003), it has not been developed in most of the existing literature on this model.

By convention, each oscillator is considered to have fired at an arbitrary point in phase space. This assumption gives the following equation which calculates the mean firing rate for a given population:

$$n_{fire}(t) = \frac{|\{j : \cos(\psi_j) > 0.99\}|}{N}. \tag{5.2}$$

The value of $n_{fire}(t)$ is at its maximum when the entire subpopulation fires at once, i.e., in pathological situations. However, a more general measure is used to calculate the degree of phase coherence in m arbitrary subpopulations (or clusters) of

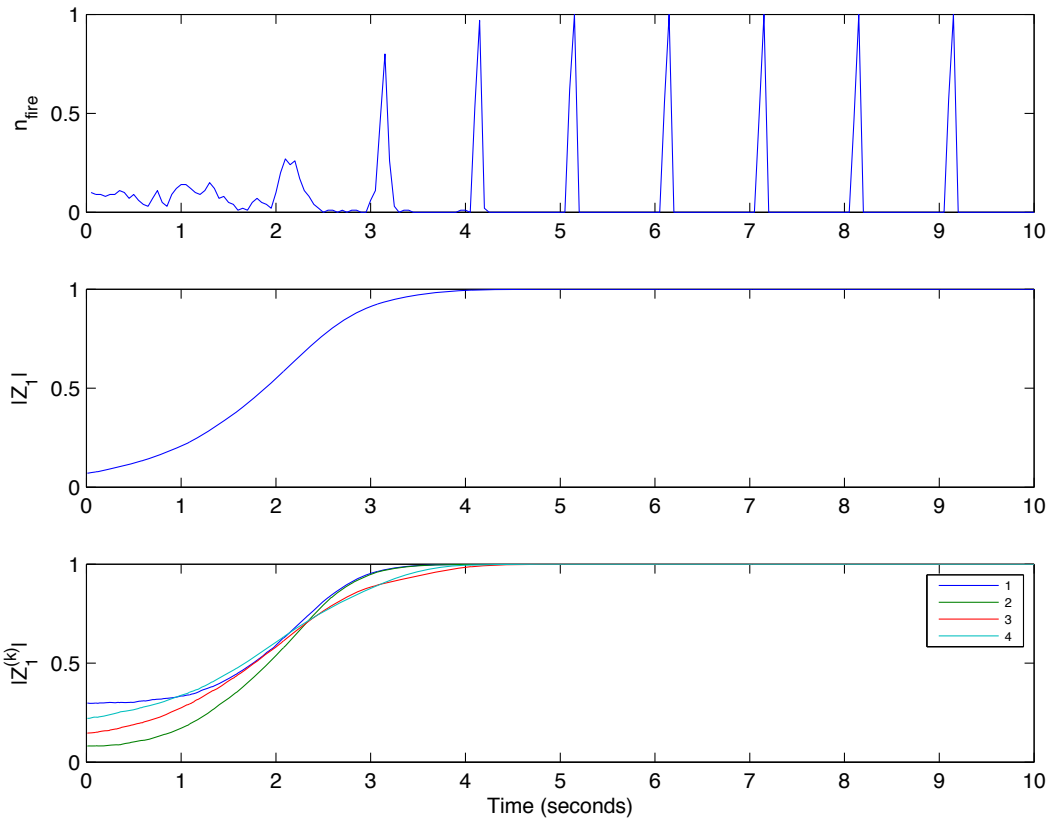

FIGURE 5–1: The standard behavior of the Tass model ($F = 1.0$ Hz). After starting in an initial incoherent state, the oscillators quickly synchronize. In the top trace, nearly 100% of the oscillators are firing simultaneously after 3-4 seconds. The amplitude of the cluster variable Z_1 climbs to a value near 1.0, indicating almost perfect synchrony. The cluster variables for the individual quadrants are also indicative of a high level of synchrony.

oscillators at each time step:

$$Z_m(t) = R_m(t)e^{i\phi_m(t)} = \frac{1}{N}\sum_{j=1}^{N} e^{im\psi_j(t)}, \tag{5.3}$$

where the value $Z_m(t)$ is termed the "cluster variable" and $R_m(t)$ and $\phi_m(t)$ are the real amplitude and phase of $Z_m(t)$ (Tass, 2003). A value of $|Z_1| = 1$ implies that every oscillator is in perfect synchrony (the entire system is acting as one coherent cluster), while a value of $|Z_m| = 0, \forall m \in \{1, 2, 3, ...\}$ describes a completely incoherent state (there are no groups of synchronized clusters in the system). Still following the notation of Tass (2003), we divide the network into four equal subpopulations or quadrants which can be stimulated independently. We observe that we can then specify the cluster variable for a specific subpopulation Λ_k by writing:

$$Z_m^{(k)}(t) = R_m^{(k)}(t)e^{i\phi_m^{(k)}(t)} = \frac{1}{|\Lambda_k|}\sum_{j\in\Lambda_k} e^{im\psi_j(t)}. \tag{5.4}$$

Since in practice these subpopulations are quadrants consisting of exactly $N/4$ adjacent oscillators, we measure the presence of global synchrony within quadrant $k = 1$ with the simplified equation:

$$Z_1^{(1)}(t) = \frac{4}{N}\sum_{j=1}^{N/4} e^{i\psi_j(t)}, \tag{5.5}$$

extending this analogously to the other three quadrants by choosing appropriate ranges for the summation (Tass, 2003). The magnitude of $Z_1^{(1)}$ can be thought of as a measure of the phase synchronization over the complete subpopulation of quadrant 1. In contrast, the value Z_4 yields the synchrony measured over an arbitrary set of four clusters of the network.

In Tass (2003), several stimulation approaches are described, but only one of these is fully developed. This stimulation paradigm consists of sets of 15 pulses delivered at 20 Hz with a 40% duty cycle (20 milliseconds on, 30 milliseconds off). The stimulation is applied to each of four equal subpopulations, or quadrants, of

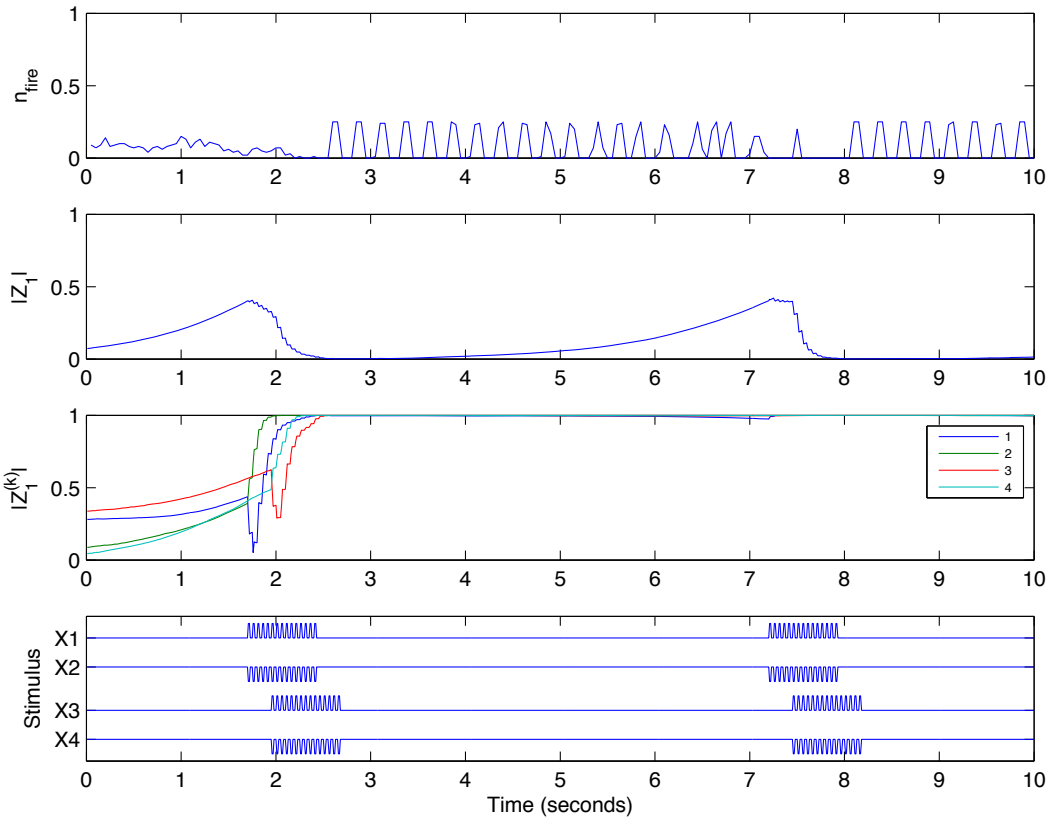

FIGURE 5–2: The Tass model with the baseline responsive control model ($F = 1.0$ Hz). A fixed stimulation pattern is triggered whenever the magnitude of Z_1 rises above 0.4. Note that the positions of stimulus signals $X1$-$X4$ along the Y-axis are offset for clarity. Note also that the cluster variables for the quadrants, $Z_1^{(k)}$, remain close to 1.0, indicating that the individual quadrants are highly synchronized.

the network. In this reported case, each pair of populations is stimulated using an identical signal of opposite polarity, with a time offset of 0.25 seconds between the two pairs. This is illustrated in Figure 5–2. The simplest version of this policy is to trigger the full train of pulses whenever the value of R_1 or $|Z_1|$ is greater than some threshold, which we label Z_{thresh}. We adopt this as our "baseline policy" for comparison, and we use these same tools in designing the policy choices available to the reinforcement learning agent.

Because the stimulation is uniform across a quadrant, desynchronization is not total, but rather consists of moving the four different subpopulations into different parts of the phase space.

Most prior work does not explicitly attempt to model the problem as the minimization of a cost function. It is therefore difficult to exactly compare our work to prior efforts. To address this, we have used our proposed cost function, Equation 5.9, to calculate the costs associated with our baseline policy for a range of parameter values. This allows us to explore, for example, how the baseline policy compares with a policy derived from a reinforcement learning algorithm, when simulation parameters vary from those originally used in Tass (2003).

5.3 Experiments

We performed a number of experiments to validate possible approaches to applying reinforcement learning on the model. Our goal with these experiments was to explore which of the RL methods would yield the best expected reward, and to examine how the RL results would compare to the baseline policy.

Except where otherwise noted, the parameter values used for the model are as follows: $\Omega = 2\pi F$ (where $F = 1.0$ Hz), $I = 30$, $K = 2$, $N = 100$, and $D = 0.4$. These values are taken directly from Tass (2003). To better validate our implementation of the model, we ran a few simulations with $N = 1000$ and observed that the dynamics remain essentially unchanged. We implemented the model using the simple Euler method for integration with a time step of 0.01 seconds. For validation purposes we experimented with smaller time steps but found no major change in the dynamics with a time step of e.g. 0.002 seconds.

In practice, we combine five integration steps of 0.01 seconds (10 milliseconds) into a single RL time step of 0.05 seconds. This is motivated by our choice of action space, as well as performance considerations. Each RL episode extends over ten seconds, yielding a total of 200 observations. We chose the 10-second episode length as being sufficient to guarantee that the uncontrolled system would typically

reach the hypersynchronous state[1] for the majority of the time, while remaining brief enough to keep computation times manageable.

We then formalize the problem in the reinforcement learning paradigm by defining a reward function, a state representation, and a set of actions, and finally, because this is a continuous state space problem, we must choose a function approximation method.

5.3.1 Reward function

The reward function design required some careful consideration. Clearly it is undesirable to have many simultaneously firing neurons, so it seems logical to create a penalty associated with excessive firing or synchrony. Because we do not wish to stimulate more than necessary, it also seems reasonable to penalize use of the stimulator. We define a value $n_{stim}(t)$ which reflects the fraction of stimulated nodes at a given time t:

$$n_{stim}(t) = \frac{|\{j : X_j(t) \neq 0\}|}{N}, \tag{5.6}$$

Preliminary experiments were performed using a reward function of the form:

$$R(t) = -n_{stim}(t) - c_s(t), \tag{5.7}$$

where $c_s(t)$ is the cost assessed to penalize synchrony in the system, and is therefore defined as:

$$c_s(t) = \begin{cases} P_{sync} & \text{if } n_{fire}(t) \geq \frac{1}{3} \\ 0 & \text{otherwise} \end{cases} \tag{5.8}$$

where P_{sync} is a constant (generally $P_{sync} = 100$).

Upon inspection of the results from these early experiments, we discovered that the optimal policy under this reward function includes situations in which *all*

[1] Where hypersynchrony is defined as the simultaneous firing of a large fraction of the oscillators.

firing is suppressed. Clearly this is neither a physiologically realistic nor a desirable outcome. We therefore expanded the reward function to include a term that gave some small positive reward for firing:

$$R(t) = -n_{stim}(t) - c_s(t) + n_{fire}(t). \tag{5.9}$$

Note that the structure of this reward function is based on the structure described in Section 3.4.4, especially Equation 3.9, where P_{sync} here is a coefficient chosen to penalize the detection of pathological symptoms. The other two terms are in the range [0, 1], so in effect we are judging hypersynchronous behavior to be P_{sync} times more "costly" to the system than stimulation, and beneficial firing is weighted just high enough to counteract some effects of the stimulation. Obviously these weights could be adjusted to achieve different behaviors with different therapeutic tradeoffs.

We initially chose to use n_{fire} in the reward function, and continued to perform experiments using this approach. It arguably would have been valid to use one of the global synchrony measures, e.g. $|Z_1|$, as an alternative. We would argue that the value of n_{fire} is probably a closer approximation of the structure of a typical electrophysiological signal, because it mimics the field potential recorded by most electrodes. Therefore it would make a somewhat more realistic analogue to a possible observation in a biological model.

5.3.2 State representation

For the state representation, it is possible to represent the full N-dimensional raw state of the oscillators. Clearly this is a difficult state to summarize using standard function approximation methods, as few function approximators are designed to work well with such high-dimensional spaces.

In this domain, the key features consist of the correlations among groups of state variables, rather than the absolute values of the individual state variables. Therefore, it seems reasonable to include either the values of Z_1 and/or R_1 in the

state representation. These values neatly summarize the state of the entire system by its most salient quality, synchronization. Given the division of the system into four quadrants, it may also be reasonable to include the Z or R values of the individual quadrants (referred to as $Z_1^{(1)}$ through $Z_1^{(4)}$ in Tass (2003). These simplifications have the drawback that in the strictest sense it changes the underlying MDP from a fully-observable to a partially-observable MDP, since we do not treat the actual oscillator phases themselves as observations. We would argue that this is not a major concern for this paradigm, as it seems clear that the individual oscillator phase variables will not be readily observable in any real physical setting, and they are not likely to serve as informative individual state features. In contrast, the cluster variable and its relatives are likely to be sufficient to describe the system in practical terms, and they may serve as realistic proxies for actual electrophysiological measurements.

Another potentially useful state feature might be the time derivative of the amplitude of the cluster variable, $\frac{d|Z_1|}{dt}$. This would provide information about whether the model was currently experiencing an increase or decrease in synchrony. In a sense this can also be considered a change from a simple first-order Markov process to a second-order Markov process, because we now incorporate information about the prior state in the current state. This change is a fairly common trick in many RL domains, for example see the "mountain-car" example in Sutton (1996).

Finally, because of the periodic nature of the domain and the known baseline policy, we include the time since the last stimulation event was applied to a specific quadrant. We denote these as $t_s^{(1)}$ through $t_s^{(4)}$. These features could presumably contribute to the ability of the Q-function to fully represent a policy with appropriate periodicity.

We performed experiments on several state vectors:

S_1 - The unidimensional state consisting of $\{|Z_1|\}$.

S_2 - The two-dimensional state consisting of $\left\{|Z_1|, \frac{d|Z_1|}{dt}\right\}$.

S_6 - The 6-dimensional state consisting of $\left\{|Z_1|, \frac{d|Z_1|}{dt}, |Z_1^{(1)}|, |Z_1^{(2)}|, |Z_1^{(3)}|, |Z_1^{(4)}|\right\}$.

S_9 - The 9-dimensional state consisting of $\left\{|Z_1|, t_s^{(1)}, t_s^{(2)}, t_s^{(3)}, t_s^{(4)}, |Z_1^{(1)}|, ..., |Z_1^{(4)}|\right\}$.

S_{10} - The 10-dimensional state consisting of $\left\{|Z_1|, \frac{d|Z_1|}{dt}, t_s^{(1)}, ..., t_s^{(4)}, |Z_1^{(1)}|, ..., |Z_1^{(4)}|\right\}$.

While the observable state vectors are as described above, the underlying state variables ψ_j were initialized to uniformly random values between 0 and 2π. This randomization was performed on each trial, which increases the difficulty of the RL problem compared with problems with a "natural" unique initial state.

5.3.3 Action space

Modeling the approach in the baseline policy (Tass, 2003), our action space consists of a choice to stimulate (or not) in one of the four quadrants in the model at a given time step. To simplify the action space yet allow the agent to mimic the baseline policy, we enforce the rule that stimulus pulses delivered in quadrants 1 and 3 always have positive voltage ($X_j = 1$), whereas pulses applied to quadrants 2 and 4 always have negative voltage ($X_j = -1$), although both use the same parameter intensity parameter I (Equation 5.1). The pulse is always delivered with a 40% duty cycle. Given our grouping of five integration steps into a single RL step, X_j is non-zero for only the first two integration steps, (20 milliseconds), then zero for the subsequent three integration steps (30 milliseconds). Given that there are four quadrants, this approach yields 2^4 or 16 possible discrete actions.

It is possible to reduce the action space to only four (or even three) choices by limiting the selections to those used in the baseline policy. In this case, a choice to stimulate in quadrant 1 implies a stimulation event in quadrant 2, and likewise for quadrants 3 and 4. In this case the four choices consist of: No stimulus, stimulus in quadrants 1 and 2, stimulus in quadrants 3 and 4, or stimulus in all four quadrants.

Finally, we also explored the possibility of using an "extended action", which exactly replicates the complete stimulus pattern defined by Tass (2003). In this case,

the action is defined as a simple Boolean value which indicates whether the agent chooses to initiate stimulation. Once the stimulation is initiated, action selection has no effect until the stimulation pattern completes. The reward is assessed at each time step according to Equation 5.9.

For clarity and brevity, we adopt the following shorthand names for these action spaces:

A_{16} - The space of all 16 possible actions.

A_4 - The space of 4 restricted actions.

A_X - The extended action only.

5.3.4 Experiments with Sarsa-λ

For experiments with Sarsa-λ, we used a range of parameters for both the environment and the learning algorithm. We attempted to hand-tune these parameters to optimize the results as much as possible.

We also explored each of the various state space dimensionalities discussed previously. In particular, we have explored the use of S_1, S_2, S_6, and S_9. We did not emphasize S_{10} because of evidence from the FQI experiments (Section 5.4.3) that suggest that the gains between S_9 and S_{10} are likely to be small. We performed experiments with only the A_X and A_4 action spaces, as the results with A_4 did not justify attempting to learn using the larger action space A_{16}.

Experiments were structured to perform 1000 episodes of learning, with each episode consisting of the standard 200 time steps. These learning intervals are interleaved with 100 steps of evaluation with no learning and greedy action selection (by temporarily setting both α and ϵ to zero). Typically the experiments were run for ten rounds, giving a total of 10000 learning episodes. Results are reported by giving the mean and standard deviation of the return observed over the evaluation intervals.

While many parameter values were tried, in most of the reported experiments we used 10 levels of discretization along each dimension, with $\alpha = 0.1$, $\lambda = 0.9$, and $\gamma = 0.98$.

5.3.5 Experiments with kNN-TD(λ)

We performed some preliminary experiments with kNN-TD(λ) using similar state and action spaces as described in the previous section, and with an identical structure using ten rounds of interleaved learning and evaluation phases.

There are some special limitations which affected the results achievable with kNN-TD(λ). kNN-TD(λ) is not practical with the 9- or 10-dimensional state vectors (S_9 and S_{10}) given a naïve discretization of the state space. A discretization of each dimension into 10 steps will yield an impractically large set of sample points. Optimizing the selection of sample points would involve either a tremendous amount of trial and error, or substantial prior knowledge of the structure of the problem.

We did attempt to incorporate some insights gleaned from the FQI experiments (Section 5.4.3) into our approach to discretizing the state space, and we report these results in the subsequent sections. In particular, we used the insight that the time dimensions ($t_s^{(k)}$), while important to the overall return of the learned policy in FQI experiments, were not sampled as often as the other dimensions. Conjecturing that this might mean that these dimensions can be adequately represented with coarser discretization, we reduced the discretization along these four dimensions to constrain the state grid to a manageable size. Rather than 10 discretization levels, as used in the Sarsa-λ experiments, we used 5 levels of discretization in most dimensions, but only 3 discretization levels in the time dimensions. For S_9, this yields a grid with 253125 points.

5.3.6 Experiments with FQI

We performed a number of experiments with FQI, modeled roughly on the approaches taken in Ernst et al. (2006) and Guez et al. (2008).

These experiments proceed in a series of rounds, as described in Section 2.5.3. At the beginning of each round, the current policy is used to generate a set of simulated trajectories consisting of tuples with the form $\langle s_t, a_t, r_t, s_{t+1} \rangle$. For each round, the policy learned in the previous round is used to select the greedy action with probability 0.85. With probability 0.15, an action is selected at random. Before the first round of training, the initial policy will simply select an action uniformly randomly. Once the trajectories are generated using these policies, the FQI algorithm is then run on all the data generated so far in the process, and a new policy is derived from the complete set of data.

In most of our experiments we used a series of exactly 10 rounds consisting of 60 episodes of 200 steps each, giving a total of 12000 tuples per round and 120000 total tuples for the final policy. The FQI algorithm was typically run for 100 iterations. However, after the first 75 iterations, the tree structures are fixed, and instead simply recalculate the output values at each node. This is done to ensure convergence (Ernst et al., 2005a).

We performed experiments with action spaces A_4 and A_{16}, as with the previous algorithms. We did not consider the "extended action" approach A_X in the FQI setting because of two considerations: practical implementation issues, because of the completely different software framework and programming language used in the FQI experiments, and second, because unlike the prior experiments, we were able to achieve good results without resorting to the extended action.

5.4 Results

As previously discussed in Section 5.3, most of our simulations were run for a simulated time of ten seconds with a network of 100 oscillators, with an RL time step of 50 milliseconds. Therefore, the reward and action selection are observed 200 times in the simulated ten second interval. We set the discount factor $\gamma = 0.98$ in each experiment, which implies that rewards anticipated after one second (20

TABLE 5–1: Observed total reward for "responsive" stimulation using the baseline policy, in which application of two time-shifted antiphase resets of pairs of subpopulations is triggered when $|Z_1| > Z_{thresh}$. Total discounted reward R_T is calculated according to Equations 5.9 and 2.1 and is averaged over 100 runs beginning at different, randomized starting states. Results include 95% confidence intervals.

Z_{thresh}	R_T $F = 1.0$ Hz	R_T $F = 1.2$ Hz	R_T $F = 1.4$ Hz	R_T $F = 1.6$ Hz
0.10	−10.426 ± 0.773	−11.113 ± 1.151	−15.457 ± 2.263	−125.962 ± 7.330
0.20	−6.193 ± 0.766	−6.917 ± 0.793	−10.937 ± 2.074	−89.874 ± 6.527
0.30	−4.820 ± 0.671	−4.777 ± 0.566	−6.599 ± 1.029	−79.238 ± 5.704
0.40	−4.064 ± 0.727	−3.627 ± 0.524	**−4.527 ± 0.830**	−66.415 ± 5.230
0.50	**−2.933 ± 0.471**	**−2.865 ± 0.710**	−4.966 ± 1.294	**−61.282 ± 5.384**
0.60	−4.033 ± 1.574	−6.543 ± 2.453	−12.511 ± 3.761	−64.588 ± 5.837
0.70	−12.884 ± 4.289	−17.571 ± 4.960	−30.737 ± 5.472	−87.120 ± 7.217
0.80	−40.105 ± 6.099	−47.521 ± 5.678	−57.770 ± 5.404	−114.203 ± 7.385
0.90	−62.382 ± 5.533	−80.110 ± 6.470	−93.042 ± 7.267	−141.644 ± 9.784

time steps) are discounted to about 2/3 the value of rewards in the immediate time step. This seems to provide a reasonable trade-off between the short and long term efficacy of the stimulation.

In the case of the model with no stimulation, the model will become nearly synchronized after roughly three seconds, yielding an average of seven hypersynchronized firing events over the ten-second run. Taking the mean over 100 trials with no stimulation gives an estimate of the expected reward of −168.5.

By comparison, using the baseline stimulus algorithm, triggered at various values of $|Z_1|$), yields the results reported in Table 5–1 for $\gamma = 0.98$.

It is clear that there is little difference in the total reward (cost) of the experiment when stimulus is triggered anywhere in the range 0.2-0.6. However, excessively high values of the threshold ($Z_{thresh} > 0.6$) tend to under-stimulate the system, resulting in an increase in hypersynchronous activity, whereas excessively low values ($Z_{thresh} < 0.2$) incur a penalty for over-stimulation. The optimal value, however, clearly shifts depending on the resonant frequency of the oscillators.

However, as the default frequency of the system increases to 1.6 Hz, there is evidence that this baseline control method begins to lose its efficacy, yielding substantially lower reward values.

