METHODS FOR REINFORCEMENT LEARNING IN
CLINICAL DECISION SUPPORT

NIRANJANI PRASAD

A DISSERTATION
PRESENTED TO THE FACULTY
OF PRINCETON UNIVERSITY
IN CANDIDACY FOR THE DEGREE
OF DOCTOR OF PHILOSOPHY

RECOMMENDED FOR ACCEPTANCE
BY THE DEPARTMENT OF
COMPUTER SCIENCE
ADVISER: PROFESSOR BARBARA E. ENGELHARDT

SEPTEMBER 2020
Abstract

The administration of routine interventions, from breathing support to pain management, constitutes a major part of inpatient care. Thoughtful treatment is crucial to improving patient outcomes and minimizing costs, but these interventions are often poorly understood, and clinical opinion on best protocols can vary significantly.

Through a series of case studies of key critical care interventions, this thesis develops a framework for clinician-in-loop decision support. The first of these explores the weaning of patients from mechanical ventilation: admissions are modelled as Markov decision processes (MDPs), and model-free batch reinforcement learning algorithms are employed to learn personalized regimes of sedation and ventilator support, that show promise in improving outcomes when assessed against current clinical practice.

The second part of this thesis is directed towards effective reward design when formulating clinical decisions as a reinforcement learning task. In tackling the problem of redundant testing in critical care, methods for Pareto-optimal reinforcement learning are integrated with known procedural constraints in order to consolidate multiple, often conflicting, clinical goals and produce a flexible optimized ordering policy.

The challenges here are probed further to examine how decisions by care providers, as observed in available data, can be used to restrict the possible convex combinations of objectives in the reward function, to those that yield policies reflecting what we implicitly know from the data about reasonable behaviour for a task, and that allow for high-confidence off-policy evaluation. The proposed approach to reward design is demonstrated through synthetic domains as well as in planning in critical care.

The final case study considers the task of electrolyte repletion, describing how this task can be optimized using the MDP framework and analysing current clinical behaviour through the lens of reinforcement learning, before going on to outline the steps necessary in enabling the adoption of these tools in current healthcare systems.
Acknowledgements

This thesis owes its existence to a number of incredible people. First and foremost, thank you to my advisor Barbara Engelhardt, for her mentorship and her commitment to tackling meaningful problems in machine learning for healthcare. She is a source of inspiration to me as both a scientist and as a pillar of support within the research community. I also want to thank Finale Doshi-Velez for her guidance and optimism in encouraging me to pursue my ideas, at crucial junctures of this dissertation.

I thank Ryan Adams, Sebastian Seung and Mengdi Wang for agreeing to serve on my thesis committee, as well as Warren Powell for his early feedback. I am also thankful to Kai Li for forging our collaborations with the Hospital of the University of Pennsylvania. I feel incredibly fortunate to have worked alongside Corey Chivers, Michael Draugelis and the rest of the data science team at Penn Medicine; this research would not have been possible without their consistent backing. I am also grateful for my discussions with physicians at Penn, in particular Krzysztof Laudanski, Gary Weissman, Heather Giannini, and Daniel Herman. Their tirelessness and confidence in the potential of machine learning to improve patient care has been heartening, and their insights have moulded my own perspectives. I also owe a great deal to my labmate and co-author, Li-Fang Cheng. Working with her in our efforts towards optimal laboratory testing in acute patient care was a wonderful experience; her steadfast and systematic approach helped me grow as a researcher.

Thank you to my officemates over the years, and the whole BEEhive, for making the lab a welcoming and engaging place to discuss anything from statistics to politics. At the same time, I am lucky to have amazing housemates, Sumegha Garg—a constant source of support and humour, and my co-conspirator in so much of our life in Princeton—and Sravya Jangareddy, who have made dissertation writing in quarantine rather more fun. Thanks also to the rest of the Musketeers for all the potlucks, hikes and game nights, keeping me from any real danger of working too hard.
Thank you to my fiancé, Cormac O’Neill, who I have found at my side in every adventure the past five years have brought my way. I am so grateful for his immeasurable patience and positivity, not to mention his role as my personal guide to the mysterious world of medical parlance. I cannot imagine this experience without him.

Above all, thank you to my family—my sister Nivedita, my parents Prasad and Vasumathi, and my paternal and maternal grandparents—for all their love and support. Their winding paths have carried me to this point, and their total conviction in my capabilities has been a constant source of strength to me. It was through that faith that I began this PhD, and it is with them that I complete it.
To Amma, Appa, Nivi, and Cormac.
# Contents

Abstract ................................................................. iii
Acknowledgements ......................................................... iv
List of Tables ............................................................. x
List of Figures ............................................................. xi

1 Introduction .......................................................... 1
   1.1 Learning from Electronic Health Records ......................... 2
   1.2 Thesis Contributions ............................................. 5
       1.2.1 Outline .................................................. 6

2 Reinforcement Learning: Preliminaries ............................ 7
   2.1 Markov Decision Processes ....................................... 8
       2.1.1 Solving an MDP .......................................... 10
   2.2 Off-Policy Policy Evaluation ................................... 16

3 An RL Framework for Weaning from Mechanical Ventilation .... 19
   3.1 Related Work .................................................. 22
   3.2 Methods ....................................................... 24
       3.2.1 MIMIC-III Dataset ....................................... 24
       3.2.2 Resampling using Multi-Output Gaussian Processes ....... 26
       3.2.3 MDP Formulation ......................................... 32
       3.2.4 Learning the Optimal Policy ............................ 34
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1.1 UPHS Dataset</td>
<td>88</td>
</tr>
<tr>
<td>6.1.2 Formulating the MDP</td>
<td>89</td>
</tr>
<tr>
<td>6.1.3 Fitted Q-Iteration with Gradient-boosted Trees</td>
<td>95</td>
</tr>
<tr>
<td>6.1.4 Reward Inference using IRL with Batch Data</td>
<td>96</td>
</tr>
<tr>
<td>6.2 Results</td>
<td>97</td>
</tr>
<tr>
<td>6.2.1 Understanding Behaviour in UPHS</td>
<td>98</td>
</tr>
<tr>
<td>6.2.2 Analysing Policies from FQI</td>
<td>100</td>
</tr>
<tr>
<td>6.2.3 Off-policy Policy Evaluation</td>
<td>103</td>
</tr>
<tr>
<td>6.3 Conclusion</td>
<td>104</td>
</tr>
<tr>
<td>7 Conclusion</td>
<td>106</td>
</tr>
<tr>
<td>7.1 Future Research Directions</td>
<td>107</td>
</tr>
<tr>
<td>7.1.1 Translation to Clinical Practice</td>
<td>109</td>
</tr>
<tr>
<td>Bibliography</td>
<td>112</td>
</tr>
</tbody>
</table>
List of Tables

3.1 Extubation guidelines at Hospital of University of Pennsylvania . . . 26

4.1 Summary statistics of key labs and vitals within selected cohort . . . 49

5.1 MDP state features for ventilation management in the ICU . . . . . 77

5.2 Top admitted weights for benchmark control tasks . . . . . . . . . 79

5.3 Effective sample size for ventilation policies with FPL . . . . . . 84

6.1 Patient state features for electrolyte repletion policies . . . . . . 91

6.2 Discretized dosage levels for K, Mg and P . . . . . . . . . . . . . 92

6.3 Reward weights inferred from IRL vs. chosen for FQI policies . . . 100
List of Figures

2.1 Agent-Environment interaction in an MDP ............................ 9
2.2 Deep Q-Network architecture ........................................... 15

3.1 Example ICU admission with invasive mechanical ventilation . . . 25
3.2 Multi-output Gaussian process imputation of key vitals ................ 31
3.3 Shaping penalties for abnormal or fluctuating vitals ..................... 35
3.4 Convergence of $Q(s,a)$ using Q-learning .............................. 39
3.5 Convergence of $Q(s,a)$ using Fitted Q-iteration ......................... 40
3.6 Gini feature importances for optimal policies ............................. 41
3.7 Evaluating mean reward and reintubations of learnt policies ............ 42

4.1 Measurement frequency of key labs and vitals ........................... 50
4.2 Feature importance scores for optimal lab ordering policies .......... 56
4.3 Recommended orders for lactate in example ICU admission ............ 58
4.4 Evaluating component-wise PDWIS estimates across ordering policies 59
4.5 Comparing information gain from lab tests (Clinician vs MO-FQI) .. 60
4.6 Evaluating lab time to treatment onset (Clinician vs MO-FQI) ........ 61

5.1 Admissibility polytopes with varying thresholds for 2D map ........... 69
5.2 Polytope size and distribution of admitted weights in benchmark tasks 80
5.3 Admissible weights for ventilation weaning task ........................ 83

6.1 Distribution of electrolyte levels at repletion events for K, Mg and P 89
6.2 Example hospital admission with potassium supplementation . . . . . 90
6.3 Penalizing abnormal electrolyte levels in reward function . . . . . . 94
6.4 UPHS vs FQI-recommended potassium repletion for sample admission 100
6.5 Distribution of recommended actions for K, Mg and P . . . . . . . 101
6.6 Shapley values of top features for PO and IV potassium repletion . . 102
6.7 Evaluation of policies for K, Mg and P using Fitted-Q Evaluation . . 104
Chapter 1

Introduction

“Medicine is a science of uncertainty and an art of probability.”

- Sir William Osler (1849-1919)

Clinical decision-making is the process of collecting and contextualizing evidence, within an evolving landscape of medical knowledge, with the intent of advancing patient health. In current practice, this requires care providers to sift through large volumes of fragmented, multi-modal data, evaluate these in the face of conflicting pressures—to minimize patient risk, manage uncertainty, and rein in costs—so as to formulate an understanding of a patient’s underlying state and decide what additional information is necessary in order to make diagnostic and therapeutic decisions.

These pressures are heightened when examining clinical decision-making for critically ill patients, that is, those in resource and data-intensive settings such as the ICU (intensive care unit), often requiring rapid judgements with high stakes. Timely and proportionate interventions are crucial to ensuring the best possible patient outcome in these cases. However, there is a severe lack of conclusive evidence on best practices for many routine interventions, particularly when serving heterogeneous patient populations [1]: multiple systematic reviews of randomized control trials of
common ICU interventions have found that less than one in seven of these were of measurable value to patient outcome [93, 106]. Coupled with human biases arising from, for example, skewed (or simply lack of) clinical experience, fatigue, or legal and procedural burdens, this often necessitates an over-reliance by clinicians on intuition or heuristics. While such heuristic decision-making may seem most practical, it can result in compounding errors with increasingly complex cases. It is estimated that more than 250,000 deaths per year in the US can be attributed to medical error [77]; within the ICU, observational studies suggest that around 45% of adverse events are preventable, with the majority of serious medical errors occurring in the ordering or execution of treatments [126]. This motivates the adoption of a more quantitative, data-centric approach to patient care, that systemically evaluates the space of possible interventions in order to determine an optimal course of action.

In this thesis, we develop a framework for clinician-in-loop decision support tools that makes use of large-scale data from electronic health records to aid the management of routine interventions in critically ill patients. We do so by considering these sequential decision-making problems through the lens of reinforcement learning (RL). We go on to demonstrate how our methods can be applied to and evaluated in critical care settings; we do so by learning policies for the management of an array of routine interventions, from the control of mechanical ventilation to the ordering of routine laboratory tests or administration of electrolyte repletion therapy.

1.1 Learning from Electronic Health Records

Electronic health records (EHRs) are digitized collections of patient data, comprising demographic information and personal statistics, medical histories, as well as data from individual hospital visits such as vitals, laboratory tests, administered drugs and procedures, radiology images, nursing notes and billing information. In the space of a
decade, the rate of adoption of EHRs in the US increased 10-fold, from just 9% in 2008 to 94% in 2017 [11, 96], driven by the need to facilitate clinical practice, streamline workflows and slow the inflation of healthcare costs. This sudden availability of rich healthcare data at scale has resulted in the proliferation of efforts to leverage state-of-the-art machine learning methods towards the analysis of this data.

The majority of these efforts have been directed towards predictive modelling, for example, forecasting the likelihood of patient mortality, length of ICU stay, onset of sepsis, and numerous other adverse clinical events [27, 33, 45]. A recurring challenge in the analysis of these forecasting methods is in justifying whether they are action-able, that is, whether they can provide clinical insights to inform early interventions and prevent patient deterioration or improve outcome. This question of actionability is addressed more directly by approaches that instead aim to directly optimize for interventions, which are then presented to clinicians. There have been numerous efforts to learn personalized treatment recommendations, from direct action-learning using simple predictive models or rule-based approaches, to those drawing on literature in control theory, contextual bandits or reinforcement learning [36, 82, 84, 124].

Learning treatment policies from observational data in the form of EHRs poses a number of challenges in practice. Firstly, much of the data present in health records is collected to facilitate the billing of hospital admissions, rather than with the view of being pedagogical for sequential decision making tasks. For instance, nursing notes and ICD-9 codes indicating diagnosed comorbidities and administered procedures are typically entered post hoc, so cannot be relied upon to be available at the time of decision-making. The timestamped data that is available is often sporadically sampled and error-ridden; time series recordings can have widely differing measurement frequencies for different physiological parameters, and are rarely missing at random. Furthermore, “normal” or reference ranges of variables hold little meaning when identifying outliers in the data, as abnormal values are likely to be omnipresent, and are
crucial to identifying patient deterioration. Great care must therefore be taken when processing the data prior to learning treatment policies.

Next, the space of possible predictors available in practice for reasoning about diagnosis or intervention is large. Certain factors may be easily observable by clinicians, but inadequately captured or difficult to infer from chart data in the EHR. These can include for instance informal assessment of a patient’s pallor, muscle weakness, or breathing difficulty. Where standardized severity tests do exist, such as for the monitoring of cognition or pain levels, these are often time-consuming to administer and subject to bias. Clinicians also have complex and seemingly arbitrary choices in intervention, and it can be difficult to discern treatment options that are systematic and evidence-based from those driven instead by available resources, local policies or physician preferences. For example, intravenous drug delivery or fluid repletion, is often restricted to fixed preparations (combinations) of drugs rather than tailored to individual patient needs, increasing risk of overdose or overcorrection of other physiological parameters in the process of treating a certain target condition.

Lastly, the inference of counterfactual outcomes given the interventions observed in patient EHRs is an intrinsic challenge to learning and evaluating policies from observational data. Counterfactual treatment effect estimation has been explored in depth in literature on causal inference. This problem is amplified when planning interventions over multiple time steps: the set of all possible treatment sequences grows exponentially with the number of decision points in the patient admission. Therefore, even with the availability of a large database of patient histories, the effective sample size for evaluating specific treatment policy—that is, the number of histories in the dataset for which decisions by the clinician match this policy—rapidly shrinks [33], demanding caution when assessing its potential value.
1.2 Thesis Contributions

The key contributions of this thesis are two-fold: firstly, the work here is fundamentally interdisciplinary, bridging the gap between the decision-making process in the hospital setting, and planning as a machine learning problem. Within the paradigm of model-free reinforcement learning, I develop definitions of state, action and reward for actors in a critical care environment that are underpinned by clinical reasoning. In doing so, I draw on canonical models in time series representation, such as Gaussian processes, and in prediction, from tree-based ensemble methods to feed-forward neural networks. I apply these methods to multiple distinct facets of inpatient care; our work on the management of invasive mechanical ventilation [98], for example, helped lay a foundation for learning and evaluating treatment policies in this paradigm.

The second area of work this thesis endeavours to push forward is methodology for reward design in practical applications. While the reward function is considered to be the most robust definition of a reinforcement learning task, the use of sparse, overarching objectives can be challenging—and sometimes misleading—to learn from. On the other hand, in a reward function that incorporates several sub-objectives that present more immediate, relevant feedback, it is often unclear how these multiple objectives should be weighted. To this end, I introduce various approaches to drawing from domain experts when deciding this trade-off: both explicitly, by developing a framework for multi-objective reinforcement learning that can be applied to extended horizon tasks, and guides clinicians in ultimately prioritizing objectives to obtain a deterministic policy [13] and implicitly, by examining available historical data to understand current clinical priorities, and using this where appropriate as an anchor in treatment policy optimization [99].
1.2.1 Outline

The remainder of this thesis is organized as follows: in Chapter 2, I introduce the fundamentals of the reinforcement learning framework for sequential decision-making.

Chapter 3 describes my efforts in the development of clinician-in-loop decision support for weaning from mechanical ventilation, outlining the formulation of this task in the reinforcement learning frame, and the use of off-policy reinforcement learning algorithms to learn an optimal sequence of actions, in terms of sedation, intubation or extubation, from sub-optimal behaviour in historical intensive care data.

Chapter 4 turns to the problem of effective reward design given multiple objectives when applying reinforcement learning to clinical decision-making tasks. It combines work in Pareto optimality in reinforcement learning with known clinical and procedural imperatives to present a flexible recommendation system for optimizing the ordering of laboratory tests for patients in critical care.

In Chapter 5, I then consider how available clinical data can be used to inform and restrict the possible convex combinations of these multiple objectives in the reward function to those that yield a scalar reward which reflects what we implicitly know from the data about reasonable behaviour for a task, and allows for robust off-policy evaluation, and apply this to reward design on synthetic domains as well as in the critical care context.

Finally, Chapter 6 explores the problem of electrolyte repletion in critically ill patients, and adapts the framework introduced to previous chapters to demonstrate how, given data from a particular healthcare system, we can understand current behavioural patterns in repletion therapy through the lens of reinforcement learning, and model this task to learn optimal repletion policies in the same way.

In my conclusion, I summarize findings from these works, and consider the steps necessary for successful adoption of data-driven decision support in clinical practice.
Chapter 2

Reinforcement Learning:

Preliminaries

The reinforcement learning paradigm is characterized by an agent continuously interacting with and learning from a stochastic environment to achieve a certain goal. This mirrors one of the fundamental ways in which we as humans learn: not with a formal teacher, but by observing cause and effect through direct sensorimotor connections to our surroundings [116]. In supervised machine learning—which dominates much of machine learning in practice—one is given data in the form of featurized inputs along with the true labels to be predicted. Unsupervised learning provides no labelled data at all, and instead aims to learn the underlying structure of observations. Reinforcement learning is distinct from either of these modalities in that we receive only feedback in the form of a reward signal, to enforce certain actions over others (rather than labels of the “correct” or best possible actions). Additionally, for most tasks with some degree of complexity, this feedback is delayed: the effects of a given action may not present themselves until several time steps into the future, or may emerge gradually over an extended period of time.
Reinforcement learning is also often distinguished by the exploration-exploitation dilemma it poses. When repeatedly interacting online with the environment, an agent can at each time step choose either the best action given existing information (exploit), or choose to gather more information (explore) that may enable better decisions in the future. While this has been a subject of intense study in reinforcement learning literature, when running RL offline—that is, with previously collected observational data, as is the case for many real-world tasks—this trade-off is to a large extent predetermined by the behaviour of the domain actors from whom the data was collected. Despite this, the ability of the reinforcement learning framework to take a holistic, goal-directed approach to learning, and to inherently capture uncertainty in observations and outcomes, makes it an attractive approach for planning in practice.

2.1 Markov Decision Processes

The simplest and most common model underlying reinforcement learning methods is the Markov decision process (MDP). Consider the setting where an agent (the learner) interacts with the environment over a series of discrete time steps, \( t = 0, 1, 2, 3, \ldots \). At each time \( t \), the agent observes the environment in some state \( s_t \), takes action \( a_t \) accordingly, and in turn receives some feedback \( r_{t+1} \) together with the next state \( s_{t+1} \) (Figure 2.1 [116]). An MDP is then defined by the tuple \( \mathcal{M} = \{ \mathcal{S}, \mathcal{A}, P_0, P, R, \gamma \} \), where \( \mathcal{S} \) is a finite state space such that the environment is in some state \( s_t \in \mathcal{S} \) at each time step \( t \), and \( \mathcal{A} \) is the space of all possible actions that can be taken by the agent, \( a_t \in \mathcal{A} \). \( P_0 \) defines the probability distribution of initial states, \( s_0 \sim P_0 \), while the transition function \( P(s_{t+1}|s_t, a_t) \) defines the probability of the next state given the current state-action pair. This essentially summarizes the dynamics of the system, and is unknown for most real-world tasks. The reward function \( R \) is the immediate feedback received at each state transition, and is typically described as a
Figure 2.1: Agent-Environment interaction in an MDP

function of the current state, action and observed next state: $r_{t+1} = R(s_t, a_t, s_{t+1})$.

Finally, the discount factor $\gamma$ determines the relative importance of immediate and future rewards for the task in question.

The Markov assumption here posits that given the full history of state transitions, $h = s_0, a_0, r_1, s_1, a_1, r_2, ..., s_t$, information on future states and rewards—and hence all information relevant in planning for the MDP—is encapsulated by the current state. This assumption of perfect information can often be unrealistic in practice. One popular generalization of the MDP framework that looks to relax this assumption is the partially observable MDP, or POMDP. In a POMDP, observations are treated as noisy measurements of the true underlying state of the environment, and used to model the probability distribution over the state space given an observation; the inferred belief state is then used to learn optimal policies for this environment. However, this inference problem is often challenging and computationally infeasible for large problems. Instead, careful design of state representation in an MDP, in a way that incorporates relevant high-level information from transition histories in order to bridge the gap to complete observability, is often more effective in practice.

It is worth noting that the full reinforcement learning problem as modelled by an MDP can be thought of as an extension to contextual multi-armed bandits in online
learning \[62\]: whereas the observation at each time step is independent and identically
distributed in the bandit setting, the observed state at each step in an MDP depends
on the previous state-action pair, as dictated by the transition function \( P \).