5.4.1 Results with Sarsa-λ

While we found consistent evidence for some optimization when using large state and action spaces (e.g., S_9 and A_4), we found that Sarsa-λ was not able to learn a policy which was competitive with the baseline policy. This was true for a wide range of parameters of the function approximator (number of tilings, discretization steps, and number of features) as well as different choices of state space, action space, and learning rate parameters such as λ or α. The best observed return, -33.6, was observed using state vector S_6 with $\alpha = 0.1$ and $\lambda = 0.9$ with a single tiling. However, many other parameter choices gave results that were within the margin of error. These results are summarized in Figure 5–3.

Sarsa-λ was able to find a good policy in the extended action case (A_X). With the simplified action space, the agent is able to find a policy that roughly equals the performance of the baseline policy. However, this is a fairly simple problem, and the agent learns to reach a value of $R_T \sim -6.0$, near that of the baseline policy, -2.93, after only two rounds. The return of the RL policy after ten rounds, $R_T = -3.17$, is within the confidence interval of the return for the baseline policy. These results are illustrated in Figure 5–4.

5.4.2 Results with kNN-TD(λ)

As previously discussed, it is not practical to use kNN-TD(λ) with large state vectors (S_9 or S_{10}) with the same naïve discretization used in the Sarsa-λ experiments. Instead, we used a coarser discretization with only 5 divisions along cluster variable dimensions (Z_1, $Z_1^{(k)}$, etc.) and 3 divisions along the time ($t_s^{(k)}$) dimensions. Despite this coarser discretization, kNN-TD(λ) was able to find policies which performed better than Sarsa-λ, even when using state vector S_9 with single nearest

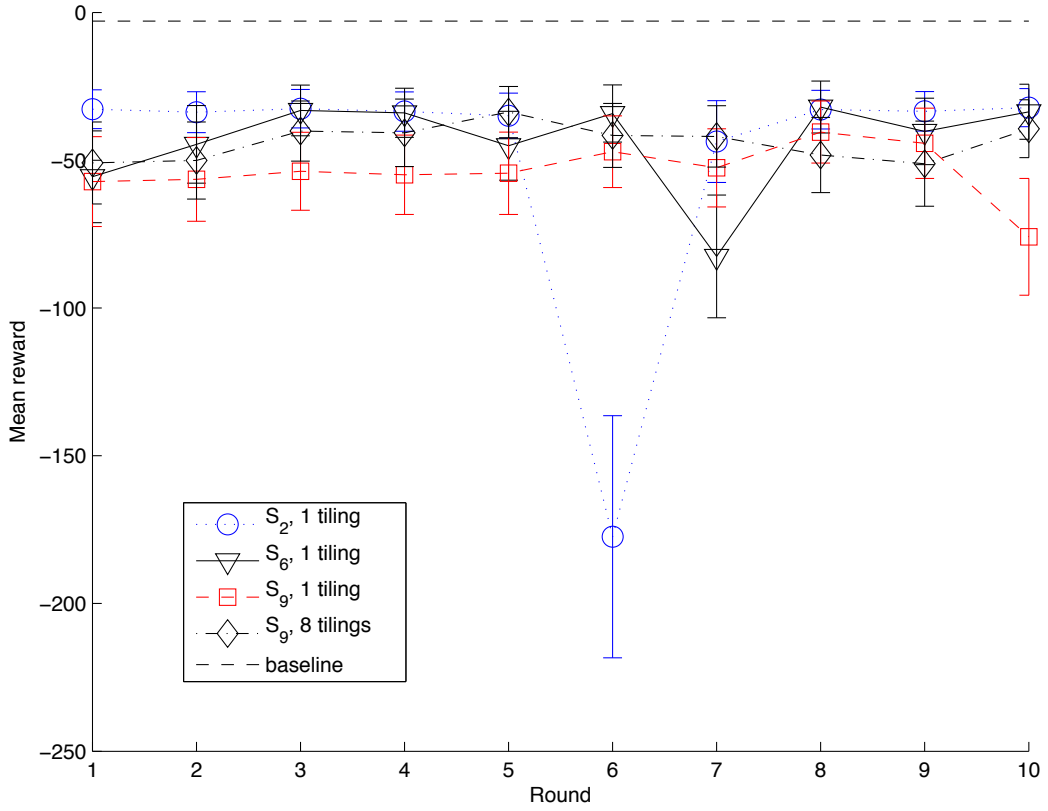

FIGURE 5–3: Comparative results for Sarsa-λ with different choices for the dimensions of the state vector, when used with action space A_4. None of the methods attempted could approach the performance of the baseline policy. The baseline return is -2.93, taken from Table 5–1.

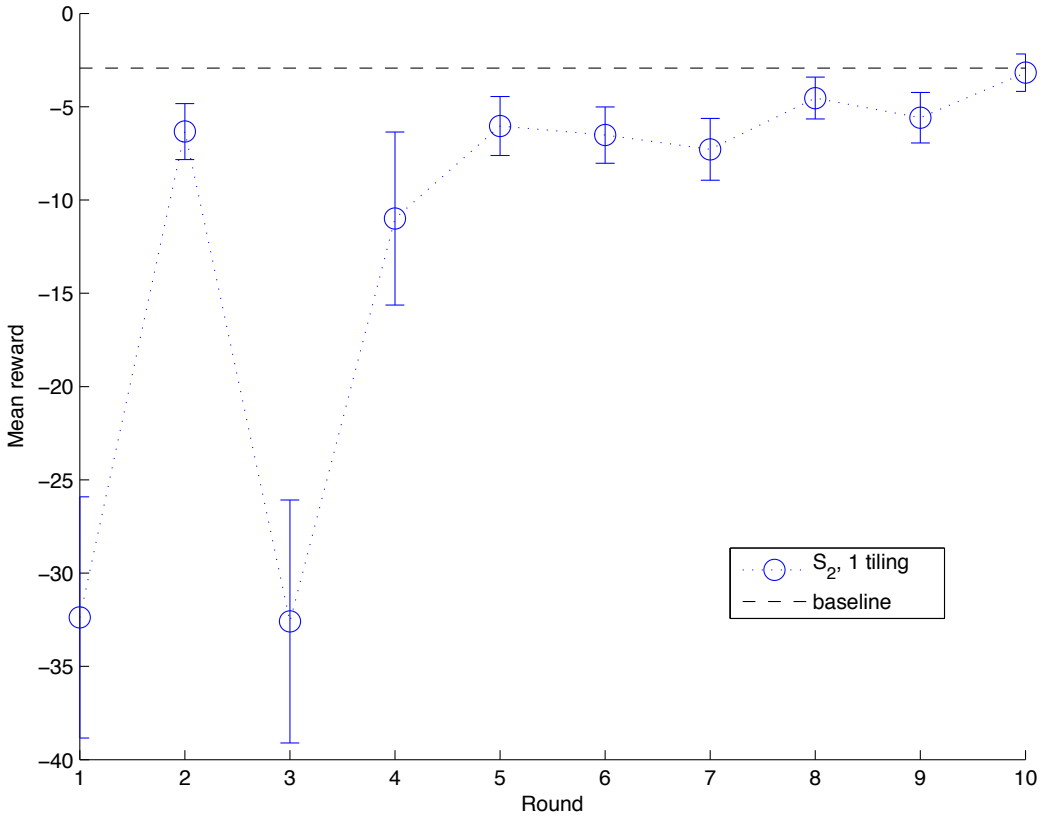

FIGURE 5–4: Example results for Sarsa-λ with state vector S_2 and action space A_X. The baseline return is -2.93, taken from Table 5–1.

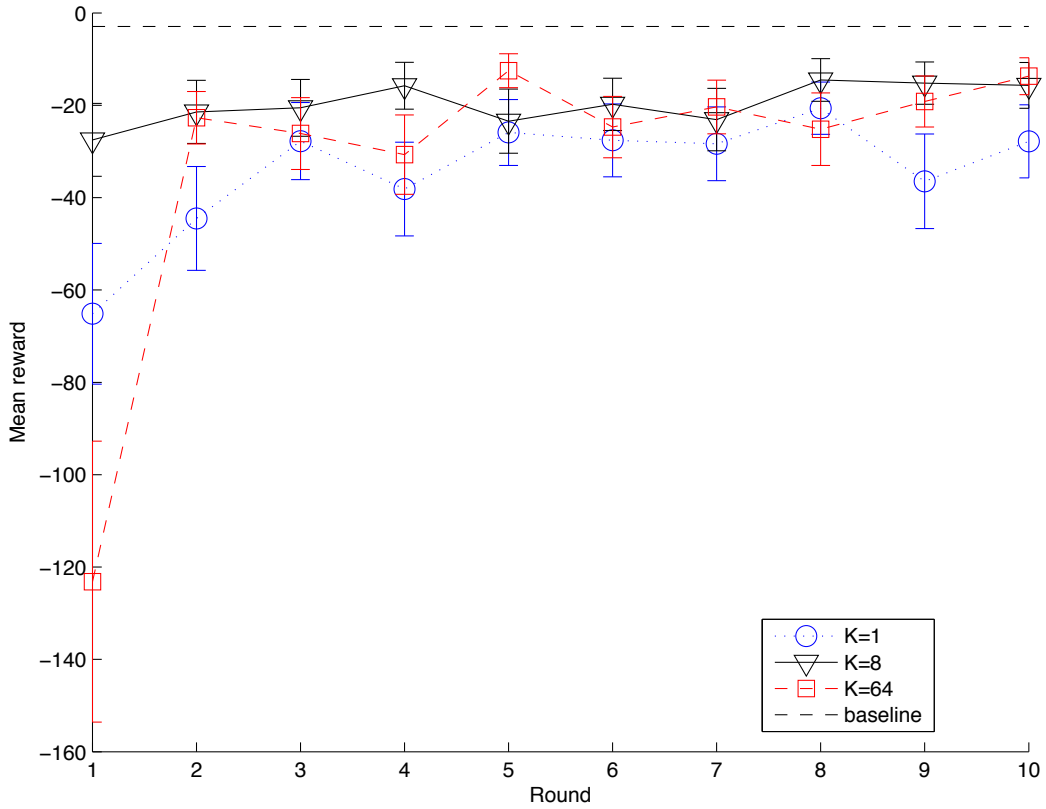

FIGURE 5–5: Example results for kNN-TD(λ) with state vector S_2 and action space A_4, while varying values of k, the number of nearest neighbors. The baseline return is -2.93, taken from Table 5–1.

neighbor $k = 1$. For example, the best R_T with Sarsa-λ was -33.6, whereas the best R_T for kNN-TD(λ) was -27.8 with $k = 1$ and -13.7 with $k = 64$. These results are summarized in Figure 5–5.

We did find that this agent, like Sarsa-λ, is able to learn a good policy quickly when using action space A_X, often in just a few thousand iterations. A typical result gives a discounted reward greater than -6.0 after only 1000 iterations. These results are illustrated in Figure 5–6.

5.4.3 Results with FQI

In general we achieved the best total reward when we used the state arrays S_{10} or S_9 described above, with the action space A_4. As can be seen in Figure 5–7, the

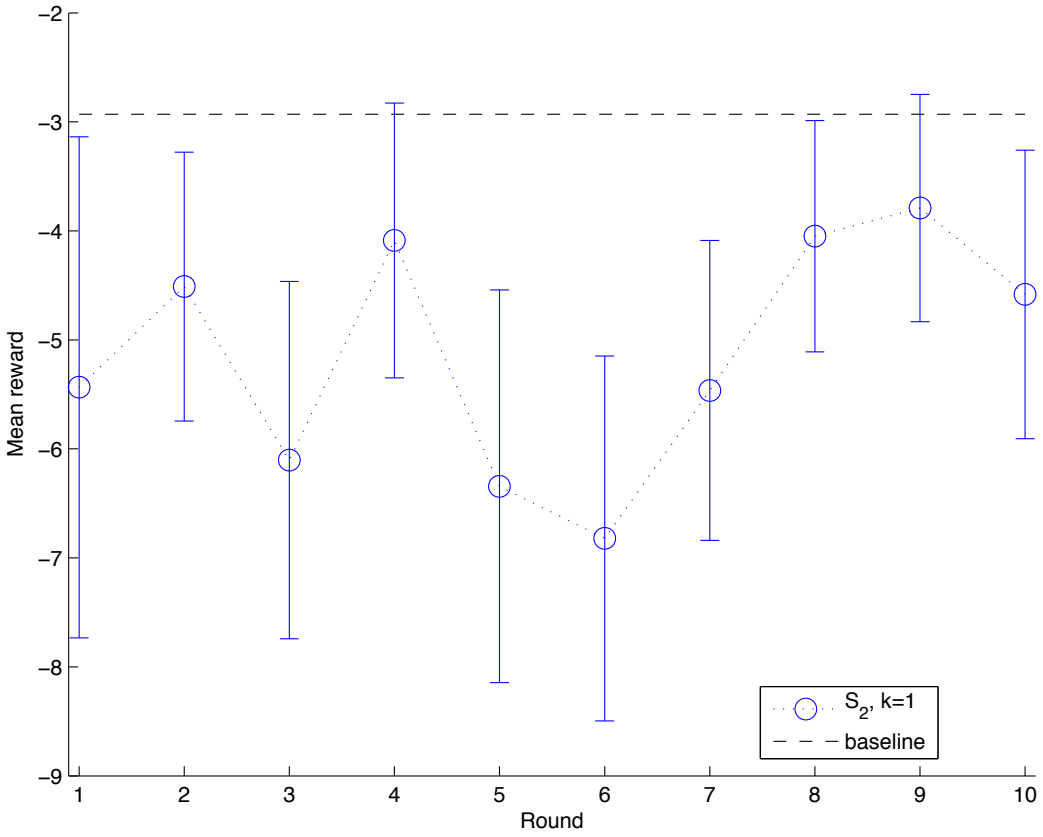

FIGURE 5–6: Example results for kNN-TD(λ) with state vector S_2 and action space A_X. The baseline return is -2.93, taken from Table 5–1.

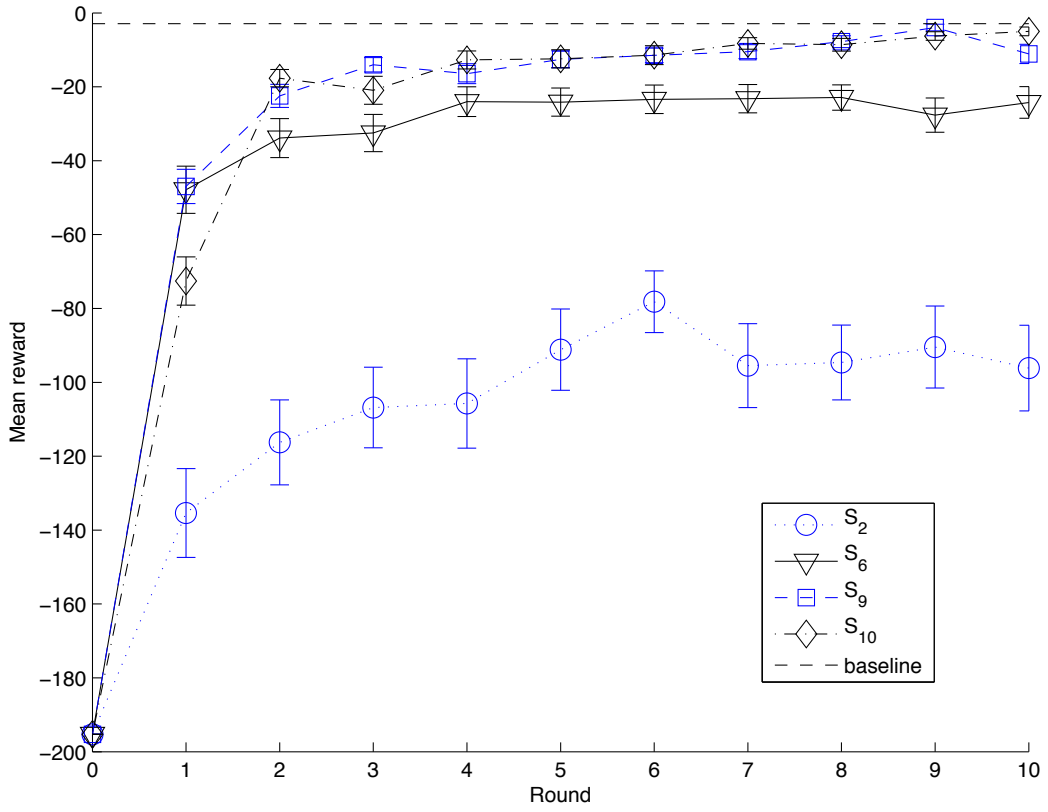

FIGURE 5–7: Comparative results for FQI with different choices for the dimensions of the state vector, as the results evolve from round 0 (the purely random policy) to round 10 (the final policy). The state vector S_{10} slightly outperforms S_9, and results with S_2 are clearly inferior. The state vector S_6 permits some improvement, but seems to quickly reach an asymptotic value well below that of the baseline policy. The baseline discounted reward is -2.93, taken from Table 5–1.

results with state vectors S_2 or S_6 did not improve quickly enough to be practical. We did not attempt experiments with a one dimensional state in the FQI setting.

It seems clear that the extra information conveyed by the time derivative of the $|Z_1|$ value is useful in finding a truly optimal policy. As can be seen in Figure 5–7, the differences were not always statistically significant, but we generally observed superior results using the state vector S_{10}, with a final $R_T = -5.00$ for S_{10} compared with $R_T = -11.10$ with S_9.

For the default oscillator frequency of 1.0 Hz, the agent was able to learn a policy which performed nearly as well as the best threshold choice applied to the

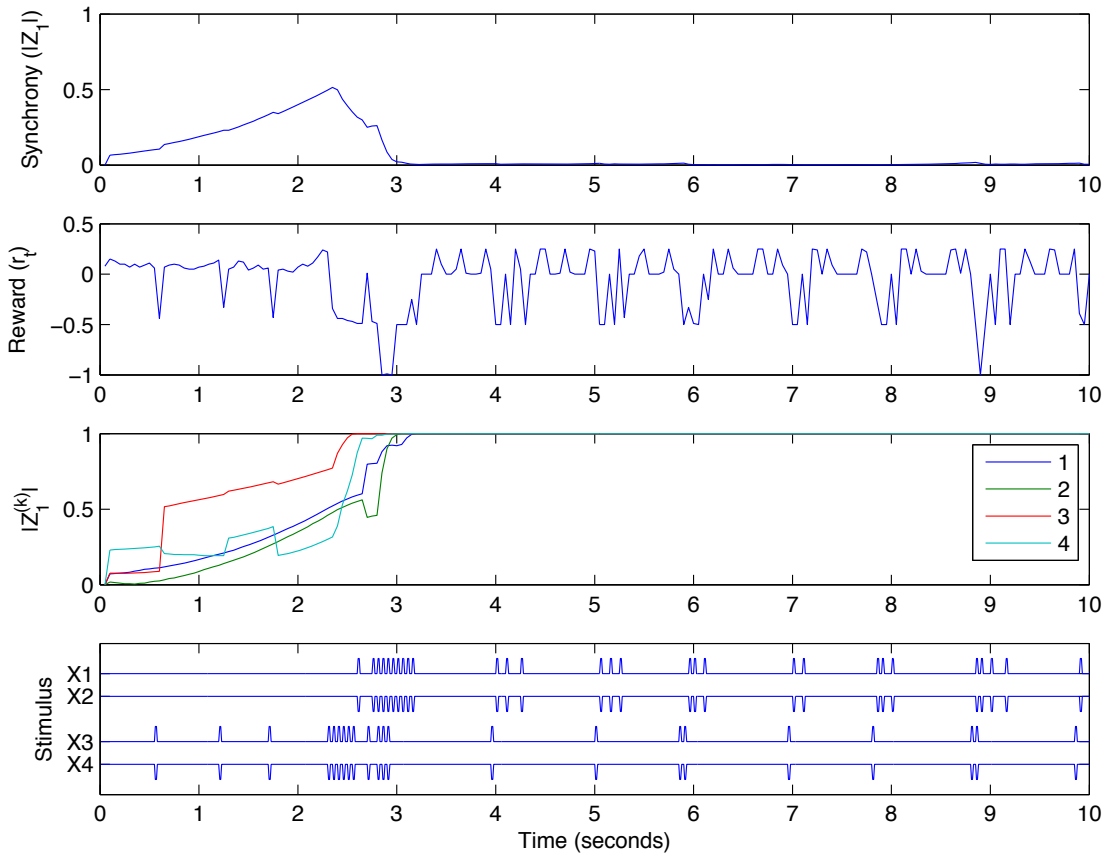

FIGURE 5–8: Example run of the Tass model with the policy learned using FQI with action space A_4 and state vector S_{10}. The frequency of the oscillators in this case is 1.0 Hz, and the total discounted reward for this episode was -1.55.

baseline policy. The learned policy for the state vector S_{10} achieved a mean discounted reward R_T of -5.00 over 60 episodes, compared with a mean R_T of -2.93 for the baseline policy. An example episode using the learned policy is shown in Figure 5–8. As with the baseline policy, the four quadrants are all in near-perfect synchrony, even if the global system is desynchronized.

However, as can be seen in Table 5–1, the baseline policy loses effectiveness where $F \geq 1.6$ Hz, with the best total discounted reward (R_T) of -61.28. In comparison, the learned policy for 1.6 Hz consistently produces a significantly higher mean reward, which approaches the results for $F = 1.0$. We show the development of the reward over the course of training in 5–9. In this example the learned policy

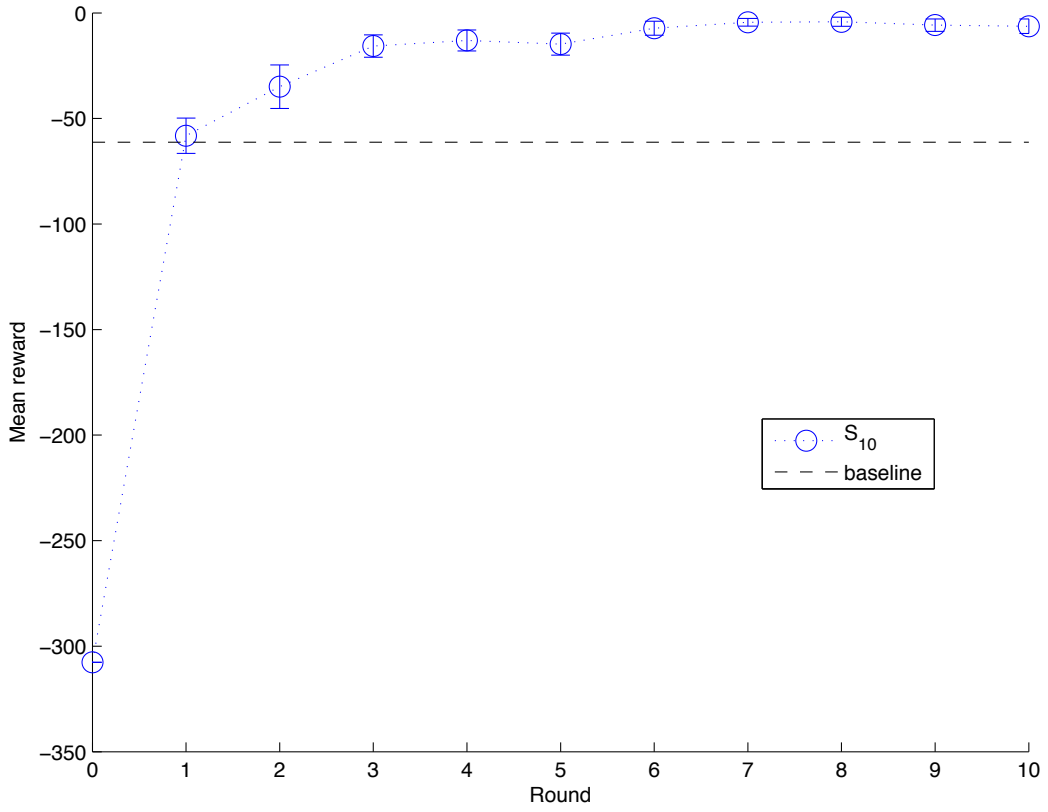

FIGURE 5–9: Results for FQI with oscillator eigenfrequencies set to 1.6 Hz, as the results evolve from round 0 (the purely random policy) to round 10 (the final policy). The policy learned using state vector S_{10} substantially outperforms the baseline after only two rounds of optimization. The best total discounted reward using the baseline policy is -61.28 (see Table 5–1).

reached a mean reward of -6.31, a substantial improvement over the baseline. An example episode of the learned policy at 1.6 Hz is shown in Figure 5–10.

Compared with the results reported in Ernst et al. (2006), we found that our FQI results benefited from a larger value of number of trees (at least 100 rather than 50) and a larger set of trajectories per round (60 rather than 30) to derive a "good" policy. In addition, we found it advantageous to use a value of n_{min} of 5 rather than 2, which has the effect of reducing the sensitivity of the model to output noise (Geurts et al., 2006). We speculate that these differences are partially necessitated by the stochastic component of our model, compared with the deterministic HIV model.

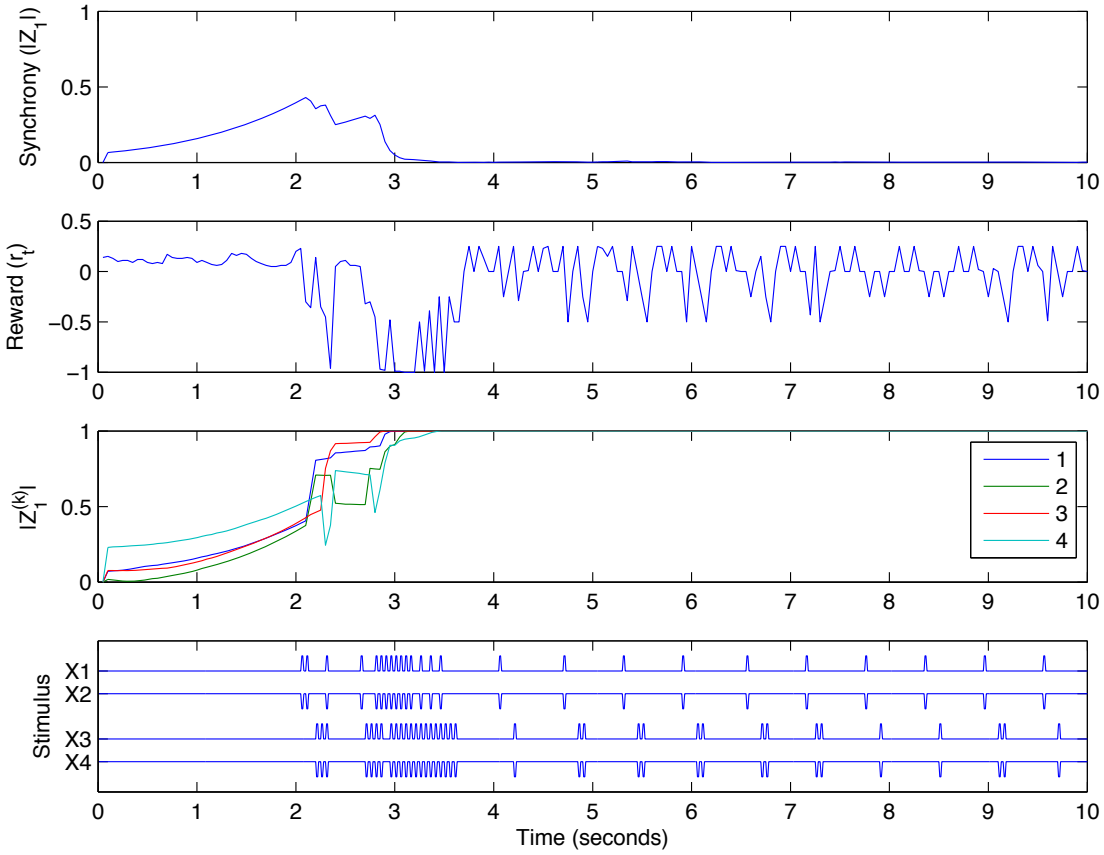

FIGURE 5–10: Example run of the Tass model with the policy learned using FQI with action space A_4 and state vector S_{10}. The frequency of the oscillators in this case is 1.6 Hz, and the total discounted reward R_T for this episode was -1.35.

Finding the best choice of parameters was, in this case, largely one of trial and error. However, one of the strengths of the method is its relative insensitivity to the parameters. While performance was affected when using slightly different values for n_{min} or M (the number of trees), these did not deviate radically from those reported in this chapter. For example, in early experiments with $M = 60$ and S_{10}, we found a total return $R_T = -9.99$, well below the value -5.00 we achieved with $M = 100$, but still clearly well above the results with other RL algorithms when used with the A_4 action space, as described in the previous sections. Going into greater detail on the effects of n_{min}, Figure 5–11 shows the varying quality of results with different choices of n_{min}. While the choice of either $n_{min} = 2$ or $n_{min} = 3$ clearly produces inferior results, any value of $n_{min} \geq 4$ produces final policies which are not readily distinguishable in quality, as they have overlapping confidence intervals.

While the best policies were derived using action space A_4, we were also able to find good evidence of learning in the A_{16} action space (Figure 5–12). A typical policy learned with A_{16} yielded a R_T of -10.6 (compared with -5.0 for A_4). Although the policies with A_{16} were inferior to those of A_4 in terms of total reward, examining the behavior of these policies is interesting, in that they show less consistent synchrony within the subpopulations. This suggests that the policies with a larger action space may be qualitatively different from those with the smaller action space. However, the policy does allow the system to become hypersynchronized at times, triggering a major penalty in the reward function (Figure 5–13).

To further explore the importance of the state variables to the problem, we examined the distribution of features selected in the nodes of a typical learned policy in the 10-dimensional case. The use of decision trees as the underlying function approximator in the FQI algorithm provides us with a simple way to examine this aspect of the derived policy. We found a distinctive pattern of frequencies with which the different features were selected for use in the tests during training. As

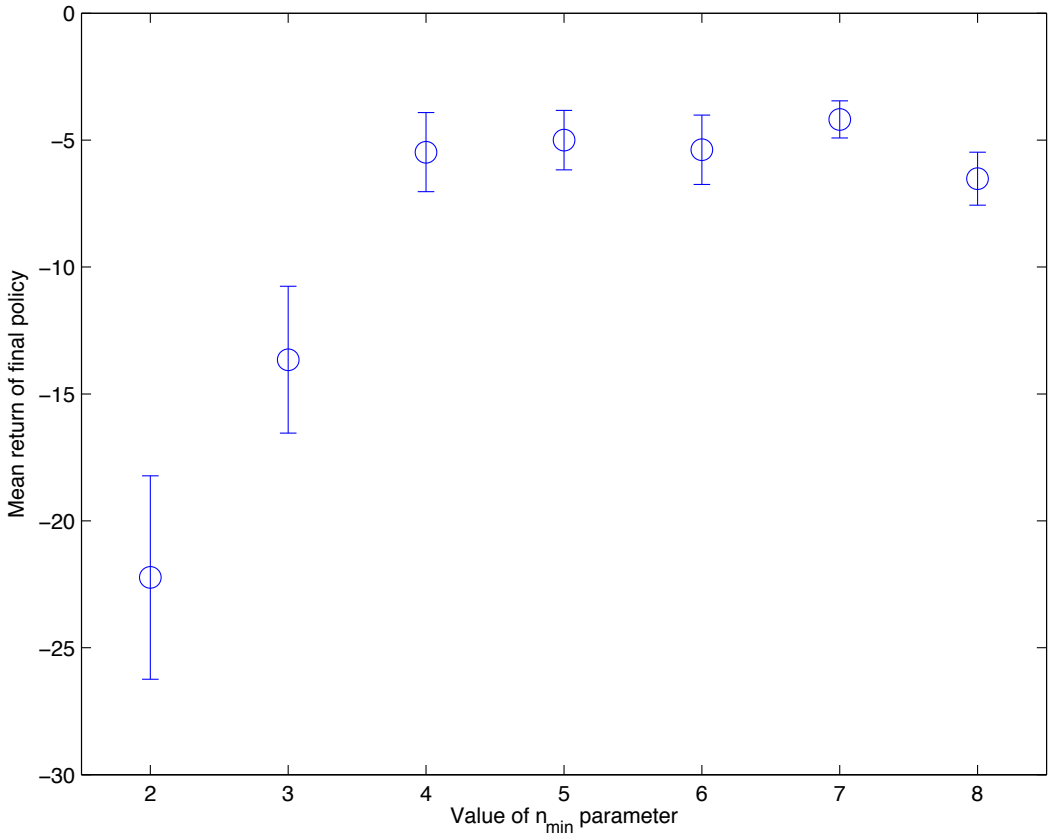

FIGURE 5–11: Mean return R_T for 200 separate evaluations of the final policy found while varying n_{min}. The parameter n_{min} is the minimum node size which can be split by the Extra tree algorithm. All other experimental parameters were held constant (e.g. A_4, S_{10}, $M = 100$ and $K = 10$). Error bars show 95% confidence intervals.

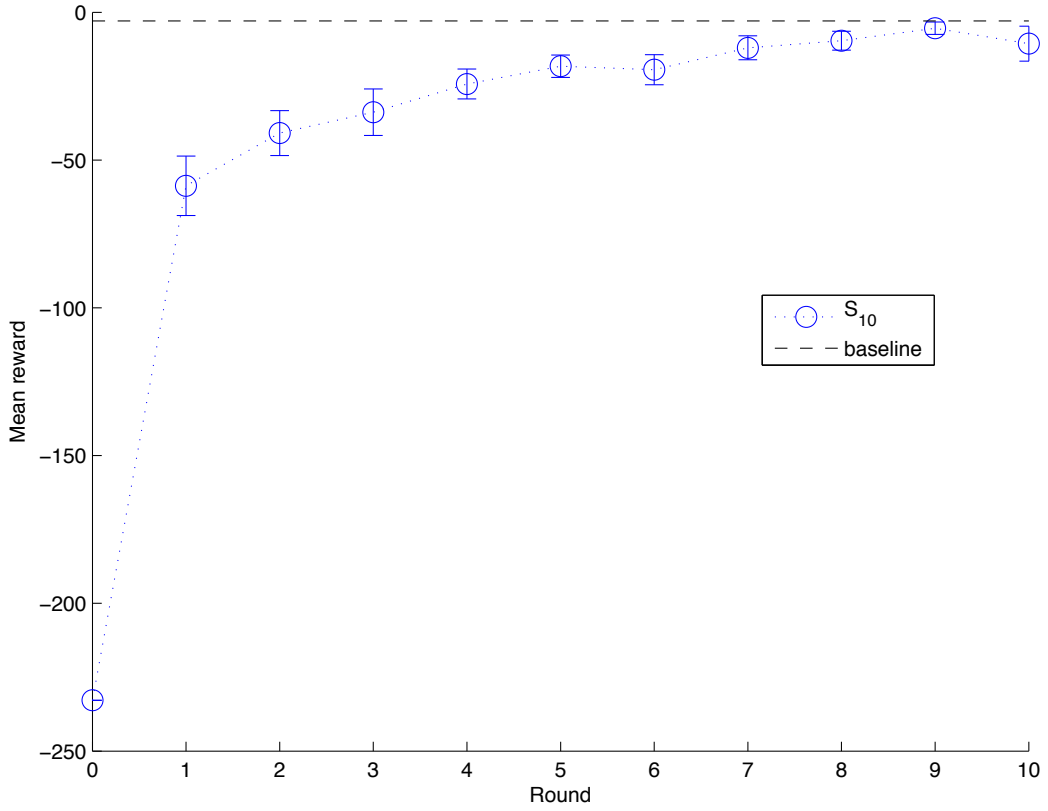

FIGURE 5–12: The mean discounted reward for FQI with action space A_{16}, using state space S_{10} and 120 trees per forest.

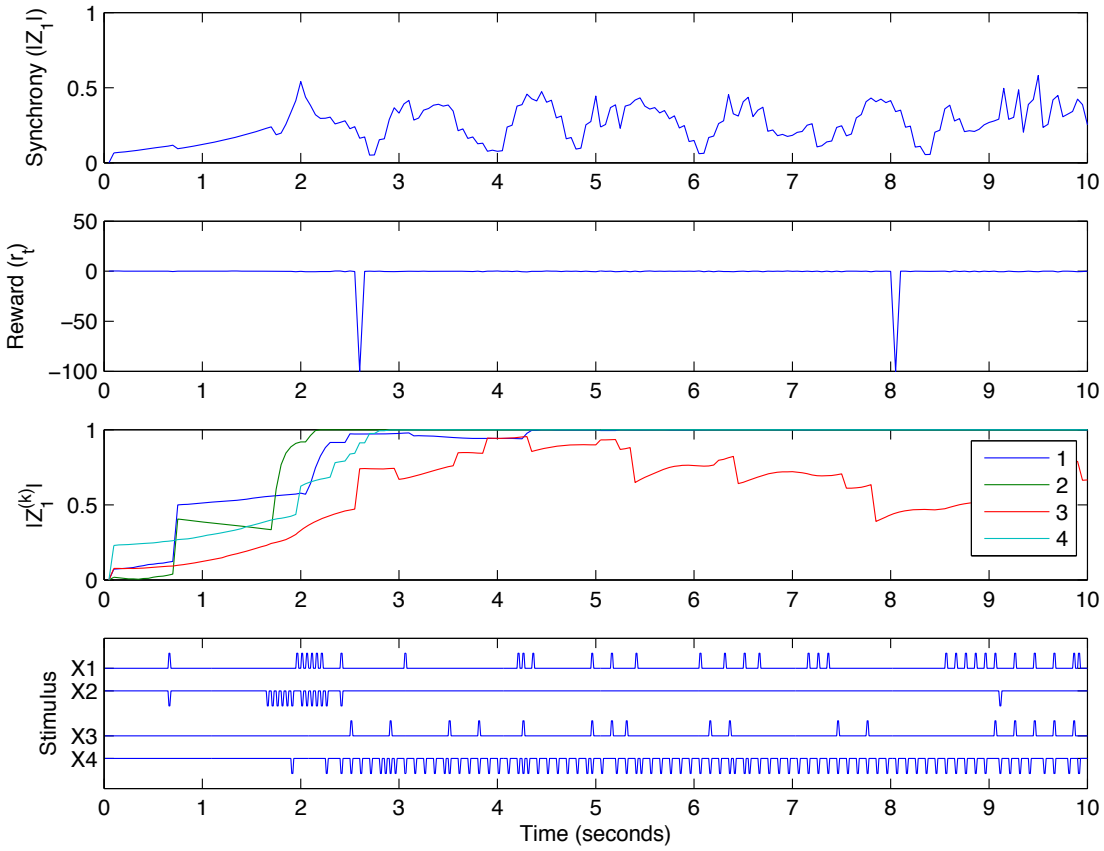

FIGURE 5–13: Example policy for FQI with action space A_{16}, using state space S_{10} and 120 trees. Note that the per-quadrant synchrony measure $Z_1^{(k)}$ is often not equal to one, but also that the policy allows a major penalty for hypersynchronization at two points in the simulation.

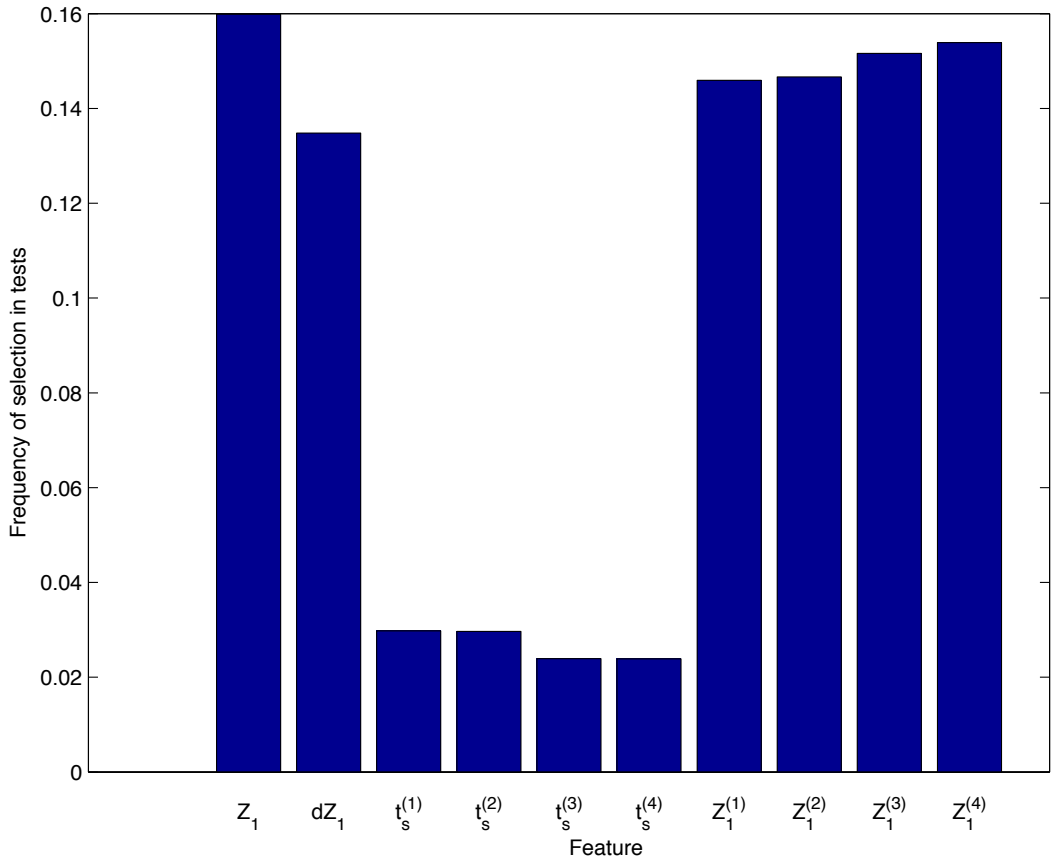

FIGURE 5–14: The distribution of features selected when growing the trees in fitted Q iteration with the extremely randomized tree algorithm. As can be seen, the most selected feature is the amplitude of the global cluster variable (Z_1), closely followed by its derivative and the amplitudes of the cluster variables of the individual quadrants. The time-based features are sampled the least often. This figure represents the results averaged over all four actions; the distributions for the individual actions have a similar structure.

can be seen in Figure 5–14, there is a substantially greater number of decision nodes using the amplitude of the cluster variable, both global and per-quadrant, as well as the derivative of the cluster variable. While the empirical evidence as shown in Figure 5–7 suggests that the time since last stimulation ($t^{(i)}$) also provides some useful information, it is nevertheless sampled less often. This would seem to imply that it may either be less important overall, or that it has only a few key discretization levels that must be represented. We used this conjecture to motivate our use of a reduced number of discretization levels in the kNN-TD(λ) case.

Another important question is whether performing the multi-round learning process is truly necessary. The intuition in many RL experiments is that data gathered with a completely random policy will be less informative than data gathered with a slightly more sophisticated policy. This is true even for algorithms which are able to learn off-policy. To test this, we ran experiments in which a single-round approach was used, using a purely random policy to generate 120,000 training samples, after which the FQI process was run over this entire set of data for 1000 iterations. This simulates the amount of data and FQI iterations used in the multi-round case, so the only significant change is the policy used to generate the data.

In this case, unsurprisingly, we found that the final learned policy was significantly worse in these single-round experiments, despite having the same number of data points and iterations of FQI. The policy yields an average discounted reward of -54.25, which is much better than a purely random policy, but significantly worse than either the baseline policy or the best multi-round policy.

5.5 Discussion

Here we will consider some implications of these results.

5.5.1 Model limitations

A fundamental question with this model is whether it truly represents a realistic model of a coupled group of neurons. One item of concern is that global desynchronization comes at the cost of local hypersynchronization. While the complete network is desynchronized globally after stimulus, each individual quadrant is forced into a hypersynchronized state. This is a core property of both the baseline policy and of our learned policies. This may not present a problem in reality, but this raises legitimate questions about the effects of hypersynchronization at differing spatial scales. We are not aware of research that discusses this, so there seems to be reason for concern that this limits the usefulness of this model to real-world applications.

5.5.2 RL experiments

Several general observations about the RL problem can be made. First, while the FQI method was able to find a policy that approached the baseline policy's total discounted reward for both the A_4 and A_{16} cases, the smaller action space typically yielded better results for the same configuration of function approximation resources. This is consistent with a general notion that a more complex state or action space will impose greater costs on the learning algorithm in either data or approximation resources, and is therefore unsurprising. However, making arbitrary simplifications to the action space may make the problem harder to solve by causing certain states to be unreachable, so one cannot conclude that a smaller action space will universally lead to better results. For this model, it would seem that the choice to use the A_4 action space does not impose a significant reduction in potential return relative to A_{16}.

Second, especially with FQI, the state variables were each useful for deriving the final policy, although it is not clear they each require the same complexity of discretization. The reinforcement learning model using the 1-dimensional or 2-dimensional state vectors clearly lacks the ability to represent the full stimulus method as described by Tass (2003). Even the 6-dimensional state space significantly underperformed the 9- and 10- dimensional vectors (Figure 5–7).

Finally, this problem may suffer in part from the lack of a clear initial start state. Many RL problems have an obvious start state, such that an agent may not in practice have to explore the entire state space to find a path to the optimal steady state, if one exists. One example of this is the HIV model of Adams et al. (2004), as implemented by Ernst et al. (2006). In this disease model, a "patient" must transition from a unique stable symptomatic state to a unique stable asymptomatic state; the optimal policy will be roughly the policy that reaches the asymptomatic state in the shortest time.

In the Tass model, there is no single obvious initial state. Although we could define one in terms of the degree of synchronization, this would permit an indefinitely large possible set of specific implementations. Also, the model can never be in a truly stable, healthy state, so there is no clear single target for a policy to reach. In our case we chose to start in a randomized state, but this means that the learning process and learned policy will have to be fully-developed over a very large fraction of the state space.

Sarsa-λ and kNN-TD(λ)

The results with Sarsa-λ and kNN-TD(λ) were generally inconclusive. In part, this is because of the limitation of methods such as tile coding and kNN-TD(λ), both of which make an implicit assumption about the smoothness of the state space. This creates a problem when the correct policy is (for example) a periodic (or quasi-periodic) policy, as there is often a need to switch between actions at nearby points in the state space. With enough prior knowledge, it might be possible to customize the discretization of the state space by using non-uniform tilings.

As can be seen from the results with the extended action and/or relatively simple 4-action model, many reinforcement learning problems benefit when the action space can be constrained. Reducing the size or functional complexity of the action space can greatly reduce the amount of data required to learn an optimal policy. In the extended action case, the RL problem reduces to optimizing a key policy parameter appropriately for a given set of environmental parameters. While this problem is "easier" than learning a more general policy class, it may still produce useful insights in practice. This is especially true in medical domains where a simple, conservative, predictable policy is likely to reduce uncertainty and facilitate regulatory approval. While it is possible that next-generation neurostimulation devices will be able to generate more complex waveforms, it is probably reasonable to assume that

certain parameters, such as maximum current, duty cycle, etc. are likely to remain under the control of clinicians rather than algorithms.

For both A_4 and A_X, it appears that there is little improvement in the learning behavior of the system when the algorithmic parameters are adapted to permit greater generalization. That is, we do not see much improvement in learning rate or outcome when we use more than one tiling (as in the case of Sarsa(λ)), or more than one nearest neighbor (in the case of kNN-TD(λ)). It is somewhat surprising that these methods are not more sensitive to the value chosen for this parameter, since we assume that in this stochastic domain there would be some value in using a function approximator with a greater tendency to generalize to previously unvisited states. Given the generally suboptimal performance of these algorithms, it may simply be that generalization performance is irrelevant, and other considerations dominate the results.

It is somewhat surprising that kNN-TD(λ) finds a superior policy with exactly the same amount of training data as Sarsa(λ) for the action space A_4 (cf. Figs. 5–5 and 5–3). While there is some empirical work on kNN-TD(λ) that supports a view of its general effectiveness, it is unclear what specific aspect of the algorithm underlies this improved performance.

5.5.3 Fitted Q iteration

The results using fitted Q iteration are more encouraging, in that the model appears to converge towards a good solution for a range of system parameters. One advantage of this algorithm is the ability to practically use larger state vectors, something which becomes impractical rapidly with many reinforcement learning methods. Another clear advantage provided by the Extra tree regression algorithm is the ability to represent discontinuities in the state/action space, which supports learning a more complex policy.

The experimental results, while hardly conclusive, suggest that the RL formulation is able to learn a good policy under conditions in which the simple baseline policy performs poorly. These results, while encouraging, are somewhat limited by the relatively simple policies possible under our current model. It may well be that more complex, or simply different, stimulation regimes could produce superior results. To investigate this question, it would be interesting to experiment with an even larger action space, for example, one which allowed either positive or negative polarity to be applied in any quadrant.

A drawback of the FQI method is the large amount of memory used to represent the trees. For the 4-action case with a 10-dimensional state, the trees for a typical run will require roughly 500 megabytes of disk storage. By comparison, simple functional approaches may be able to represent a policy with only a few bytes of data.

Another drawback of FQI is the relatively large expense in computation time. A typical run of FQI with 100 trees and four actions can require over twelve hours on a modern 1.7 GHz Intel Core i5 processor, even when implemented in C++ and taking advantage of the inherent parallelism of the algorithm. In contrast, experiments with Sarsa-λ take less than two hours to run, even though they generate many more trajectories and are written in Java.

However, it is important to note that FQI learns a very good policy with only 120000 data samples (60 trajectories of 200 time steps over 10 rounds). This is in contrast with both Sarsa-λ and kNN-TD(λ), which produce inferior results with two million samples (10 rounds of 1000 trajectories of 200 time steps each). In any likely real-world medical domain with empirical data from either experiments or clinical trials, it clearly would be appealing to start modeling the problem with FQI, in order to take advantage of the superior data efficiency of this method. The

greater data efficiency of the FQI method would outweigh any advantage in computational time efficiency, especially given the continually declining costs associated with processing cores and memory.

The other important consideration is the choice of policy used to collect the data for FQI. As we have shown in our results, data produced with a purely random behavior policy will not produce as good a final policy as can be achieved with the 10-round bootstrap approach. We have observed a similar result with our own FQI experiments with the HIV model described in Ernst et al. (2006), in that the multi-round approach consistently yields a good policy, but a single-round approach does not. This has important implications in any real medical study, as it suggests that there may be greater value in re-running any clinical trial or other experiment with incrementally better policies, rather than relying on a purely exploratory policy.

The fact that the model lacks well-defined stable start and/or stop states probably accounts for our observation that this environment seems to benefit from both a larger data set and larger forests than was required for the HIV model (Adams et al., 2004; Ernst et al., 2006). The larger forests strengthen the variance reduction in the policy (Geurts et al., 2006), while the larger data set provides greater coverage of the very large and poorly structured state space.

5.6 Related work

Tass (1999) introduced the model to demonstrate the feasibility of coordinated phase resetting of populations of neurons. Tass (2003) demonstrated the effectiveness of the phase resetting approach in the model and laid out the simple stimulation paradigm we have followed here. These approaches are relatively "coarse-grained" in that they assume access to a very simple stimulus function, which is simply turned on or off in a demand-controlled manner. This paper does consider the possibility of varying the time and duration of the stimulus, but retains the basic quadrant geometry and use of antiphase stimulation in related quadrants.