### 2.1.1 Solving an MDP

The goal of the agent in reinforcement learning is to learn a policy function \( \pi : S \to A \)
mapping from the state space to the action space of the MDP, such that this policy
maximizes expected return, that is, the expected cumulative sum of rewards received
by the agent over time. Denoting this optimal policy \( \pi^* \),

\[
\pi^* = \arg\max_{\pi \in \Pi} \mathbb{E}_{s_0 \sim P_0} \left[ \lim_{T \to \infty} \sum_{t=0}^{T-1} \gamma^t r_{t+1} \big| \pi \right]
\]  

(2.1)

for an infinite horizon MDP, where \( 0 \leq \gamma \leq 1 \). Setting discount factor \( \gamma = 0 \) models
a myopic agent, looking only to maximize immediate rewards; as \( \gamma \) approaches 1,
future rewards increasingly contribute to the expected return.

The use of discounted sum of rewards as the objective when solving an MDP
for optimal policies is both mathematically convenient—ensuring finite returns with
\( \gamma < 1 \)—and a reasonable model for most tasks in practice, which immediate feedback
tends to be most reflective of the action taken at the current state, and there is
increasing uncertainty about distant rewards. It can also be seen as a softening of
finite horizon or \textit{episodic} MDPs with fixed horizons \( T \), as the contribution of rewards
far into the future to the objective function is negligible.

### The Bellman Optimality Equation

Given an infinite horizon MDP with finite state and action spaces, bounded reward
\( R \) and discount \( \gamma < 1 \), the value \( V^\pi(s) \) of state \( s \) is defined simply as the expected
return when starting from $s$ and following of policy $\pi$ from that point onwards:

$$V^\pi(s) = \mathbb{E} \left[ \lim_{T \to \infty} \sum_{t=0}^{T-1} \gamma^t r_{t+1} \mid \pi, s_0 = s \right]$$

(2.2)

A fundamental property of value functions in reinforcement learning is that they can be written recursively, such that given current state $s$, action $a$ and next state $s'$, the value of the current state can be written as the sum of the immediate reward $r = R(s, a, s')$ and the discounted expected value of the next state:

$$V^\pi(s) = \mathbb{E}[r + \gamma V^\pi(s')]$$

(2.3)

This is the Bellman recursive equation for value function $V^\pi(s)$. An optimal policy $\pi^*$ is therefore one for which $V^*(s) = \max_\pi V^\pi(s) \ \forall s \in \mathcal{S}$. Substituting the recursive definition above gives us the Bellman optimality equation:

$$V^*(s) = \max_\pi \mathbb{E}[r + \gamma V^\pi(s')]$$

$$\leq \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} P(s, a, s') [R(s, a, s') + \gamma V^\pi(s')]$$

$$= \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} P(s, a, s') [R(s, a, s') + \gamma V^*(s')]$$

(2.4)

We can interpret this as stating that the value of a given state under an optimal policy is necessarily the expected discounted return when taking the best possible action from that state [1116]. It has been shown that $V^*(s)$ is in fact a unique solution of Equation 2.4, [102]; it follows that the deterministic policy

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} P(s, a, s') [R(s, a, s') + \gamma V^*(s')]$$

(2.5)

is optimal. Here, a deterministic policy is one which maps from any given state to a single action; a randomized or stochastic policy on the other hand maps from
states to a probability distribution over the action space. These can be useful in adversarial settings or when tackling the exploration-exploitation trade-off in online reinforcement learning, but are of less interest in the case of human-in-loop decision support, where we wish to recommend a single, optimal action to the user.

**Value Function Approximation in Off-policy RL**

Optimal policies also share the same action-value function: 

\[ Q^*(s, a) = \max_{\pi} Q^\pi(s, a) \]

where \( Q^\pi(s, a) \) is the expected return when taking action \( a \) at state \( s \), and following policy \( \pi \) thereafter, such that \( V^*(s) = \max_a (Q^*(s, a)) \). The Q-function in effect caches the result of one-step lookahead searches for the value of each action at any given state, simplifying the process of choosing optimal actions. The corresponding Bellman optimality equation for \( Q^*(s, a) \) is given by:

\[
Q^*(s) = \sum_{s' \in S} P(s, a, s')[R(s, a, s') + \max_{a' \in A} \gamma V^*(s')] 
\]  

This forms the basis of one of the most popular classes of reinforcement learning algorithms, namely value-based methods such as Q-learning and its variants. **Q-learning** \[\text{125}\] is a reinforcement learning algorithm that uses one-step temporal differences to successively bootstrap on current estimates for the value of each state-action pair. Starting from some initial state and an arbitrary approximation \( \hat{Q}(s, a) \), we perform an update using the observed immediate reward at each state transition according to the following update rule, based on the Bellman equation:

\[
\hat{Q}(s, a) \leftarrow \alpha(r + \gamma \max_{a' \in A} \hat{Q}(s', a')) + (1 - \alpha) \hat{Q}(s, a) 
\]

Our new estimate of \( Q \) is a convex sum of the previous estimate, and the expected return given the reward received at the current transition. The learning rate \( \alpha \) determines the relative weights of the new and old estimates in this update. We repeat
this over a fixed number of iterations, or until the LHS and the RHS of the above equation are approximately equal. It has been shown that this procedure provides guaranteed convergence to the true value of $Q$ in the tabular setting—that is, with discrete state and action spaces—given that all state-action pairs in this space are repeatedly sampled and updated.

Now that we have our estimate of $Q$, the optimal policy $\pi^*$ is simply the action maximizing $Q$ at each state:

$$\pi^*(s) = \arg\max_{a \in A} Q(s, a) \quad \forall s \in S$$

(2.8)

The Q-learning algorithm is both model-free, requiring no prior knowledge of the transition or reward dynamics of the system, and off-policy: it learns an optimal policy from experience collected whilst following a different behaviour policy.

In order to extend from the tabular case to large or continuous state spaces, we must combine Q-learning with some form of parametric function approximation, such as linear models or neural networks. These take as input a vector representation of state and action, and learn the mapping to the corresponding action-value. In practice, updating the function approximator with each new state transition can cause significant instability in learning the Q-function: sampled transitions are typically sparse in comparison with the full state space, and an update based on a single observation in a certain part of the space can disproportionately affect our estimate of $Q$ in a very different region, and in turn lead to extremely slow convergence.

**Fitted Q-iteration** (FQI) is a batch-mode reinforcement learning algorithm that addresses this instability by treating $Q$-function estimation in an infinite-horizon MDP as a sequence of supervised learning problems, where each iteration extends the optimization horizon by one time step. Given a dataset of transition tuples of the form $D = \{(s_n, a_n, r_n, s'_n)\}_{n=1:|D|}$ and initializing $\hat{Q}_0(s, a) = 0 \ \forall s, a$, the training set for the
The $k^{th}$ iteration of FQI is given by:

$$
\left\{ \langle s_n, a_n \rangle, r_n + \gamma \max_{a' \in \mathcal{A}} Q_{k-1}(s'_n, a') \right\}
$$

(2.9)

We can see that at the first iteration, solving this regression problem yields an approximation for the immediate reward given a state-action pair, that is, we solve for the 1-step optimization problem. It follows that running this over $k$ iterations gives us the expected return over a $k$-step optimization horizon; the number of iterations required for convergence in an infinite horizon MDP is effectively determined by the discount factor $\gamma$. The algorithm uses all available experience at each iteration to learn the action-value function, and in turn the optimal policy. This efficient use of information makes it popular in settings with limited data, or where additional experience is expensive to collect, as is the case in healthcare. It can also be applied in conjunction with any function approximator, from tree-based methods or kernel functions [21] to neural networks [105], and provides convergence guarantees for several common regressors.

Many of the recent successes of reinforcement learning have been achieved through the adaptation of existing action-value approximation methods in the form of deep Q-networks (DQNs) [51]. Rather than updating estimates following each observed transition (as in Q-learning) or training function approximators from scratch at each iteration using the entire set of collected experience, DQNs take a mini-batch approach, with several key deviations from prior methods in order to stabilize and speed up training. The first is the use of experience replay: the agent maintains a data buffer $\mathcal{D}$ of randomized, decorrelated recent experience and draws a mini-batch of tuples $\langle s, a, r, s' \rangle \sim U(\mathcal{D})$ uniformly from this buffer. The Q-learning update is then applied to this mini-batch by running gradient descent with the following loss
function:

$$L_i(\theta_i) = \mathbb{E}_{e \sim U(D)} \left[ (r + \gamma \max_{a'} Q(s', a'; \theta_i') - Q(s, a; \theta_i))^2 \right]$$ (2.10)

The above definition highlights another key variation: DQN maintains two separate networks, a target network with parameters $\theta'$, and the actual Q-network parametrized by $\theta$. These parameters are only copied over to the target network periodically, in order to reduce temporal correlations between the Q-value used in action evaluation and in the target.

Finally, while traditional Q-function approximators take as input the state and action, and output Q-value; the DQN takes just the state as input, and outputs a vector of Q-values for each action (Figure 2.2)—necessitating, in the case of continuous state spaces, access to the data-generating process in order to simulate all possible state-action pairs. This speeds up training by allowing us to estimate Q for all actions with a single forward pass through the network.

Much of the performance gains afforded by DQNs comes from the convolutional neural network architecture used to learn state representations in settings with unstructured, high-dimensional observations such as raw image inputs in Atari [81], and are likely to have limited impact when handling unstructured EHR data. This,
combined with issues of sample efficiency and dependence on the availability of a simulator, makes DQNs less suited to reinforcement learning in clinical settings.

2.2 Off-Policy Policy Evaluation

A fundamental challenge of reinforcement learning using batch data, in settings where it is infeasible to either build a functional simulator of system dynamics or to collect additional experience, is in evaluating the efficacy of a proposed policy. This can be viewed as a problem of counterfactual interference: given observed outcomes following a certain behaviour policy, we wish to estimate what would have happened if instead we follow a different policy.

Observational data in practice is rarely generated with pedagogical intent, and the distribution of states and actions represented in these datasets can be starkly different from the policies we want to evaluate. The majority of approaches to off-policy evaluation (OPE) in these settings are founded on either importance sampling or the training of approximate models, or a combination of the two. Importance sampling based approaches draw from methods in classical statistics for handling mismatch between target and sampling distributions: given a dataset of trajectories \( \mathcal{D} = \{ h^{(i)} \}_{i=1:N} \) sampled from some behaviour policy \( \pi_b(a|s) \), and a policy \( \pi_e(a|s) \) that we wish to evaluate, importance sampling re-weights each trajectory \( h^{(i)} = \{ s_0, a_0, r_1, s_1, ... \}^{(i)} \) according to its relative likelihood under the new policy. We define importance weights \( \rho_t \),

\[
\rho_T = \prod_{t=0}^{T-1} \frac{\pi_e(a^h_t|s^h_t)}{\pi_b(a^h_t|s^h_t)} 
\] (2.11)
as the probability ratio of $T$ steps of trajectory $h$ under policy $\pi_e$ versus $\pi_b$ [100]. It follows that value of the new policy $\pi_e$ can be estimated by:

$$\hat{V}_{IS}(\pi_e) = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T-1} \gamma^t \rho_{T-1}^{(i)} r_{t+1}$$  \hspace{1cm} (2.12)

This yields a consistent, unbiased estimate of the value of a given policy, but can be incredibly high variance in practice, as a result of the product term in $\rho_T$. This is amplified in tasks with extended horizons. Two common extensions that attempt to mitigate this explosion of variance are the per-decision importance sampling (PDIS) and the per-decision weighted (PDWIS) estimators [100], defined as follows:

$$\hat{V}_{PDIS}(\pi_e) = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T} \gamma^t \rho_t^{(i)} r_{t+1}$$  \hspace{1cm} (2.13)

$$\hat{V}_{PDWIS}(\pi_e) = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T} \gamma^t \frac{\rho_t^{(i)}}{\sum_{i=1}^{N} \rho_t^{(i)}} r_{t+1}$$  \hspace{1cm} (2.14)

The intuition behind the per-decision estimator is to weight each reward along a trajectory according to the likelihood of the trajectory only up to that time step, rather than the relative probability of the complete trajectory. However, the variance of the PDIS estimator from importance weights $\rho$ can still often be unacceptably high. To address this, the weighted variant normalizes $\rho$, dividing by the sum of all importance weights during each trajectory. While this introduces bias in our estimated policy value, it still yields a consistent and lower variance estimator, in comparison with alternative approaches.

The second class of approaches to off-policy evaluation rely on directly learning regressors for the expected return, by first fitting a model $M$ for the MDP using available transition data, and then taking the estimated parameters $\hat{P}$ and $\hat{R}$ for the transition and reward function respectively, substituting these into the Bellman equation (Equations 2.3 - 2.5) in order to estimate the value $V^{\pi_e}$ of the policy in question.
However, it is challenging to train models that can generalize well in most real-world problems, composed of large or continuous state spaces and many combinations of state-action pairs that are never observed in the data. Function approximation in these settings can introduce significant bias in the estimated parameters of the MDP, limiting the credibility of the policy value estimates returned.

Doubly robust estimators for off-policy evaluation in sequential decision-making problems [48] look to leverage both the low bias of importance sampling and the low variance of model-based approaches to achieve the best possible estimates for the value of a given policy:

\[
\hat{V}_{DR}^0 = 0; \quad \hat{V}_{DR}^{T-t+1}(\pi_e) = \hat{V}_{AM}(s_t) + \rho_T(r_t + \gamma \hat{V}_{DR}^{t} - \hat{Q}_{AM}(s_t, a_t))
\]

(2.15)

where \(\hat{V}_{AM}\) and \(\hat{Q}_{AM}\) are the state and action value estimates respectively according the approximate model of the MDP, and \(\rho_T\) is the importance weight given available trajectories (Equation 2.11). The quality of the doubly robust estimator \(\hat{V}_{DR}\) is then dependent of the robustness of the best of the IS or AM estimates.

In recent years, several extensions to both importance sampling and model-based methods have been introduced for off-policy evaluation in reinforcement learning. These include importance sampling applied to state visitation distributions rather than state transition sequences to tackle exploding variance in tasks with extended horizons [70], efforts to draw from treatment effect estimation in causal reasoning to estimate individual policy values [71], and variations of model-based or supervised learning approaches [34, 47]. In particular, fitted Q-evaluation (FQE) [64] for batch reinforcement learning, which adapts the iterative supervised learning approach of FQI to the evaluation of learnt policies, has been shown to be data-efficient and outperform prior approaches in high-dimensional reinforcement learning settings.
Chapter 3

An RL Framework for Weaning from Mechanical Ventilation

Mechanical ventilation is one of the most widely used interventions in admissions to the intensive care unit (ICU): around 40% of patients in the ICU are supported on invasive mechanical ventilation at any given hour, accounting for 12% of total hospital costs in the United States \cite{3,130}. These are typically patients with acute respiratory failure or compromised lung function caused by some underlying condition such as pneumonia, sepsis or heart disease, or cases in which breathing support is necessitated by neurological disorders, impaired consciousness or weakness following major surgery. As advances in healthcare enable more patients to survive critical illness or surgery, the need for mechanical ventilation during recovery has risen.

Closely coupled with ventilation in the care of these patients is sedation and analgesia, which are crucial to maintaining physiological stability and controlling pain levels of patients while intubated. The underlying condition of the patient, as well as factors such as obesity or genetic variations, can have a significant effect on the pharmacology of drugs, and cause high inter-patient variability in response to a given sedative \cite{17}, lending motivation to a personalized approach to sedation strategies.
Weaning refers to the process of liberating patients from mechanical ventilation. The primary diagnostic tests for determining whether a patient is ready to be extubated involve screening for resolution of the underlying disease, monitoring haemodynamic stability, assessment of current ventilator settings and level of consciousness, and finally a series of spontaneous breathing trials (SBTs) ascertaining that the patient is able to cope with reduced support. Prolonged ventilation—and in turn over-sedation—is associated with post-extubation delirium, drug dependence, ventilator-induced pneumonia and higher patient mortality rates [44], in addition to inflating costs and straining hospital resources. Physicians are often conservative in recognizing patient suitability for extubation, however, as failed breathing trials or premature extubations that necessitate reintubation within the space of 48 to 72 hours can cause severe patient discomfort, and result in even longer ICU stays [59]. Efficient weaning of sedation levels and ventilation is therefore a priority both for improving patient outcomes and reducing costs, but a lack of comprehensive evidence and the variability in outcomes between individuals and across subpopulations means there is little agreement in clinical literature on the best weaning protocol [18, 32].

In this work, we aim to develop a decision support tool that leverages available information in the data-rich ICU setting to alert clinicians when a patient is ready for initiation of weaning, and recommend a personalized treatment protocol. We explore the use of off-policy reinforcement learning algorithms, namely fitted Q-iteration (FQI) with different regressors, to determine the optimal treatment at each patient state from sub-optimal historical patient treatment profiles. The setting fits naturally into the framework of reinforcement learning as it is fundamentally a sequential decision making problem rather than purely a prediction task: we wish to choose the best possible action at each time—in terms of sedation drug and dosage, ventilator settings, initiation of a spontaneous breathing trial, or extubation—while capturing
the stochasticity of the underlying process, the delayed effects of actions and the uncertainty in state transitions and outcomes.

The problem poses a number of key challenges: firstly, there are a multitude of factors that can potentially influence patient readiness for extubation, including some not directly observed in ICU chart data, such as a patient’s inability to protect their airway due to muscle weakness. The data that is recorded can itself be sparse, noisy and irregularly sampled. In addition, there is potentially an extremely large space of possible combinations of sedatives (in terms of drug, dosage and delivery method) and ventilator settings, such as oxygen concentration, tidal volume and system pressure, that can be manipulated during weaning. We are also posed with the problem of interval censoring, as in other intervention data: given past treatment and vitals trajectories, observing a successful extubation at time $t$ provides us only with an upper bound on the true time to extubation readiness, $t_e \leq t$; on the other hand, if a breathing trial was unsuccessful, there is uncertainty how premature the intervention was. This presents difficulties both during learning and when evaluating policies.

The rest of this chapter is organized as follows: Section 3.1 explores prior efforts towards the use of reinforcement learning in clinical settings. In Section 3.2 we describe the data and methods applied here, and Section 4 presents the results achieved. Finally, conclusions and possible directions for further work are discussed in Section 3.3.

**Prior Publication:** Niranjani Prasad, Li-Fang Cheng, Corey Chivers, Michael Draugelis, and Barbara E. Engelhardt. *A reinforcement learning approach to weaning of mechanical ventilation in intensive care units.* Proceedings of 33rd Conference on Uncertainty in Artificial Intelligence, (UAI) 2017 [98].
3.1 Related Work

The widespread adoption of electronic health records paved the way for a data-driven approach to healthcare, and recent years have seen a number of efforts towards personalized, dynamic treatment regimes. Reinforcement learning in particular has been explored across various settings, particularly in the management of chronic illness. These range from determining the sequence of drugs to be administered in HIV therapy or cancer treatment, to minimizing risk of anaemia in haemodialysis patients and insulin regulation in diabetics.

These efforts are typically based on estimating the value, in terms of clinical outcomes, of different treatment decisions given the state of the patient. For example, Ernst et al. [22] apply fitted Q-iteration with a tree-based ensemble method to learn the optimal HIV treatment in the form of structured treatment interruption strategies, in which patients are cycled on and off drug therapy over several months. The observed reward here is defined in terms of the equilibrium point between healthy and unhealthy blood cells in the patient as well as the time spent on drug therapy, such that the RL agent learns a policy that minimizes viral load (the fraction of unhealthy cells) as well as drug-induced side effects.

Zhao et al. [133] use Q-learning to learn optimal individualized treatment regimens for nonsmall cell lung cancer. The objective is to choose the optimal first and second lines of therapy and optimal initiation time for the second line treatment such that the overall survival time is maximized. The Q-function with time-indexed parameters is approximated using a modification of support vector regression (SVR) that explicitly handles right-censored data. In this setting, right-censoring arises in measuring the time of death from start of therapy: given that a patient is still alive at the time of the last follow-up, we merely have a lower bound on the exact survival time.

Escandell-Montero et al. [23] compare the performance of both Q-learning and fitted Q-iteration with current clinical protocol for informing the administration of
erythropoiesis-stimulating agents (ESAs) for treating anaemia. The drug administration strategy is modeled as an MDP, with the state space expressed by current and change in haemoglobin levels, the most recent ESA dosages, and the patient subpopulation group. The action space here comprises a set of four discretized ESA dosages, and the reward function is designed to maintain haemoglobin levels within a healthy range, while avoiding abrupt changes.

On the problem of administering anaesthesia in the acute care setting, Moore et al. [82] apply Q-learning with eligibility traces to the administration of intravenous propofol, modeling patient dynamics according to an established pharmacokinetic model, with the aim of maintaining some level of sedation or consciousness. Padmanabhan et al. [94] also use Q-learning, for the regulation of both sedation level and arterial pressure (as an indicator of physiological stability) using propofol infusion rate. All of the aforementioned work rely on model-based approaches to reinforcement learning, and develop treatment policies on simulated patient data. More recently however, Nemati et al. [86] consider the problem of heparin dosing to maintain blood coagulation levels within some well-defined therapeutic range, modeling the task as a partially observable MDP, using a dynamic Bayesian network trained on real ICU data, and learning a dosing policy with neural fitted Q-iteration (NFQ).

There exists some literature on machine learning methods for the problem of ventilator weaning: Mueller et al. [83] and Kuo et al. [11] look at prediction of weaning outcomes using supervised learning methods, and suggest that classifiers based on neural networks, logistic regression, or naive Bayes, trained on patient ventilator and blood gas data, show promise in predicting successful extubation. Gao et al. [28] develop association rule networks for naive Bayes classifiers, in order to analyze the discriminative power of different feature categories toward each decision outcome class, to help inform clinical decision making.
The approach described in this chapter is novel in its use of reinforcement learning methods to directly provide actionable recommendations for the management of ventilation weaning, the incorporation of a larger number of possible predictors of wean readiness in the patient state representation compared with previous work—which limit features for classification to a few key vitals—and the design of a reward function informed by current clinical protocols.

3.2 Methods

3.2.1 MIMIC III Dataset

We use the Multi-parameter Intelligent Monitoring in Intensive Care (MIMIC III) database \([49]\), a freely available source of de-identified critical care data for 53,423 adult admissions and 7,870 neonates. The data includes patient demographics, recordings from bedside monitoring of vital signs, administration of fluids and medications, results of laboratory tests, observations and notes charted by care providers, as well as information on diagnoses, procedures and prescriptions for billing.

We extract from this database a set of 8,860 admissions from 8,182 unique adult patients undergoing invasive ventilation. In order to train and test our weaning policy, we further filter this dataset to include only those admissions in which the patient was kept under ventilator support for more than 24 hours. This allows us to exclude the majority of episodes of routine ventilation following surgery, which are at minimal risk of adverse extubation outcomes. We also filter out admissions in which the patient in not successfully discharged from the hospital by the end of the admission, as in cases where the patient expires in the ICU, this is largely due to factors beyond the scope of ventilator weaning, and again, a more informed weaning policy is unlikely to have a significant influence on outcomes. Failure in our problem setting is instead defined as prolonged ventilation, administration of unsuccessful spontaneous breathing
Figure 3.1: Example ICU admission comprising mechanical ventilation and accompanying sedation, with time-stamped measurements of key vitals.
Table 3.1: Core extubation guidelines at Hospital of University of Pennsylvania

<table>
<thead>
<tr>
<th>Physiological Stability</th>
<th>Oxygenation Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Respiratory Rate ≤ 30</td>
<td>PEEP (cm H₂O) ≤ 8</td>
</tr>
<tr>
<td>Heart Rate ≤ 130</td>
<td>$SpO₂$ (%) ≥ 88</td>
</tr>
<tr>
<td>Arterial pH ≥ 7.3</td>
<td>Inspired $O₂$ (%) ≤ 50</td>
</tr>
</tbody>
</table>

Before extubation, some patients may experience complications, such as extubation shock, extubation failure and reintubation, or are at risk of extubation failure. Trials, or reintubation within the same admission—all of which are associated with adverse outcomes for the patient. A typical patient admission episode is illustrated in Figure 3.1: we can see ventilation times, a number of administered sedatives, both as continuous IV drips and discrete boli, as well as nurse-verified recordings of patient physiological parameters, measured at a widely varying sampling intervals.

Preliminary guidelines for the weaning protocol, in terms of the desired ranges of major physiological parameters (heart rate, respiratory rate and arterial pH) as well as approximate constraints at time of extubation on the inspired $O₂$ fraction ($FiO₂$), oxygenation pulse oxymetry ($SpO₂$) and the setting of positive end-expiratory pressure (PEEP), were obtained by referencing criteria in current practice at the Hospital of University of Pennsylvania, and are summarized in Table 3.1. These ranges are used in designing the feedback received by our reinforcement learning agent, to facilitate the learning of an optimal weaning policy.

### 3.2.2 Resampling using Multi-Output Gaussian Processes

Measurements of vitals and lab results in the ICU data can be irregular, sparse and error-ridden. Non-invasive measurements such as heart rate or respiratory rate are taken several times an hour, while tests for arterial pH or oxygen pressure, which involve more distress to the patient, may only be carried out every few hours as needed. This wide discrepancy in measurement frequency is typically handled by resampling with means in hourly intervals, and using sample-and-hold interpolation.
where data is missing. However, patient state—and therefore the need to update management of sedation or ventilation—can change within the space of an hour, and naive methods for interpolation are unlikely to provide the necessary accuracy at higher temporal resolutions. We therefore explore methods that can enable further fine-tuning of policy estimation. One of the most commonly used techniques to resolve missing data and irregular sampling is Gaussian processes (GPs, \cite{20, 30, 115}), a function-based method well-suited to medical time series data. GPs can be thought of as distributions over arbitrary functions; a collection of random variables is said to form a Gaussian process if for any finite subset of these random variables there is a joint Gaussian distribution. In the case of time-series modeling, given a dataset with inputs denoted by a set of $T$ time steps $t = [t_1 \ldots t_T]^T$ and corresponding observations of some vital sign $v = [v_1 \ldots v_T]^T$, we can model

$$v = f(t) + \varepsilon,$$ (3.1)

where $\varepsilon$ vector represents i.i.d Gaussian noise, and $f(t)$ are the latent noise-free values we would like to estimate. Equivalently, this can thought of as placing a GP prior on the latent function $f(t)$:

$$f(t) \sim \mathcal{GP}(m(t), \kappa(t, t')),$$ (3.2)

where $m(t)$ is the mean function and $\kappa(t, t')$ is the covariance function or kernel:

$$m(t) = \mathbb{E}[f(t)]$$ (3.3)

$$\kappa(t, t') = \mathbb{E}[f(t - m(t)), f(t' - m(t'))]$$ (3.4)

Together, the mean and kernel functions fully describe the Gaussian process. Properties such as smoothness and periodicity of $f(t)$ are dependent on the kernel used.
Prior approaches to modeling physiological time series typically rely on univariate Gaussian processes, treating each signal as independent. However, this assumption may result in considerable loss of information—there are known correlations between several common vitals—and limit the accuracy of imputation for more sparsely sampled vitals. In this work, we instead learn a multi-output GP (MOGP) to account for temporal correlations between physiological parameters during interpolation; MOGPs have shown improvements over the univariate case in medical time series for both imputation and forecasting. We adapt the framework in to impute the physiological signals jointly by exploring covariance structures between them, excluding the sparse prior settings: for the $i^{th}$ patient is our out dataset, the time series of the $d^{th}$ covariate (that is, vital sign or laboratory test) is denoted by a vector of time points $t_{i,d}$ and corresponding values $v_{i,d}$. The time series data for this patient over all $D$ covariates can then be written as:

\[ t_i = [t_{i,1}^T \ t_{i,2}^T \ ... \ t_{i,D}^T]^T \]  
\[ v_i = [v_{i,1}^T \ v_{i,2}^T \ ... \ v_{i,D}^T]^T \]  

where $t_i, v_i \in \mathbb{R}^{T_i \times 1}$, and $T_i = \sum_d T_{i,d}$. Denoting $\mathcal{F}_i$ as a multi-output time series function for patient $i$, we now have:

\[ v_i = \mathcal{F}_i(t_i) + \epsilon_i \]  

where $\mathcal{F}_i(t_i)$ is drawn from a patient specific Gaussian process, $\mathcal{GP}_i$ such that

\[ \mathcal{F}_i(t_i) \sim \mathcal{GP}_i(\mu_i(t), \kappa_i(t, t')) \]  

Here, we set $\mu(t) = 0$ without loss of generality so the Gaussian process is completely defined by secondary statistics alone. In designing a kernel function $\kappa(t, t')$
that captures covariance structure in clinical time series, we adapt the linear model
of coregionalization (LMC) framework, originally applied to prediction over vector-valued data in geostatistics [51]. In the linear model of coregionalization, outputs are modeled as a weighted combination of independent random functions, which we refer to as basis kernels. We denote this set of $Q$ basis kernels used to model our $D$ covariates as $\kappa_q(t, t')_{q=1}^Q$, such that the full joint kernel for a given patient $i$ can be written as a structured linear mixture of these $Q$ kernels:

$$
\kappa_i(t_i, t_i') = \sum_{q=1}^Q \begin{bmatrix}
    b_{q,(1,1)}\kappa_q(t_{i,1}, t_{i,1}') \\
    \vdots \\
    b_{q,(D,1)}\kappa_q(t_{i,D}, t_{i,1}') \\
    \vdots \\
    b_{q,(1,D)}\kappa_q(t_{i,1}, t_{i,D}') \\
    \vdots \\
    b_{q,(D,D)}\kappa_q(t_{i,D}, t_{i,D}')
\end{bmatrix} \in \mathbb{R}^{T_i \times T_i}
$$

(3.9)

where weights $b_{q,(d,d')}$ scale the covariance (as described by $\kappa_q$) between covariates $d$ and $d'$. These weights can be rewritten as a set of matrices $\{B_q\}_{q=1}^Q$ where each $B_q$ is a symmetric positive definite matrix defined by:

$$
B_q = \begin{bmatrix}
    b_{q,(1,1)} & \cdots & b_{q,(1,D)} \\
    \vdots & \ddots & \vdots \\
    b_{q,(D,1)} & \cdots & b_{q,(D,D)}
\end{bmatrix} \in \mathbb{R}^{D \times D}.
$$

(3.10)

In cases where we have the same input time vector for each of our covariates, the LMC in Equation 3.9 can be further simplified using the Kronecker product ($\otimes$), such that:

$$
\kappa_i(t_i, t_i') = \sum_{q=1}^Q B_q \otimes \kappa_q(t_{i,*}, t_{i,*}')
$$

(3.11)

where $t_{i,*}$ represents the time vector of each covariate. Note that this is not true for the irregular sampled vitals and lab tests in the clinical time series modeled here. In practice, we compute each sub-block $\kappa_q(t_{i,d}, t_{i,d'})$ given any pair of input time $t_{i,d}$ and $t_{i,d'}$ from two signals, indexed by $d$ and $d'$. 
For our setting, we parametrize the basis kernel as a spectral mixture kernel \([128]\):

\[
\kappa_q(t, t') = \exp(-2\pi^2 \nu_q \tau^2) \cos(2\pi \nu_q \tau)
\]

(3.12)

where \(\tau = |t - t'|\), allowing us to model smooth transitions in time or circadian rhythm of these vital signs and lab results. The use of this model for GP regression requires that our covariance matrix \(\kappa(t, t')\) is positive definite for all \(t, t'\); as each basis kernel is positive definite, we simply need to ensure that every \(B_q\) is also positive definite. We do so by parametrizing:

\[
B_q = A_q A_q^T + \begin{bmatrix}
\lambda_{q,1} & 0 & \cdots & 0 \\
0 & \lambda_{q,2} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \lambda_{q,D}
\end{bmatrix} = A_q A_q^T + \text{diag}(\lambda_q)
\]

(3.13)

where \(A_q \in \mathbb{R}^{D \times R_q}\) and \(\lambda_q \in \mathbb{R}^{D \times 1}\); \(R_q\) is therefore the rank of \(B_q\) when \(\lambda_q = 0\).

In this work, we set the number of basis kernels \(Q = 2\) and \(R_q = 5 \forall q\), to jointly model 12 selected physiological signals (\(D = 12\)). In choosing these signals, we exclude vitals that take discrete values, such as ventilator mode or the RASS sedation scale. For each patient, one structured GP kernel is estimated using the implementation in \([12]\). We then impute the time series with the estimated posterior mean given all the observations across all chosen physiological signals within that patient. For those vitals that are not imputed this way, we simply resample with means and apply sample-and-hold interpolation. After preprocessing, we obtain complete data for each patient, at a temporal resolution of 10 minutes, from admission time to discharge time. Imputation in the training set uses all known measurements, while for the test set we use only those measurements before current time step; our forecast values converge
Figure 3.2: Sample trajectories of 8 vitals in an ICU admission, with Gaussian
Process imputation. A total of 12 vital signs are jointly modeled by the MOGP.
towards the population mean with increasing time since the last known measurement.
An example of imputed vital signs for a single patient is shown in Figure 3.2.

3.2.3 MDP Formulation

A Markov Decision Process or MDP is defined by the following key components:

(i) A finite **state space** \( \mathcal{S} \) such that at each time \( t \), the environment (here, the patient as observed through the EHR) is in state \( s_t \in \mathcal{S} \),

(ii) An **action space** \( \mathcal{A} \): at each time \( t \), the reinforcement learning agent chooses some action \( a_t \in \mathcal{A} \), which influences the next state, \( s_{t+1} \),

(iii) A **transition function** \( \mathcal{P}(s_{t+1}|s_t, a_t) \), which defines the dynamics of the system and typically unknown, and

(iv) A **reward** \( r_{t+1} = R(s_t, a_t, s_{t+1}) \) observed at each time step, which defines the immediate feedback received following a state transition.

The goal of the reinforcement learning agent is to learn a policy, or mapping \( \pi(s) \to a \) from states to actions, that maximizes the value \( V^\pi(s) \) defined as expected accumulated reward, over horizon length \( T \) with discount factor \( \gamma \):

\[
V^\pi(s_t) = \lim_{T \to \infty} \mathbb{E}_\pi \left[ \sum_{t}^{T-1} \gamma^t R(s_t, a_t, s_{t+1}) \right] \tag{3.14}
\]

where \( \gamma \) determines the relative weight of immediate and long-term rewards.

Patient response to sedation and readiness for extubation can depend on a number of different factors, from demographic characteristics, pre-existing conditions and comorbidities to specific time-varying vitals measurements, and there is considerable variability in clinical opinion on the extent of influence of different factors. Here, in defining each patient state within an MDP, we look to incorporate as many reliable
and frequently monitored features as possible, and allow the algorithm to determine the relevant features. The state at each time $t$ is a 32-dimensional feature vector that includes fixed demographic information (patient age, weight, gender, admit type, ethnicity) as well any relevant physiological measurements, ventilator settings, level of consciousness (given by the Richmond Agitation Sedation Scale, or RASS), current dosages of different sedatives or analgesic agents, time into ventilation, and the number of intubations so far in the admission. For simplicity, categorical variables admit type and ethnicity are binarized according to emergency/non-emergency and white/non-white admissions respectively.

In designing the action space, we develop an approximate mapping of a set of six commonly used sedatives into a single dosage scale, and choose to discretize this scale to four different levels of sedation. The action $a_t \in \mathcal{A}$ at each time step is chosen from a finite two-dimensional set of eight actions, where $a_t[0] \in \{0, 1\}$ indicates having the patient off or on the ventilator respectively, and $a_t[1] \in \{0, 1, 2, 3\}$ corresponds to the level of sedation to be administered over the next 10-minute interval:

$$\mathcal{A} = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 2 \\ 3 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \right\}$$ (3.15)

Finally, we associate a reward $r_{t+1}$ with each state transition—defined by the tuple $(s_t, a_t, s_{t+1})$—to encompass (i) effective cost of time spent under invasive ventilation $r_{t+1}^{\text{intub}}$, (ii) feedback from failed SBTs or need for reintubation $r_{t+1}^{\text{extub}}$, and (iii) penalties for physiological stability, i.e. when vitals are highly fluctuating or outside reference ranges, $r_{t+1}^{\text{vitals}}$. The feedback at each timestep is defined by a combination of sigmoid, piecewise-linear and threshold functions that reward closely regulated vitals and successful extubation while penalizing adverse events:

$$r_{t+1} = r_{t+1}^{\text{intub}} + r_{t+1}^{\text{extub}} + r_{t+1}^{\text{vitals}}$$ (3.16)
where each component in the above summation is defined as follows:

\[ r_{t+1}^{intub} = \mathbb{1}_{[\alpha_t(0)=1]} \left[ C_1 \mathbb{1}_{[\alpha_{t-1}(0)=1]} - C_2 \mathbb{1}_{[\alpha_{t-1}(0)=0]} \right] \]  
(3.17)

\[ r_{t+1}^{extub} = \mathbb{1}_{[\alpha_t(0)=0]} \left[ C_3 \mathbb{1}_{[\alpha_{t-1}(0)=1]} + C_4 \mathbb{1}_{[\alpha_{t-1}(0)=0]} - C_5 \sum_{v_{ext}} \mathbb{1}_{[v_{ext} > v_{ext max}]} \mathbb{1}_{[v_{ext} < v_{ext min}]} \right] \]  
(3.18)

\[ r_{t+1}^{vitals} = \sum_v \left[ \frac{C_6}{1 + e^{-(v_t - v_{min})}} - \frac{C_6}{1 + e^{-(v_t - v_{min})}} + \frac{C_6}{2} - C_7 \max \left( 0, \frac{|v_{t+1} - v_t|}{c} - \frac{1}{5} \right) \right] \]  
(3.19)

where positive constants \( C_1 \) to \( C_7 \) determine the relative importance of these reward signals. The system therefore accumulates negative rewards \( C_1 \) at intubation, and \( C_2 \) for each additional time step spent on the ventilator. A large positive reward \( C_3 \) is observed at the time of extubation, along with additional positive feedback \( C_4 \) while remaining off the ventilator. Vitals \( v_{t,ext}^{ext} \) comprise the subset of parameters directly associated with readiness for extubation (\( FiO_2 \), \( SpO_2 \) and PEEP set) with weaning criteria defined by the ranges \([v_{ext min}, v_{ext max}]\). A fixed penalty \( C_5 \) is applied for each criterion not met when off invasive support.

Finally, values \( v_t \) are the measurements of those vitals \( v \) (included in the state representation \( s_t \)) believed to be indicative of physiological stability at time \( t \), with desired ranges \([v_{min}, v_{max}]\). The penalty for exceeding these ranges at each time step is given by a truncated sigmoid function, illustrated in Figure 3.3a. The system also receives negative rewards when consecutive measurements see a change greater than 20\% (positive or negative) in value, as shown in Figure 3.3d. These two sources of feedback are scaled by constants \( C_6 \) and \( C_7 \) respectively.

### 3.2.4 Learning the Optimal Policy

The majority of reinforcement learning algorithms are based on estimation of the \( Q \)-function, that is, the expected value of state-action pairs \( Q^\pi(s, a) : S \times A \to \mathbb{R} \),
(a) Exceeding threshold values  
(b) High fluctuation in values

Figure 3.3: Shape of reward function penalising instability in vitals, \( r_{\text{vitals}}(v_t) \)

to determine the optimal policy \( \pi \). Of these, the most widely used is Q-learning, an off-policy reinforcement learning algorithm in which we start with some initial state and arbitrary approximation of the Q-function, and update this estimate using the reward from the next transition using the Bellman recursion for Q-values:

\[
\hat{Q}(s_t, a_t) = \hat{Q}(s_t, a_t) + \alpha (r_{t+1} + \gamma \max_{a \in A} \hat{Q}(s_{t+1}, a) - \hat{Q}(s_t, a_t)) \tag{3.20}
\]

where the learning rate \( \alpha \) determines the weight given to each new transition seen, and \( \gamma \) is the discount factor.

Fitted Q-iteration (FQI), on the other hand, is a form of off-policy batch-mode reinforcement learning that uses a set of one-step transition tuples:

\[
\mathcal{F} = \{(s^n_t, a^n_t, s^n_{t+1}, r^n_{t+1}), n = 1, ..., |\mathcal{F}|\} \tag{3.21}
\]

to learn a sequence of function approximators \( \hat{Q}_1, \hat{Q}_2, ..., \hat{Q}_K \) of the value of state-action pairs, by iteratively solving supervised learning problems. Both FQI and Q-learning belong to the class of model-free reinforcement learning methods, which assumes no
knowledge of the dynamics of the system. In the case of FQI, there are also no
assumptions made on the ordering of tuples; these could correspond to a sequence
of transitions from a single admission, or randomly ordered transitions from multiple
histories. FQI is therefore more data-efficient, with the full set of samples used by the
algorithm at every iteration, and typically converges much faster than Q-learning.

The training set at the $k^{th}$ supervised learning problem is given by $\mathcal{T}\mathcal{S} = \{(s^n_t, a^n_t), \hat{Q}_k(s^n_t, a^n_t)) \mid n = 1, ..., |\mathcal{F}|\}$. As before, the Q-function is updated at each iteration according to the Bellman equation:

$$\hat{Q}_k(s_t, a_t) \leftarrow r_{t+1} + \gamma \max_{a \in \mathcal{A}} \hat{Q}_{k-1}(s_{t+1}, a)$$  \hspace{1cm} (3.22)

where $\hat{Q}_1(s_t, a_t) = r_{t+1}$. The optimal policy after $K$ iterations is then given by:

$$\hat{\pi}^*(s) = \arg\max_{a \in \mathcal{A}} \hat{Q}_K(s, a)$$  \hspace{1cm} (3.23)

A variant of this procedure is outlined in Algorithm 1 for Fitted Q-iteration with
sampling, where a batch of transitions are sampled from the full dataset (uniformly,
or by prioritizing certain experience) without replacement at each iteration. This
allows us to speed up training of the Q-function given very large datasets, assigning
greater weight more informative transitions as necessary.