Popovych et al. (2005) considered a similar model, showing a method for desynchronization based on delayed feedback that does not require extensive calibration. However, this model involves the generation of a control signal which strongly depends on the state and phase of the oscillator's output to form the feedback signal for desynchronization. While the authors argue that their mechanism is in a strict sense optimal for the given model, it is unclear how realistic it is to detect and generate such a complex and dynamic feedback signal in a real population of neurons.

Echauz et al. (2009) explored the same model as a proxy for early termination of epileptic seizure by desynchronization, proposing additional control mechanisms for the model. They point out that even open-loop stimulation devices are subject to a combinatorial explosion of stimulation parameters which threaten to make optimization impractical. As an alternative, they argue for a functional, closed loop control mechanism which can be tuned automatically to a specific patient using a simple set of parameters. They demonstrate several mechanisms which are of possible interest, but while some are formulated using an explicit notion of optimization, the objective functions used do not include cost of the stimulus itself as a term. This results in several cases where proposed policies are (admittedly) unsafe or physically impossible.

Of course, it is not entirely clear that such simple functional forms will be obvious or sufficient for real systems.

5.7 Contributions

We have presented the first attempt to formulate this decade-old model as a reinforcement learning problem, with a rigorous cost model that includes the cost of stimulus. We have shown that at a minimum, an RL agent is able to optimize the key parameter in the baseline policy, and that at best, it can learn a policy which

can outperform the baseline once the system moves away from the parameter settings for which the baseline policy was designed. While not much discussed in the prior development of the model, it seems unlikely that a simple, universal constant eigenfrequency is a realistic assumption in any real neural system. Certainly, no argument has been offered to explain why a constant 1.0 Hz frequency is physiologically realistic, as neuron firing frequencies commonly deviate from 1.0 Hz (Dayan & Abbott, 2001). In our opinion it is important to consider the behavior of the model for a much wider range of parameters, and it is in these cases that we see the RL approach can potentially contribute novel insights into the possible choices for alternative policies.

We have also demonstrated the viability of fitted Q iteration with extremely randomized trees in a relatively complex medical domain. This algorithm brings a number of advantages, so validation of this approach in a new, non-trivial context represents an important step toward real-world applications.

5.8 Future work

Currently, the reward function parameters are chosen such that the cost associated with synchronization should be much greater than the cost associated with stimulation. We have used fairly arbitrary values here, it would be useful to review this reward function with more consideration for the actual physiological costs and benefits of hypersynchronization vs. stimulation.

It would be more physiologically realistic to represent the coupling between the oscillators as a full matrix of connection strengths, possibly with a time-varying component. Similarly, the natural frequency Ω of the oscillators is likely to be a distribution rather than a single value, so it could be represented as a constant or time varying vector. Existing work has already implemented some of these extensions (Popovych et al., 2005). These elaborations would yield a more realistic

model of an actual network of neurons, in the presence of long and short term synaptic strength variations. Such a model might provide additional support for the idea of using an on-line RL method to address the likelihood that such a system would not converge to a unique stationary solution, but would rather benefit from an algorithm that could dynamically track changes in the system response, and which can also represent likely variations among patients.

Of course, with added model complexity comes the problem of selecting appropriate values for newly created parameters; this is a fundamental problem of modeling, as discussed in Chapter 3.

It would be useful to consider a class of policies similar to those explored in Echauz et al. (2009). Like the extended action (A_x) in this work, these policy classes might permit a relatively compact policy representation. In the cited work, there does not appear to be enough detail given for these policy classes to guide their implementation in an RL framework. Given what we do know, optimizing these policies might benefit from a very different learning paradigm, such as an explicit actor-critic or policy iteration method, especially one that exploits policy search (Ng & Jordan, 2000) or policy gradient methods (Peters & Schaal, 2008). In any case, a simply-parameterized policy might allow for faster convergence and better data efficiency.

As we mentioned in the beginning of this chapter, there is increasing clinical evidence that responsive systems for neurostimulation can produce superior results with reduced symptoms for a variety of neurological diseases (Sun et al., 2008; Smith et al., 2010; Rosin et al., 2011). While responsive devices use some form of closed-loop control, they do not necessarily have the means to optimize their treatment for a specific patient. Algorithms developed using the approach described in this chapter might adapt automatically to the needs of specific patients, by training them using historical data from the patient. Even further in the future, these

algorithms might be able to continuously adapt their policy to track changes in the patient's condition.

CHAPTER 6
Fractionation scheduling for radiation therapy

This chapter describes our preliminary efforts to investigate the potential of reinforcement learning to find optimal scheduling algorithms for radiation therapy. As with the previous chapters, we use a computational model, in this case one that models radiation effects in cell cultures. We parameterize the model using data from clinical experiments, as well as values gleaned from the literature. This model is then used as the basis for our experiments. Unlike the models previously described in this thesis, within some constraints we are able to exhaustively enumerate the policies in this model, making it possible to determine the optimal policy.

6.1 Problem description

Radiation therapy uses ionizing radiation such as X-rays to selectively destroy cancer cells. It is a well-established method for treating a variety of cancers, estimated to be used to treat roughly 52% of cancer patients (Joiner et al., 2009). The method relies on the observation that most tumor cells are more susceptible to radiation damage than are healthy cells. Recent innovations in radiation therapy have increased its specificity by increasing the ability of clinicians to target cancerous tissue while sparing normal tissue. The technologies use complex hardware and software to perform tasks such as target identification and beam forming.

Because of the inherent risks posed by exposure to ionizing radiation, there is an understandable concern about dangers posed by operator errors and software problems, e.g. the Therac-25 (Leveson & Turner, 1993). One method which is widely used to manage risks and improve outcomes is *fractionation*. If the total dose required to destroy a tumor is estimated to be N grays (Gy), the dosage will typically be divided into many equal, smaller treatments delivered over a period of

several weeks, with at least several hours between treatments (Fowler, 2011). The primary motivation for this is the empirical evidence that shorter doses will kill tumor cells effectively while (relatively) sparing normal tissue. This effect reflects, in part, both the superior ability of normal cells to repair sub-lethal lesions and the increase in repair rates observed as cells are exposed to radiation (Joiner et al., 2009).

The standard approach to fractionation assumes that the same dose can be applied during every treatment, so if a total dose of N is to be applied over M treatments, the dose per treatment will simply be $\frac{N}{M}$ (Joiner et al., 2009). We choose to apply reinforcement learning to explore whether there might be better approaches to fractionation than simple division of the total does into equal fractions. There exists some evidence that temporal variation of the dosages may have an effect on clinical outcomes (Altman et al., 2006). Given the likelihood that the underlying system has a high complexity that is incompletely understood, it seems reasonable to conjecture that there might be some, if small, advantage to a more dynamic schedule of treatments.

6.2 Data and modeling

Given the risks associated with radiation therapy, studies using humans or even animals are typically quite expensive and difficult. Cell culture experiments are more readily accessible, but still require a large investment of time and skill, with considerable measurement error, biological variability, and limitations on the kinds of manipulations that are practical. In typical cell culture experiments, a sample taken from a specific cell line is exposed to several radiation doses which are evenly-spaced with an inter-exposure gap of roughly one week.

To address these problems, we have again implemented a computational model that attempts to capture the dynamics of the radiation therapy process, especially as observed in cell culture experiments.

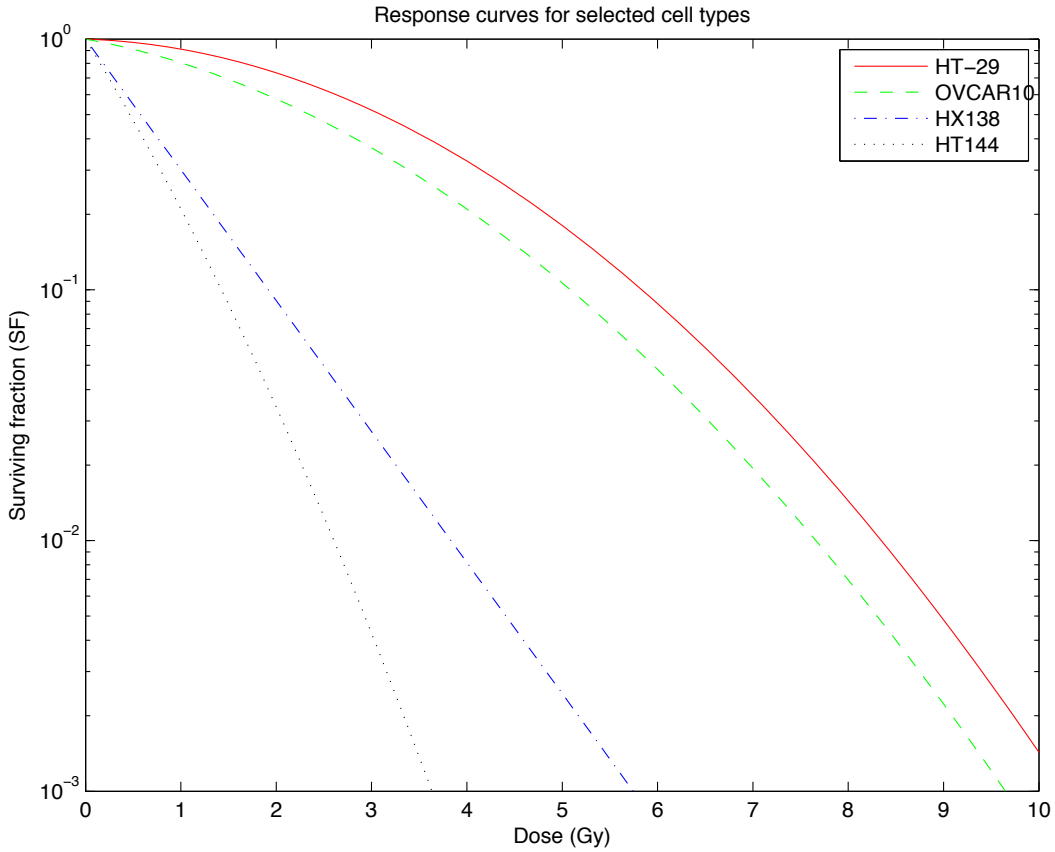

FIGURE 6–1: Dose-response curves for the LQ model for four different tumor types. Constants from Chapman (2003).

6.2.1 Linear-quadratic model

The most basic model of radiation response is the linear-quadratic (LQ) model:

$$\sigma = \exp\left(-\alpha D - \beta D^2\right), \tag{6.1}$$

where σ is the probability that a given cell survives the treatment (sometimes called the "surviving fraction"), α and β are constants characteristic of a particular class of cells, and D is the total radiation dosage in grays (Gy). The constant α can be considered in natural logarithm of the probability of a cell being non-repairably damaged per Gy, whereas β is the probability of a cell being repairably damaged per Gy^2 (Fowler, 2011). The behavior of the LQ model is illustrated in Figure 6–1.

Dividing the total dose D into n fractions of individual dose $d = D/n$, and assuming that each fraction is equally effective, we can derive a minor extension of the LQ model for fractionated radiotherapy:

$$\sigma_n = (\sigma_1)^n = \left[\exp(-\alpha d - \beta d^2)\right]^n, \tag{6.2}$$

or:

$$\sigma_n = \exp\left(-\alpha n d - \beta n d^2\right). \tag{6.3}$$

However, even this extended LQ model does not consider a number of important secondary effects. These include dose rate ($R = \frac{dD}{dt}$) effects, lesion repair rates, and cell regrowth. It also fails to provide a detailed picture of the evolution of the system through time. For an RL application, it is necessary to construct a differential model that reflects the evolution of the system through time.

6.2.2 Differential model

For our differential model, we implement the Γ-LQ model developed by Scheidegger et al. (2011). This model provides a complete picture of the evolution of cell populations over the entire duration of the treatment. The model also captures cell regrowth and dose rate effects. It consists of the differential equation:

$$\frac{dN}{dt} = -(\alpha + 2\beta\Gamma)RN + k_N N, \tag{6.4}$$

where N is the cell count, R the instantaneous dose rate, and k_N is a growth constant (which could also be dose-dependent). Γ is the "dose equivalent" and is proportional to the mean number of non-lethal lesions per cell. We chose to implement the model such that Γ obeys the equation:

$$\frac{d\Gamma}{dt} = R - \tilde{\gamma}\Gamma^2. \tag{6.5}$$

The constant $\tilde{\gamma}$ is the kinetic constant of repair. We have used this quadratic formulation of the repair process throughout this chapter. There is some evidence

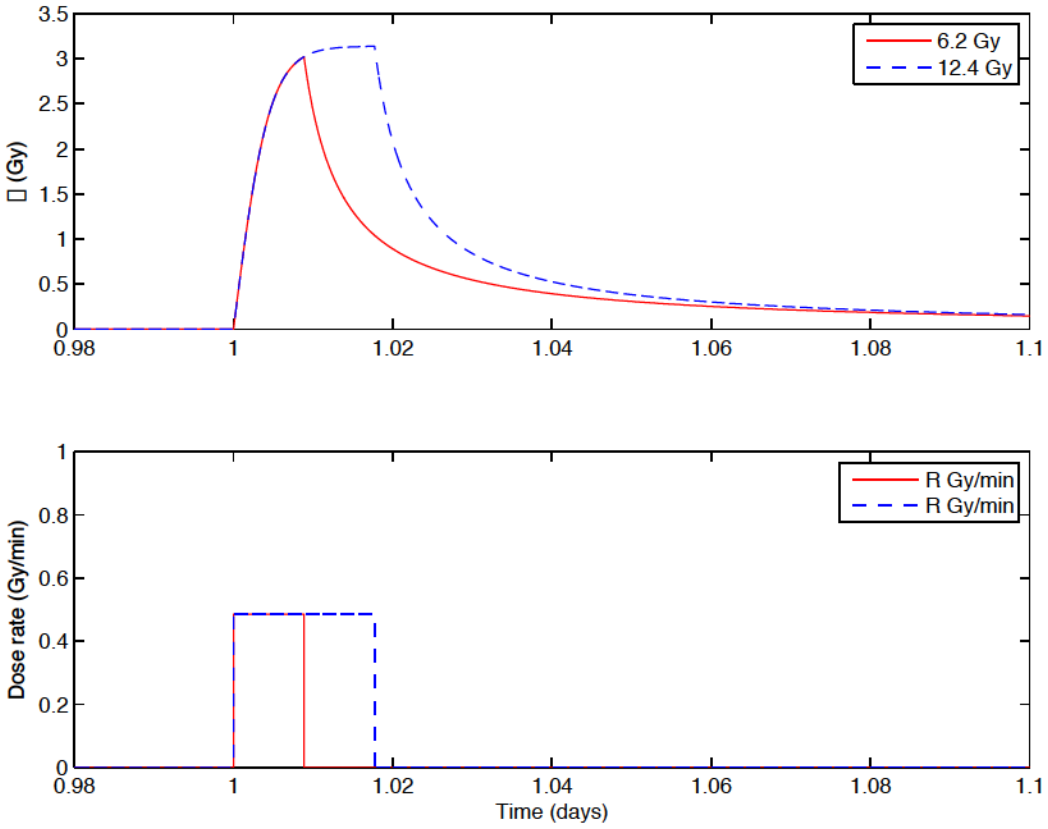

FIGURE 6–2: This figure illustrates the dynamics of the variable Γ in the model. The red solid curves illustrate the applied radiation (lower graph) and effective dose (Γ, upper graph) for a total dosage of 6.2 Gy. The blue dashed curves illustrate the response for a total dosage of 12.4 Gy. The treatment is initiated at $t = 1$ day, $\tilde{\gamma} = 71$ Gy^{-1} day^{-1}, $R = 0.486$ Gy/min. (Scheidegger et al., 2011). By equation 6.6, Γ will not exceed 3.1396 Gy.

supporting the assumption of a second order repair process, but Equation 6.5 is still a simplification (Scheidegger et al., 2011).

During irradiation, the dose-equivalent Γ behaves as shown in Figure 6–2. It can be thought of as a transient dosage value that increases with the dose rate R, but which is decreased by repair processes represented by the second term, $\tilde{\gamma}\Gamma^2$. The equation clearly has a zero where $R = \tilde{\gamma}\Gamma^2$, therefore Γ has a maximum value of

$$\Gamma = \sqrt{\frac{R}{\tilde{\gamma}}}. \tag{6.6}$$

A key advantage of this differential model is its use of the same α and β parameters used in the LQ model. Estimates for these parameters can be found throughout

the literature, or can be derived from new experiments. In contrast, other differential models generally require other rate constants which are not trivially derived from α and β (Curtis, 1986; Carlone et al., 2005).

The α and β constants used in our model are derived from experiments performed in the El Naqa lab during the winter of 2012 (El Naqa, 2012), or from other sources in the literature. Following Scheidegger et al. (2011), we generally used a value of $k_N = 0.15$ day^{-1} and $\tilde{\gamma} = 71$ Gy^{-1} day^{-1}. However, we performed experiments with other values to model repair rates in different tissue types.

6.2.3 Details of the complete model

To represent the problem of destroying a tumor while preserving normal tissue, we run two simultaneous copies of the model, one using the expected parameters of a tumor cell, and another using parameters of a normal cell. The dynamics of each of the cell types are therefore entirely independent of the dynamics of the other - they are tied together only in that both cell types receive the same simulated dosage at any time.

We implement the model using Euler integration with an integration time step of one second. The initial value of N is $N_0 = 10^{11}$, and the initial value for Γ is zero.

We start by assuming a constant total number of fractions F. We assume that each radiation fraction is applied to the model beginning at some regular time interval T_F at some constant dose rate R_0. The instantaneous dose rate R is therefore either R_0 or zero at any step. The model assumes that the dosage per fraction is controlled by choosing the length of time that the sample is exposed to radiation, rather than by manipulating the dose rate.

In our experiments, the time T_F, the interval between fractions, is typically either one day or one week, depending on the type of phenomena we wish to model. In clinical settings, the delay between treatments is typically on the order of several hours to one day, with 30-70 fractions and a cumulative total dose of 50-80

Gy (O'Rourke et al., 2009; Fowler, 2011). However, in our experimental cell culture data, T_F is typically one week, with four fractions and a total dose of 5 Gy (1.25 Gy/fraction) (El Naqa, 2012).

The total dosage for a given fraction D_f can therefore be calculated from the product of the instantaneous dose rate R and the time that the application was non-zero:

$$D_f = \sum_{t=0}^{T_F} RX(t)\Delta t, \tag{6.7}$$

where $X(t)$ is an indicator function whose value is one when the therapy is turned on and zero when it is turned off.

The total dosage D is accumulated over the F fractions by summation over the elapsed fractions:

$$D = \sum_{f=1}^{F} D_f. \tag{6.8}$$

Figure 6–3 illustrates the behavior of the model in this setting, using values for α and β derived from the literature. This shows a somewhat ideal case of excellent preservation of normal cells while destroying most of the tumor cells.

We also evaluated some additions to the model as presented in Scheidegger et al. (2011). First, we implemented a delay in the onset of regrowth of cells after irradiation, something which is commonly observed in real biological experiments (Fowler, 2011). We represent this by extending Equation 6.4 as follows:

$$\frac{dN}{dt} = \begin{cases} -(\alpha + 2\beta\Gamma)RN + k_N N & \text{if } T_E \geq T_k, \\ -(\alpha + 2\beta\Gamma)RN & \text{otherwise,} \end{cases} \tag{6.9}$$

where T_E is the elapsed time since the beginning of the last radiation treatment, and T_k is a parameter which sets the delay before compensatory regrowth begins. In other words, the regrowth rate parameter k_N is in effect forced to zero at the onset of a fraction and remains zero until T_k days have elapsed. We generally had to estimate possible values of T_k, but the typical range is on the order of a few days for tumors,

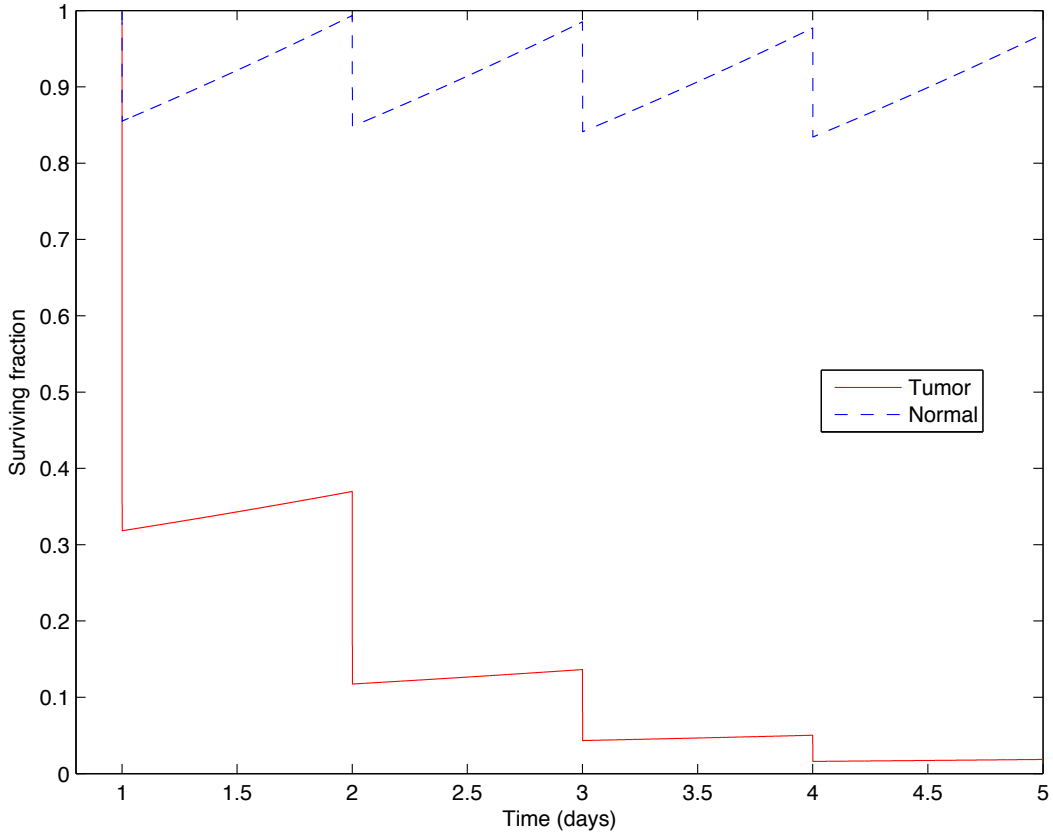

FIGURE 6–3: Illustration of the time evolution of the model for two different cell types, with a total dose of 3 Gy delivered in four daily fractions. The "normal" cells have $\alpha = 0.15$, $\beta = 0.079$, taken from the values for subcutaneous cells in Bentzen et al. (1989). The tumor cells have $\alpha = 1.43$, $\beta = 0.13$, taken from the values for HT144 melanoma in Chapman (2003). Dose rate R_0 is 0.64 Gy/min, regrowth rate k_N is 0.15 day^{-1}. Final surviving fractions are 0.019 for tumor cells and 0.97 for normal cells.

but likely to be much shorter for normal cells, however, normal cells will generally have a slower regrowth rate (e.g. a smaller k_N) (Fowler, 2011).

A second addition to the model provides an upper bound to the maximum cell population, by adding another extension to Equation 6.5, as follows:

$$\frac{dN}{dt} = \begin{cases} -(\alpha + 2\beta\Gamma)RN + k_N N\left(1 - \frac{N}{N_{max}}\right) & \text{if } T_E \geq T_k, \\ -(\alpha + 2\beta\Gamma)RN & \text{otherwise,} \end{cases} \tag{6.10}$$

where N_{max} is a "carrying capacity" or maximum population of a cell. This avoids the problem of unbounded growth implied by the regrowth term ($k_N N$) in Equations 6.4 and 6.9, while stopping short of a complete implementation of a traditional population model such as the competitive Lotka-Volterra model (Lotka, 1932).

Using the same constants as in our Figure 6–3, but incorporating more realistic settings of $N_{max} = 2 \times 10^{11}$ and a time delay T_k of two days for tumor cells, we find that we can achieve similar results with a somewhat lower overall dosage, as is shown in Figure 6–4.

6.3 Experiments

We performed a number of reinforcement learning experiments using the model of the prior section, using each of the RL algorithms we have previously detailed. We explored the model using both reinforcement learning and, where possible, exhaustive evaluation of all possible policies. Exhaustive policy evaluation is practical in this domain where $F \leq 5$. With a space of 11 discrete actions, we have 11^F possible different policies, or 14,641 for $F = 4$ and 161,051 for $F = 5$. Our software is capable of evaluating about 20 simulated days per second on a 1.7 GHz Intel Core i5 CPU. This allows us to evaluate all feasible policies where $F = 4$ and $T_F = 1$ day in about an hour. Knowing the true optimal policy for a given number of fractions permits us to compare the best policy found using reinforcement learning methods.