FQI guarantees convergence for many commonly used regressors, including kernel-
based methods \cite{92} and decision trees. In particular, fitted-Q with extremely random-
ized trees or extra-trees (FQIT) \cite{21,29}, a tree-based ensemble method that extends
on random forests by introducing randomness in the thresholds chosen at each split,
has been applied in the past to learning large or continuous Q-functions in clinical
settings \cite{22,23}. Neural Fitted-Q (NFQ) \cite{105} on the other hand, looks to lever-
age the representational power of neural networks as regressors to fitted Q-iteration.
Nemati et al. \cite{86} use NFQ to learn optimal heparin dosages, mapping the patient
Algorithm 1 Fitted Q-iteration with sampling

**Input:**

One-step transitions \( \mathcal{F} = \{ (s^n_t, a^n_t, s^n_{t+1}), r^n_{t+1} \}_{n=1:|\mathcal{F}|} \)

Regression parameters \( \theta \);

Action space \( \mathcal{A} \); subset size \( N \)

**Initialize** \( Q_0(s_t, a_t) = 0 \) \( \forall s_t \in \mathcal{F}, a_t \in \mathcal{A} \)

for iteration \( k = 1 \rightarrow K \) do

\( \text{subset}_N \sim \mathcal{F} \)

\( S \leftarrow [] \)

for \( i \in \text{subset}_N \) do

\( Q_k(s_i, a_i) \leftarrow r_{i+1} + \gamma \max_{a' \in \mathcal{A}} (\text{predict}(s_{i+1}, a'), \theta)) \)

\( S \leftarrow \text{append}(S, ((s_i, a_i); Q(s_i, a_i))) \)

end

\( \theta \leftarrow \text{regress}(S) \)

end

**Result:** \( \theta \)

\( \pi \leftarrow \text{classify}(s^n_t, a^n_t) \)

hidden state to expected return. Neural networks hold an advantage over tree-based methods in iterative settings in that it is possible to simply update weights in the network at each iteration, rather than rebuilding entirely.

### 3.3 Results

After extracting relevant ventilation episodes from ICU admissions in the MIMIC III database as described in Section 3.2.1, and splitting these into training and test data, we obtain a total of 1,800 distinct admissions in our training set and 664 admissions in our test set. We interpolate a set of 12 key time-varying vitals measurements using Gaussian processes, sampling at 10-minute intervals; missing values in the remaining components of the state space are imputed using sample-and-hold interpolation. This yields of the order of 1.5 million one-step transition tuples of the form \( (s_t, a_t, s_{t+1}, r_{t+1}) \) in the training set and 0.5 million in the test set respectively, where each state in the
tuple is a 32-dimensional continuous representation of patient physiology, each action is two-dimensional and can take one of eight discrete values, and the scalar rewards indicate the “goodness” of each transition with respect to patient outcome. In our policy optimization, we use discount factor $\gamma = 0.9$, such that rewards observed 24 hours in the future then have approximately one tenth the weight of immediate rewards, when determining the optimal action at a given state.

As an initial baseline, we look to apply Q-learning on the training data to learn the mapping of continuous states to Q-values, with function approximation using a simple three-layer feedforward neural network. The network is trained using Adam, an efficient stochastic gradient-based optimizer $\text{Adam}$, and $l_2$ regularization of weights. Each patient admission $k$ is treated as a distinct episode, with of the order of thousands of state transitions in each, and the network weights are incrementally updated following each transition. The change between successive episodes in the predicted Q-values for all state-action pairs in the training set is plotted in Figure 3.4—it is unclear whether the algorithm succeeds in converging within the 1,800 training episodes.

We then explore the use of Fitted Q-iteration instead to learn our Q-function, first running with an Extra-Trees regressor. In our implementation, each iteration of FQI is performed on a sampled subset of 10% of all transitions in the training set, as described in Algorithm 2, such that on average, each sample is seen in a tenth of all iterations. Though sampling increases the total number of iterations required for convergence, it yields significant speed-ups in building trees at each iteration, and hence in total training time. The ensemble regressor learns 50 trees, with regularization in the form of a minimum leaf node size of 20 samples. We present here results with FQI performed for a fixed number of 100 iterations, though it is possible to use a convergence criterion of the form $\Delta(Q_k, Q_{k-1}) \leq \varepsilon$ for early stopping, in order to speed up training further.
The same methods are then used to run FQI with neural networks (NFQ) in place of tree-based regression: we train a feedforward network with architecture and techniques identical to those applied in function approximation with Q-learning. Convergence of the estimated Q-function for both regressors, measured by the mean change in the estimate \( \hat{Q} \) for transitions in the training set, is plotted in Figure 3.5; we can see that the algorithm takes roughly 60 iterations to converge in both cases. However, NFQ yields approximately a four-fold gain in runtime speed, as expected, since with neural networks we can incrementally update weights rather than retraining the network with a cold start at each iteration.

The estimated Q-functions from FQI with Extra-Trees (FQIT) and from NFQ are then used to evaluate the optimal action, i.e. that which maximizes the value of the state-action pair, for each state in the training set. We can then train policy functions \( \pi(s) \) mapping a given patient state to the corresponding optimal action \( a \in \mathcal{A} \). To
allow for clinical interpretation of the final policy, we choose to train an Extra-Trees classifier comprising an ensemble of 100 trees to represent the policy function.

Figure 3.6 gives the relative weight assigned to the top 24 features in the state space for the policy trees learnt, when training on optimal actions from both FQIT and NFQ. Feature importances are obtained using the Gini or mean decrease in impurity importance score. The five vitals ranking highest in importance across the two policies are arterial $O_2$ pressure, arterial pH, $FiO_2$, $O_2$ flow and PEEP set. These are as expected—arterial pH, $FiO_2$ and PEEP all feature in our preliminary HUP guidelines for extubation criteria, and there is considerable literature suggesting blood gases are an important indicator of readiness for weaning [42]. On the other hand, oxygen saturation pulse oxymetry ($SpO_2$) which is also included in HUP’s current extubation criteria, is fairly low in ranking. This may be because these measurements are highly correlated with other factors in the state space, for example arterial $O_2$
pressure\cite{17}, that account for its influence on weaning more directly. The limited importance assigned to heart rate and respiratory rate are also likely to be explained by this dependence between vitals. In terms of demographics, weight and age play a significant role in the weaning policy learnt: weight is likely to influence our sedation policy specifically, as dosages are typically adjusted for patient weight, while age can be strongly correlated with a patient’s speed of recovery, and hence the time needed on ventilator support.

In order to evaluate the performance of the policies learnt, we compare the algorithm’s recommendations against the true policy implemented by the hospital. Considering ventilation and sedation separately, the policies learnt with FQIT and NFQ achieve similar accuracies in recommending ventilation (both matching the true policy in approximately 85% of transitions), while FQIT far outperforms NFQ in the case of sedation policy (achieving 58% accuracy compared with just 28%, barely outperformed random dosage level), perhaps due to overfitting of the neural network on this...
dataset—it is likely that more data is necessary to develop a meaningful sedation policy with NFQ. We therefore concentrate further analysis of policy recommendations to those produced by FQIT.

Given the long horizons of MDPs in this task, and the size of the action space, traditional off-policy evaluation estimators such as importance sampling yield incredibly high variance estimates of performance. Instead, we consider applying a variant of simpler rejection-sampling approaches, detailed here. We divide the 664 test admissions into six groups according to the fraction of FQI policy actions that differ from the hospital’s policy: $\Delta_0$ comprises admissions in which the true and recommended policies agree perfectly, while those in $\Delta_5$ show the greatest deviation. Figure 3.7a and 3.7b plot the distribution of the number of reintubations and the mean accumulated reward over patient admissions respectively, for all patients in each set; we can see
that those admissions in set $\Delta_0$ undergo no reintubation, and in general the average number of reintubations increases with deviation from the FQIT policy, with up to seven distinct intubations observed in admissions in $\Delta_5$. This effect is emphasised by the trend in mean rewards across the six admission groups, which serve primarily as an indicator of the regulation of vitals within desired ranges and whether certain criteria were met at extubation: we can see that mean reward over a set is highest (and the range lowest) for admissions in which the policies match perfectly, and decreases with increasing divergence of the two policies. A less distinct but very much comparable pattern is seen when grouping admissions instead by similarity of the sedation policy to the true dosage levels administered by the hospital; Figure 3.7c and 3.7d illustrate the trends in the number of reintubations and in mean rewards respectively.

3.4 Conclusion

In this chapter, we propose a data-driven approach to the optimization of weaning from mechanical ventilation of patients in the ICU. We model patient admissions as Markov decision processes, developing novel representations of the problem state, action space and reward function in this framework. Reinforcement learning with fitted Q-iteration using different regressors is then used to learn a simple ventilator weaning policy from examples in historical ICU data. We demonstrate that the algorithm is capable of extracting meaningful indicators for patient readiness and shows promise in recommending extubation time and sedation levels, on average outperforming clinical practice in terms of regulation of vitals and reintubations.

There are a number of challenges that must be overcome before these methods can be meaningfully implemented in a clinical setting, however: firstly, in order to generate robust treatment recommendations, it is important to ensure policy invariance to
reward shaping: the current methods display considerable sensitivity to the relative weighting of various components of the feedback received after each transition. A more principled approach to the design of the reward function, can help tackle this sensitivity. In addition, addressing the question of censoring in sub-optimal historical data and explicitly correcting for the bias that arises from the timing of interventions is crucial to fair evaluation of learnt policies, particularly where they deviate from the actions taken by the clinician. Finally, effective communication of the best action, expected reward, and the associated uncertainty, calls for a probabilistic approach to estimation of the Q-function, which can perhaps be addressed by pairing regressors such as Gaussian processes with Fitted Q-iteration.

Possible avenues for future work also include increasing the sophistication of the state space, for example by handling long term effects more explicitly using second-order statistics of vitals, applying techniques in inverse reinforcement learning to feature engineering [67], or modeling patient admissions as a partially observable MDP, in which raw observations of the patient physiology are drawn from some true underlying state. Extending the action space to include continuous dosages of specific drug types and explicit settings such as the inspired oxygen fraction or the value of PEEP set can also facilitate directly executable policy recommendations, and enable better informed decisions in critical care.
Chapter 4

Optimizing Laboratory Tests with Multi-objective RL

Precise, targeted patient monitoring is central to improving treatment in an ICU, allowing clinicians to detect changes in patient state and to intervene promptly and only when necessary. While basic physiological parameters that can be monitored bedside (e.g., heart rate) are recorded continually, those that require invasive or expensive laboratory tests (e.g., white blood cell counts) are more intermittently sampled. These lab tests are estimated to influence up to 70% percent of diagnoses or treatment decisions, and are often cited as the motivation for more costly downstream care [7, 134]. Recent medical reviews raise several concerns about the over-ordering of lab tests in the ICU [74]. Redundant testing can occur when labs are ordered by multiple clinicians treating the same patient or when recurring orders are placed without reassessment of clinical necessity. Many of these orders occur at time intervals that are unlikely to include a clinically relevant change or when large panel testing is repeated to detect a change in a small subset of analyses [58]. This leads to inflation in costs of care and in the likelihood of false positives in diagnostics, and also causes unnecessary discomfort to the patient. Moreover, excessive phlebotomies (blood tests)
can contribute to risk of hospital-acquired anaemia; around 95% of patients in the ICU have below normal haemoglobin levels by day 3 of admission and are in need of blood transfusions. It has been shown that phlebotomy accounts for almost half the variation in the amount of blood transfused [46].

With the disproportionate rise in lab costs relative to medical activity in recent years, there is a pressing need for a sustainable approach to test ordering. A variety of approaches have been considered to this end, including restrictions on the minimum time interval between tests or the total number of tests ordered per week. More data-driven approaches include an information theoretic framework to analyze the amount of novel information in each ICU lab test by computing conditional entropy and quantifying the decrease in novel information of a test over the first three days of an admission [65]. In a similar vein, a binary classifier was trained using fuzzy modeling to determine whether or not a given lab test contributes to information gain in the clinical management of patients with gastrointestinal bleeding [15]. An “informative” lab test is one in which there is significant change in the value of the tested parameter, or where values were beyond certain clinically defined thresholds; the results suggest a 50% reduction in lab tests compared with observed behaviour. More recent work looked at predicting the results of ferritin testing for iron deficiency from information in other labs performed concurrently [76]. The predictability of the measurement is inversely proportional to the novel information in the test. These past approaches underscore the high levels of redundancy that arise from current practice. However, there are many key clinical factors that have not been previously accounted for, such as the low-cost predictive information available from vital signs, causal connection of clinical interventions with test results, and the relative costs or feasibility constraints associated with ordering various tests.

In this chapter, we introduce a reinforcement learning (RL) based method to tackle the problem of developing a policy to perform actionable lab testing in ICU patients.
Our approach is two-fold: first, we build an interpretable model to forecast future patient states based on past observations, including uncertainty quantification. We adapt multi-output Gaussian processes (MOGPs; [12, 30]) to learn the patient state transition dynamics from a patient cohort including sparse and irregularly sampled medical time series data, and to predict future states of a given patient trajectory. Second, we model patient trajectories as a Markov decision process (MDP). In doing so, we draw from the framework introduced in Chapter 4 to efficiently wean patients from mechanical ventilation [98], as well as other work on recommendation of treatment strategies for critical care patients in a variety of different settings [88, 103]. We design the state and reward functions of the MDP to incorporate relevant clinical information, such as the expected information gain, subsequent administered interventions, and the costs of actions (namely, requesting and performing a lab test).

A major challenge is designing a reward function that can trade off multiple, often opposing, objectives. There has been initial work on extending the MDP framework to composite reward functions [85]. Specifically, fitted Q-iteration (FQI) has been used to learn policies for multi-objective MDPs with vector-valued rewards, for the sequence of interventions in two-stage clinical antipsychotic trials [72]. A variation of Pareto domination was then used to generate a partial ordering of policies and extract all policies that are optimal for some scalarization function, leaving the choice of parameters of the scalarization function to decision makers.

Here, we look to translate these principles to the problem of lab test ordering. Specifically, we focus on blood tests relevant in the diagnosis of sepsis or acute renal failure, two conditions with high prevalence in the ICU, as well as high associated mortality risk in the ICU. These tests are white blood cell count (WBC), blood lactate level, serum creatinine, and blood urea nitrogen (BUN); abnormalities in the first two markers are commonly used in diagnosis of severe sepsis, while the latter are associated with compromised kidney function. We present our methods within a
flexible framework that can in principle be adapted to a patient cohort with different diagnoses or treatment objectives, influenced by a distinct set of lab results. Our proposed framework integrates prior work on off-policy RL and Pareto learning with practical clinical constraints to yield policies that are close to intuition demonstrated in historical data. Again, we demonstrate our approach using a publicly available database of ICU admissions, evaluating the estimated policy against the policy followed by clinicians using both importance sampling based estimators for off-policy policy evaluation and by comparing against multiple clinically inspired objectives, including onset of clinical treatment that was motivated by the lab results.


*Much of the work detailed in this chapter was developed jointly with Li-Fang Cheng. I sincerely thank her for her contribution.*

### 4.1 Methods

#### 4.1.1 MIMIC Cohort Selection and Preprocessing

We extract our cohort of interest from the MIMIC III database [19], which includes de-identified critical care data from over 58,000 hospital admissions. From this database, we first select adult patients with at least one recorded measure for each of 20 vital signs and lab tests commonly ordered and reviewed by clinicians (for instance, results reported in a complete blood count or basic metabolic panel). We further filter patients by their length-of-stay, keeping only those in the ICU for between one and twenty days, to obtain a final set of 6,060 patients. Table [14] summarizes key statistics for patient physiological parameters in this filtered cohort.
Table 4.1: Total number of nurse-verified recordings, measurement mean and standard deviation (SD) for covariates in selected cohort.

<table>
<thead>
<tr>
<th>Covariate</th>
<th>Count</th>
<th>Mean</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Respiratory Rate (RR)</td>
<td>1,046,364</td>
<td>20.1</td>
<td>5.7</td>
</tr>
<tr>
<td>Heart Rate (HR)</td>
<td>964,804</td>
<td>87.5</td>
<td>18.2</td>
</tr>
<tr>
<td>Mean Blood Pressure (Mean BP)</td>
<td>969,062</td>
<td>77.9</td>
<td>15.3</td>
</tr>
<tr>
<td>Temperature, °F</td>
<td>209,499</td>
<td>98.5</td>
<td>1.4</td>
</tr>
<tr>
<td>Creatinine</td>
<td>67,565</td>
<td>1.5</td>
<td>1.2</td>
</tr>
<tr>
<td>Blood Urea Nitrogen (BUN)</td>
<td>66,746</td>
<td>31.0</td>
<td>21.1</td>
</tr>
<tr>
<td>White Blood Cell Count (WBC)</td>
<td>59,777</td>
<td>11.6</td>
<td>6.2</td>
</tr>
<tr>
<td>Lactate</td>
<td>39,667</td>
<td>2.4</td>
<td>1.8</td>
</tr>
</tbody>
</table>

Included in the 20 physiological traits we filter for are eight that are particularly predictive of the onset of severe sepsis, septic shock, or acute kidney failure. These traits are included in the SIRS (System Inflammatory Response Syndrome) and SOFA (Sequential Organ Failure Assessment) criteria [78]. The average daily measurements or lab test orders across the chosen cohort for these eight traits is highly variable (Figure 4.1). Of these eight traits, the first three are vitals measured using bedside monitoring systems for which approximately hourly measurements are recorded; the latter four are labs requiring phlebotomy and are typically measured just 2–3 times each day. We find the frequency of orders also varies across different labs, possibly due in part to differences in cost; for example, WBC (which is relatively inexpensive to test) is on average sampled slightly more often than lactate.

In order to apply our proposed RL algorithm to this sparse, irregularly sampled dataset, we adapt the multi-output Gaussian process (MOGP) framework [12] to obtain hourly predictions of patient state with uncertainty quantified, on 17 of the 20 clinical traits. For three of the vitals, namely the components of the Glasgow Coma Scale, we impute with the last recorded measurement.
Figure 4.1: Mean recorded measurements per day, of eight key vitals and lab tests.

### 4.1.2 Designing a Multi-Objective MDP

Each patient admission is modelled as a Markov decision process defined by: (i) state space \( \mathcal{S} \), where \( s_t \in \mathcal{S} \) is patient physiological state at time \( t \); (ii) action space \( \mathcal{A} \) from which the clinician’s action \( a_t \) is chosen; (iii) unknown transition function \( \mathcal{P}(s,a) \) that determines the patient dynamics; and (iv) reward function \( r_{t+1} = r(s_t, a_t) \) which determines observed clinical feedback for this action. The objective of the RL agent is to learn an optimal policy \( \pi^* : \mathcal{S} \to \mathcal{A} \) that maximizes the expected discounted (with some factor \( \gamma \)) accumulated reward over the course of an admission:

\[
\pi^* = \arg\max_{\pi} \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t r_t | \pi \right]
\]

We start by describing the state space of our MDP for ordering lab tests. We first re-sample the raw time series using a multi-objective Gaussian process with a sampling period of one hour. The patient state at time \( t \) is defined by:

\[
s_t = \begin{bmatrix} m_{t}^{\text{SOFA}}, & m_{t}^{\text{vitals}}, & m_{t}^{\text{labs}}, & y_{t}^{\text{labs}}, & \Delta_{t}^{\text{labs}} \end{bmatrix}^T
\]

(4.1)

Here, \( m_t \) denotes the predictive means and standard deviations respectively of each of the vitals and lab tests. For the predictive SOFA score \( m_t^{\text{SOFA}} \), we compute the
value using its clinical definition, from the predictive means on five traits—mean BP, bilirubin, platelet, creatinine, $FiO_2$—along with GCS and related medication history (e.g., dopamine). Vitals include any time-varying physiological traits that we consider when determining whether to order a lab test. Here, we look at four key physiological traits—heart rate, respiratory rate, temperature, and mean blood pressure—and four lab tests—creatinine, BUN, WBC, and lactate. The values $y_t$ are the last known measurements of each of the four labs, and $\Delta_t$ denotes the elapsed time since each was last ordered. This formulation results in a 21-dimensional state space. Depending on the labs that we wish to learn recommendations for testing, the action space $A$ is a set of binary vectors whose 0/1 elements indicate whether or not to place an order for a specific lab. These actions can be written as $a_t \in A = \{1, 0\}^L$, where $L$ is the number of labs. In our experiments, we learn policies for each of the four labs independently, such that $L = 1$, but this framework could be easily extended to jointly learning recommendations for multiple labs.

In order for our RL agent to learn a meaningful policy, we need to design a reward function that provides positive feedback for the ordering of tests where necessary, while penalizing the over- or under-ordering of any given lab test. In particular, the agent should be encouraged to order labs when the physiological state of the patient is abnormal with high probability, based on estimates from the MOGP, or when a lab is predicted to be informative (in that the forecasted value is significantly different from the last known measurement) due to a sudden change in disease state. In addition, the agent should incur some penalty whenever a lab test is taken, decaying with elapsed time since the last measurement, to reflect the effective cost (both economic and in terms of discomfort to the patient) of the test. We formulate these ideas into a vector-valued reward function $r_t \in \mathbb{R}^d$ of the state and action at time $t$, as follows:

$$r_t = \begin{bmatrix} r_t^{SOFA} ; r_t^{treat} ; r_t^{info} ; -r_t^{cost} \end{bmatrix}^\top$$  (4.2)
**Patient state:** The first element, $r^{SOFA}$, uses the recently introduced SOFA score for sepsis [112] which assesses severity of organ dysfunction in a potentially septic patient. Our use of SOFA is motivated by the fact that, in practice, sepsis is more often recognized from the associated organ failure than from direct detection of the infection itself [122]. The raw SOFA score ranges from 0 to 24, with a maximum of four points assigned each to symptom of failure in the respiratory system, nervous system, liver, kidneys, and blood coagulation. A change in SOFA score $\geq 2$ is considered a critical index for sepsis [112]. This rule of thumb is used to define the first reward term:

$$r^t_{SOFA} = 1_{a_t \neq 0} \cdot 1_{f(\cdot) \geq 2}, \text{ where } f(\cdot) = m^t_{SOFA} - m^{SOFA}_{t-1}.$$  \hspace{1cm} (4.3)

The raw score $m^t_{SOFA}$ at each $t$ is evaluated using current patient labs and vitals [122].

**Treatment onset:** The second term is an indicator variable for rewards capturing whether or not there is some treatment or intervention initiated at the next time step:

$$r^t_{treat} = 1_{a_t \neq 0} \cdot \sum_{i \in M} 1_{s_{t+1}(\text{treatment } i \text{ was given})},$$ \hspace{1cm} (4.4)

where $M$ denotes the set of disease-specific interventions of interest. Again, the reward term is positive if a lab is ordered; this is based on the rationale that, if a lab test is ordered and immediately followed by an intervention, the test is likely to have provided actionable information. Possible interventions include antibiotics, vasopressors, dialysis or ventilation.

**Lab redundancy:** The term $r^t_{info}$ denotes the feedback from taking one or more lab tests with novel information. We quantify this by using the mean absolute difference between the last observed value and predictive mean from the MOGP as a proxy for
the information available:

\[ r_t^{info} = \sum_{\ell=1}^{L} \max(0, g(\cdot) - c_\ell) \cdot 1_{a_t[\ell]=1}, \text{ where } g(\cdot) = \left| \frac{m_t^{(\ell)} - y_t^{(\ell)}}{\sigma_t^{(\ell)}} \right|, \quad (4.5) \]

where \( \sigma_t^{(\ell)} \) is the normalization coefficient for lab \( \ell \), and the parameter \( c_\ell \) determines the minimum prediction error necessary to trigger a reward; in our experiments, this is set to the median prediction error for labs ordered in the training data. The larger the deviation from current forecasts, the higher the potential information gain, and in turn the reward if the lab is taken.

**Lab cost:** The last term in the reward function, \( r_t^{cost} \), adds a penalty whenever any test is ordered to reflect the effective “cost” of taking the lab at time \( t \).

\[ r_t^{cost} = \sum_{\ell=1}^{L} \exp \left( -\frac{\Delta_t^{(\ell)}}{\Gamma_\ell} \right) \cdot 1_{a_t[\ell]=1}, \quad (4.6) \]

where \( \Gamma_\ell \) is a decay factor that controls the how fast the cost decays with the time \( \Delta_t \) elapsed since the last measurement. In our experiments, we set \( \Gamma_\ell = 6 \forall \ell \in L \).

### 4.1.3 Solving for Deterministic Optimal Policy

Once we extract sequences of states, actions, and rewards from the ICU data, we can generate a dataset of one-step transition tuples of the form \( D = \{ (s^n_t, a^n_t, s^n_{t+1}, r^n_t) \}, \ n = 1 \ldots |D| \). These tuples can then be used to learn an estimate of the Q-function, \( \hat{Q} : S \times A \rightarrow \mathbb{R}^d \) —where \( d = 4 \) is the dimensionality of the reward function—to map a given state-action pair to a vector of expected cumulative rewards. Each element in the Q-vector represents the estimated value of that state-action pair according to a different objective. We learn this Q-function using a variant of Fitted Q-iteration (FQI) with extremely randomized trees [21, 28]. FQI is a batch off-policy reinforcement learning algorithm that is well-suited to clinical applications where we have
limited data and challenging state dynamics. The algorithm adapted here to handle vector-valued rewards is based on Pareto-optimal Fitted-Q [72].

In order to scale from the two-stage decision problem originally tackled to the much longer admission sequences here ($\geq 24$ time steps), we define a stricter pruning of actions: at each iteration we eliminate any dominated actions for a given state—those actions that are outperformed by alternatives for all elements of the Q-function—and retain only the set $\Pi(s) = \{a : \exists a' (\forall d, \hat{Q}_d(s, a) < \hat{Q}_d(s, a'))\}$ for each $s$. Actions are further filtered for consistency: we might consider feature consistency to be defined as rewards being linear in each feature space [72]. Here, we relax this idea to filter out only those actions from policies that cannot be expressed by our non-linear tree-based classifier. The function will still yield a non-deterministic policy (NDP) as, in most cases, there will not be a strictly optimal action that achieves the highest $Q_d$ for all $d$. We suggest one possible approach for reducing the NDP to give a single best action for any given state based on practical considerations in the next section.

4.2 Results

Following the extraction of our 6,060 admissions and resampling in hourly intervals using the forecasting MOGP, we partitioned the cohort into training and test sets of 3,636 and 2,424 admissions respectively. This gave approximately 500,000 one-step transition tuples of the form $\langle s_t, a_t, s_{t+1}, r_t \rangle$ in the training set, and over 350,000 in the test set. We then ran batched FQI with these samples for 200 iterations with discount factor $\gamma = 0.9$. Each iteration took 100,000 transitions, sampled from the training set, with probability inversely proportional to the frequency of the action in the tuple. The vector-valued outputs of estimated Q-function were then used to obtain a non-deterministic policy for each lab considered (Section 4.1.3). We chose
Algorithm 2 Multi-Objective Fitted Q-iteration with strict pruning

Input:
One-step transitions $\mathcal{F} = \{(s^n_t, a^n_t, s^n_{t+1}, r^n_{t+1})_{n=1:|\mathcal{F}|}\}$
Regression parameters $\theta$; action space $\mathcal{A}$; subset size $N$

Initialize $Q^{(0)}(s_t, a_t) = 0 \in \mathbb{R}^d \; \forall s_t \in \mathcal{F}, a_t \in \mathcal{A}$

for iteration $k = 1 \rightarrow K$ do
  Sample subset $\mathcal{F}_N \sim \mathcal{F}$; initialize $S \leftarrow []$
  for $i \in \text{subset}_N$ do
    Generate set $\Pi(s_i)$ using $Q^{(k-1)}$
    Initialize classification parameters $\phi$
    $\phi \leftarrow \text{classify}(s_i, a_i)$
    for $\pi_i \in \Pi :$ do
      $a' \leftarrow \pi_i(s_{i+1}) \cap \text{predict}(s_{i+1}, \phi)$
      $Q^{(k)}(s_i, a_i) \leftarrow r_{i+1} + \gamma Q^{(k-1)}(s_{i+1}, a')$
    end
    $S \leftarrow \text{append}(S, ((s_i, a_i), Q^{(k)}(s_i, a_i)))$
  end
  $\theta \leftarrow \text{regress}(S)$
end

Result: $\theta$

to collapse this set to a practical deterministic policy as follows:

$$\Pi(s) = \begin{cases} 
1, & \hat{Q}_d(s, a = 0) < \hat{Q}_d(s, a = 1) + \varepsilon_d, \; \forall d \\
0, & \text{otherwise.} 
\end{cases} \quad (4.7)$$

In particular, a lab should be taken ($\Pi(s) = 1$) only if the action is optimal, or estimated to outperform the alternative for all objectives in the Q-function. This strong condition for ordering a lab is motivated by the fact that one of our primary objectives here is to minimize unnecessary ordering; the variable $\varepsilon_d$ allows us to relax this for certain objectives if desired. For example, if cost is a softer constraint, setting $\varepsilon_{\text{cost}} > 0$ is an intuitive way to specify this preference in the policy. In our
experiments, we tuned $\varepsilon_{\text{cost}}$ such that the total number of recommended orders of each lab approximates the number of actual orders in the training set.

With a deterministic set of optimal actions, we could train our final policy function $\pi : S \rightarrow A$; again, we used extremely randomized trees. The estimated Gini feature importances of the policies learnt show that in the case of lactate the most important features are the mean and measured lactate, the time since last lactate measurement ($\Delta$) and the SOFA score (Figure 4.2). These relative importance scores are expected: a change in SOFA score may indicate the onset of sepsis, and in turn warrant a lactate test to confirm a source of infection, fitting typical clinical protocol. For the other three policies (WBC, creatinine, BUN) again the time since last measurement of the respective lab tends be prominent in the policy, along with the $\Delta$ terms for the other two labs. This suggests an overlap in information in these tests: For example, abnormally high white blood cell count is a key criteria for sepsis; severe sepsis often cascades into renal failure, which is typically diagnosed by elevated BUN and creatinine levels [56].
Once we have trained our policy functions, an additional component is added to our final recommendations: we introduce a budget that suggests taking a lab at the end of every 24 hour period for which our policy recommends no orders. This allows us to handle regions of very sparse recommendations by the policy function, and reflects clinical protocols that require minimum daily monitoring of key labs. In the policy for lactate orders in a typical patient admission, looking at the timing of the actual clinician orders, recommendations from our policy, and suggested orders from the budget framework, the actions are concentrated where lactate values are increasingly abnormal, or at sharp rises in SOFA score (Figure 4.3).

4.2.1 Off-Policy Evaluation

We evaluated the quality of our final policy recommendations in a number of ways. First, we implemented the per-step or per decision weighted importance sampling (PDWIS) estimator to calculate the value of the policy \( \pi_e \) to be evaluated:

\[
\hat{V}_{PDWIS}(\pi_e) = \sum_{i=1}^{n} \sum_{t=0}^{T-1} \gamma_{WIS}^t \left[ \frac{\rho_t^{(i)}}{\sum_{i=1}^{n} \rho_t^{(i)}} \right] r_t^{(i)}, \quad \text{where} \quad \rho_t = \prod_{j=0}^{t-1} \frac{\pi_e(s_j|a_j)}{\pi_b(s_j|a_j)},
\]

given data collected from behaviour policy \( \pi_b \). The behaviour policy was found by training a regressor on real state-action pairs observed in the dataset. The discount factor was set to \( \gamma_{WIS} = 1.0 \), so all time steps contribute equally to the value of a trajectory.

We then compared estimates for our policy (MO-FQI) against the behaviour policy and a set of randomized policies as baselines. These randomized policies were designed to generate random decisions to order a lab, with probabilities \( p = \{0.01, p_{emp}, 0.5\} \), where \( p_{emp} \) is the empirical probability of an order in the behaviour policy. For each \( p \), we evaluated ten randomly generated policies and averaged performance over these. We observed that MO-FQI outperforms the behaviour policy across all reward com-
Figure 4.3: Demonstration of recommended lactate ordering policy for example admission; shaded green region denotes normal lactate range (0.5–2 mmol/L).

ponents, for all four labs (Figure 4.4). Our policy also consistently approximately matches or outperforms the other policies in terms of cost—note that for absolute cost, the best policy corresponds to that with the lowest estimated value—even with the inclusion of the slack variable $\varepsilon_{cost}$ and the budget framework. Across the remaining objectives, MO-FQI outperforms the random policy in at least two of three components for all but lactate. This may be due in part to the relatively sparse orders for lactate resulting in higher variance value estimates.

In addition to evaluating using the per-step WIS estimator, we looked for more intuitive measures of how the final policy influences clinical practice. We computed three metrics here: (i) estimated reduction in total number of orders, (ii) mean information gain of orders taken, and (iii) time intervals between labs and subsequent treatment onsets.

In evaluating the total number of recommended orders, we first filter a sequence of recommended orders to the just the first (onset) of recommendations if there are no clinician orders between them. We argue that this is a fair comparison as subsequent recommendations are made without counterfactual state estimation, i.e., without assuming that the first recommendation was followed the clinician. Empirically, we find that the total number of recommendations is considerably reduced. For instance, in the case of recommending WBC orders, our final policy reports 12,358 orders in the
Figure 4.4: Evaluating $\hat{V}_d(\pi_\varepsilon)$ for each reward component $d$, across policies for four labs. The (⋆) indicates the best performing policy for each reward component. Error bars for randomized policies show standard deviations across 10 trials.

test set, achieving a reduction of 44% from the number of true orders (22,172). In the case of lactate, for which clinicians’ orders are the least frequent (14,558), we still achieved a reduction of 27%.

We also compared the approximate information gain of the actions taken by the estimated policy, in comparison with the policy used in the collected data. To do this, we defined the information gain at a given time by looking at the difference between the approximated true value of the target lab, which we impute using the MOGP model given all the observed values, and the forecasted value, computed using only the values observed before the current time. The distribution of aggregate information gain for orders recommended by our policy and actual clinician’s orders in the test set shows consistently higher expected mean information gain following ordering policies learnt from MO-FQI, across all four labs (Figure 4.5).
Lastly, we considered the time to onset of critical interventions, which we define to include initiation of vasopressors, antibiotics, mechanical ventilation or dialysis. We first obtained a sequence of treatment onset times for each test patient; for each of these time points, we traced back to the earliest observed or recommended order taking place within the past 48 hours, and computed the time between these: $\Delta t = t_{\text{treatment}} - t_{\text{order}}$. The distribution of time-to-treatment for labs taken by the clinician in the true trajectory against that for recommendations from our policy, for all four labs, shows that the recommended orders tend to happen earlier than the actual time of an order by the clinician—on average over an hour in advance for lactate, and more that four hours in advance for WBC, creatinine, and BUN (Figure 4.6).

4.3 Conclusion

In this work, we propose a reinforcement learning framework for decision support in the ICU that learns a compositional optimal treatment policy for the ordering of lab tests from sub-optimal histories. We do this by designing a multi-objective reward function that reflects clinical considerations when ordering labs, and adapting meth-
ods for multi-objective batch RL to learning extended sequences of Pareto-optimal actions. Our final policies are evaluated using importance-sampling based estimators for off-policy evaluation, metrics for improvements in cost, and reducing redundancy of orders. Our results suggest that there is considerable room for improvement on current ordering practices, and the framework introduced here can help recommend best practices and be used to evaluate deviations from these across care providers, driving us towards more efficient health care. Furthermore, the low risk of these types of interventions in patient health care reduces the barrier of testing and deploying clinician-in-the-loop machine learning-assisted patient care in ICU settings.

Figure 4.6: Evaluating time to treatment onset for lab orders by the clinician against MO-FQI, across all labs. The mean time intervals are as follows: 9.1 vs 13.2 for WBC; 7.9 vs 12.5 for creatinine; 8.0 vs 12.5 for BUN; 14.4 vs 15.9 for lactate.
Chapter 5

Constrained Reward Design for Batch RL

One fundamental challenge of reinforcement learning (RL) in practice is specifying the agent’s reward. Reward functions implicitly define policy, and misspecified rewards can introduce severe, unexpected effects, from reward gaming to irreversible changes in parts of the environment we do not want to influence. However, it can be difficult for domain experts to distil multiple (and often implicit) requisites for desired behaviour into a single scalar feedback signal. This is exemplified by efforts towards the application of reinforcement learning to decision-making in healthcare; in RL, an agent aims to choose the best action within a stochastic process given inherent time delay in feedback from a decision, making it an attractive framework for learning clinical treatment policies. However, this feedback can be received over various time scales and represent clinical implications—such as treatment efficacy, side effects or patient discomfort—with widely different, and uncertain, priorities. Existing approaches to representing this scalar feedback in healthcare tasks range from taking reward to be a sparse, high-level signal such as mortality or rewards based on a
single physiological variable or severity score of interest \[88, 111\] to relatively ad hoc weighting of clinically derived objectives, as in Chapter 3.

Much work in reward design \[113, 114\] or inference using inverse reinforcement learning \[11, 9, 37\] focuses on online, interactive settings in which the agent has access either to human feedback \[14, 73\] or to a simulator with which to evaluate policies and compare against human performance. Here, we focus on reward design for batch RL: we assume access only to a set of past trajectories collected from sub-optimal experts, with which to train our policies. This is common in many real-world scenarios where the risks of deploying an agent are high but logging current practice is relatively easy, as in healthcare, as well as education or finance \[6, 19\].

Batch RL is distinguished by two key preconditions when performing reward design. First, as we assume that data are expensive to acquire, we must ensure that policies found using the reward function can be evaluated given existing data. Regardless of the true objectives of the designer, there exist fundamental limitations on reward functions that can be optimized and that also provide guarantees on performance. There have been a number of methods presented in the literature for safe, high-confidence policy improvement from batch data given some reward function, treating behaviour seen in the data as a baseline \[31, 63, 107, 118\]. In this work, we turn this question around to ask: What is the class of reward functions for which high-confidence policy improvement is possible?

Second, we typically assume that batch data are not random but produced by domain experts pursuing biased but reasonable policies. Thus if an expert-specified reward function results in behaviour that diverges substantially from past trajectories, we must ask whether that divergence was intentional or, as is more likely, simply because the designer omitted an important constraint, causing the agent to learn unintentional behaviour. This assumption can be formalized by treating the batch data as \(\varepsilon\)-optimal with respect to the true reward function, and searching for rewards
that are consistent with this assumption \cite{foster2008learning}. Here, we extend these ideas to incorporate the uncertainty present when evaluating a policy in the batch setting, where trajectories from the estimated policy cannot be collected.

We note that these two constraints are not equivalent; the extent of overlap in reward functions satisfying these criteria depends, for example, on the homogeneity of behaviour in the batch data: if consistency is measured with respect to average behaviour in the data, and agents deviate substantially from this average—as may be across clinical care providers—then the space of policies that can be evaluated given the batch data may be larger than the space consistent with the average expert.

In this chapter, we combine these two conditions to construct tests for admissible functions in reward design using available data. This yields a novel approach to the challenge of high-confidence policy evaluation given high-variance importance sampling-based value estimates over extended decision horizons—typical of batch RL problems—and encourages safe, incremental policy improvement. We illustrate our approach on several benchmark control tasks with continuous state spaces, and in reward design for the task of weaning a patient from a mechanical ventilator.


### 5.1 Preliminaries and Notation

A Markov decision process (MDP) is a tuple of the form $M = \{S, A, P_0, P, R, \gamma\}$, where $S$ is the set of all possible states, and $A$ are the available actions. $P_0(s)$ is the distribution over the initial state $s \in S$; $P(s'|s,a)$ gives the probability of transition to $s'$ given current state $s$ and action $a \in A$. The function $R(s,a,s')$ defines the
reward for performing action $a$ in state $s$, and observing new state $s’$. Lastly, the discount factor $\gamma \leq 1$ determines the relative importance of immediate and longer-term rewards received by the reinforcement learning agent.

Our objective is to learn a policy function $\pi^* : \mathcal{S} \rightarrow \mathcal{A}$ mapping states to actions that maximize the expected cumulative discounted reward—that is, $\pi^* = \arg\max_{\pi} \mathbb{E}_{s \sim P_0}[V^\pi(s)|M]$—where the value function $V^\pi(s)$ is defined as:

$$V^\pi = \mathbb{E}_{P_0, P, \pi}\left[ \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) \right].$$

(5.1)

In batch RL, we have a collection of trajectories of the form $h = \{s_0, a_0, r_0, \ldots, s_T, a_T, r_T\}$. We do not have access to the transition function $P$ or the initial state distribution $P_0$. Without loss of generality, we express the reward as a linear combination of some arbitrary function of the observed state transition: $R = w^T \phi(s, a, s’)$, where $\phi \in \mathbb{R}^k$ is a vector function of state-action features relevant to learning an optimal policy, and $\|w\|_1 = 1$, to induce invariance to scaling factors in reward specification [4]. The value $V^\pi$ of a policy $\pi$ with reward weight $w$ can then be written as:

$$V^\pi = \mathbb{E}_{P_0, P, \pi}\left[ \sum_{t=0}^{\infty} \gamma^t w^T \phi(\cdot) \right] = w^T \mu^\pi, \text{ where}$$

$$\mu^\pi = \mathbb{E}_{P_0, P, \pi}\left[ \sum_{t=0}^{\infty} \gamma^t \phi(\cdot) \right].$$

(5.2)

where the vector $\mu^\pi$ denotes the feature expectations [4] of policy $\pi$, that is, the total expected discounted time an agent spends in each feature state. Thus, $\mu^\pi$ provides a representation of the state dynamics of a policy that is entirely decoupled from the reward function of the MDP.

To quantify confidence in the estimated value $V^\pi$ of policy $\pi$, we adapt the empirical Bernstein concentration inequality [80] to get a probabilistic lower bound $V_{lb}$ on the estimated value [119]: consider a set of trajectories $h_n$, $n \in 1\ldots N$ and let $\hat{V}_n$
be the value estimate for trajectory \( n \). Then, with probability at least 1 – \( \delta \):

\[
V_{lb} = \frac{1}{N} \sum_{n=1}^{N} \hat{V}_n - \frac{1}{N} \sqrt{\frac{\ln(\frac{2}{\delta})}{N-1}} \sum_{n,n'=1}^{N} (\hat{V}_n - \hat{V}_{n'})^2 - \frac{7b \ln(\frac{2}{\delta})}{3(N-1)},
\]

(5.3)

where \( b \) is the maximum achievable value of \( V(\pi) \).

## 5.2 Admissible Reward Sets

We now turn to our task of identifying admissible reward sets – that is, defining the space of reward functions that yield policies that are consistent in performance with available observational data, as well as possible to evaluate off-policy for high-confidence performance lower bounds. In Sections 5.2.1 and 5.2.2, we define two sets of weights \( \mathcal{P}_C \) and \( \mathcal{P}_E \) to be the consistent and evaluable sets, respectively, show that they are closed and convex, and define their intersection \( \mathcal{P}_C \cap \mathcal{P}_E \) as the set of admissible reward weights. In Sections 5.2.3 and 5.2.4, we describe how to test whether a given reward lies in the intersection of these polytopes, and, if not, how to find the closest points within this space of admissible reward functions given some initial reward proposed by the designer of the RL agent.