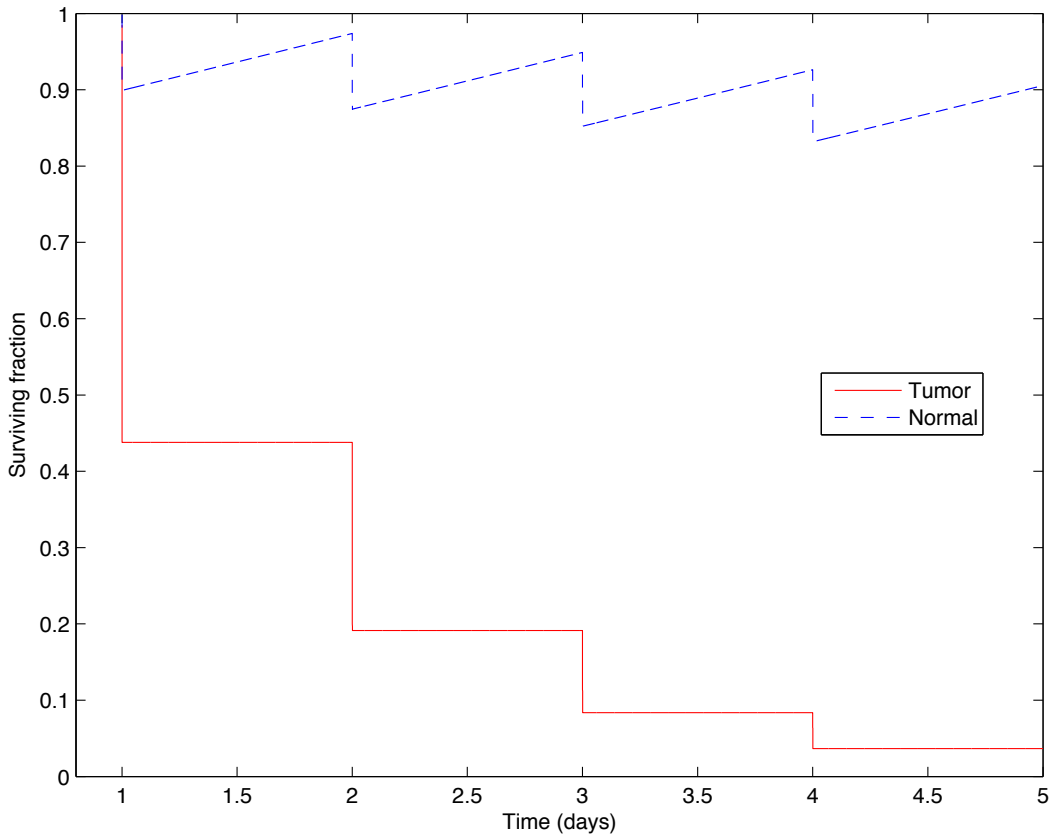

FIGURE 6–4: Illustration of the time evolution of the extended model (Equation 6.10), with a total dose of 2.2 Gy delivered in four daily fractions. Constants are the same as in Figure 6–3 except for using the extended model with the following constants: For tumor cells, $N_{max} = 2N_0$, and $T_k = 2$ days. Final surviving fractions are 0.037 for tumor cells and 0.91 for normal cells.

In each case, we chose a value of 0.98 for the discount factor (the parameter γ, as in Equation 2.1). Because radiation therapy always terminates after a fixed number of steps, this domain is inherently finite. Therefore, it is not strictly necessary to use a discount factor less than one. In practice, FQI requires a discount factor less than one (Ernst et al., 2005a). However, given that we have a domain with a single delayed reward, we select a value for γ very close to 1.0 in order to emphasize the value of the final reward.

As discussed earlier, the model equations are integrated over a time step (Δt) of 1 second to capture the evolution of the differential equations with sufficient precision for these simulations. However, we use the full inter-fraction time, T_F, as our time step from a reinforcement learning perspective– that is, we chose an action at the beginning of each fraction, and evaluated the state and reward immediately before choosing and applying the next action.

6.3.1 Model parameter settings

We considered a wide range of possible parameter settings for the models, in an attempt to best mimic likely clinical scenarios. As in many cases, this goal is complicated somewhat by a lack of a source for standardized parameters for these models. We derived parameters both from sources in the literature and from data provided to us by the El Naqa lab in the McGill University Department of Oncology (El Naqa, 2012).

With respect to the LQ model and related models, tissues are typically characterized by the α/β ratio, which is inversely proportional to a tissue's sensitivity to the dosage per fraction (O'Rourke et al., 2009). Tissues with high values for α/β (in the range 10-20 Gy) are often rapidly-dividing cells with low capacity for repair, which is a common characteristic of tumor cells (O'Rourke et al., 2009). These are often termed "acute responding" tissues. In contrast, tissues with low

TABLE 6–1: Raw data from cell culture experiments using HN47 HPV+ cells and primary keratinocyte (PK) cells. The cultures were exposed to 1.25 Gy of radiation at 0.64 Gy/min at the beginning of each fraction, giving a total dose of 5 Gy.

Fraction	0	1	2	3	4
σ_{HN47}	1.0	0.46	0.14	0.022	0.0032
σ_{PK}	1.0	0.69	0.48	0.16	0.049

TABLE 6–2: Parameters derived from a least-squares fit of Equation 6.1 to the data in Table 6–1. We also give the 95% confidence interval for each parameter.

Cell type	α	β
HN47 HPV+	0.422 ± 0.0223	0.152 ± 0.0130
PK	0.158 ± 0.2532	0.077 ± 0.0885

values for α/β (in the range 1-3 Gy) are typically slowly-proliferating cells with a much higher repair capacity (O'Rourke et al., 2009).

The α/β ratio is more commonly reported than the specific values of α or β, but the ratio does not provide enough information to adequately parameterize the model. Therefore we were limited to consideration of sources which provided strong guidance for specific values for these parameters. In this case, the paper by Chapman (2003) is especially valuable for tumor cell types.

From our experimental data, we derived parameters for two cell types, HN47 HPV+ (a head and neck tumor, positive for human papillomavirus) and a primary keratinocyte (a normal cell type which is a major constituent of the epidermis).

We used the data in Table 6–1 to estimate the α and β parameters by performing nonlinear fits to the standard LQ model (Equation 6.1) using the MATLAB optimization toolkit (MATLAB, 2013). The parameter values derived from this data are given in Table 6–2.

The α parameter for the PK cells has a very wide 95% confidence interval, however, the values agree well with those derived for other classes of skin cells (Bentzen et al., 1989). The parameters for the tumor cells, HN47, have a relatively narrow confidence interval, but a surprisingly low α/β of 2.8. We performed a number of experiments using these parameters, in order to mimic the specific experimental

TABLE 6–3: Parameters derived from the literature, specifically HT144 melanoma (Chapman, 2003) and subcutaneous cells (Bentzen et al., 1989).

Cell type	α	β
HT144 melanoma	1.43	0.13
subcutaneous cells	0.15	0.079

setup used to collect our preliminary data. However, for our RL experiments, we used parameters derived from the literature (see Figure 6–3).

In addition to the α and β parameters, we need to choose the values for $\tilde{\gamma}$, which is a measure of response of the cell's repair mechanisms. Higher values of $\tilde{\gamma}$ correspond to faster repair rates and therefore a lower maximum dose-equivalent Γ (see Equation 6.6). In Scheidegger et al. (2011), the value of $\tilde{\gamma}$ is 71 Gy^{-1} day^{-1} in all cases. Because tumor cells are generally believed to have slower, less-efficient repair processes (Fowler, 2011), we used lower values for $\tilde{\gamma}$ in the tumor model, typically 40 Gy^{-1} day^{-1}.

For most of our experiments we chose the α and β values as used to create Figure 6–3, which provide a much greater contrast between α/β ratios than we found with the parameters derived from the experimental data. These parameters are given in Table 6–3. These values were used in most of our experiments, and all RL experiments.

In all experiments we used $k_N = 0.15$ day^{-1}. The dose rate R_0 is chosen to be 0.64 Gy/minute during the simulated treatments, based on the typical values used in lab experiments by our collaborators (El Naqa, 2012).

6.3.2 Reward function

We chose to express the reward function in terms of the surviving ratios of normal and/or tumor cells, σ_N and σ_T, respectively (we use σ_X to refer to either value generically). These values seemed to represent the most salient measures of the overall clinical outcome, while being easily derived from the model equations.

The other elements which obviously could participate in the reward function would be either the cumulative total dosage, or the dosage per fraction. Presumably the reward function could assess a penalty if, for example, the cumulative dosage exceeded some preset limit. However, the surviving fraction values should capture the necessary information, given their close relationship to the cumulative dosage.

In all cases, we restrict the surviving fraction to a maximum value of one, to avoid giving extra reward in cases where the final cell population is greater than the initial cell population, N_0, because this condition seems to reflect a failure of the model rather than a desirable clinical outcome:

$$\sigma_X = \begin{cases} 1 & \text{if } N_x \geq N_0, \\ \frac{N_X}{N_0} & \text{otherwise,} \end{cases} \tag{6.11}$$

where, as mentioned previously, we use σ_X and N_X to refer to the calculation for tumor (σ_T, N_T) or normal cells (σ_N, N_N) generically.

Initial experiments were performed using the following reward function:

$$R(t) = \begin{cases} \sigma_N^2 - \sigma_T & \text{if } f = F, \\ 0 & \text{otherwise.} \end{cases} \tag{6.12}$$

In this function, the reward is delayed until the final fraction was applied; the results are only assessed at the end of treatment. Intermediate time steps receive a reward of zero. The intent of this function was to emphasize the importance of preserving normal tissue, since the quadratic term will magnify the penalty for even fairly small deviations below 100% tissue preservation.

Alternative reward functions we evaluated included the ratio of surviving fraction of normal cells to the surviving fraction of cancerous cells:

$$R(t) = \frac{\sigma_N}{\sigma_T}. \tag{6.13}$$

This equation has two fairly obvious problems. First, the value is unbounded as σ_T approaches zero. Second, it is purely a function of the relative survival of the two tissue types. This can lead to the perverse result that an arbitrarily low surviving fraction of normal tissue can yield a relatively high reward as long as the tumor tissue is more severely effected. This is presumably never a reasonable outcome in a real clinical setting. Therefore we abandoned experiments with this function and they are not reported in this chapter.

Another form of the reward function was derived after experiments using Equation 6.12 were found to have limited sensitivity near the "optimal" values. The agent would not effectively differentiate between cases where the surviving fraction of tumor cells was 0.1 or 0.01, for example. Therefore we used an alternative form which instead uses the square root of the surviving fraction of tumor cells, as follows:

$$R(t) = \begin{cases} \sigma_N - \sqrt{\sigma_T} & \text{if } f = F, \\ 0 & \text{otherwise.} \end{cases} \tag{6.14}$$

By using the value of the surviving fraction of normal tissue as a linear term, this function is somewhat less sensitive to changes in this value compared to Equation 6.12. Compared to Equation 6.12, Equation 6.14 magnifies the importance of changes in σ_T for values close to zero. Like Equation 6.12, the reward may be non-zero only after the final fraction.

In some of our more recent experiments, we used yet another form of reward function:

$$R(t) = \begin{cases} -1 & \text{if } \sigma_N \leq 0.9, \\ 1 - \sigma_T & \text{if } \sigma_N > 0.9 \text{ and } f = F, \\ 0 & \text{otherwise.} \end{cases} \tag{6.15}$$

The goal of this formulation was to enforce a hard lower limit on the surviving fraction of normal cells after each fraction. Because the penalty for constraint violation

is greater than the possible reward, no policy which violates the constraint can have a positive reward. This approach also presents a possible source of optimization; solutions that violate this constraint can be pruned immediately, reducing the total amount of computation required.

6.3.3 State representation

To frame the model as a reinforcement learning problem, we model a competition between two different cell types, so that the state consists of at least two dimensions, the total number of normal (N_N) and tumor (N_T) cells alive at a time step. In either case we follow the approach of Scheidegger et al. (2011) and assume an initial value N_0 of 10^{11} for the cell counts. In some experiments we randomized the initial value of the tumor cell population by selecting the value from a Gaussian with mean N_0 and standard deviation $0.02N_0$. This introduces a small stochastic element in what is otherwise a deterministic model. The model assumes a fixed number of fractions F.

We used a 4-dimensional state vector in most experiments. In addition to the values for N_N and N_T, our state vector includes $f \in \{0, 1, ..., F\}$, the current fraction number, and D_f, the cumulative dose up to fraction f.

We did not typically include the value of Γ as a state variable in the system. This decision was made because Γ returns to zero over the inter-fraction time T_F, so it would not be informative given our choice of T_F as the reinforcement learning time step. We did perform some experiments which included Γ, but found it did nothing to improve our results.

6.3.4 Action space

While the action space consists of a potentially infinite range of doses, in practice it is not realistic to assume that the dose per fraction will vary over the entire possible range. Instead we used a small set $\mathcal{A}$ of eleven actions, fractional dosages

ranging from 0 to some maximum value D_{max} in fixed intervals:

$$\mathcal{A} = \left\{ \frac{nD_{max}}{10}, \forall n \in \{0, 1, ..., 10\} \right\}, \tag{6.16}$$

where the parameter D_{max} was typically 1 Gy or 5 Gy. We used smaller values of D_{max} in experiments with large numbers of fractions with short values for the fraction interval, to mimic clinical treatment settings. We used larger values of D_{max}, with few fractions and longer values for the fraction interval, to mimic the cell culture experiments.

6.3.5 Experiments using exhaustive policy search

We performed initial experiments using exhaustive policy search where $F = 4$ and $F = 5$, using the α and β parameters given in Table 6–1, and a $T_F = 7$ days and $D_{max} = 5$. These values reflect the choices made in the experimental data we initially sought to mimic.

As we refined our model and reward function, we chose to continue the experiments using parameters from the literature, as shown in Tables 6–3 and 6–4. We felt these choices better reflected the expected values in a clinical setting. We also chose to set $T_F = 1$ day for both the practical reason that it reduced calculation time per fraction, and because a shorter interval is more consistent with clinical applications. The reduced value of T_F implied that we also reduce the maximum dosage per fraction, to $D_{max} = 1$ Gy.

We chose to use a larger value of N_{max} for tumor cells than normal cells. In effect, this choice will cause tumor cells to proliferate more rapidly than normal cells once the regrowth delay expires. Because we use a regrowth delay which is equal to the time between fractions, this has the effect of imposing a penalty on choosing a dosage of zero Gy during a fraction, as this would give the tumor cells a chance to take advantage of their higher growth rate.

TABLE 6–4: Parameter settings for the model as used in all RL experiments. All experiments were performed with a $T_F = 1$ day and a dose rate $R_0 = 0.64$ Gy/min unless otherwise specified.

Symbol	Description	Tumor value	Normal value	Units
N_{max}	Carrying capacity	$2N_0$	N_0	cells
$\tilde{\gamma}$	Kinetic constant of repair	40	71	$\text{Gy}^{-1}\ \text{days}^{-1}$
k_N	Regrowth rate	0.15	0.15	days^{-1}
T_k	Regrowth delay	1	0	days

6.3.6 Experiments with FQI

We performed a number of experiments with fitted Q iteration (Ernst et al., 2005a). For our reinforcement learning experiments we used the reward function in Equation 6.15.

As described in the previous section, the α and β parameters used were as shown in Table 6–3 and other parameters were chosen as in 6–4.

In each RL experiment, we used a 1-day treatment interval, to make it practical to evaluate treatments with tens of fractions, and to bring our results closer to those we might see in a clinical setting rather than in a laboratory experiment.

As in Chapter 5, the optimization took place in a series of 10 rounds of simulated "treatments" beginning with a uniformly random policy. In this case, each round included $S = 100$ different episodes (analogous to 100 different "subjects") of F fractions, where we tested $F = 4$, $F = 10$, and $F = 30$. In subsequent rounds, the Q-function and policy learned in the prior round is used to gather a new set of data. The agent continues to explore using an ε-greedy policy, where it selects the best action with probability $(1 - \varepsilon) = 0.80$ or a uniformly random action with probability 0.20. All data are used in subsequent rounds, so each round has access to $100FN_r$ data samples, where N_r is the round number from 0 to 9. The FQI algorithm performed 50 iterations per round, with the tree structures held fixed for the final 13 iterations.

Once the final Q-function is determined, we must evaluate the resulting policy against our model. For the standard, fully deterministic model, we do this by running one instance of the model in which we perform no exploration. In cases where we used the randomized initialization of the tumor cell population, we averaged over a full set of S model instances (typically ~ 100).

We performed systematic variation of some parameters to identify the best choices of M and n_{min} for this domain. We also explored a range of values for the exploration probability ε and the number of FQI iterations. In contrast, we did not feel that it was as important to vary K, as in our experience we have found that the algorithm tends to give good results when K is equal to the number of dimensions in the state space.

6.3.7 Experiments with other RL algorithms

We implemented the model within our framework for Sarsa(λ) (Rummery & Niranjan, 1994; Sutton, 1996) and kNN-TD(λ) (Martín H. & de Lope, 2009), as well as an experimental implementation of UCT (Kocsis & Szepesvari, 2006). While we found some evidence that these methods were able to optimize the problem for some parameter settings, the results were worse than those found with FQI, with substantially higher variance in the outcome and greater evidence for divergence from the optimum under some parameter settings. Therefore, we have omitted them from further discussion.

6.4 Results

We now present the results of our experiments, especially with respect to the discovery of optimal policies for our parameter choices.

6.4.1 Model validation and adjustment

As mentioned above, for a total number of fractions $F \leq 5$, it is practical to exhaustively search the policy space for the best policy using simple brute-force

methods. We performed extensive evaluations of the model in these settings, partially to evaluate the effect of the reward functions and overall model.

Preliminary results were obtained with reward functions given in Equations 6.12 and 6.14. Because these preliminary attempts were conceived of as mimicking the cell culture experiments, we used the parameter values derived from the lab experiments (Table 6–2), an inter-fraction time $T_F = 7$ days and a $D_{max} = 5$ Gy. The model equation used is the basic one, Equation 6.4. We found that this yielded optimal policies which were permutations of the sequence $\{1, 4.5, 4.5, 0\}$ for a total dosage of 10 Gy. This yielded a final reward of 0.996 using the reward function Equation 6.12, yielding a discounted reward of 0.94 (recall that the discount factor is 0.98). In contrast, the final reward is 0.94 using reward function Equation 6.14, yielding a discounted reward 0.88. In either case the final value of σ_N was 1, and the final value of σ_T was 0.004. We compared this to the result for the uniform policy with exactly $10/F$ Gy/fraction and found that the resulting policy yielded an inferior discounted reward of 0.79 using reward function 6.14, with a final σ_T of 0.025.

While this result may seem encouraging initially, the actual model is quite unrealistic. This was discovered by inspecting the full time course of the simulation, as depicted in Figure 6–5. It can be readily seen that σ_N exceeds 2 after the first fraction of treatment, implying that the normal cell population has more than doubled. In addition, σ_N also drops as low as 0.14 after the third fraction. Neither of these facts is particularly realistic or desirable in a cell culture experiment. More importantly it would be extremely problematic in a clinical setting.

This result subsequently inspired the use of the reward function in Equation 6.15, to force the system to keep the value of σ_N within a safe range. We also added the carrying capacity term to the model, Equation 6.10 with $N_{max} = N_0$, to avoid unrealistic proliferation of cells, and a regrowth delay of 1 day for tumor

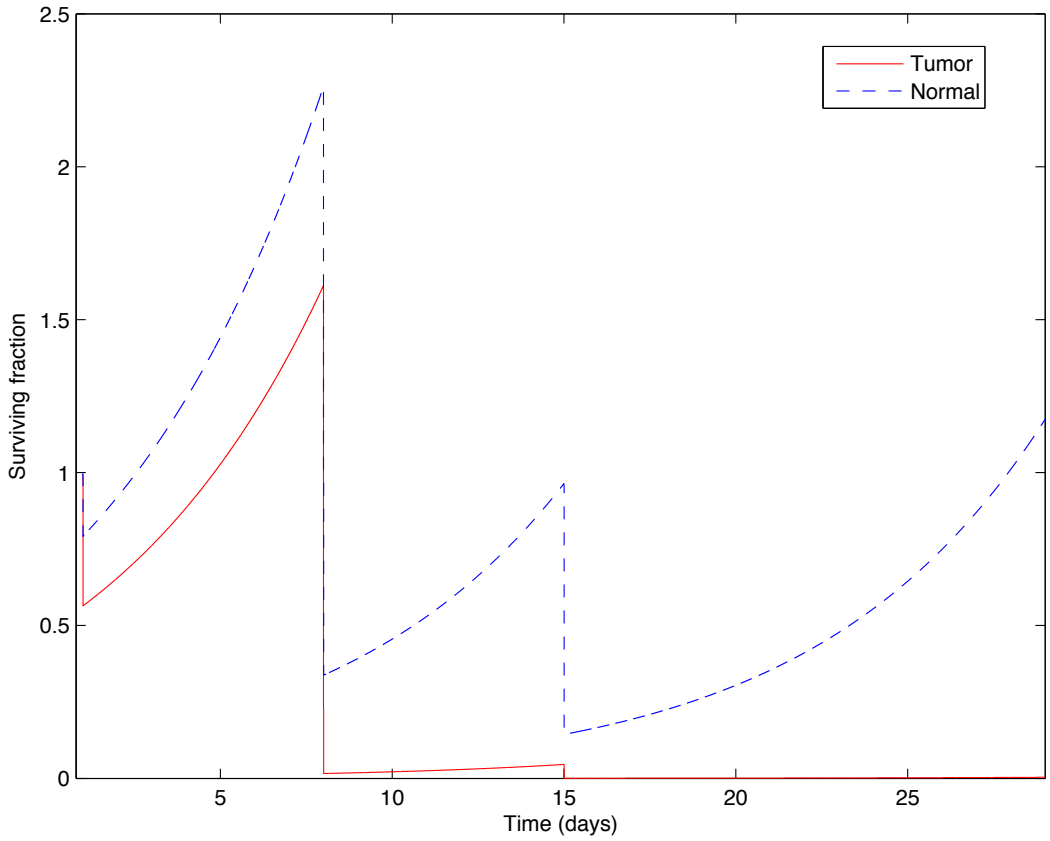

FIGURE 6–5: Results demonstrating the lack of realism in the model when used to simulate cell culture experiments with an unlimited total cell population.

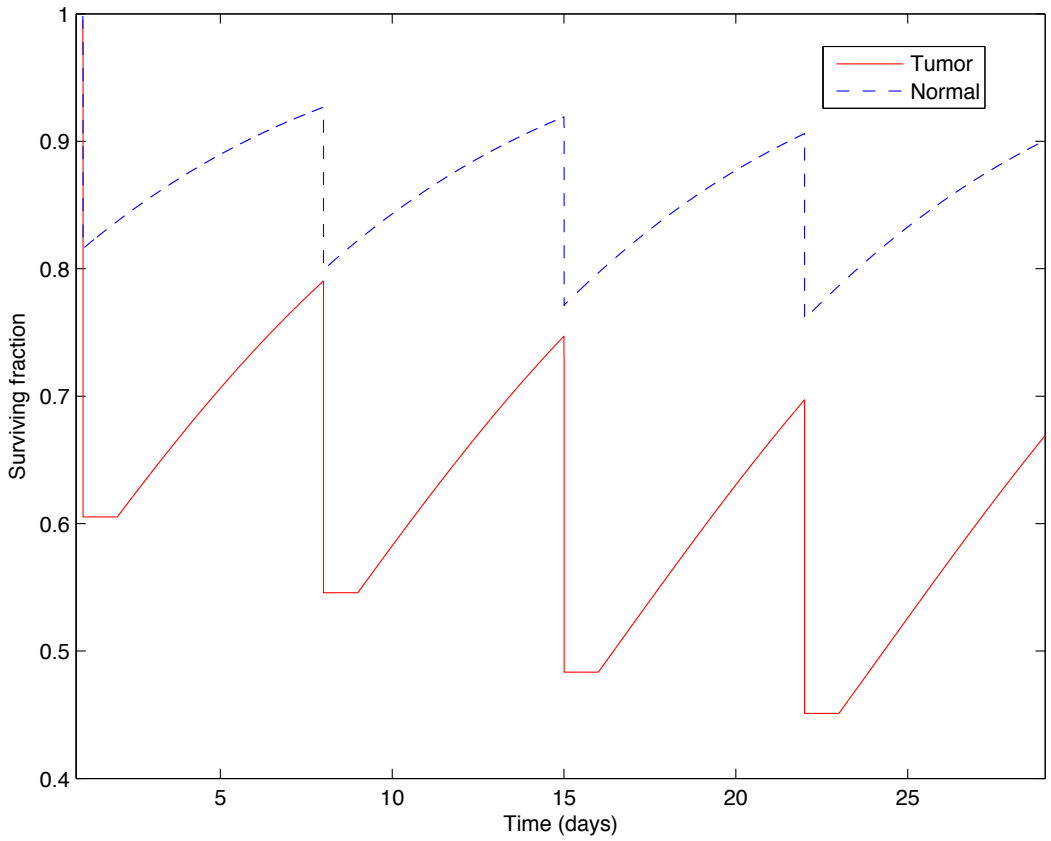

FIGURE 6–6: Results using the model with reward function 6.15 and a maximum carrying capacity of 10^{11} cells to prevent the over-proliferation seen in Figure 6–5. There is also a delay of 1 day added between the application of radiation and the onset of regrowth of tumor cells.

cells. Finally, we adjusted D_{max} to 1 Gy to account for the greater sensitivity of the model when using the reward function in Equation 6.15. Using the exhaustive policy evaluation for a four-step policy with a 7 day treatment interval we found the best policy to be $\{0.9, 0.7, 0.8, 0.8\}$ for a total dosage of only 3.2 Gy. While the resulting population curves are probably more realistic, the final surviving fraction of tumor cells is unacceptably high for any clinical application at 0.67 and discounted reward $R_T = 0.32$. The behavior of this model under this policy is illustrated in Figure 6–6.

In comparison, if the model is run with uniform dosage (0.8 per fraction), the discounted return is negative, because the surviving percentage of normal cells dips below the 0.9 threshold specified in Equation 6.15 after the final time step.

6.4.2 Characterizing the final model

Using the insights from the prior experiments, we settled on the model parameters described in Tables 6–3 and 6–4. Using these parameters and exhaustive policy evaluation for $F = 4$, we found an optimal policy of $\{0.6, 0.1, 0.1, 0.1\}$ with an $R_T = 0.6950$ and final σ_T of 0.26. In contrast, a uniform treatment plan of $0.225 = 0.9/4$ Gy/fraction yields a negative R_T, again because the constraint on the survival rate of normal cells is violated.

In this formulation, the optimal policy is unique among the 4^{11} (14,641) policies. This has implications for the learnability of the overall problem, as the likelihood of sampling a "good" policy is relatively small if all policies are sampled with equal probability. For the 4-fraction problem, only 252 (1.7%) of the possible policies give returns of 0.5 or better, and only 12 policies give returns of 0.69 or better. The distribution of returns is illustrated in Figure 6–7. Each of the best 14 policies, have a total dosage of 0.9 Gy, but only the optimal policy begins with an initial fractional dosage of 0.6 Gy. Again, this implies that an RL agent may have difficulty learning the best policy.

We performed similar experiments with $F = 5$, which yields a total of 161,051 possible policies. The optimal policy in this case is $\{0.6, 0.1, 0.1, 0.1, 0.1\}$, with a final $\sigma_T = 0.23$ and a discounted total return of 0.71. The fact that the optimal policy is a simple extension of the 4-step policy is unsurprising, but certainly not obvious. Also as with the 4-step case, the optimal policy is unique, and the distribution of policy values appears to be very similar.

Of course, it is not practical to evaluate all possible policies where $F = 10$ or $F = 30$, but it seems possible that the pattern will continue in those cases, at

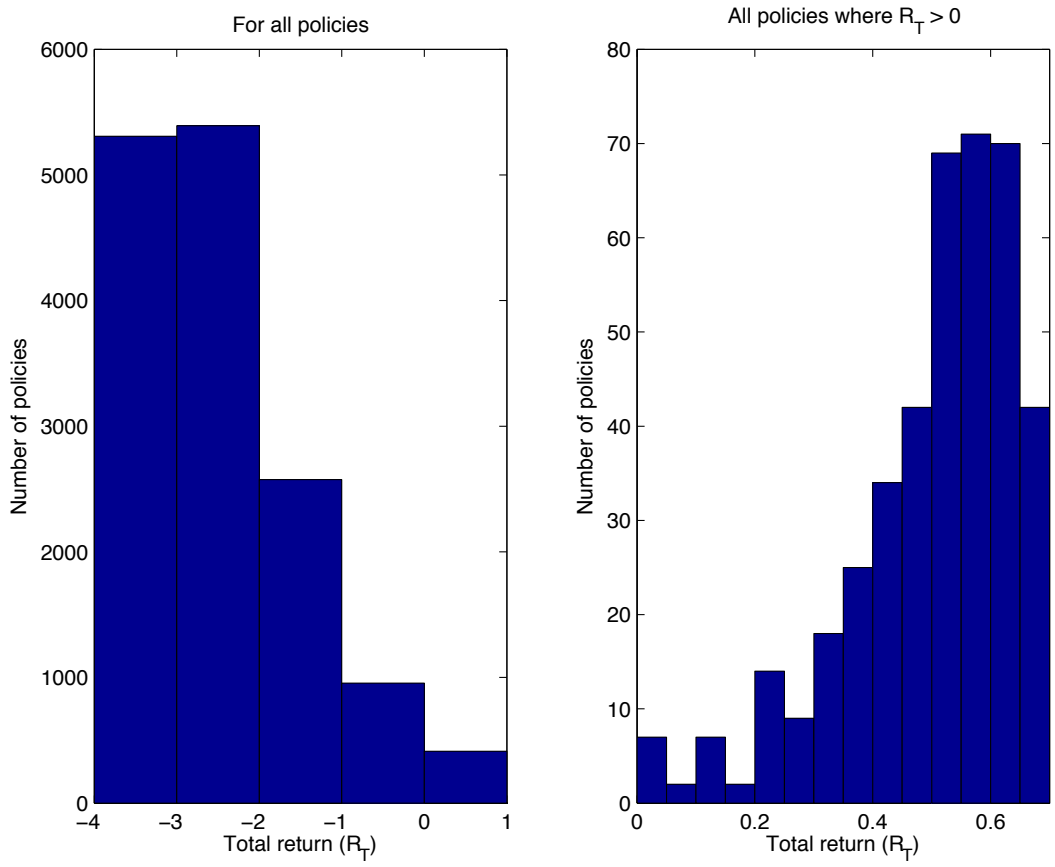

FIGURE 6–7: Histograms of total discounted return (R_T) for all possible policies in the 4-fraction treatment plan. The bar graph on the left gives the distribution over the entire range, the bar graph on the right shows the detail of all policies where $R_T \geq 0$. Only 42 policies have $R_T \geq 0.65$. For these simulations, the parameters were as in Table 6–3.

least for these parameter settings. In fact, when we evaluate the analogous policies (one fraction of 0.6 Gy followed by $F - 1$ fractions of 0.1 Gy), we find that they continue to perform well, yielding $\sigma_T = 0.11$ with $R_T = 0.7425$ for $F = 10$ and $\sigma_T = 0.006$ with $R_T = 0.5533$ for $F = 30$. In both cases, the constraint $\sigma_N > 0.9$ is never violated at the end of the time step, but in the $F = 30$ case, the discount factor significantly decays the value of the final reward.

6.4.3 Sensitivity to model parameter choices

To investigate the sensitivity of these policies to the model parameters, we performed a series of experiments varying several key parameters in the exhaustive policy search case.

Sensitivity to dose rate

We explored the effects of dose rate on the optimal policy for the model. We performed the same exhaustive policy evaluation as before, but setting the dose rate to either $R_0 = 0.2$ Gy/min or $R_0 = 0.9$ Gy/min. Each of these alternative scenarios yields the same optimal policy as when $R_0 = 0.64$ Gy/min, with only small differences ($< 0.1\%$) in the surviving populations and final total discounted return, R_T.

Sensitivity to $\tilde{\gamma}$

We also varied the parameter $\tilde{\gamma}$ to explore whether it had significant effects on the best policy, given our other parameter choices. While our standard value for tumor cells was 40 day^{-1}, we also considered values of either 20 day^{-1} or 71 day^{-1} (the default value from Scheidegger et al. (2011). As with the dose rate, we found that the optimal policy was identical in these alternative cases, again with only small differences ($< 0.1\%$) in the surviving populations and final R_T.

TABLE 6–5: Results for systematic variation of Extra tree parameters, taking the mean and 95% confidence intervals of the discounted reward for 20 trials. The best result is highlighted in boldface.

		M				
		1	10	30	50	70
n_{min}	2	**0.66 ± 0.02**	0.63 ± 0.01	0.63 ± 0.01	0.63 ± 0.02	0.62 ± 0.02
	3	0.61 ± 0.11	0.62 ± 0.02	0.61 ± 0.02	0.62 ± 0.02	0.62 ± 0.01
	4	0.57 ± 0.13	0.62 ± 0.01	0.61 ± 0.02	0.61 ± 0.02	0.62 ± 0.02
	5	0.49 ± 0.22	0.62 ± 0.02	0.63 ± 0.02	0.62 ± 0.01	0.61 ± 0.02
	6	0.57 ± 0.16	0.62 ± 0.01	0.61 ± 0.02	0.60 ± 0.02	0.61 ± 0.02

Sensitivity to variations in α and β

As a final form of validation, we varied the α and β parameters for tumor cells by 5% from their normal values ($\alpha = 1.43$ and $\beta = 0.1296$). Again we found that these variations produced only negligible changes in the optimal solution.

Our conclusion, then, is that the results appear to be quite stable within a region nearby the parameters we have chosen for our RL experiments.

6.4.4 FQI results for $F = 4$

As mentioned previously we began by systematically varying the parameters M, for the number of trees per forest, and n_{min}, the minimum node size that can be split. We chose M from the set $\{1, 10, 30, 50, 70\}$ and $n_{min} = \{2, ..., 6\}$, giving a total of 25 combinations. For each of the possible combinations we performed 20 different experiments by varying the random seed used in the tree randomization, and observed the mean reward of the final policy learned in this manner. The results are summarized in Table 6–5.

The mean results are much less than the known optimal results, presumably because we are sampling only a small portion of the potential policy space. It is somewhat surprising that the best results would be the case where $M = 1$ and $n_{min} = 2$, as this is the mode which corresponds to a single, fully developed tree for each action. However, we suspect that this result is at least in part a byproduct of the rigidly deterministic domain. We know that, even with a relatively large set of

TABLE 6–6: Results for systematic variation of Extra tree parameters, taking the mean and 95% confidence intervals of the discounted reward for 20 trials. In these experiments the initial value of the state variable N_T was chosen using a Gaussian with mean N_0 and standard deviation of $0.02N_0$. The best result is highlighted in boldface.

		M				
		1	10	30	50	70
n_{min}	2	0.60 ± 0.08	0.57 ± 0.10	0.62 ± 0.01	0.62 ± 0.01	0.61 ± 0.02
	3	0.62 ± 0.06	0.61 ± 0.02	0.61 ± 0.02	0.62 ± 0.01	0.63 ± 0.01
	4	$\mathbf{0.66 \pm 0.01}$	0.60 ± 0.04	0.62 ± 0.01	0.62 ± 0.03	0.58 ± 0.09
	5	0.61 ± 0.08	0.54 ± 0.10	0.62 ± 0.01	0.62 ± 0.01	0.62 ± 0.02
	6	0.47 ± 0.20	0.55 ± 0.11	0.60 ± 0.01	0.61 ± 0.01	0.62 ± 0.02

11 actions, there is a finite number of possible states that can be encountered in this domain, the fact we exploited to perform the exhaustive evaluation. Because of this deterministic nature, it follows that the variance reduction offered when $M > 1$ may not be of great importance.

To explore how the problem would change in the presence of a non-deterministic model, we performed an equivalent set of experiments using the randomized perturbation of the initial value of the tumor cell count, as described in Section 6.3.3. In all other respects the experiments were identical. The results of this set of experiments are summarized in Table 6–6.

As expected, the results suggest that random initialization causes the problem to become somewhat more difficult, although the average performance of $R_T = 0.60$ is very close to that of the fixed deterministic case, $R_T = 0.61$. The standard deviation of the policy values found over the 25 different parameter settings are also very similar, being 0.148 in the deterministic case and 0.146 in the random initialization case. While there is some clear evidence for increased approximation error and reduced policy effectiveness for cases where M is small and n_{min} relatively large, we otherwise observe that the results do not show much systematic variation over this large range of parameter settings. We also see some clear evidence for

variance reduction as M grows, consistent with the literature on tree bagging and other randomized approaches to multi-tree classifiers (Breiman, 2001).

However, examining the structure of the trees created shows one way in which the result has changed. In the deterministic case, the trees are quite small. In the case where $M = 1$, an average of 419 tests are required to implement the function approximator with 11 single-tree forests (recall that we have one forest per action). In the comparable case with random initialization, the same function approximator requires an average of 1689 tests.

We examined the policies created on typical runs using the same method we described in Chapter 5. We examined the policies with respect to which features were recruited for the tests at each node in a typical policy. We found a consistent pattern that is exemplified by Figure 6–8. Here, the deterministic case recruits the fraction number F in about ten percent of cases, and recruits the number of tumor and normal cells in roughly equal proportions. The story changes when we examine the random initialization experiment. Here, the number of tumor cells is by far the most-recruited feature, reducing the proportion of each of the other three features. The increased recruitment of the number of tumor cells, N_T, by the function approximator confirms the intuition that, by randomizing the initial value of this variable, we are creating many more state combinations that much be tracked to account for variations in the Q-function. By comparison, the information content of the other, deterministic state variables is reduced relative to N_T.

6.4.5 FQI results for larger values of F

Finally, we performed experiments to see if the FQI procedure could identify the optimal policies for the cases where $F = 10$ and $F = 30$. As previously discussed, intuition suggests that the obvious extension to the 10-stage and 30-stage versions of the 5-stage optimal policy would also be optimal, as long as the constraints of the reward function are not violated. As suggested by our analysis, we

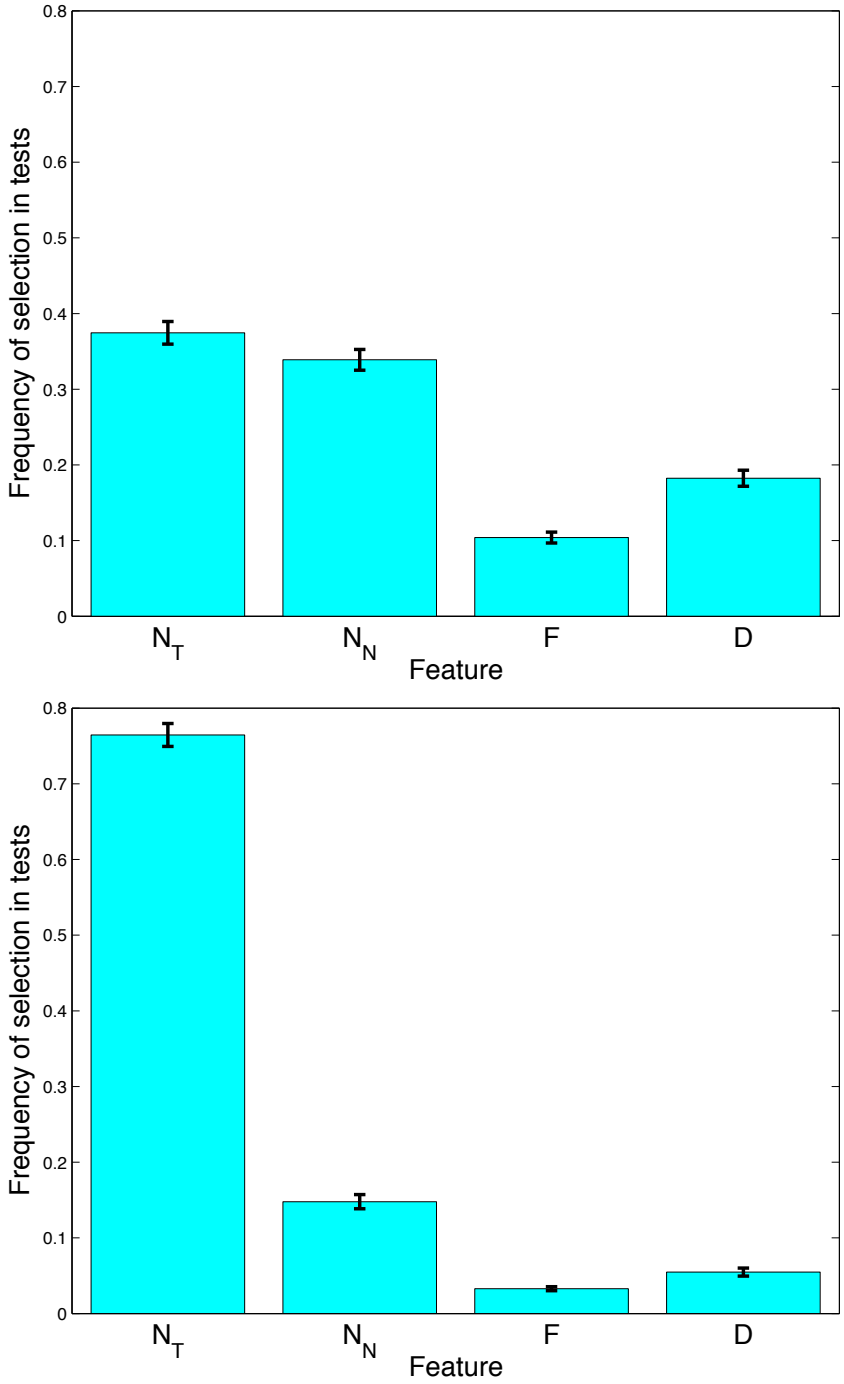

FIGURE 6–8: Comparison of features used in the final Q-function for the fully deterministic (top) and random initialization (bottom) cases. The bars show the fraction of tree nodes for which a particular feature was chosen by the Extremely randomized trees algorithm, averaged over twenty runs with different random number sequences, with error bars showing the 95% confidence interval. In both cases, the Extra tree algorithm parameters were $M = 1$, $K = 4$, and $n_{min} = 2$.

also have reason to believe that the optimal policy may either be unique or statistically rare in these cases, as it is with the shorter schedules.

For these experiments, we used 100 rather than 50 FQI iterations to allow the delayed reward to be fully propagated through the FQI process. In all other respects we used the same parameter choices as in the previous experiments, choosing $M = 1$ and $n_{min} = 2$ with deterministic initialization.

For $F = 10$, the best final policy found in 20 experiments yielded an $R_T = 0.7417$ and a $\sigma_T = 0.11$, only very slightly worse than the policy we derived by extension of the optimal result from the $F = 4$ and $F = 5$ cases.

For $F = 30$, the best final policy found in 20 experiments had an $R_T = 0.53$ and a final $\sigma_T = 0.06$, which is substantially worse than our "known good" policy, $\{0.6, 0.1, 0.1, ...\}$, which yields a $\sigma_T = 0.006$. To achieve even these results, we used a population of 200 rather than 100 model runs per round of FQI. This has the effect of allowing the FQI process to explore a larger part of the state space. Given the size of the policy space, we would not expect to find the optimal policy without a more directed method for searching the space.

Examining the overall pattern of results where of $M = 1$ and $n_{min} = 2$, the results show a much greater variance and much lower mean reward than when $F = 4$ or $F = 5$. When $F = 10$ case, we have a mean of only $R_T = 0.0962$ over 20 different runs of the full FQI algorithm, with a standard deviation of 0.961. With $F = 30$, the mean is $R_T = -1.7363$ with a standard deviation of 2.504.

We explored whether we can change or even improve this somewhat by increasing M, ε, or S. We would predict that increasing M would tend to reduce variance in the results. In contrast, we expect that increasing ε or S might increase the probability of sampling a policy that is closer to the optimal value.

For $F = 10$, increasing the value of ε did not improve the results, yielding a mean $R_T = -0.328$ with standard deviation of 0.947. In contrast, setting $M = 30$

had a major effect, yielding an $R_T = 0.589$ with standard deviation 0.054. Setting $M = 70$ provided only a very small improvement to $R_T = 0.597$ with standard deviation 0.052. Increasing the number of subjects by setting the number of independent "subjects" ($S = 200$) provides roughly the same mean return ($R_T = 0.595$), but further decreases the standard deviation to 0.036.

Similar manipulations with $F = 30$ improved results somewhat. Increasing the number of subjects to $S = 200$, holding the rest of the model constant, gave a mean $R_T = -0.635$ with standard deviation 1.417.

6.5 Discussion

These preliminary results strongly suggest that, at least for some parameter settings, there is value in using a non-uniform policy during radiation therapy. However, there are a number limitations in this research, so we would hesitate to claim that the evidence is overwhelming.

One debatable aspect of the model is the inclusion of the time step (or fraction number in this case) in the state vector. Given that the default model is deterministic, the fraction number should be a sufficient statistic for the optimal value function; that is, given a finite task, a fixed initial state, and a deterministic state transition model, the optimal policy can be expressed as a function of one variable, the time step. Arguably, a "perfect" method for learning the deterministic version of our domain could rely completely on the F state and ignore the other state features. The FQI approach we used does a fair job of modeling the problem despite failing to discover this property of the state features. Once we introduce the random perturbation of the initial state, the variable F clearly loses its status as a sufficient statistic, as is reflected by the change in the distribution of state variables shown in Figure 6–8. In practice, we would argue that the inclusion of this feature in the state is reasonable, because, even in real clinical settings, fractionated radiation therapy

is an example of a finite horizon problem with a number of time steps that is fixed at the onset of treatment.

It is therefore not terribly surprising that the best results with the fully deterministic model occur when the parameter M, the number of trees per forest, is one. Since increased values of M are typically expected to reduce the variance in the Extra trees algorithm (Geurts et al., 2006), this strongly suggests that the system is so predictable that there is no value in reducing variance. As we showed, introducing even a very small stochastic perturbation in the initial conditions has a major effect on the optimal value of the parameter, and reduces the apparent usefulness of the state feature corresponding to the fraction number.

Another questionable aspect of the RL formulation is our assumption, in the reward function, of a hard lower limit on the surviving percentage of normal tissue. While there is some clinical and intuitive justification for this, the use of a hard limit seems somewhat artificial, and introduces a discontinuity into the reward function, which is apparently reflected in the value function. A formulation that preserved continuity might be more justifiable from a biological and clinical standpoint, and may also more tractable from an RL perspective.

While the results with longer, more realistic fractionation schedules are not definitive, it is encouraging that we do see evidence that policies very near the likely optimum can be identified for these longer policies. However, it is unfortunate that the current RL model seems to have great difficulty optimizing longer policies. One aspect of the problem is the large increase in the size of the possible policy space as the length of treatment increases. For a given value of S, we only sample at most $10S$ policies. For $F = 4$, this is already only about 7 percent of the policy space. For $F = 10$, we are sampling less than 4×10^{-6} percent of the policy space, and for $F = 30$ the percentage is vanishingly small. It is very likely that more principled methods for exploration could yield superior results, however we are not aware of

extensive experimental or theoretical work with alternative exploration policies in fitted Q iteration.

As we implied in Sections 6.4.2 and 6.4.5, it is tempting to suppose that our simple 5-step optimal policy might be extended to the 10 or 30 step case in a straightforward manner. In our opinion, this idea merits additional theoretical and empirical investigation. It would be worthwhile to examine whether there might be principled methods for extending policies learned over a short time horizon, and under what conditions such methods might be useful.

Regarding the model itself, there are a number of interesting things to note. Our parameters are somewhat intentionally chosen to give a good contrast between the normal and tumor tissues. Naturally, this will not be the case in some clinical scenarios. In practice, the use of beam forming systems in intensity-modulated radiation therapy (IMRT) means that normal tissue rarely experiences the same dosage as tumor tissue (Boyer, 2002). Therefore, our assumption that the tumor tissue is exposed to the same radiation dosage as the normal tissue would not apply in a true clinical setting.

Perhaps one of the most important results from this work is its demonstration that the fitted Q iteration algorithm is able to perform well on novel domains, over a wide range of parameter choices. We have consistently found that this method yields more predictable results than the on-line methods we have implemented. Our informal results with Sarsa-λ showed much larger variance in the results and greater sensitivity to parameters with many times more data.

6.6 Related Work

While the basic LQ model of radiation therapy is well-established (Fowler, 2011), it has the limitation that it is not a differential model and therefore does not allow for integration of the full time course of radiation therapy. A number of differential models have been proposed, such as those of Tobias (1985), Curtis

(1986), Dale et al. (1999) and Carlone et al. (2005). Most of these models differ from that of Scheidegger et al. (2011) in that they explicitly model the number of lesions caused by radiation, rather than implicitly modeling this with the concept of a "dose equivalent."

Curtis (1986) proposed a model which reduces to the LQ model for low dose rates. While it is hypothetically possible to parameterize the Curtis model with α and β under the low dose rate assumption, this is not straightforward, and estimates of parameters for real cell lines are difficult to obtain. However, the model is arguably more complete than ours in that it explicitly models independent stochastic processes for both radiation-induced lesions and cellular repair (or misrepair). The model also assumes that the time available for repair is finite, and is also expressed as a parameter.

Carlone et al. (2005) proposed a similar, if slightly simpler model. Like Curtis (1986) the model explicitly represents both the lesion and repair processes, and therefore requires additional parameters. Unlike the Curtis model, it assumes an infinite time is available for repair. It is also not straightforward to express the model parameters in terms of α and β.

A number of recent papers have examined the use of adaptive methods for radiation therapy. None have specifically considered reinforcement learning as a paradigm for optimization, but Kim et al. (2009) develops a very closely related model of the problem of fractionation scheduling as a Markov decision process, specifically to explore whether non-uniform fractional doses might have some benefit for the patient. Their results rely on fully specifying the MDP in order to solve for the optimal policy using value iteration. An earlier paper by Ferris and Voelker (2004) examined optimization of radiation therapy fractionation, primarily as a means to compensate for errors that inevitable occur during treatment, rather than considering intrinsic dynamics of radiation response at the cellular level.

Another important line of related research is exemplified by Altman et al. (2006) and Schmitt et al. (2012). In practice, modern IMRT systems produce complex fields which naturally vary during a particular fraction. These studies use modeling techniques and clinical data to argue that optimization of the dosage pattern within a fraction can improve overall probability of tumor control.

The work of de la Zerda et al. (2007) showed a number of possible formulations for closed-loop control of radiation therapy, examining the broader problem of adaptive treatment including replanning the treatment based on updating computed tomography images to reevaluate the patient geometry before delivering each fractional dose. The goal is primarily to overcome problems with errors in dose delivery and patient positioning in prior fractions. It is unclear whether the value of additional imaging steps is well established, given their likely costs.

6.7 Contributions

We have proposed and implemented novel extensions of a differential model for radiation therapy, which takes into account some expected effects on cell proliferation resulting from carrying capacity effects as well as radiation therapy timing. This model implements a richer view of the evolution of tumor and normal cell populations during radiation therapy than is typical of prior models, while continuing to use standard clinical parameters. In time, such models will hopefully become meaningful complements to cell culture experiments.

In addition, we have used this model to formulate the problem of control of tumor cells as a reinforcement learning problem. We have investigated several possible reward functions and solutions methods. Our preliminary findings suggest that, at least for some tissue types, there may be an advantage in using non-uniform fractionation schedules.