### 5.2.1 Consistent Reward Polytope

Given near-optimal expert demonstrations, the polytope of consistent rewards \( \mathbf{R} \) may be defined as the set of all weight vectors \( w \) defining reward function \( R = w^T \phi(s) \), that are consistent with the agent’s existing knowledge. In the setting of learning from demonstrations, this knowledge is the assumption that demonstrations achieve \( \varepsilon \)-optimal performance with respect to the “true” reward. We denote the behaviour policy of experts as \( \pi_b \) with policy feature expectations \( \mu_b \), where \( V(\pi_b) = w^T \mu_b \). The consistent weight vectors for this expert demonstration setting are then all \( w \) such
that $w^T \mu \leq w^T \mu_b + \varepsilon$, $\mu \in \mathcal{P}_F$, where $\mathcal{P}_F$ is the space of all possible policy feature representations. It has been shown that this set is convex, given access to an exact MDP solver \cite{13}.

Translating this to the batch reinforcement learning setting, with a fixed set of sub-optimal trajectories, requires adaptations to both the constraints and their computation. First, we choose to constrain the relative rather than absolute difference in performance of the observed trajectories and that of the learnt optimal policy, in order to better handle high variance in the magnitudes of estimated values. Second, we make our constraint symmetric such that the value of the learnt policy can deviate equally above or below the value of the observed behaviour. This reflects the use of this constraint as a way to place metaphorical guardrails on the deviation of the behaviour of the learnt policy from the policy in the batch trajectories—rather than to impose optimality assumptions that only bound performance from above. That is, we want a reward that results in performance similar to the observed batch trajectories, where performance some factor $\Delta_c$ greater than or less than this established baseline should be equally admissible. Our new polytope $\mathcal{P}_C$ for the space of weights satisfying this is then:

$$\mathcal{P}_C = \left\{ w : \frac{1}{\Delta_c} \leq \frac{w^T \mu_b}{w^T \mu} \leq \Delta_c \right\}, \quad (5.4)$$

where $\mu$ are the feature expectations of the optimal policy when solving an MDP with reward weights $w$, and value estimates are constrained to be positive, $w^T \mu > 0 \; \forall \mu \in \mathcal{P}_F$. The parameter $\Delta_c \geq 1$ that determines the threshold on the consistency polytope is tuned according to our confidence in the batch data; trajectories from severely biased experts may warrant larger $\Delta_c$.

The batch setting also requires changes to the computation of these constraints, as we do not have access to a simulator to calculate exact feature expectations $\mu$;
we must instead estimate them from available data. We do so by adapting off-policy evaluation methods to estimate the representation of a policy in feature space. Specifically, we use per-decision importance sampling (PDIS) to get a consistent, unbiased estimator of $\mu$:

$$\hat{\mu} = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=0}^{T} \gamma^t \rho_t^{(n)} \phi(s_t^{(n)})$$

(5.5)

where importance weights $\rho_t^{(n)} = \prod_{i=0}^{t} (\pi(a_i^n|s_i^n)/\pi_b(a_i^n|s_i^n))$. Together with the feature expectations of the observed experts (obtained by simple averaging across trajectories), we can evaluate the constraint in Equation 5.4.

**Proposition 1.** The set of weights $P_C$ defines a closed convex set, given access to exact MDP solver.

**Proof.** The redefined constraints in Equation 5.4 can be rewritten as: $w^T(\mu - \Delta_b \mu_b) \leq 0$; $w^T(\mu_b - \Delta_c \mu) \leq 0$, where $\mu = \max_{\mu' \in \mathcal{P}_F} w^T \mu'$ is the feature expectations of the optimal policy obtained from the exact MDP solver. As these constraints are still linear in $w$—that is, of the form $w^T A \leq b$—the convexity argument in [43] holds.

In Section 5.2.3, we discuss how this assumption of convexity changes given the presence of approximation error in the MDP solver and in estimated feature expectations.

**Illustration.** We first construct a simple, synthetic task to visualize a polytope of consistent rewards. Consider an agent in a two-dimensional continuous environment, with state defined by position $s_t = [x_t, y_t]$ for bounded $x_t$ and $y_t$. At each time $t$, available actions are steps in one of four directions, with random step size $\delta_t \sim \mathcal{N}(0.4, 0.1)$. The reward is $r_t = [0.5, 0.5]^T s_t$: the agent’s goal is to reach the top-right corner of the 2D map. We use fitted-Q iteration with tree-based approximation [21] to learn a deterministic policy $\pi_b$ that optimizes the reward, then we sample 1000
trajectories from a biased policy (move left with probability \( \epsilon \), and \( \pi_b \) otherwise) to obtain batch data.

We then train policies \( \pi_w \) optimizing for reward functions \( r_t = w^T \phi(s) \) on a set of candidate weights \( w \in \mathbb{R}^2 \) on the unit \( \ell_1 \)-norm ball. For each policy, a PDIS estimate of the feature expectations \( \hat{\mu}_w \) is obtained using the collected batch data. The consistency constraint (Equation 5.4) is then evaluated for each candidate weight vector, with different thresholds \( \Delta_c \) (Figure 5.1). Prior to evaluating constraints, we ensure our estimates \( w^T \mu \) for discounted cumulative reward are positive, by augmenting \( w \) and \( \phi(s) \) with a constant positive bias term: \( w' = [w, 1], \phi'(s) = [\phi(s), B] \) where \( B = 14.0 \) for this task. For large \( \Delta_c (\Delta_c \geq 17) \), the set of consistent \( w \) includes approximately half of all test weights: given these thresholds, all \( w \) for which at least one dimension of the state vector was assigned a significant positive weight (greater than 0.5) in the reward function were determined to yield policies sufficiently close to the batch data, while vectors with large negative weights on either coordinate are rejected. When \( \Delta_c \) is reduced to 3.0, only the reward originally optimized for the batch data, \( (w = [0.5, 0.5]) \) is admitted by \( \mathcal{P}_C \).

![Figure 5.1: Consistency and evaluability polytopes with different thresholds \( \Delta_c > 1.0 \) and \( \Delta_e < 1.0 \) respectively, given true reward \( r_t = [0.5, 0.5]^T[x_t, y_t] \). Increasing \( \Delta \) corresponds to relaxing constraints and expanding the satisfying set of weights \( w \).](image-url)
5.2.2 Evaluable Reward Polytope

Our second set of constraints on reward design stem from the need to be able to confidently evaluate a policy in settings when further data collection is expensive or infeasible. We interpret this as a condition on confidence in the estimated policy performance: given an estimate for the expected value $\mathbb{E}[\hat{V}^\pi] = w^T \hat{\mu}$ of a policy $\pi$ and corresponding probabilistic lower bound $V_{lb}^\pi$, we constrain the ratio of these values to lie within some threshold $\Delta_e \geq 0$. A reward function with weights $w$ lies within the polytope of evaluable rewards if $V_{lb}^\pi \geq (1 - \Delta_e) w^T \hat{\mu}$, where $\hat{\mu} \in \mathcal{P}_F$ is our PDIS estimate of feature expectations. To formulate this as a linear constraint in the space of reward weights $w$, the value lower bound $V_{lb}^\pi$ must be rewritten in terms of $w$.

This is done by constructing a combination of upper and lower confidence bounds on the policy feature expectations, denoted $\mu^{lb}$. Starting from the empirical Bernstein concentration inequality (Equation 5.3):

\[
V_{lb} = \frac{1}{N} \sum_{n=1}^{N} \hat{V}_n - \frac{1}{N} \left[ \ln\left(\frac{2}{\delta}\right) \frac{\sum_{n,n'=1}^{N} (\hat{V}_n - \hat{V}_{n'})^2}{c_1} \right] - \frac{7b \ln\left(\frac{2}{\delta}\right)}{3(N-1)}
\]

\[
= \frac{1}{N} \sum_{n=1}^{N} \hat{\mu}^{(n)} - sgn(w) \cdot \frac{1}{N} \sqrt{c_1 \sum_{n,n'=1}^{N} (\hat{\mu}^{(n)} - \hat{\mu}^{(n')})^2} - c_2 \quad (5.6)
\]

\[
= \frac{1}{N} \sum_{n=1}^{N} \hat{\mu}^{(n)} - sgn(w) \cdot \frac{1}{N} \sqrt{c_1 \sum_{n,n'=1}^{N} (\hat{\mu}^{(n)} - \hat{\mu}^{(n')})^2} - c_2 \quad (5.7)
\]

where the $k^{th}$ element of $\hat{\mu}^{lb}$—that is, the value of the $k^{th}$ feature that yields the lower bound in the value of the policy—is dependent on the sign of the corresponding
element of the weights, \( w[k] \):

\[
\hat{\mu}_{lb}[k] = \begin{cases} 
\frac{1}{N} \sum_{n=1}^{N} \hat{\mu}^{(n)} - \frac{c_1}{N} \sum_{n,n'=1}^{N} (\hat{\mu}^{(n)} - \hat{\mu}^{(n')})^2 & w[k] \geq 0 \\
\frac{1}{N} \sum_{n=1}^{N} \hat{\mu}^{(n)} + \frac{c_1}{N} \sum_{n,n'=1}^{N} (\hat{\mu}^{(n)} - \hat{\mu}^{(n')})^2 & w[k] < 0
\end{cases}
\] (5.8)

The definition in Equation 5.8 allows us to incorporate uncertainty in \( \hat{\mu} \) when evaluating our confidence in a given policy: a lower bound for our value estimate requires the lower bound of \( \hat{\mu} \) if the weight is positive, and the upper bound if the weight is negative. Thus, the evaluable reward polytope can be written as:

\[
\mathcal{P}_\mathcal{E} = \{ w : w^T \mu_{lb} \geq (1 - \Delta_e) w^T \mu \} 
\] (5.9)

where \( \mu = \max_{\mu' \in \mathcal{P}_\mathcal{F}} w^T \mu' \) is the expectation of state features for the optimal policy obtained from solving the MDP with reward weights \( w \), and \( \mu_{lb} \) is the corresponding lower bound. The constant \( c_2 \) in the performance lower bound (Equation 5.7) is absorbed by threshold parameter \( \Delta_e \) on the tightness of the lower bound.

**Proposition 2.** The set of weights \( \mathcal{P}_\mathcal{E} \) defines a closed convex set, given access to an exact MDP solver.

**Proof.** The set \( \mathcal{P}_\mathcal{E} \) contains all weights \( w \) that satisfy constraints linear in \( w \):

\[
w^T (\mu_{lb} - (1 - \Delta_e) \mu) \leq 0.
\]

As in the case of \( \mathcal{P}_\mathcal{C} \), it follows from [13] that the set described by these constraints is convex. \( \square \)

**Illustration.** In order to visualize an example polytope for evaluable rewards (Equation 5.9), we return to the two-dimensional map described in Section 5.2.1. As before, we begin with a batch of trajectories collected by a biased \( \epsilon \)-greedy expert policy trained on the true reward. We use these trajectories to obtain PDIS estimates
Algorithm 3 Separation oracle $SO_{adm}$ for admissible $w$

**Input:**
Proposed weights $w \in \mathbb{R}^k$, Behaviour policy $\mu_b$, Threshold parameters $\Delta_c, \Delta_e$

1. Solve MDP with weights $w$ for optimal policy features $\mu = \text{argmax}_\mu w^T \mu'$
2. Evaluate lower bound $\mu^{lb}$ for estimated policy features

```plaintext
if $w^T (\mu - \Delta_c \mu_b) > 0$ then
    $w \notin \mathcal{P}_C \Rightarrow$ REJECT $w$
    OUTPUT: Halfspace \{ $w^T (\mu - \Delta_c \mu_b) \leq 0$ \}
else if $w^T (\mu_b - \Delta_c \mu) > 0$ then
    $w \notin \mathcal{P}_C \Rightarrow$ REJECT $w$
    OUTPUT: Halfspace \{ $w^T (\mu_b - \Delta_c \mu) \leq 0$ \}
else if $w^T ((1 - \Delta_e) \mu - \mu_b) > 0$ then
    $w \notin \mathcal{P}_E \Rightarrow$ REJECT $w$
    OUTPUT: Halfspace \{ $w^T ((1 - \Delta_e) \mu - \mu_b) \leq 0$ \}
else
    $w \in \mathcal{P}_C \cap \mathcal{P}_E = \mathcal{P}_{adm} \Rightarrow$ ACCEPT $w$
```

$\hat{\mu}$ for policies trained with a range of reward weights $w$ on the $\ell_1$-norm ball. We then evaluate $\hat{\mu}^{lb}$, and in turn the hyperplanes defining the intersecting half-spaces of the evaluable reward polytope, for each $w$. Plotting the set of evaluable reward vectors for different thresholds $\Delta_e$, we see substantial overlap with the consistent reward polytope in this environment, though neither polytope is a subset of the other (Figure 5.1b). We also find that in this setting, the value of the evaluability constraint is asymmetric about the true reward—more so than the consistency metric—such that policies trained on penalizing $x_i (w[0] < 0)$, hence favouring movement left, can be evaluated to obtain a tighter lower bound than weights that learn policies with movement down, which is rarely seen in the biased demonstration data (Figure 5.1d).

Finally, tightening the threshold further to $\Delta_e = 0.1$ (Figure 5.1d) the set of accepted weights is again just the true reward, as for the consistency polytope.
5.2.3 Querying Admissible Reward Polytope

Given our criteria for consistency and evaulability of reward functions, we need a way to access the sets satisfying these constraints. These sets cannot be explicitly described as there are infinite policies with corresponding representations \( \mu \), and so infinite possible constraints; instead, we construct a separation oracle to access points in this set in polynomial time (Algorithm 3). A separation oracle tests whether a given point \( w' \) lies in polytope of interest \( P \), and if not, outputs a separating hyperplane defining some half-space \( w^T A \leq b \), such that \( P \) lies inside this half-space and \( w' \) lies outside of it. The separation oracle for the polytope of admissible rewards evaluates both consistency and evaulability to determine whether \( w' \) lies in the intersection of the two polytopes, which we define as our admissible polytope \( P_{\text{adm}} \). If a constraint is not met, it outputs a new hyperplane accordingly.

It should be noted that the RL problems of interest to us are typically large MDPs with continuous state spaces, as in the clinical setting of managing mechanical ventilation in the ICU, and moreover, because we are optimizing policies given only batch data, we know we can only expect to find approximately optimal policies. The use of PDIS estimates \( \hat{\mu} \) of the true feature expectations in the batch setting introduces an additional source of approximation error. It has been shown that Algorithm 3 with an approximate MDP solver produces a weird separation oracle \( [43] \), one that does not necessarily define a convex set. However, it does still accept all points in the queried polytope, and can thus still be used to test whether a proposed weight vector \( w \) lies within this set.

Returning to our 2D map (Figure 5.1), the admissible reward polytope \( P_{\text{adm}} \) is the set of weights accepted by both the consistent and evaulable polytopes. The choice of thresholds \( \Delta_c \) and \( \Delta_e \) respectively is important in obtaining a meaningfully restricted, non-empty set to choose rewards from. These thresholds will depend on the extent of exploratory or sub-optimal behaviour in the batch data, and the level
of uncertainty acceptable when deploying a new policy. We find that in this toy 2D map setting, there is considerable overlap between the two polytopes defining the admissible set, though this is not always the case; from our earlier intuition, as the behaviour policy from which trajectories were generated is the same for all trajectories, there is limited “exploration”, or deviation from average behaviour across trajectories, and the therefore the evaluability constraints admit reward weights that largely overlap with those consistent with average behaviour.

5.2.4 Finding the Nearest Admissible Reward

With a separation oracle $SO_{\text{adm}}$ for querying whether a given $w$ lies in the admissible reward polytope, we optimize linear functions over this set using, e.g., the ellipsoid method for exact solutions or—as considered here—the iterative follow-perturbed-leader (FPL) algorithm for computationally efficient approximate solutions [52]. To achieve our goal of aiding reward specification for a designer with existing but imperfectly known goals, we pose our optimization problem as follows (Algorithm 4): given initial reward weights $w_0$ proposed by the agent designer, we first test whether $w_0$, with some small perturbation, lies in the admissible polytope $P_{\text{adm}}$, which we define by training a policy $\pi_0$ approximately optimizing this reward. If it does not lie in $P_{\text{adm}}$, we return new weights $w \in P_{\text{adm}}$ that minimize distance $\|w - w_{\text{init}}\|_2$ from the proposed weights. This solution is then perturbed and tested in turn. We note that constraints posed based on the behaviour $\mu_b$ observed in the available batch trajectories are encapsulated by this minimization over weights in set $P_{\text{adm}}$, that is, solving a constrained linear optimization defined by the linear constraints on $w$ from Equations 5.4 and 5.9. The constraints at each iteration do not fully specify $P_{\text{adm}}$, but instead give us a half-space to optimize over, at each step.

The constrained linear program solved at each iteration scales in constant time with the dimensionality of $w$; although we only present results with functions $\phi(s)$
Algorithm 4 Follow-perturbed-leader FOR ADMISSIBLE w.

INPUT: Initial weights $w_0 \in \mathbb{R}^k$, # iterations $T$, perturbation $\delta = \frac{1}{k\sqrt{T}}$

$t = 0$

while $t \leq T$ do

1. Let $r_t = \sum_{i=1}^{t-1}(w_i + p_t) \cdot \phi(\cdot)$, where $p_t \sim \mathcal{U}[0, \frac{1}{\delta}]^k$
2. Solve for $\pi_t = \arg\max_{\pi} V_{\pi}^r |r_t$
3. Let $\mu_t = \mu(\pi_t) + q_t$, where $q_t \sim \mathcal{U}[0, \frac{1}{\delta}]^k$
4. Evaluate constraints defining $\mathcal{P}_{adm}$
5. Solve for $w_t := \arg\min_{w \in \mathcal{P}_{adm}} \|w - w_{init}\|_2$
6. $t := t + 1$

end

OUTPUT: $\bar{\pi}_{final} = \frac{1}{T} \sum_{t=1}^{T} \pi_t; \bar{w} = \frac{1}{T} \sum_{t=1}^{T} w_t$

of dimensionality at most 3, for the sake of visualization, the iterative algorithm presented can be scaled to higher dimensional $\phi(s)$, as the complexity of the linear program solved at each iteration is dependent only on the number of constraints. Our final reward weights and a randomized policy are the average across the approximate solutions in each iteration. This policy optimizes a reward that is the closest admissible reward to the original goals of the designer of the RL agent.

5.3 Experiment Design

5.3.1 Benchmark Tasks

We illustrate our approach to determining admissible reward functions on three benchmark domains with well-defined objectives: classical control tasks Mountain Car and Acrobot, and a simulation-based treatment task for HIV patients. The control tasks, implemented using OpenAI Gym [8], both have a continuous state space and discrete action space, and the objective is to reach a terminal goal state. To explore how the constrained polytopes inform reward design for these tasks, an expert behaviour
policy is first trained with data collected from an exploratory policy receiving a reward of \(-1\) at each time step, and \(0\) once the goal state is reached. A batch of 1000 trajectories is collected by following this expert policy with Boltzmann exploration, mimicking a sub-optimal expert. Given these trajectories, our task is to choose a reward function that allows us to efficiently learn an optimal policy that is i) consistent with the expert behaviour in the trajectories, and ii) evaluable with acceptably tight lower bounds on performance. We limit the reward function \(r_t = w^T \phi(s_t)\) in each task to a weighted sum of three features, \(\phi(s) \in \mathbb{R}^3\), chosen to include sufficient information to learn a meaningful policy while allowing for visualization. For Mountain Car, we use quantile-transformed position, velocity, and an indicator \(\pm 1\) of whether the goal state has been reached. For Acrobot, \(\phi(s)\) comprises the quantile-transformed cosine of the angle of the first link, angular velocity of the link, and an indicator \(\pm 1\) of whether the goal link height is satisfied. We sweep over weight vectors on the 3D \(\ell_1\)-norm ball, training policies with the corresponding rewards, and filtering for admissible \(w\).

The characterization of a good policy is more complex in our third benchmark task, namely treatment recommendation for HIV patients, modeled by a linear dynamical system [22]. Again, we have a continuous state space and four discrete actions to choose from: no treatment, one of two possible drugs, or both in conjunction. The true reward in this domain is given by: 
\[
R = -0.1V + 10^3 E - 2 \cdot 10^4 (0.7d_1)^2 - 2 \cdot 10^3 (0.3d_2)^2,
\]
where \(V\) is the viral count, \(E\) is the count of white blood cells (WBC) targeting the virus, and \(d_1\) and \(d_2\) are indicators for drugs 1 and 2 respectively. We can rewrite this function as \(r = w^T \phi(s)\), where \(\phi(s) = [V, c_0E, c_1d_1 + c_2d_2] \in \mathbb{R}^3\), with constants \(c_0, c_1\) and \(c_2\) set such that weights \(w = [-0.1, 0.5, 0.4]\) reproduce the original function. Again, the low dimensionality of \(\phi(s)\) is simply for the sake of interpretability. An expert policy is trained using this true reward, and a set of sub-optimal trajectories
Table 5.1: MDP state features taken as input for learning an optimal policy for management of mechanical ventilation in ICU.