6.8 Future Work

There are a number of areas in which this research could be extended. These involve both the computational model itself and the reinforcement learning approach used in this chapter.

It is clear that the differential model itself can be extended in many significant ways. To suggest one example, the delay in regrowth might be dependent on the dosage applied in a fraction, i.e.: T_k. Another possibility would involve implementing the full Lotka-Volterra competitive model, which is probably a more realistic model of cells competing for limited resources. Additionally, as described above, real IMRT systems are capable of varying the dosage delivery pattern within a particular fraction (Altman et al., 2006; Schmitt et al., 2012). It might therefore be practical to apply the methods we used in this chapter to the problem of temporal optimization within a single fraction. Finally, it might be interesting to investigate the implications of relaxing our model's unrealistic assumption that the radiation dosage is uniform across tissue types. By design, most clinical systems will focus a much larger dose on the tumor cells (Fowler, 2011). While we have not had time to explore this idea in detail, preliminary investigations using an arbitrary percentage reduction in the dosage applied to normal tissue suggest that this effect will produce results which are much more consistent with actual tumor control in a clinical setting.

In clinical practice, radiation therapy is often delivered 5 days per week, with a two-day gap during weekends (Kim et al., 2009). It may be useful to expand the definition of the RL task to accommodate this extra practical constraint.

Another useful contribution would involve the extension of the research to consider a fuller range of possible biological constants, especially if such work focuses on understanding critical points in the parameter space where results may

differ qualitatively. For example, we conjecture that there may exist regimes in the parameter space in which radically different policies might be optimal.

Regarding the RL formulation, the reward function still provides less than satisfactory results. A function which was continuous might be easier to optimize, at least in some settings. It would be interesting to explore alternatives here, especially with a more sophisticated model. And we would definitely like to extend this work by examining the use of softmax (Sutton & Barto, 1998) or UCB1 (Auer et al., 2007) as FQI exploration policies in this domain. Improving the reward function and exploration strategies would hopefully lead to faster convergence by focusing policy search on the region of the policy space near the optimum, which, as we described in Section 6.4.2, appears to be quite small relative to the entire policy space.

CHAPTER 7
Conclusion

Adaptive treatment strategies appear to have the potential for improving outcomes in many clinical settings. Advances in medical treatment are being driven by a number of other trends in technology - "big data" and data science, increasingly sophisticated statistical methods, and our ever increasing ability to collect and store data. As we acquire ever more sophisticated methods for data collection and analysis, it becomes increasingly possible to imagine using algorithms to enhance customization and personalization of any number of aspects of human life. Sadly, this is currently most discussed in the contexts of data mining in surveillance (e.g. the NSA) or the targeting of advertising and marketing (e.g. Facebook and Google). In coming years, hopefully, we will see more cases of these techniques being applied in more unambiguously useful contexts, such as medicine. However, many issues remain before these techniques can be widely adopted.

Our work has examined specific problems within the medical literature as case studies for further development as reinforcement learning problems. Perhaps most importantly, we hope that we have provided some practical insights into how reinforcement learning can become a useful framework for developing adaptive treatment strategies. However, a number of major issues remain which may slow or even prevent clinical adoption of RL methods.

7.1 Clinical issues

We have touched on a number of major issues raised in the development of adaptive treatment strategies, but probably none are more important than the specific clinical issues of safety and robustness that need to be addressed before adoption of any medical procedure or device based on reinforcement learning principles.

7.1.1 Safety

Obviously, safety is of paramount importance in medical settings. For good moral and legal reasons there is great reluctance to deploy novel medical treatments for their own sake - they must first be shown to be safe in a very wide range of settings and applications, and any risks must be quantified with great care.

Reinforcement learning methods are inherently at odds with some of these requirements. Practical RL methods are heavily dependent on complex function approximation methods which may not yield readily interpretable policies that can be quickly verified by a human. There are two important methods for improving this situation, first by developing better tools for the empirical analysis of RL methods, but also by developing methods which lend themselves to such analysis. This remains an ongoing area of research in the RL community (Fonteneau et al., 2009; Păduraru et al., 2012; Păduraru, 2013).

Another important method for addressing safety issues would be by enforcing strong constraints on the policy. Nothing precludes setting global limits on the possible behavior of the policy. For example, in the case of implantable electrical stimulation devices, we could place absolute constraints on the total current delivered per unit time, or the maximum voltage or duty cycle. While such constraints may be encoded in a reward function, it might be more reasonable to encode them directly in the policy itself (Altman, 1999; Mannor & Tsitsiklis, 2013).

7.1.2 Robustness

One of the strengths of reinforcement learning is its ability to optimize stochastic processes. RL methods are inherently intended to optimize outcomes in the presence of noise in the form of uncertainty, generally expressed as a distribution over state transitions. What is more difficult is handling incomplete or noisy state variables. Here the reinforcement learning problem moves into the realm of partial

observability, with well-known increases in the difficulty of the problem, in both computational complexity and data requirements.

However, in practice, it seems reasonable to argue that state representations based on clinical measurements and evaluations can be sufficiently informative to be used to form a state representation. What is somewhat less clear is exactly how a policy can be generalized from the data collected for training. Ideally we might like to state with high confidence that the outcomes for an arbitrary patient will be at least as good as some baseline treatment. At present, the theoretical and empirical tools to make this assessment are somewhat lacking, although there has been some recent work on robust MDP solutions which account for the uncertainty of the MDP parameters (Xu & Mannor, 2012).

7.1.3 Interpretability

Perhaps one of the most profound issues with modern machine learning methods, including reinforcement learning, is the lack of clearly interpretable results. Many statistical learning methods are sufficiently complex and obscure in the details of their operation that it is difficult to check them against human expertise, but this is clearly a very important criteria when working in medical domains. It would be extremely useful for researchers to concentrate on systematic methods for expressing RL policies in a manner conducive to interpretation by experts in the relevant application areas. This could be approached in many ways, with some mix of formal evaluation frameworks to introduce better performance bounds, policy summarization and pruning to reduce complexity, and identification of key decision points with significant effects on policy value. Some interesting work on this problem, with some practical examples, can be found in Khan et al. (2009)

Whatever the approach, we believe that without major effort, interpretability will remain a significant barrier to adoption of RL policies in medical domains.

7.2 Modeling of disease processes

Our results demonstrate the value of computational models as a tool for early stage exploration of adaptive treatment strategies. We strongly feel that the continued development of computational models of disease processes is an extremely important field that deserves increased support.

One key implication of this is the need for models which can be useful in practical, engineering settings. The model of epileptiform activity presented in this thesis makes a step in this direction by addressing our model to the specific characteristics of the biological model we used in our experiments. Our extended model of cell population response to radiation therapy also makes improvements towards this general goal.

Obviously, computational models of biological processes are imperfect, as they are subject to various difficult scientific and computational constraints. They are routinely used as examples of the minimum requirements to mimic a particular experimental result, and in this setting they can be quite useful as platforms for the prediction of outcomes of future experimental work. However, in our application, we are more interested in developing models which are as realistic as feasible for their own sake, to guarantee that we are capturing both the large-scale but also the fine-scale structure of a problem, insofar as it is necessary to capture these aspects of the model in our state representation. It is currently still quite difficult to claim that any computational model of a biological phenomenon is sufficiently detailed to yield data that can be directly used to learn a policy that could be subsequently applied to a human patient in a clinical setting. We strongly hope that work continues on both RL and modeling methods which can begin to address these limitations. We are especially encouraged to see the development of models for diabetes which have been approved for use in some preclinical trial settings (Kovatchev et al., 2009).

7.3 RL optimization

Our experimental results provide some useful evidence for further exploration with reinforcement learning in medical problems. We have provided some preliminary evidence that these problems can be approached fruitfully using RL, and hopefully we have described some general methods that might help guide the development of RL approaches to other medical domains.

Our lab's work in epilepsy has provided a demonstration that fully adaptive systems can optimize treatment more effectively than other fixed strategies (Pineau et al., 2009), at least in an *in vitro* model. Recently, devices which implement a responsive stimulation strategy (Kossoff et al., 2004; Sun et al., 2008; Smith et al., 2010) have been given pre-market approval from the FDA (NeuroPace, Inc., 2013), but such systems are presently still quite limited in their actual adaptability, relying on fairly simple paradigms of detection and subsequent stimulation to achieve good results. We feel strongly that further work in this area is likely to show that a more dynamic, fully adaptive approach can further improve clinical results.

We have now shown that RL optimization approaches similar to those applied in our epilepsy research can be used to optimize stimulation in a related but substantially different model of neuronal hypersynchrony in Parkinson's disease. Again, we find some evidence that increasing the degrees of freedom available to the control algorithm may provide superior control over a wider range of patient states than would be achieved using previous techniques.

Further, we adapt the same methodology and show that it may be possible to optimize a radically different type of medical treatment, radiation therapy, using the same family of methods. Our results suggest that, in at least some clinical situations, there may be reasons to consider an adaptive strategy that varies dosage in response to the patient's state. Our approach is very simple, by necessity. We tend to think that, as these model limitations are lifted, we will show that there are in fact larger

gains to be made in the full clinical setting. Our work here has also contributed to understanding this problem by showing some limitations of existing computational models, and exploring some ways in which these might be addressed.

As we have mentioned, our work and others leaves us with strong evidence that the choice of exploration policy can have a significant effect on the quality of the final policy when using fitted Q iteration. We believe that there may be several approaches to improve results with FQI by using methods other than ε-greedy policies with fixed ε. The simplest variation would be to anneal the process by reducing ε on each round of FQI, for example. Alternative methods, however, might include the use of stochastic policies such as softmax (Sutton & Barto, 1998), which balance exploration and exploitation in a more principled and flexible manner. In some settings, it may also be possible to apply bandit algorithms such as UCB1 (Auer et al., 2007) to FQI in a manner similar to its application in Monte Carlo tree search (Kocsis & Szepesvari, 2006).

More generally, we have provided additional evidence of the robustness of batch reinforcement learning methods such as fitted Q iteration. This method, and possibly others like it, is clearly able to derive good policies even from highly randomized data. In fact, increasing the randomization can even lead to improved results by some measures, as we have shown in Chapter 6. We have also discovered hints as to some areas where this method might be improved, such as in the choice of exploration policy.

Finally, we wish to underscore the value of an iterative approach to the design of the state, action, and reward components of an RL formulation. As we showed in Chapter 5, refinements of the action space can have important effects on the size of the policy space and therefore the performance and viability of optimization algorithms. In Chapter 6, we show that the refinement of the model and reward function can be important to avoid a clinically implausible control policy. Computational

models can play an especially important role in this kind of exploratory design process. We think this approach deserves greater systematic attention. We would be curious to explore these practical aspects of solution development, which we suspect are often under-reported in the rush to publish positive findings.

7.4 Summary

We have great hope that computational modeling methods can be successfully applied to reinforcement learning in medical domains. Our research has shown that, at least in some limited ways, such studies can provide guidance for a range of study parameters, such as possible algorithmic methods and the relative importance of some clinical variables. Computational models will most likely never be a substitute for clinical trials, but perhaps they can be used as a low-cost means to guide the design and creation of better adaptive treatment strategies.

We believe that reinforcement learning can be used to great effect to build treatment strategies that may one day be fully adaptive and robust to patient and disease variations. Many technical issues remain, as do many challenges in bridging the cultural differences between medical practitioners and computer scientists. When the time comes, navigating the approvals process for an adaptive treatment strategies derived from reinforcement learning methods will certainly be challenging. Encouraging machine learning researchers to place more emphasis on robustness, safety, and interpretability will be especially important to making this type of research relevant in clinical settings.

References

Abbott, L. F. (1999). Lapicque's introduction of the integrate-and-fire model neuron (1907). *Brain Research Bulletin*, *50*, 303–304.

Abbott, L. F., A., V. J., Sen, K., & Nelson, S. B. (1997). Synaptic depression and cortical gain control. *Science*, *275*, 221–224.

Abdulsalam, H., Skillicorn, D. B., & Martin, P. (2007). Streaming random forests. *11th International Database Engineering and Applications Symposium* (p. 225).

Ackley, D. H., & Littman, M. L. (1990). Generalization and scaling in reinforcement learning. *Advances in Neural Information Processing Systems*, *2*, 550–557.

Adams, B. M., Banks, H. T., Davidian, M., Kwon, H.-D., Tran, H. T., Wynne, S. N., & Rosenberg, E. S. (2005). HIV dynamics: Modeling, data analysis, and optimal treatment protocols. *Journal of Computational and Applied Mathematics*, *184*, 10–49. Special Issue on Mathematics Applied to Immunology.

Adams, B. M., Banks, H. T., Kwon, H.-D., & Tran, H. T. (2004). Dynamic multidrug therapies for HIV: optimal and STI control approaches. *Math Biosci Eng*, *1*, 223–41.

Albus, J. S. (1971). A theory of cerebellar function. *Mathematical Biosciences*, *10*, 25–61.

Altman, E. (1999). *Constrained Markov decision processes*, vol. 7. CRC Press.

Altman, M. B., Chmura, S. J., Deasy, J. O., & Roeske, J. C. (2006). Optimization of the temporal pattern of radiation: an IMRT based study. *Int. J. Radiat. Oncol. Biol. Phys.*, *66*, 898–905.

Amari, S. (1998). Natural gradient works efficiently in learning. *Neural Computation*, *10*, 251.

Antos, A., Munos, R., & Szepesvári, C. (2007). Fitted Q-iteration in continuous action-space MDPs. In J. C. Platt, D. Koller, Y. Singer and S. Roweis (Eds.), *Advances in neural information processing systems 20*, 9–16. Cambridge, MA: MIT Press.

Atkeson, C. G., Moore, A. W., & Schaal, S. (1997). Locally weighted learning for control. *Artificial Intelligence Review*, *11*, 75–113.

Auer, P., Ortner, R., & Szepesvári, C. (2007). Improved rates for the stochastic continuum-armed bandit problem. *Learning Theory*, 454–468.

Avoli, M., D'Antuono, M., Louvel, J., Köhling, R., Biagini, G., Pumain, R., D'Arcangelo, G., & Tancredi, V. (2002). Network and pharmacological mechanisms leading to epileptiform synchronization in the limbic system in vitro. *Prog. Neurobiol.*, *68*, 167–207.

Baird, III, L. C., & Klopf, A. H. (1993). *Reinforcement learning with high-dimensional continuous actions* (Technical Report US Air Force WL-TR-93-1147). Wright Laboratory, Wright-Patterson Air Force Base, OH.

Barreto, A. M. S. (2014). Tree-based on-line reinforcement learning. *Proceedings of the 28th AAAI Conference*. Quebec City.

Bazhenov, M., Timofeev, I., Steriade, M., & Sejnowski, T. J. (2004). Potassium model for slow (2-3 Hz) in vivo neocortical paroxysmal oscillations. *J. Neurophysiol.*, *92*, 1116–32.

Bellman, R. (1957). *Dynamic programming*. Princeton, NJ: Princeton University Press.

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, *18*, 509.

Bentzen, S. M., Thames, H. D., & Overgaard, M. (1989). Latent-time estimation for late cutaneous and subcutaneous radiation reactions in a single-follow-up clinical study. *Radiotherapy and Oncology*, *15*, 267.

Beuter, A., & Modolo, J. (2012). Closed loop control of brain rhythms. In E. Mosekilde, O. Sosnovtseva and A. Rostami-Hodjegan (Eds.), *Biosimulation in biomedical research, health care and drug development*. Wien: Springer-Verlag.

Biswal, B., & Dasgupta, C. (2002a). Neural network model for apparent deterministic chaos in spontaneously bursting hippocampal slices. *Physical Review Letters*, *88*, 088102.

Biswal, B., & Dasgupta, C. (2002b). Stochastic neural network model for spontaneous bursting in hippocampal slices. *Physical Review E*, *66*, 051908.

Boada, M. J. L., Egido, V., Barber, R., & Salichs, M. A. (2002). Continuous reinforcement learning algorithm for skills learning in an autonomous mobile robot. *Proceedings of the 28th Annual Conference of the IEEE Industrial Electronics Society*, *4*, 2611–2616.

Bonet, B., & Geffner, H. (2006). Learning depth-first search: A unified approach to heuristic search in deterministic and non-deterministic settings, and its application to MDPs. *International Conference on Automated Planning and Scheduling* (pp. 142–151).

Bothe, M. K., Dickens, L., Reichel, K., Tellmann, A., Ellger, B., Westphal, M., & Faisal, A. A. (2013). The use of reinforcement learning algorithms to meet the challenges of an artificial pancreas. *Expert Rev Med Devices*, *10*, 661–73.

Boutilier, C., Dean, T., & Hanks, S. (1999). Decision theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, *11*, 1–94.

Boutilier, C., Dearden, R., & Goldszmidt, M. (2000). Stochastic dynamic programming with factored representations. *Artificial Intelligence*, *121*, 49.

Boyer, A. L. (2002). The physics of intensity-modulated radiation therapy. *Physics Today*, *55*, 38–44.

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

Breiman, L. (2001). Random forests. *Machine learning*, *45*, 5–32.

Bush, K., & Pineau, J. (2009). Manifold embeddings for model-based reinforcement learning under partial observability. *Advances in Neural Information Processing Systems*, *22*, 189–197.

Busoniu, L. (2009). *Reinforcement learning in continuous state and action spaces*. Doctoral dissertation, TU Delft, Delft, Netherlands.

Carlone, M., Wilkins, D., & Raaphorst, P. (2005). The modified linear-quadratic model of Guerrero and Li can be derived from a mechanistic basis and exhibits linear-quadratic-linear behaviour. *Physics in Medicine and Biology, Volume 50, Issue 10, pp. L9-L13 (2005).*, *50*, L9–L13.

Chapman, J. D. (2003). Single-hit mechanism of tumour cell killing by radiation. *International Journal of Radiation Biology*, *79*, 71.

Chiu, A. W. L., & Bardakjian, B. L. (2004). Control of state transitions in an in silico model of epilepsy using small perturbations. *IEEE transactions on biomedical engineering*, *51*, 1856–9.

Cressman, J. R., Ullah, G., Ziburkus, J., Schiff, S. J., & Barreto, E. (2009). The influence of sodium and potassium dynamics on excitability, seizures, and the stability of persistent states: I. Single neuron dynamics. *Journal of computational neuroscience*, *26*, 159–70.

Crites, R. H., & Barto, A. G. (1998). Elevator group control using multiple reinforcement learning agents. *Machine Learning*, *33*, 235.

Crowder, J. M., Croucher, M. J., Bradford, H. F., & Collins, J. F. (1987). Excitatory amino acid receptors and depolarization-induced Ca^{2+} influx into hippocampal slices. *J. Neurochem.*, *48*, 1917–24.

Curtis, S. B. (1986). Lethal and potentially lethal lesions induced by radiation–a unified repair model. *Radiat. Res.*, *106*, 252–70.

da Silva, F. L., Blanes, W., Kalitzin, S. N., Parra, J., Suffczynski, P., & Velis, D. N. (2003). Epilepsies as dynamical diseases of brain systems: Basic models of the transition between normal and epileptic activity. *Epilepsia*, *44*, 72.

Dale, R. G., Fowler, J. F., & Jones, B. (1999). A new incomplete-repair model based on a 'reciprocal-time' pattern of sublethal damage repair. *Acta Oncologica*, *38*, 919–929.

D'Arcangelo, G., Panuccio, G., Tancredi, V., & Avoli, M. (2005). Repetitive low-frequency stimulation reduces epileptiform synchronization in limbic neuronal networks. *Neurobiol. Dis.*, *19*, 119–28.

Dayan, P., & Abbott, L. F. (2001). *Theoretical neuroscience: Computational and mathematical modeling of neural systems*. Cambridge, MA: MIT Press.

de la Zerda, A., Armbruster, B., & Xing, L. (2007). Formulating adaptive radiation therapy (ART) treatment planning into a closed-loop control framework. *Phys Med Biol*, *52*, 4137–53.

Destexhe, A., Mainen, Z. F., & Sejnowski, T. J. (1998). Kinetic models of synaptic transmission. In C. Koch and I. Segev (Eds.), *Methods in neuronal modeling: From ions to networks*, 1–25. Cambridge, Massachusetts: Massachusetts Institute of Technology.

Dietterich, T. G. (2002). Machine learning for sequential data: A review. *Structural Syntactic and Statistical Pattern Recognition* (pp. 15–30). Berlin Heidelberg: Springer-Verlag.

Durand, D., & Bikson, M. (2001). Suppression and control of epileptiform activity by electrical stimulation: A review. *Proceedings of the IEEE*, *89*, 1065.

During, M. J., & Spencer, D. D. (1992). Adenosine: a potential mediator of seizure arrest and postictal refractoriness. *Ann. Neurol.*, *32*, 618–24.

Echauz, J., Firpi, H., & Georgoulas, G. (2009). Intelligent control strategies for neurostimulation. In K. P. Valavanis (Ed.), *Applications of intelligent control to engineering systems*, vol. 39, chapter 10, 247–263. Springer Netherlands.

El Naqa, I. (2012). personal communication.

Engel, Y., Mannor, S., & Meir, R. (2005). Reinforcement learning with Gaussian processes. *Proceedings of the 22nd international conference on Machine learning* (pp. 201–208).

Ernst, D., Geurts, P., & Wehenkel, L. (2005a). Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, *6*, 503–556.

Ernst, D., Glavic, M., Geurts, P., & Wehenkel, L. (2005b). Approximate value iteration in the reinforcement learning context. application to electrical power system control. *International Journal of Emerging Electric Power Systems*, *3*, 1066–1.

Ernst, D., Stan, G.-B., Gonçalves, J., & Wehenkel, L. (2006). Clinical data based optimal STI strategies for HIV: A reinforcement learning approach. *15th Machine Learning Conference of Belgium and The Netherlands (Benelearn 2006)*. Ghent, Belgium.

Fard, M. M., Grinberg, Y., Farahmand, A. M., Pineau, J., & Precup, D. (2013). Bellman error based feature generation using random projections on sparse spaces. *Neural Information Processing Systems* (pp. 3030–3038).

Ferris, M. C., & Voelker, M. M. (2004). Fractionation in radiation treatment planning. *Mathematical Programming*, *101*, 387–413.

Fonteneau, R., Murphy, S., Wehenkel, L., & Ernst, D. (2009). Inferring bounds on the performance of a control policy from a sample of trajectories. *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning* (p. 117).

Fowler, J. F. (2011). Practical time–dose evaluations, or how to stop worrying and learn to love linear quadratics. In S. H. Levitt, J. A. Purdy, C. A. Perez and

P. Poortmans (Eds.), *Technical basis of radiation therapy*, 3–50. Berlin Heidelberg: Springer-Verlag.

Franaszczuk, P. J., Kudela, P., & Bergey, G. K. (2003). External excitatory stimuli can terminate bursting in neural network models. *Epilepsy Res.*, *53*, 65–80.

Fröhlich, F., Bazhenov, M., Iragui-Madoz, V., & Sejnowski, T. J. (2008). Potassium dynamics in the epileptic cortex: New insights on an old topic. *The Neuroscientist*, *14*, 422–33.

Fröhlich, F., Sejnowski, T. J., & Bazhenov, M. (2010). Network bistability mediates spontaneous transitions between normal and pathological brain states. *J. Neurosci.*, *30*, 10734–43.

Fujikawa, D. G., Kim, J. S., Daniels, A. H., Alcaraz, A. F., & Sohn, T. B. (1996). In vivo elevation of extracellular potassium in the rat amygdala increases extracellular glutamate and aspartate and damages neurons. *Neuroscience*, *74*, 695–706.

Gaskett, C., Wettergreen, D., & Zelinsky, A. (1999). Q-learning in continuous state and action spaces. *Australian Joint Conference on Artificial Intelligence*, 417–428.

Gaweda, A. E., Muezzinoglu, M. K., Aronoff, G. R., Jacobs, A. A., Zurada, J. M., & Brier, M. E. (2007). Using clinical information in goal-oriented learning. *IEEE Engineering in Medicine and Biology Magazine*, *26*, 27–36.

Gelly, S., & Silver, D. (2007). Combining online and offline knowledge in UCT. *Proceedings of the International Conference on Machine Learning* (pp. 273–280).

Geramifard, A., Doshi, F., Redding, J., Roy, N., & How, J. (2011). Online discovery of feature dependencies. *Proceedings of the 28th International Conference on Machine Learning* (pp. 881–888).

Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, *63*, 3–42.

Ghilezan, M., Yan, D., & Martinez, A. (2010). Adaptive radiation therapy for prostate cancer. *Seminars in radiation oncology* (pp. 130–7).

Golomb, D., & Amitai, Y. (1997). Propagating neuronal discharges in neocortical slices: computational and experimental study. *J. Neurophysiol.*, *78*, 1199–211.

Golomb, D., Shedmi, A., Curtu, R., & Ermentrout, G. B. (2006). Persistent synchronized bursting activity in cortical tissues with low magnesium concentration: a modeling study. *J. Neurophysiol.*, *95*, 1049–67.

Gordon, G. J. (1999). *Approximate solutions to Markov decision processes*. Doctoral dissertation, Carnegie-Mellon University, Pittsburgh, PA.

Gosavi, A. (2004). Reinforcement learning for long-run average cost. *European Journal of Operational Research*, *155*, 654–674.

Guestrin, C., Koller, D., Parr, R., & Venkataraman, S. (2003). Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research*, *19*, 399–468.

Guez, A. (2010). Adaptive control of epileptic seizures using reinforcement learning. Master's thesis, McGill University, Montreal, Quebec, Canada.