<table>
<thead>
<tr>
<th>State Features</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Demographics</strong></td>
</tr>
<tr>
<td>Age, Gender, Ethnicity, Admission Weight, First ICU Unit</td>
</tr>
<tr>
<td><strong>Vent Settings</strong></td>
</tr>
<tr>
<td>Ventilator mode, Inspired $O_2$ fraction ($FiO_2$), $O_2$ Flow Positive End-Expiratory Pressure (PEEP) set</td>
</tr>
<tr>
<td><strong>Measured Vitals</strong></td>
</tr>
<tr>
<td>Heart Rate, Respiratory Rate, Arterial pH, $O_2$ saturation pulse oxymetry ($SpO_2$), Richmond-RAS Scale, Non Invasive Blood Pressure (systolic, diastolic, mean), Mean Airway Pressure, Tidal Volume, Peak Insp. Pressure, Plateau Pressure, Arterial $CO_2$ Pressure, Arterial $O_2$ pressure</td>
</tr>
<tr>
<td><strong>Input Sedation</strong></td>
</tr>
<tr>
<td>Propofol, Fentanyl, Midazolam, Dexmedetomidine, Morphine Sulfate, Hydromorphone, Lorazepam</td>
</tr>
<tr>
<td><strong>Other</strong></td>
</tr>
<tr>
<td>Consecutive duration into ventilation ($D$), Number of reintubations ($N$)</td>
</tr>
</tbody>
</table>

are collected by following this policy with Boltzmann exploration. Policies are then trained over weights $w, ||w||_1 = 1$ to determine the set of admissible rewards.

5.3.2 Mechanical Ventilation in ICU

We use our methods to aid reward design for the task of managing invasive mechanical ventilation in critically ill patients, as described in Chapter 3. Mechanical ventilation refers to the use of external breathing support to replace spontaneous breathing in patients with compromised lung function. It is one of the most common, as well as most costly, interventions in the ICU [108]. Timely weaning, or removal of breathing support, is crucial to minimizing risks of ventilator-associated infection or over-sedation, while avoiding failed breathing tests or reintubation due to premature weaning. Expert opinion varies on how best to trade off these risks, and clinicians
tend to err towards conservative estimates of patient wean readiness, resulting in extended ICU stays and inflated costs.

We look to design a reward function for a weaning policy that penalizes prolonged ventilation, while weighing the relative risks of premature weaning such that the optimal policy does not recommend strategies starkly different from clinician behaviour, and the policies can be evaluated for acceptably robust bounds on performance using existing trajectories. We train and test our policies on data filtered from the MIMIC III data \[49\] with 6,883 ICU admissions from successfully discharged patients following mechanical ventilation, preprocessed and resampled in hourly intervals. The MDP for this task is adapted from that introduced in Section 3.2.3: the patient state \(s_t\) at time \(t\) is a 32-dimensional vector that includes demographic data, ventilator settings, and relevant vitals (Table 5.1). We learn a policy with binary action space \(a_t \in [0, 1]\), for keeping the patient off or on the ventilator, respectively. The reward function \(r_t = w^T \phi(s_t, a_t)\) with \(\phi(s, a) \in \mathbb{R}^3\) includes (i) a penalty for more than 48 hours on the ventilator, (ii) a penalty for reintubation due to unsuccessful weaning, and (iii) a penalty on physiological instability when the patient is off the ventilator based on abnormal vitals:

\[
\phi = \left[ -\min(0, \tanh 0.1(D_t - 48)) \cdot 1[a_t = 1] \right.
\]

\[
-1[\exists t' > t \text{ such that } N_{t'} > N_t] \cdot 1[a_t = 0]
\]

\[
-\frac{1}{|V|} \sum_{v} \left( v < v_{\min} \mid v < v_{\max} \right) \cdot 1[a_t = 0]
\]

where \(D_t\) is duration into ventilation at time \(t\) in an admission, \(N_t\) is the number of reintubations, \(v \in V\) are physiological parameters each with normal range \([v_{\min}, v_{\max}]\), and \(V = \{\text{Ventilator settings, Measured vitals}\}\). The three terms in \(\phi(\cdot)\) represent penalties on duration of ventilation, reintubation, and abnormal vitals, respectively. Our goal is to learn the relative weights of these feedback signals to produce a consis-
Table 5.2: Analysing top three admitted weights $w$ for each of the three benchmark control enivronments. Admissibility polytope thresholds are set by choosing a small $\Delta_c$ and required corresponding threshold $\Delta_e$ for an admissible set of size $|P_{adm}| = 3$.

<table>
<thead>
<tr>
<th>Task</th>
<th>Top 3 Admissible weights</th>
<th>$\Delta_c (P_c)$</th>
<th>$\Delta_e (P_\epsilon)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mountain Car</td>
<td>$[0.0, 0.2, 0.8]^T, [0.2, 0.2, 0.6]^T, [0.4, -0.4, 0.2]^T$</td>
<td>1.10</td>
<td>0.27</td>
</tr>
<tr>
<td>Acrobot</td>
<td>$[-0.2, 0.0, 0.8]^T, [-0.8, -0.2, 0.0]^T, [-0.2, -0.2, 0.6]^T$</td>
<td>1.10</td>
<td>0.29</td>
</tr>
<tr>
<td>HIV Simulator</td>
<td>$[0.0, 0.4, 0.6]^T, [-0.6, 0.2, 0.2]^T, [-0.2, 0.4, -0.4]^T$</td>
<td>1.20</td>
<td>0.28</td>
</tr>
</tbody>
</table>

tent, evaluable reward function and learn a policy optimizing this reward. As before, we train our optimal policies using Fitted Q-iteration (FQI) with function approximation using extremely randomized trees [21]. We partition our dataset into 3,000 training episodes and 3,883 test episodes, and run FQI over 100 iterations on the training set, with discount factor $\gamma = 0.9$. We then use the learnt Q-function to train our binary treatment policy.

5.4 Results and Discussion

5.4.1 Benchmark Control Tasks

Admissible $w$ are clustered near true rewards.

We analyze reward functions from the sweep over weight vectors on the $\ell_1$-norm unit ball for each benchmark task (Section 5.3.1) by first visualizing how the space of weights accepted by the consistency and evaluability polytopes—and therefore the space $P_{adm}$ at the intersection of these polytopes—changes with the values of thresholds $\Delta_c$ and $\Delta_e$. Alongside this, we plot the set of admitted weights produced by
Figure 5.2: Admissible polytope size for varying thresholds on consistency ($\Delta_c$) and evaluability ($\Delta_e$), and distribution of admitted weights for fixed $\Delta_c, \Delta_e$, in: (a) Mountain Car (b) Acrobot (c) HIV Simulator. Note that admitted rewards for each task typically correspond to positive weights on the goal state.
arbitrarily chosen thresholds (Figure 5.2). In all three tasks, we find that the admitted weights form distinct clusters; these are typically at positive weights on goal states in the classic control tasks, and at positive weights on WBC count for the HIV simulator, in keeping with the rewards optimized by the batch data in each case. We could therefore use this naive sweep over weights to choose a vector within the admitted cluster that is closest to our initial proposed function, or to our overall objective. For instance, if in the HIV task we want a policy that prioritizes minimization of side effects from administered drugs, we can choose specifically from admissible rewards with negative weight on the treatment term.

**Analysis of admissible w can lend insight into reward shaping for faster policy learning.**

We may wish to shortlist candidate weights by setting more stringent thresholds for admissibility. We mimic this design process as follows: prioritizing evaluability in each of our benchmark environments, we choose the smallest possible $\Delta_e$ and large $\Delta_c$ for an admissible set of exactly three weights (Table 5.2). This reflects a typical batch setting, in which we want high-confidence performance guarantees; we also want to allow our policy to deviate when necessary from the available sub-optimal expert trajectories. For Mountain Car, our results show that two of the three vectors assign large positive weights to reaching the goal state; all assign zero or positive weight to the position of the car. The third, $w = [0.4, -0.4, 0.2]$ is dominated by a significant positive weight on position and a significant negative weight on velocity; this may be interpreted as a kind of reward shaping: the agent is encouraged to first move in reverse to achieve a negative velocity, as is necessary to reach the goal state in the under-powered mountain car problem. The top three $w$ for Acrobot also place either positive weights on the goal state, or negative weights on the position of the first link.
Again, the latter reward definition likely plays a shaping role in policy optimization by rewarding link displacement.

**FPL can be used to correct biased reward specification in the direction of true reward.**

We use the HIV treatment task to explore how iterative solutions for admissible reward (Algorithm 4) can improve a partial or flawed reward specified by a designer. For instance, a simple first attempt by the designer at a reward function may place equal weights on each component of $\phi(s)$, with the polarity of weights—whether each component should elicit positive feedback or incur a penalty—decided by the designer’s domain knowledge; here, the designer may suggest $w_0 = \frac{1}{3}[-1, 1, -1]^T$.

We run Algorithm 4 for twenty iterations with this initial vector and thresholds $\Delta_c = 2.0, \Delta_e = 0.8$ and average over the weights from each iteration. This yields weights $\bar{w} = [-0.11, 0.57, -0.32]^T$, redistributed to be closer to the reward function being optimized in the batch data. This pattern is observed with more extreme initial rewards functions too; if e.g., the reward proposed depends solely on WBC count, $w_0 = [0, 1, 0]$, then we obtain weights $\bar{w} = [-0.14, 0.83, -0.04]$ after twenty iterations of this algorithm such that appropriate penalties are introduced on viral load and administered drugs.

### 5.4.2 Mechanical Ventilation in ICU

**Admissible $w$ may highlight bias in expert behaviour.**

We apply our methods to choose a reward function for a ventilator weaning policy in the ICU, given that we have access only to historical ICU trajectories with which to train and validate our policies. When visualizing the admissible set, with $\Delta_c = 1.8, \Delta_e = 0.4$, we find substantial intersection in the consistent and evaluable polytopes (Figure 5.3). Admitted weights are clustered at large negative weights on
the duration penalty term favouring policies that are conservative in weaning patients (that is, those that keep patients longer on the ventilator), which is the direction of bias we expect in the past clinical behaviour. We can tether a naive reward that instead penalizes duration on the ventilator, $w = [1, 0, 0]$ to the space of rewards that are consistent with this conservative behaviour as follows: using FPL to search for a reward within the admissible set given this initial vector yields $\hat{w} = [0.72, 0.14, 0.14]$, introducing non-zero penalties on reintubation and physiological instability when off ventilation. This allows us to learn behaviour that is averse to premature extubation (consistent with historical clinical behaviour) without simply rewarding long durations on the ventilator.

FPL improves effective sample size for learnt policies.

To verify whether weights from the admissible polytope enable higher confidence policy evaluation, we explore a simple proxy for variance of an importance sampling-
Table 5.3: Mechanical ventilation in the ICU: Influence of FPL algorithm on Kish effective sample size of learnt policies.

<table>
<thead>
<tr>
<th>Initial $w$</th>
<th>$N_{eff}$</th>
<th>Final $w$</th>
<th>$N_{eff}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1., 0., 0.]</td>
<td>8</td>
<td>[0.72, 0.14, 0.14]</td>
<td>14</td>
</tr>
<tr>
<td>[0., 1., 0.]</td>
<td>304</td>
<td>[-0.07, 0.77, 0.16]</td>
<td>352</td>
</tr>
<tr>
<td>[0., 0., 1.]</td>
<td>32</td>
<td>[0.15, -0.21, 0.66]</td>
<td>37</td>
</tr>
<tr>
<td>$\frac{1}{3}$[1., 1., 1.]</td>
<td>16</td>
<td>[0.24, 0.51, 0.25]</td>
<td>33</td>
</tr>
</tbody>
</table>

Based estimate of performance: the effective sample size $N_{eff} = (\sum_{n}^{N} \rho_{n})^2 / \sum_{n}^{N} \rho_{n}^2$ of the batch data, where $\rho_{n}$ is importance weight of trajectory $n$ for a given policy.

In order to evaluate the Kish effective sample size $N_{eff}$ for a given policy, we subsample admissions in our test data to obtain trajectories of approximately 20 timesteps in length, and calculate importance weights $\rho_{n}$ for the policy considered using these subsampled trajectories. Testing a number of naive initializations of $w$, we find that effective sample size is consistently higher for weights following FPL (Table 5.3). This indicates that the final weights induce an optimal policy that is better represented in the batch data than the policy from the original weights.

### 5.5 Conclusion

In this work, we present a method for reward design in reinforcement learning using batch data collected from sub-optimal experts. We do this by constraining rewards to those yielding policies within some distance of the policies of domain experts; the policies inferred from the admissible rewards also provide reasonable bounds on performance. Our experiments show how rewards can be chosen in practice from the space of functions satisfying these constraints, and illustrate this on the problem of weaning clinical patients from mechanical ventilation.
Effective reward design for RL in safety-critical settings is necessarily an iterative process of deployment and evaluation, to push the space of observed behaviour incrementally towards policies consistent and evaluable with respect to our ideal reward. There are a number of ways in which the methods here could be extended however, to better use the information available in existing data on what constitutes a safe policy, and in turn what reward function can ensure this. For instance, different care providers in clinical settings likely follow policies with different levels of precision, or perhaps even optimize for different reward functions; modeling this heterogeneity in behaviour and weighting experts appropriately can enable learnt behaviour closer to the best, rather than the average, expert. In addition, going beyond the use of summary statistics provided by policy feature expectations to explore more complex representations of behaviour that are still decoupled from rewards, and in turn better metrics for similarity in behaviour, can aid in more meaningful choices in reward.
Chapter 6

Guiding Electrolyte Repletion in Critical Care using RL

The replacements of electrolyte levels is an ubiquitous part of healthcare delivery in hospitalized and critically ill patients. Electrolytes are charged minerals found in the blood, such as potassium, sodium, magnesium, calcium or phosphate, that are essential in supporting the normal function of cells and tissues. They play a key role in electrical conduction in the heart, muscle and nervous system, and in intracellular signalling; it follows that electrolyte insufficiency is associated with higher morbidity and mortality rates in critical care.

Disturbances in electrolyte levels can arise from a range in underlying causes, including reduced kidney or liver function, endocrine disorders, or concurrently administered drugs such as diuretics. Although the standardized institutional protocols are typically in place to guide electrolyte replacement, adherence to published guidelines is poor, and the repletion process is instead largely driven by individual care providers. There is evidence that experiential bias from this provider-directed approach is prone to significant errors, both in terms of more missed episodes of low electrolyte levels and—increasingly—high rates of superfluous replacements, con-
tributing to unnecessary expenditure by way of prescription of medications, ordering of laboratory tests, as well as clinician and nursing time spent [50, 123].

There have been several studies in recent literature that highlight the prevalence of ineffectual electrolyte repletion therapy. Considering the regulation of potassium in particular, as many as 20% of hospitalized patients experience episodes of hypokalaemia, where blood serum levels of potassium are below the reference normal range. The majority of patients receiving (predominantly non-potassium sparing) diuretics go to become hypokalaemic [121]. However, only in 4-5% of patients has this been found to be clinically significant [2]. In investigating rule-of-thumb potassium repletions, Hammond et al. [38] found that just over a third of repletions achieved potassium levels within reference range. Lancaster et al. [61] demonstrate that potassium supplementation is not effective as a preventative measure against atrial fibrillation, while magnesium supplementation can in fact increase risk.

In this chapter, we aim to develop a clinician-in-loop decision support tool for electrolyte repletion, focusing on the management of potassium, magnesium and phosphate levels in hospitalized patients. While there have been few efforts to take a personalized, data-driven approach to electrolyte repletion, machine learning methods have been applied to the closely related problem of fluid resuscitation, in order to manage hypotension in critically ill patients. For example, Celi et al. [10] consider a Bayesian network to predict need for fluid replacement based on historical data, while Komorowski et al. [57] describe a reinforcement learning approach to the administration of fluids and vasopressors in patients with sepsis, using Q-learning with discretized state and action spaces to learn a policy minimizing the risk of patient mortality. Here, we translate the reinforcement learning framework introduced in Chapter 3—using batch reinforcement learning methods with continuous patient state representations—to learning policies for targeted electrolyte repletion. We seek to understand the clinical priorities that shape current provider behaviour.
through methods based on inverse reinforcement learning, and adapt these priorities to learn policies that minimize the costs associated with repletion while maintaining electrolyte levels within their reference ranges.

6.1 Methods

6.1.1 UPHS Dataset

The data used in this work is drawn from a set of over 450,000 acute care admissions between 2010 and 2015, across three centres within the University of Pennsylvania health system (UPHS). For each admission, we have de-identified electronic health records comprising demographics, details of the hospital and unit the patients are admitted to, identification numbers of their care providers, nurse-verified physiological parameters, administered medications and procedures, and patient outcomes. From this rich dataset, we select for all adult patients for which we have high-level information on the admission (including age, gender and admission weight), a minimum of one lab test result for each of potassium, magnesium and phosphate levels (often available jointly as part of a basic electrolyte panel) as well as recorded measurements for other key vitals and lab tests, including all commonly tested electrolytes.

This yields a cohort of 13,164 unique patient visits, of which 7,870 are administered potassium at least once, 8,342 are administered magnesium, and 1,768 are administered phosphates. Figure 6.1 plots the distribution of measured serum electrolyte levels both prior to and post repletion events, along with the target (normal) range in each case. We can see that while the majority of phosphate repletions occur when measurements fall below the reference range, this is not true for potassium or magnesium; in the case of potassium, 4% of all repletions are ordered while the last known measurement is above the target range, which appears to lend support to claims in the literature regarding unnecessary potassium supplementation.
Each patient hospital visit is our chosen cohort is resampled into 6-hour intervals. This relatively large window is chosen as it reflects the minimum frequency with which lab tests for electrolyte levels are generally ordered, and in turn the duration between reassessment of the need for electrolyte supplements; in standard practice, electrolyte repletion is typically reviewed three times a day. In sampling patient vitals and lab tests, outliers (recorded measurements that are not clinical viable) are filtered out, and the mean of remaining measures is taken as representative of the value at each 6 hour interval. Missing values are imputed using simple feed-forward imputation, up to a maximum of 48 hours since the last known measurement, and otherwise imputed with the value of the population mean.

6.1.2 Formulating the MDP

As in previous chapters, we frame the clinical decision-making problem here as a Markov decision process (MDP), \( \mathcal{M} = \{S, A, P, P_0, R, \gamma\} \) parametrized by some fi-
Figure 6.2: Example hospital admission with potassium supplementation

Finite state space $\mathcal{S}$, $s_t \in \mathcal{S}$, finite set of actions $\mathcal{A}$, $a_t \in \mathcal{A}$, an unknown transition function $P(s_{t+1}|s_t, a_t)$, distribution $P_0$ over initial states $s_0 \in \mathcal{S}$, a reward function $R(s_t, a_t, s_{t+1})$, and a scalar discount factor $\gamma$ defining the relative importance of immediate and long-term rewards. Our objective is learn an optimal policy function $\pi^*(s) : \mathcal{S} \rightarrow \mathcal{A}$ that maximizes the discounted cumulative reward:

$$\pi^*(s_t) = \arg\max_{a_t \in \mathcal{A}} \mathbb{E}_{P,P_0} \left[ \sum_t \gamma^t R(s_t, a_t, s_{t+1}) \right]$$  \hspace{1cm} (6.1)

**State representation** The relative risk posed by electrolyte levels outside the reference range, and the initiation of strict regulation of potassium, magnesium and phosphate levels, is influenced by a number of different factors. These include demographic characteristics, patient physiological stability, and interaction with concurrently administered drugs. For example, Figure 6.2 illustrates repletion events for a single hospital visit, along with available measurements of serum potassium level, and administration events of non-potassium sparing diuretics. We can see that oral potassium repletion is routinely ordered as a prophylaxis—that is, as a preventive measure against hypokalaemia—in conjunction with diuretics, even when potassium levels are within the target range.

In defining our state space, we therefore include static features defining patient admissions such as age, weight, gender and whether the admission is to the ICU or to a regular inpatient ward on the hospital floor (as an proxy for patient severity of illness). We also incorporate imputed measurements at each 6-hour interval for
Table 6.1: Selected 52 features for patient state representation in electrolyte repletion.

<table>
<thead>
<tr>
<th>Features</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>STATIC</strong></td>
<td>Age, Gender, Weight, Floor/ICU</td>
</tr>
<tr>
<td><strong>VITALS</strong></td>
<td>Heart rate, Respiratory rate, Temperature, O\textsubscript{2} saturation pulseoxymetry (\textit{SpO\textsubscript{2}}), Urine output Non-invasive blood pressure (systolic, diastolic)</td>
</tr>
<tr>
<td><strong>LABS</strong></td>
<td>K, Mg, P, Na, Ca (Ionized), Chloride, Anion gap, Creatinine, Hemoglobin, Glucose, BUN, WBC, CPK, LDH, ALT, AST, PTH</td>
</tr>
<tr>
<td><strong>DRUGS</strong></td>
<td>K-IV, K-PO, Mg-IV, Mg-PO, P-IV, P-PO, Ca-IV, Ca-PO, Loop diuretics, Thiazides, Acetazolamide, Spironolactone, Fluids, Vasopressors, Beta Blockers, Ca Blockers, Dextrose, Insulin, Kayazlate, TPN, PN, PO Nutrition</td>
</tr>
<tr>
<td><strong>PROCEDURES</strong></td>
<td>Packed cell transfusion, Dialysis</td>
</tr>
</tbody>
</table>

seven common vitals (including urine output) and eleven labs. In addition, seven rarer labs are represented with an indicator of whether a measurement is available from the past 24 hours, as the ordering of these lab tests by clinicians can in itself be informative. In terms of medication, the patient state includes the administered dose of both intravenous (IV) and oral (PO) potassium, magnesium or phosphate. Several additional key classes of drugs are represented as indicator variables, taking a value of 1 if the drug is administered over the 6-hour interval, and 0 if not. Fluids, diuretics, parenteral nutrition, etc. fall within this category. Finally, we include indicators of whether patient is administered packed cell transfusions (as these can increase risk of hyperkalaemia) or dialysis, which aims to correct electrolyte imbalances resulting from kidney failure. This yields a 52-dimensional state space (Table 6.1) encompassing all available information relevant in learning an optimal repletion policy.
Table 6.2: Discretized dosage levels for K, Mg and P.