Guez, A., Vincent, R. D., Avoli, M., & Pineau, J. (2008). Adaptive treatment of epilepsy via batch-mode reinforcement learning. *Innovative Applications of Artificial Intelligence* (pp. 1671–1678).

Heinemann, U., Lux, H. D., & Gutnick, M. J. (1977). Extracellular free calcium and potassium during paroxysmal activity in the cerebral cortex of the cat. *Experimental Brain Research*, *27*, 237–243.

Hodgkin, A. L., & Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol. (Lond.)*, *117*, 500–44.

Hoey, J., St-Aubin, R., Hu, A., & Boutilier, C. (1999). SPUDD: Stochastic planning using decision diagrams. *Proceedings of the 15th Conference on Uncertainty in*

Artificial Intelligence, 279–288.

Huguenard, J. R., & Prince, D. A. (1992). A novel T-type current underlies prolonged Ca^{2+}-dependent burst firing in GABAergic neurons of rat thalamic reticular nucleus. *J. Neurosci.*, *12*, 3804–17.

Igelström, K. M. (2013). Is slack an intrinsic seizure terminator? *The Neuroscientist*, *19*, 248–254.

Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE transactions on neural networks*, *14*, 1569–72.

Izhikevich, E. M. (2004). Which model to use for cortical spiking neurons? *IEEE transactions on neural networks*, *15*, 1063–70.

Jahr, C. E., & Stevens, C. F. (1990). A quantitative description of NMDA receptor-channel kinetic behavior. *J. Neurosci.*, *10*, 1830–7.

Jensen, M. S., Azouz, R., & Yaari, Y. (1994). Variant firing patterns in rat hippocampal pyramidal cells modulated by extracellular potassium. *J. Neurophysiol.*, *71*, 831–9.

Jensen, M. S., Cherubini, E., & Yaari, Y. (1993). Opponent effects of potassium on $GABA_A$-mediated postsynaptic inhibition in the rat hippocampus. *J. Neurophysiol.*, *69*, 764–71.

Joiner, M. C., van der Kogel, A. J., & Steel, G. G. (2009). Introduction: the significance of radiobiology and radiotherapy for cancer treatment. In van der A. J. Kogel and M. C. Joiner (Eds.), *Basic clinical radiobiology*, 1–10. Hodder Arnold.

Kaelbling, L. P., Littman, M. L., & Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, *101*, 99–134.

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, *4*, 237–285.

Kager, H., Wadman, W. J., & Somjen, G. G. (2000). Simulated seizures and spreading depression in a neuron model incorporating interstitial space and ion concentrations. *J. Neurophysiol.*, *84*, 495–512.

Kalyanakrishnan, S., & Stone, P. (2007). Batch reinforcement learning in a complex domain. *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems - AAMAS '07* (p. 1).

Kearns, M., & Singh, S. (1998). Near-optimal reinforcement learning in polynomial time. *Proceedings of the 15th International Conference on Machine Learning*, 260–268.

Kearns, M., & Singh, S. (2002). Near-optimal reinforcement learning in polynomial time. *Machine Learning*, *49*, 209–232.

Khan, O. Z., Poupart, P., & Black, J. P. (2009). Minimal sufficient explanations for factored Markov decision processes. *19th International Conference on Automated Planning and Scheduling*. Thessaloniki, Greece.

Kim, M., Ghate, A., & Phillips, M. H. (2009). A Markov decision process approach to temporal modulation of dose fractions in radiation therapy planning. *Phys Med Biol*, *54*, 4455–76.

Klatt, J., Feldwisch-Drentrup, H., Ihle, M., Navarro, V., Neufang, M., Teixeira, C., Adam, C., Valderrama, M., Alvarado-Rojas, C., Witon, A., Le Van Quyen, M., Sales, F., Dourado, A., Timmer, J., Schulze-Bonhage, A., & Schelter, B. (2012). The EPILEPSIAE database: an extensive electroencephalography database of epilepsy patients. *Epilepsia*, *53*, 1669–76.

Klopf, A. H. (1972). *Brain function and adaptive systems: a heterostatic theory* (Technical Report AFCRL-72-0164). Air Force Cambridge Research Laboratories, Bedford, MA.

Kocsis, L., & Szepesvari, C. (2006). Bandit based Monte-Carlo planning. *European Conference on Machine Learning* (p. 282).

Kolter, J. Z., & Ng, A. Y. (2009). Regularization and feature selection in least-squares temporal difference learning. *Proceedings of the 26th annual International Conference on Machine Learning* (pp. 521–528).

Koo, B., Ham, S. D., Sood, S., & Tarver, B. (2001). Human vagus nerve electrophysiology: a guide to vagus nerve stimulation parameters. *Journal of clinical neurophysiology*, *18*, 429–33.

Kossoff, E. H., Ritzl, E. K., Politsky, J. M., Murro, A. M., Smith, J. R., Duckrow, R. B., Spencer, D. D., & Bergey, G. K. (2004). Effect of an external responsive neurostimulator on seizures and electrographic discharges during subdural electrode monitoring. *Epilepsia*, *45*, 1560–7.

Kovatchev, B., Breton, M., & Johnson, B. (2012). In silico models of alcohol dependence and treatment. *Front Psychiatry*, *3*, 4.

Kovatchev, B. P., Breton, M., Dalla Man, C., & Cobelli, C. (2009). In silico preclinical trials: a proof of concept in closed-loop control of type 1 diabetes. *Journal of diabetes science and technology*, *3*, 44–55.

Krishnan, G. P., & Bazhenov, M. (2011). Ionic dynamics mediate spontaneous termination of seizures and postictal depression state. *J. Neurosci.*, *31*, 8870–82.

Kwan, P., & Brodie, M. J. (2000). Early identification of refractory epilepsy. *N. Engl. J. Med.*, *342*, 314–9.

Lagoudakis, M. G., & Parr, R. (2003). Least-squares policy iteration. *Journal of Machine Learning Research*, *4*, 1107.

Läuger, P. (1991). *Electrogenic ion pumps*. Distinguished lecture series of the Society of General Physiologists. Sunderland, MA: Sinauer.

Lazaric, A., Restelli, M., & Bonarini, A. (2007). Reinforcement learning in continuous action spaces through sequential Monte Carlo methods. *Advances in Neural Information Processing Systems*, *20*.

Leveson, N., & Turner, C. (1993). An investigation of the Therac-25 accidents. *Computer*, *26*, 18.

Li, T., Thongphiew, D., Zhu, X., Lee, W. R., Vujaskovic, Z., Yin, F.-F., & Wu, Q. J. (2011). Adaptive prostate IGRT combining online re-optimization and re-positioning: a feasibility study. *Phys Med Biol*, *56*, 1243–58.

Liem, E. B., Lin, C.-M., Suleman, M.-I., Doufas, A. G., Gregg, R. G., Veauthier, J. M., Loyd, G., & Sessler, D. I. (2004). Anesthetic requirement is increased in redheads. *Anesthesiology*, *101*, 279–83.

Lizotte, D. J., Gunter, L., Laber, E., & Murphy, S. A. (2008). Missing data and uncertainty in batch reinforcement learning. *NIPS-08 Workshop on Model Uncertainty and Risk in Reinforcement Learning*.

Localio, A. R., Berlin, J. A., Ten Have, T. R., & Kimmel, S. E. (2001). Adjustments for center in multicenter studies: an overview. *Annals of internal medicine*, *135*, 112–123.

Löscher, W., & Köhling, R. (2010). Functional, metabolic, and synaptic changes after seizures as potential targets for antiepileptic therapy. *Epilepsy Behav*, *19*, 105–13.

Lotka, A. J. (1932). The growth of mixed populations: two species competing for a common food supply. *J. Wash. Acad. Sci.*, *22*, 461–469.

Lytton, W. W., & Omurtag, A. (2007). Tonic-clonic transitions in computer simulation. *Journal of clinical neurophysiology*, *24*, 175–81.

Mahadevan, S. (1996). Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine learning*, *22*, 159–195.

Malof, J. M., & Gaweda, A. E. (2011). Optimizing drug therapy with reinforcement learning: The case of anemia management. *The 2011 International Joint Conference on Neural Networks* (p. 2088).

Mannor, S., & Tsitsiklis, J. N. (2013). Algorithmic aspects of mean–variance optimization in Markov decision processes. *European Journal of Operational Research*, *231*, 645–653.

Mansley, C., Weinstein, A., & Littman, M. L. (2011). Sample-based planning for continuous action Markov decision processes. *Proc. 21st Int. Conf. Automat. Plan. Sched., Freiburg, Germany* (pp. 335–338).

Martín H., J. A., & de Lope, J. (2009). Ex$\langle a \rangle$: An effective algorithm for continuous actions reinforcement learning problems. *35th Annual Conference of IEEE Industrial Electronics* (p. 2063).

Martín H., J. A., de Lope, J., & Maravall, D. (2011). Robust high performance reinforcement learning through weighted k-nearest neighbors. *Neurocomputing*, 1251–1259.

Matarić, M. J. (2001). Learning in behavior-based multi-robot systems: Policies, models, and other agents. *Cognitive Systems Research*, *2*, 81–93.

MATLAB (2013). *Version 8.1.0.604*. Natick, Massachusetts: The Mathworks, Inc.

McKay, J. R. (2009). *Treating substance use disorders with adaptive continuing care*. Washington, DC: American Psychological Association.

Millán, J. R., Posenato, D., & Dedieu, E. (2002). Continuous-action Q-learning. *Machine Learning*, *49*, 247–265.

Molenberghs, G., & Kenward, M. G. (2007). *Missing data in clinical studies*. Statistics in Practice. Wiley.

Moore, A. W., & Atkeson, C. G. (1993). Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, *13*, 103.

Murphy, S. A. (2005). An experimental design for the development of adaptive treatment strategies. *Statistics in medicine*, *24*, 1455–81.

Murphy, S. A., Lynch, K. G., Oslin, D., McKay, J. R., & TenHave, T. (2007). Developing adaptive treatment strategies in substance abuse research. *Drug and*

alcohol dependence, *88 Suppl 2*, S24–30.

Netoff, T. I., Clewley, R., Arno, S., Keck, T., & White, J. A. (2004). Epilepsy in small-world networks. *J. Neurosci.*, *24*, 8075–83.

NeuroPace, Inc. (2013). FDA grants premarket approval (PMA) for the NeuroPace RNS system to treat medically refractory epilepsy. Press release.

Ng, A. Y., Harada, D., & Russell, S. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. *Proceedings of the 16th International Conference on Machine Learning* (pp. 278–287).

Ng, A. Y., & Jordan, M. (2000). PEGASUS: A policy search method for large MDPs and POMDPs. *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, 406–415.

Ohayon, E., Kwan, H., Burnham, W., Suffczynski, P., & Kalitzin, S. (2004). Emergent complex patterns in autonomous distributed systems: mechanisms for attention recovery and relation to models of clinical epilepsy. *2004 IEEE International Conference on Systems Man and Cybernetics* (p. 2066).

Ormoneit, D., & Sen, S. (2002). Kernel-based reinforcement learning. *Machine Learning*, *49*, 161–178.

O'Rourke, S. F. C., McAneney, H., & Hillen, T. (2009). Linear quadratic and tumour control probability modelling in external beam radiotherapy. *J Math Biol*, *58*, 799–817.

Panuccio, G., Guez, A., Vincent, R., Avoli, M., & Pineau, J. (2013). Adaptive control of epileptiform excitability in an in vitro model of limbic seizures. *Exp. Neurol.*, *241*, 179–83.

Pazis, J., & Lagoudakis, M. (2009). Learning continuous-action control policies. *IEEE International Symposium on Adaptive Dynamic Programming and Reinforcement Learning*. Nashville, TN.

Perreault, P., & Avoli, M. (1991). Physiology and pharmacology of epileptiform activity induced by 4-aminopyridine in rat hippocampal slices. *J. Neurophysiol.*, *65*, 771–85.

Peters, J., & Schaal, S. (2008). Natural actor-critic. *Neurocomputing*, *71*, 1180.

Pineau, J., Bellemare, M. G., Rush, A. J., Ghizaru, A., & Murphy, S. A. (2007). Constructing evidence-based treatment strategies using methods from computer science. *Drug and alcohol dependence*, *88 Suppl 2*, S52–60.

Pineau, J., Gordon, G., & Thrun, S. (2003). Point-based value iteration: An anytime algorithm for POMDPs. *International joint conference on Artificial Intelligence* (pp. 1025–1032).

Pineau, J., Guez, A., Vincent, R., Panuccio, G., & Avoli, M. (2009). Treating epilepsy via adaptive neurostimulation: A reinforcement learning approach. *International Journal of Neural Systems*, *19*, 227–240.

Poolos, N. P., & Kocsis, J. D. (1990). Elevated extracellular potassium concentration enhances synaptic activation of N-methyl-D-aspartate receptors in hippocampus. *Brain Res.*, *508*, 7–12.

Popovych, O. V., Hauptmann, C., & Tass, P. A. (2005). Effective desynchronization by nonlinear delayed feedback. *Physical Review Letters*, *94*, 164102.

Potti, A., Schilsky, R. L., & Nevins, J. R. (2010). Refocusing the war on cancer: the critical role of personalized treatment. *Science translational medicine*, *2*, 28–13.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (2007). *Numerical recipes: The art of scientific computing*. Cambridge University Press. Third edition.

Păduraru, C. (2013). *Off-policy evaluation in Markov decision processes*. Doctoral dissertation, McGill University, Montreal, Quebec, Canada.

Păduraru, C., Precup, D., & Pineau, J. (2012). A framework for computing bounds for the return of a policy. In *Recent advances in reinforcement learning*, 201–212.

Springer.

Puterman, M. L. (2005). *Markov decision processes: discrete stochastic dynamic programming*. Wiley-Interscience.

Rivera, C., Voipio, J., Thomas-Crusells, J., Li, H., Emri, Z., Sipilä, S., Payne, J. A., Minichiello, L., Saarma, M., & Kaila, K. (2004). Mechanism of activity-dependent downregulation of the neuron-specific K-Cl cotransporter KCC2. *J. Neurosci.*, *24*, 4683–91.

Robbins, H., & Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, 400–407.

Rosin, B., Slovik, M., Mitelman, R., Rivlin-Etzion, M., Haber, S. N., Israel, Z., Vaadia, E., & Bergman, H. (2011). Closed-loop deep brain stimulation is superior in ameliorating Parkinsonism. *Neuron*, *72*, 370–84.

Ross, S., Pineau, J., Chaib-draa, B., & Kreitmann, P. (2011). A Bayesian approach for learning and planning in partially observable Markov decision processes. *Journal of Machine Learning Research*, 1729–1770.

Rubin, D. B. (1976). Inference and missing data. *Biometrika*, *63*, 581–592.

Rulkov, N. F. (2008). Oscillations and synchrony in large-scale cortical network models. *Journal of Biological Physics*, *34*, 279.

Rummery, G. A., & Niranjan, M. (1994). *On-line Q-learning using connectionist systems* (Technical Report CUED/F-INFENG/TR 166). Cambridge University Engineering Department.

Rutecki, P. (1990). Anatomical, physiological, and theoretical basis for the antiepileptic effect of vagus nerve stimulation. *Epilepsia*, *31*, S1–S6.

Santamaria, J. C., Sutton, R. S., & Ram, A. (1997). Experiments with reinforcement learning in problems with continuous state and action spaces. *Adaptive Behavior*, *6*, 163.

Scheidegger, S., Lutters, G., & Bodis, S. (2011). A LQ-based kinetic model formulation for exploring dynamics of treatment response of tumours in patients. *Z Med Phys*, *21*, 164–73.

Schiff, S. J., & Sauer, T. (2008). Kalman filter control of a model of spatiotemporal cortical dynamics. *Journal of neural engineering*, *5*, 1–8.

Schindler, K., Elger, C. E., & Lehnertz, K. (2007). Increasing synchronization may promote seizure termination: evidence from status epilepticus. *Clin Neurophysiol*, *118*, 1955–68.

Schmitt, J. D., Warren, G. W., & Wang, I. Z. (2012). Potential increase in biological effectiveness from field timing optimization for stereotactic body radiation therapy. *Med Phys*, *39*, 2956–63.

Shoeb, A., Pang, T., Guttag, J., & Schachter, S. (2009). Non-invasive computerized system for automatically initiating vagus nerve stimulation following patient-specific detection of seizures or epileptiform discharges. *Int J Neural Syst*, *19*, 157–172.

Shortreed, S. M., Laber, E., Lizotte, D. J., Stroup, T. S., Pineau, J., & Murphy, S. A. (2011). Informing sequential clinical decision-making through reinforcement learning: an empirical study. *Mach Learn*, *84*, 109–136.

Shorvon, S., Guerrini, R., Cook, M., & Lhatoo, S. (Eds.). (2013). *Oxford textbook of epilepsy and epileptic seizures*. Oxford Textbooks in Clinical Neurology. Oxford, United Kingdom: Oxford University Press.

Silver, D., & Veness, J. (2010). Monte-Carlo planning in large POMDPs. *Advances in Neural Information Processing Systems*, *23*, 2164–2172.

Singh, S. P., & Sutton, R. S. (1996). Reinforcement learning with replacing eligibility traces. *Machine Learning*, *22*, 123.

Smart, W. D., & Kaelbling, L. P. (2000). Practical reinforcement learning in continuous spaces. *Proceedings of the Seventeenth International Conference on Machine Learning*, 903–910.

Smith, A. J. (2002). Applications of the self-organising map to reinforcement learning. *Neural Networks*, *15*, 1107–1124.

Smith, J. R., Fountas, K. N., Murro, A. M., Park, Y. D., Jenkins, P. D., Morrell, M., Esteller, R., & Greene, D. (2010). Closed-loop stimulation in the control of focal epilepsy of insular origin. *Stereotact Funct Neurosurg*, *88*, 281–7.

Sondik, E. J. (1971). *The optimal control of partially observable Markov processes.* Doctoral dissertation, Stanford University, Stanford, California.

Sterne, P. J. (2004). Reinforcement sailing. Master's thesis, University of Edinburgh.

Strehl, A. L., & Littman, M. L. (2005). A theoretical analysis of model-based interval estimation. *Proceedings of the 22nd International Conference on Machine Learning* (pp. 856–863).

Strens, M. (2000). A Bayesian framework for reinforcement learning. *Proceedings of the Seventeenth International Conference on Machine Learning* (pp. 943–950).

Su, H., Alroy, G., Kirson, E. D., & Yaari, Y. (2001). Extracellular calcium modulates persistent sodium current-dependent burst-firing in hippocampal pyramidal neurons. *J. Neurosci.*, *21*, 4173–82.

Suffczynski, P., Kalitzin, S., & Lopes Da Silva, F. H. (2004). Dynamics of non-convulsive epileptic phenomena modeled by a bistable neuronal network. *Neuroscience*, *126*, 467–84.

Sun, F. T., Morrell, M. J., & Wharen, R. E. (2008). Responsive cortical stimulation for the treatment of epilepsy. *Neurotherapeutics*, *5*, 68–74.

Sutton, R. S. (1996). Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Advances in neural information processing systems*, *8*, 1038–1044.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.

Swartz, M. S., Perkins, D. O., Stroup, T. S., McEvoy, J. P., Nieri, J. M., & Haak, D. C. (2003). Assessing clinical and functional outcomes in the clinical antipsychotic trials of intervention effectiveness (CATIE) schizophrenia trial. *Schizophrenia bulletin*, *29*, 33–43.

Tass, P. A. (1999). *Phase resetting in medicine and biology: stochastic modelling and data analysis*, vol. 172 of *Springer Series in Synergetics*. Berlin Heidelberg New York: Springer. First edition.

Tass, P. A. (2003). A model of desynchronizing deep brain stimulation with a demand-controlled coordinated reset of neural subpopulations. *Biol Cybern*, *89*, 81–8.

Tass, P. A., & Majtanik, M. (2006). Long-term anti-kindling effects of desynchronizing brain stimulation: a theoretical study. *Biol Cybern*, *94*, 58–66.

Tateno, K., Hayashi, H., & Ishizuka, S. (1998). Complexity of spatiotemporal activity of a neural network model which depends on the degree of synchronization. *Neural Networks*, *11*, 985–1003.

Tesauro, G. (1992). Practical issues in temporal difference learning. *Machine Learning*, *8*, 257–277.

Tesauro, G. (1995). Temporal difference learning and TD-Gammon. *Communications of the ACM*, *38*, 58–68.

Tobias, C. A. (1985). The repair-misrepair model in radiobiology: Comparison to other models. *Radiation Research*, *104*, –77.

Traub, R. D., Bibbig, R., Piechotta, A., Draguhn, R., & Schmitz, D. (2001). Synaptic and nonsynaptic contributions to giant IPSPs and ectopic spikes induced by 4-aminopyridine in the hippocampus in vitro. *J. Neurophysiol.*, *85*, 1246–56.

Traub, R. D., Colling, S. B., & Jefferys, J. G. (1995). Cellular mechanisms of 4-aminopyridine-induced synchronized after-discharges in the rat hippocampal slice. *J. Physiol. (Lond.)*, *489 (Pt 1)*, 127–40.

Traub, R. D., Contreras, D., Cunningham, M. O., Murray, H., LeBeau, F. E. N., Roopun, A., Bibbig, A., Wilent, W. B., Higley, M. J., & Whittington, M. A. (2005). Single-column thalamocortical network model exhibiting gamma oscillations, sleep spindles, and epileptogenic bursts. *J. Neurophysiol.*, *93*, 2194–232.

Traub, R. D., Miles, R., & Jefferys, J. G. (1993). Synaptic and intrinsic conductances shape picrotoxin-induced synchronized after-discharges in the guinea-pig hippocampal slice. *J. Physiol. (Lond.)*, *461*, 525–47.

Traub, R. D., & Wong, R. K. (1981). Penicillin-induced epileptiform activity in the hippocampal slice: a model of synchronization of CA3 pyramidal cell bursting. *Neuroscience*, *6*, 223–30.

Traub, R. D., & Wong, R. K. (1982). Cellular mechanism of neuronal synchronization in epilepsy. *Science*, *216*, 745–7.

Tsakalis, K., & Iasemidis, L. D. (2006). Control aspects of a theoretical model for epileptic seizures. *International Journal of Bifurcation and Chaos*, *16*, 2013–2027.

Tsitsiklis, J. N., & Van Roy, B. (1999). Average cost temporal-difference learning. *Automatica*, *35*, 1799–1808.

Ullah, G., Cressman, J. R., Barreto, E., & Schiff, S. J. (2009). The influence of sodium and potassium dynamics on excitability, seizures, and the stability of persistent states: II. Network and glial dynamics. *Journal of computational neuroscience*, *26*, 171–83.

Uthman, B. M., Reichl, A. M., Dean, J. C., Eisenschenk, S., Gilmore, R., Reid, S. A., Roper, S. N., & Wilder, B. J. (2004). Effectiveness of vagus nerve stimulation in epilepsy patients: A 12 year observation. *Neurology*, *63*, 1124–1126.

Valerio Jr., L. G. (2009). In silico toxicology for the pharmaceutical sciences. *Toxicology and applied pharmacology*, *241*, 356–370.

Vincent, R. D. (2008). Detection, simulation and control in models of epilepsy. Master's thesis, McGill University, Montreal, Quebec.

Vincent, R. D., Courville, A., & Pineau, J. (2011). A bistable computational model of recurring epileptiform activity as observed in rodent slice preparations. *Neural Networks*, *24*, 526–37.

Wang, X. J., & Buzsáki, G. (1996). Gamma oscillation by synaptic inhibition in a hippocampal interneuronal network model. *J. Neurosci.*, *16*, 6402–13.

Wasserman, L. (2004). *All of statistics: a concise course in statistical inference*. Springer.

Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. *Machine Learning*, *8*, 279–292.

Wendling, F. (2008). Computational models of epileptic activity: a bridge between observation and pathophysiological interpretation. *Expert review of neurotherapeutics*, *8*, 889–96.

Wendling, F., Hernandez, A., Bellanger, J.-J., Chauvel, P., & Bartolomei, F. (2005). Interictal to ictal transition in human temporal lobe epilepsy: insights from a computational model of intracerebral EEG. *Journal of clinical neurophysiology*, *22*, 343–56.

Wiering, M., & Schmidhuber, J. (1998). Efficient model-based exploration. *Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior* (pp. 223–228).

Williams, R. J., & Baird, III, L. C. (1993). *Tight performance bounds on greedy policies based on imperfect value functions* (Technical Report NUCCS-93-14). Northeastern University, College of Computer Science, Boston, MA.

Wynants, L., Timmerman, D., Bourne, T., Van Huffel, S., & Van Calster, B. (2013). Screening for data clustering in multicenter studies: the residual intraclass correlation. *BMC Med Res Methodol*, *13*, 128.

Xiong, Z. Q., Saggau, P., & Stringer, J. L. (2000). Activity-dependent intracellular acidification correlates with the duration of seizure activity. *J. Neurosci.*, *20*, 1290–6.

Xu, H., & Mannor, S. (2012). Distributionally robust Markov decision processes. *Mathematics of Operations Research*, *37*, 288–300.

Zafra-Cabeza, A., Rivera, D. E., Collins, L. M., Ridao, M. A., & Camacho, E. F. (2011). A risk-based model predictive control approach to adaptive interventions in behavioral health. *Control Systems Technology, IEEE Transactions on*, *19*, 891–901.

Ziemann, A. E., Schnizler, M. K., Albert, G. W., Severson, M. A., Howard, M. A., Welsh, M. J., & Wemmie, J. A. (2008). Seizure termination by acidosis depends on ASIC1a. *Nat. Neurosci.*, *11*, 816–22.