<table>
<thead>
<tr>
<th></th>
<th>ORAL (PO)</th>
<th>INTRAVENOUS (IV)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PO1 PO2 PO3</td>
<td>IV1 IV2 IV3 IV4 IV5 IV6</td>
</tr>
<tr>
<td>K</td>
<td>0 20 40 60</td>
<td>20mEq 40mEq 60mEq 20mEq 40mEq 60mEq</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2h 4h 6h 1h 2h 3h</td>
</tr>
<tr>
<td>Mg</td>
<td>0 400 800 1200</td>
<td>0.5mEq 1mEq 1mEq 1mEq</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1h 1h 2h 3h</td>
</tr>
<tr>
<td>P</td>
<td>0 250 500 750</td>
<td>1mEq 2mEq 2mEq</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1h 3h 6h</td>
</tr>
</tbody>
</table>

**Action representation** For each of the three electrolytes considered here, supplements are administered either via fast-acting intravenous drugs (at various rates and infusion times) or with tablets at different doses. In designing our action space, we discretize the dosage rates such that the set of actions we choose from are in line with most common practice in the UPHS data. In the case of potassium, this yields the following options: no repletion, PO repletion at one of three discretized levels: 0-20, 20-40 or 40-60 mEq, IV repletion at one of six possible rates: 0-10 mEq/hr infused over 1, 2 or 3 hours; 10-20 mEq/hr over 2, 4 or 6 hours, or some combination of both intravenous and oral supplements.

In order to effectively learn treatment recommendations over this space of actions, we choose to learn three independent policies for each electrolyte, each with a distinct action space. Specifically, we first learn optimal policy recommendations for the route of electrolyte administration, $\pi^{route}: S \rightarrow A^{route}$ where

$$A^{route} = \left\{ \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \right\}$$

(6.2)
such that \( a_t^{route}[0] = 1 \) indicates an IV repletion event at time \( t \), and \( a_t^{route}[1] = 1 \) indicates administration of PO repletion. We then learn separate policies for PO and IV dosage that map patient state to action spaces \( \mathcal{A}_{PO} \) and \( \mathcal{A}_{IV} \) respectively, where the actions in each set are represented by one-hot encodings of each dosage level in that category. The size of the action space in each case is therefore given by the total number of possible dosage rates, plus an additional action representing no repletion.

For potassium, this yields action spaces of size \( |\mathcal{A}_{IV}| = 7 \) and \( |\mathcal{A}_{PO}| = 4 \) respectively. The set of action spaces for magnesium and phosphate repletion are defined in the same way; the complete list of discretized dosage levels is summarized in Table 6.2.

This subdivision of action spaces mimics the likely decision-making process in clinical practice: the provider must first decide whether to administer a supplement, and if so, by what route, before choosing the most appropriate dosage of this supplement.

**Reward function**  The objective of our electrolyte repletion policy is to optimize patient clinical outcome while minimizing unnecessary repletion events. To this end, we look to incorporate the following elements in our reward function: (i) the effective cost of an IV repletion, which can be thought of as encompassing the prescription cost, care-provider time, and the cost of the drug itself, (ii) the effective cost of PO repletion, given these same considerations, (iii) a penalty for electrolyte levels above the reference range, and (iv) a penalty for electrolyte levels below this range. The reward function for potassium can then be written as:

\[ r_{t+1} = w \cdot \phi_t(s_t, a_t, s_{t+1}), \]

where \( \phi \) is a four-dimensional vector function such that:

\[
\phi_t(\cdot) = \left[ \begin{array}{c} -1_{a_t^{route}[0]} \\ -1_{a_t^{route}[1]} \\ -1_{s_{t+1}[K]>K_{max}} \cdot 10 \left[ 1 + \exp^{-\sigma(K-K_{max})} \right]^{-1} \\ -1_{s_{t+1}[K]<K_{min}} \cdot 10 \left[ 1 - \left( 1 + \exp^{-\sigma(K-K_{min})} \right)^{-1} \right] \end{array} \right] \in \{0, -1\} \begin{bmatrix} 0, -1 \\ (-10, 0) \end{bmatrix} \]  

(6.3)
Figure 6.3: Penalizing abnormal potassium levels in reward function: $\sigma = 3.5$

and $w$, $||w||_1 = 1$ determines the relative weighting of each penalty. In the above equation, K is the last known measurement of potassium; $K_{\text{max}}$ and $K_{\text{min}}$ define the upper and lower bounds respectively of the target potassium value. Penalties for values above and below the reference range are applied independently in order to allow for asymmetric weighting of the risks posed by hypokalaemia when compared with hyperkalaemia. The sigmoid function used to model penalties on abnormal vitals (Figure 6.3) can be justified as follows: the maximum and minimum thresholds for electrolyte reference ranges are fairly arbitrary, and can vary considerably across hospitals. While patients with abnormal values near these thresholds are likely to be asymptomatic or experience few adverse effects, more severe electrolyte imbalance becomes increasingly harmful to patient outcome, until irrevocable. The parameter $\sigma$ in the definition of this function effectively determines the sharpness of this threshold, and can be set according to the width of the reference range and our confidence in the threshold value. The maximum value of the sigmoid penalties in $\phi$ are scaled the lie between 0 and -10, in order that the mean non-zero penalties of all four terms lie within approximately the same order of magnitude, to aid subsequent analysis and the choice of weights $w$ for the final reward function.

The vector function $\phi$ for both magnesium and phosphate are defined in much the same way, with elements corresponding to IV repletion cost, PO repletion cost, abnormally high and abnormal low electrolyte levels respectively.
6.1.3 Fitted Q-Iteration with Gradient-boosted Trees

Now that we have defined our Markov decision processes, we can extract a sequence of one-step transition tuples from each hospital admission to produce a dataset $D = \{ (s^n_t, a^n_t, s^n_{t+1}, \phi^n_{t+1}) \}_{t=0, T_n-1} \}_{n=1:N}$, where $N$ is the number of distinct hospital visits, and $|D| = \sum_n T_n$, and solve for optimal treatment policies using batch reinforcement learning methods, namely Fitted Q-iteration (FQI) [21]. FQI is a data-efficient value function approximation algorithm that learn an estimator for value $Q(s, a)$ of each state-action pair in the MDP—that is, the expected discounted cumulative reward starting from $(s, a)$—through a sequence of supervised learning problems. FQI also offers flexibility in the use of any regression method to solve the supervised problems at each iteration.

Here, we fit our estimate of $Q$ at each iteration of FQI using gradient boosting machines (GBMs) [26]. This is an ensemble method in which weaker predictive models, such as decision trees, are built sequentially by training on residual errors, rather building all trees concurrently as is the case for random forests or extremely randomized trees. This allows models to learn higher-order terms and more complex interactions amongst features [132]. Gradient boosted trees have been increasingly used for function approximation in clinical supervised learning tasks, and have been demonstrated to have strong predictive performance [45, 129]. Boosting has been previously explored in conjunction with FQI by Tosatto et al. [120], who propose an additive model of the Q-function in which a weak learner is built at each iteration from the Bellman residual error in the previous estimate of $Q$. In this work, we instead output a fully fitted GBM at the end of each iteration of FQI.

Treating repletion strategy as a hierarchical decision-making task, we estimate the following three Q-functions for each electrolyte: $\hat{Q}^{\text{route}}(s, a) : S, A^{\text{route}} \to \mathbb{R}$ which gives the action value estimates for each repletion route, $\hat{Q}^{\text{PO}}(s, a) : S, A^{\text{PO}} \to \mathbb{R}$ and $\hat{Q}^{\text{IV}}(s, a) : S, A^{\text{IV}} \to \mathbb{R}$, corresponding to value estimates of different doses of oral
and IV supplements respectively. For each $\hat{Q}$, we train an optimal policy such that:

$$
\pi^{route}(s) = \arg\max_{a \in A^{route}} \hat{Q}(s, a) 
$$

(6.4)

$$
\pi^{PO}(s) = \arg\max_{a \in \{A^{PO} \setminus A_0^{PO}\}} \hat{Q}^{PO}(s, a)
$$

(6.5)

$$
\pi^{IV}(s) = \arg\max_{a \in \{A^{IV} \setminus A_0^{IV}\}} \hat{Q}^{IV}(s, a)
$$

(6.6)

where $A_0^{PO}$ and $A_0^{IV}$ denote elements in these action spaces corresponding to a dose of 0, that is, no repletion. In order to obtain our final treatment recommendations, we first query $\pi^{route}(s)$ for current state $s$. If this policy recommends one or both modes of repletion, we query policies $\pi^{PO}$ and $\pi^{IV}$ accordingly to select the most appropriate non-zero dosage level for the corresponding repletion route.

### 6.1.4 Reward Inference using IRL with Batch Data

In order to better understand current clinical practice, and in turn determine how to set priorities in clinical objectives for our optimal policy, we first apply an inverse reinforcement learning (IRL)-based approach to inference of the weights in the reward function $r = w \cdot \phi$, as optimized by clinicians in the historical UPHS data. The fundamental strategy of most algorithms for inverse reinforcement learning are as follows [5]: we first initialize a reward function parametrized by some weights $w$, and solve the MDP given this reward function for an optimal policy. We then estimate some representation of the dynamics of the RL agent when following this optimal policy, such as the state visitation frequency [89, 135]. Finally, we compare this estimate with the dynamics observed in the available batch data, and update $w$ to shift learnt policies towards to this behaviour, iterating until the learnt optimal policy
is sufficiently close. Here, we use policy feature expectations $\mu^\pi$, where:

$$\mu^\pi = \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t \phi_t(\cdot) \right]$$

(6.7)

to obtain a representation of agent dynamics that is decoupled from the reward function. For the behaviour policy, this is evaluated by simple averaging over patient trajectories in the dataset. However, in order to estimate the feature expectations for the learnt optimal policy given only this batch data, we turn to estimators for off-policy evaluation (as in Chapter 5), specifically per-decision importance sampling:

$$\hat{\mu}_{PDIS}^\pi = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=0}^{T} \gamma^t \rho_t^{(n)} \phi \left( s_t^{(n)} \right) \text{ where } \rho_t^{(n)} = \prod_{i=0}^{t} \frac{\pi(a_i^{(n)}|s_i^{(n)})}{\pi_0(a_i^{(n)}|s_i^{(n)})}$$

(6.8)

At each epoch of IRL, we use the difference in the $\ell_1$-normalized feature expectations of the behaviour and evaluation policy in order to update the reward weights $w$, as our objective is to infer the relative values of the elements in $w$, $||w||_1 = 1$, and optimal policies are invariant to scaling in the reward function. The complete procedure is outlined in Algorithm 5.

### 6.2 Results

For the experiments described in this section, we take the full cohort of 13,164 patient visits described in Section 6.1.1 and divide these into a training set of 7,000 and a test set of 6,134 visits. Of those in the training set, the number of hospital visits comprising electrolyte repletion events is 4,109, 4,430 and 867 for potassium (K), magnesium (Mg) and phosphate (P) respectively, where the mean length each visit is approximately four days. Sampling at 6-hour intervals, these admissions yield one-step transition tuple sets of size 54,228, 59,775, and 15,863; these make up the training sets for the treatment policies of each electrolyte. Learnt policies are then evaluated.


**Algorithm 5 Linear IRL with Batch Data**

**Input:** $D = \{(s^n_t, a^n_t, s^n_{t+1}, \phi^n_{t+1})_{t=0:T_n-1}\}_{n=1:N}$ where $\phi \in \mathbb{R}^d$; behaviour policy $\pi_b$

Reward weights $w_0 \in \mathbb{R}^d$, $\|w_0\|_1 = 1$; discount $\gamma$; epochs $E$; learning rate $\alpha$

**Initialize** $w = w_0$

$\mu_b = \frac{1}{N} \sum_n \sum_t \gamma^t \phi^n_{t+1}$

**for** epoch $i = 1 \rightarrow E$ **do**

$\pi_w \leftarrow FQI(D, w, \gamma)$

$\hat{\mu}_w \leftarrow PDIS(\pi_w, \pi_b, D, \gamma)$

$\nabla = \frac{\mu_b}{\|\mu_b\|_1} - \frac{\hat{\mu}_w}{\|\hat{\mu}_w\|_1}$

$w \leftarrow w + \alpha \cdot \nabla$

$w = \frac{w}{\|w\|_1}$

**end**

**Result:** $w$ 

*Return final weights for reward function $r(\cdot) = w \cdot \phi(\cdot)$*

using cohorts of size 3,440, 4,233, and 901 for K, Mg and P, selected in the same way from the test partition.

### 6.2.1 Understanding Behaviour in UPHS

Our first set of experiments aim to infer the reward function optimized by providers in the UPHS dataset, in order to gain insight into incentives underlying existing patterns of behaviour. For each electrolyte, we initialize our reward function with weights $w_0 = \frac{1}{4}[1 1 1 1]^T$, assigning equal priority to each of the four elements of $\phi(\cdot)$, and run the procedure described in Algorithm 5 over twenty epochs with a learning rate of 0.2. At each epoch, the current weights $w$ are used to learn a policy for the route of electrolyte supplementation $\pi^{route}(s)$, given the transitions in the training set. The first column in Table 6.3 summarizes the final weights, averaging over three independent runs of IRL, for cohorts K, Mg and P. We find that in the case of potassium, we obtain small negative weights on both the cost of IV and the cost PO repletion, that is, agents in the UPHS data appear to be rewarding the
administration of potassium supplements. This provides some evidence in support of concerns that care providers either tend to order potassium supplements reflexively (without fully considering cost or clinical necessity) or are unnecessarily conservative in avoiding potassium deficiency. This is emphasised by the fact that the penalty incurred for hypokalaemia (low potassium levels) is significantly higher relative to that for hyperkalaemia. In order to try to correct for this, we train our optimal policies with FQI using rewards with small positive penalties on repletion, while maintaining the relative weights of the remaining penalties at approximately the same value.

On the other hand, inferred weights for both magnesium and phosphate repletions suggest that a cost is incurred by the agent for both intravenous and oral repletion in these cases, with a larger penalty on IV. This is more in line with what we would expect. In particular, a higher effective cost on IV repletion can be justified in a number of ways: in the cost of the prescription itself in the necessary form for intravenous delivery, in the provider time taken to initiate and monitor delivery of the drug, and increased risk of overcorrection when setting the infusion rate, as well as bruising, clotting or infection at the infusion site.

Additionally, for both Mg and P, greater penalties are placed on above normal values. This may be because the risks posed to patients by excess magnesium or phosphate levels are considered to be more critical, or simply due to the fact these electrolytes are less likely to be over-corrected; both hypermagnesemia and hyperphosphatemia are rare in the dataset. While this may be reasonable, we attempt to avoid reinforcing this behaviour in our learnt optimal policies by shifting weights to penalize abnormally high and low values approximately equally, and doing to same with IV and oral repletion, when running FQI.
Table 6.3: Inferred behavioural priorities from IRL versus chosen reward weights for optimal policy, for electrolyte repletion route recommendations for K, Mg and P.

<table>
<thead>
<tr>
<th></th>
<th>UPHS POLICY $\pi^\text{route}_b$</th>
<th>FQI POLICY $\pi^\text{route}_w$</th>
</tr>
</thead>
<tbody>
<tr>
<td>K</td>
<td>$[-0.05 \ -0.08 \ 0.20 \ 0.67]^T$</td>
<td>$[0.07 \ 0.04 \ 0.15 \ 0.74]^T$</td>
</tr>
<tr>
<td>Mg</td>
<td>$[0.09 \ 0.08 \ 0.56 \ 0.25]^T$</td>
<td>$[0.29 \ 0.29 \ 0.21 \ 0.21]^T$</td>
</tr>
<tr>
<td>P</td>
<td>$[0.31 \ 0.11 \ 0.32 \ 0.26]^T$</td>
<td>$[0.17 \ 0.17 \ 0.33 \ 0.33]^T$</td>
</tr>
</tbody>
</table>

Figure 6.4: UPHS vs FQI-recommended potassium repletion for sample admission

6.2.2 Analysing Policies from FQI

With our chosen reward functions (as parametrized by the FQI policy weights in Table 6.3), we learn policies for repletion route, oral and IV dosage for each of the three electrolytes. Figure 6.4 illustrates the recommended repletion for potassium for a single hospital visit, obtained through construction of a hierarchical treatment recommendation strategy as outlined in Section 6.1.3. This is plotted for comparison along with the true repletion events in the UPHS data, and the measured potassium values. We can see from this example that, while there are multiple instances of IV repletion, when potassium levels drop but are still well within reference range, the FQI policy shows a preference for oral repletions, all repletion recommendations occur when potassium is below the reference range, and IV repletions are only recommended with the patient is significantly hypokalaemic.
Figure 6.5: Distribution of recommended actions for K, Mg and P

Figure 6.5 compares the distribution of actions taken in the UPHS histories with those recommended by our policies from FQI. We find that for potassium, the learnt policy recommends 75% fewer intravenous supplements and 50% fewer oral supplements. Where repletion is recommended, our policy tends to favour higher effective dosage of either oral and IV potassium. This strategy can be justified by the fact that in current practice, repletion events often fail to bring potassium levels into the target range (Figure 6.1), suggesting that smaller doses are often either unnecessary or not cost effective. The total number of recommended repletions for both phosphate, though the distribution of recommended doses for repletion approximately matches than in the historical data. For magnesium, as in the case of potassium, we see a shift in preference towards higher IV doses, as well as more frequent oral repletion, while the total number of recommendations recommended remains roughly unchanged. The lack of significant reduction in repletions may again be partly attributed to the fact that over-correction of magnesium is highly unlikely.

In order to investigate that factors influencing recommendations in our output policy, we train a pair of classification policies for PO and IV repletion respectively, mapping from patient states to binary actions. Figure 6.6 plots the Shapley values for these classifiers in the case of potassium. Shapley values evaluate the contribu-
Figure 6.6: Shapley values of top 10 features for (a) K-PO and (b) K-IV repletion. The SHAP value represents the impact of each feature in the state representation in pushing the predicted probability of repletion away from the population mean prediction, along with the direction of influence. Therefore, a high Shapley value associated with a feature can be interpreted as higher probability of recommended repletion. In addition to population-level feature importances, it allows for individual-level explanations of predictions.

As expected, current potassium values are among the most influential features in both cases, with low potassium levels associated with higher probability of repletion. We also find that oral repletion is more likely to be recommended at high levels of sodium (which is typically inversely correlated with potassium) and at high urine output. The latter is likely the direct result of the administration of diuretics, causing increased rates of potassium loss and necessitating repletion (as we noted in Figure 6.2). Creatinine also features highly in policies for both oral and IV potassium, with high levels of creatinine associated with increased probability of recommended IV repletion. A possible mechanism that may explain this is that accumulation of creatinine is commonly used to diagnose kidney failure, and typically necessitates
dialysis. Serum potassium can drop significantly following dialysis, with roughly 45% of patients presenting with post-dialysis hypokalaemia [11]. The optimal range of potassium levels for patients undergoing dialysis is therefore often higher [111], motivating urgent repletion.

6.2.3 Off-policy Policy Evaluation

Finally, we can produce a quantitative estimate of the value of our learnt policies offline using Fitted Q evaluation (FQE) [64]. FQE adapts the iterative Q-value estimation problem solved in FQI through a series of supervised learning problems, to the task of off-policy policy evaluation. Given our dataset of one-step transitions, and policy \( \pi_e \) to be evaluated, each iteration \( k \) of FQE takes as input all state-action pairs of the form \( \langle s_t, a_t \rangle \) in the dataset. The targets are then given by

\[
\hat{Q}_k^{\pi_e}(s_t, a_t) = r_{t+1} + \gamma \hat{Q}_{k-1}(s_{t+1}, \pi_e(s_{t+1}))\forall t, i,
\]

such that the value of a given state-action pair is given by an estimate of the immediate reward plus the expected discounted value of following policy \( \pi_e \) from this point onwards. Solving this regression task at each iteration yields a sequence of estimates \( \hat{Q}_k \).

Figure 6.7 plots the distribution over all state-action pairs \( \langle s, \pi_e(s) \rangle \) of Q-values estimated in this way for the optimal repletion route policies of potassium, magnesium and phosphate, over their corresponding test cohorts. Note these values represent estimates of the discounted cumulative weighted repletion cost and penalties for electrolyte imbalance, starting from each \( \langle s, \pi_e(s) \rangle \). We find that in all three polices, the mean value of the FQI policy is greater than that of the estimated behaviour policy followed by clinicians in the historical data. While gains are marginal in the case of magnesium (in line with the fact that the percentage reduction in recommended Mg repletion is relatively modest), estimates for both potassium and phosphate suggest significant improvement over current practice.
6.3 Conclusion

This chapter presents a data-centric approach to electrolyte repletion therapy in hospital. Patient admissions in a multi-centre dataset from the University of Pennsylvania health system are modelled as Markov decision processes, and used to learn strategies for efficient repletion of potassium, magnesium and phosphate levels through batch reinforcement learning methods. The proposed policies suggest fewer repletion events overall, with a shift towards oral supplements at higher doses. These recommendations have the potential to ease the burden presented by the ordering of prescriptions for the pharmacist, reduce demands on the clinician in terms of periodic re-evaluation of electrolyte levels and in the administration of supplements, and lower costs for the hospital, without compromising on patient outcome.

The work here outlines the first phase of ongoing work. As a next step, we look to verify the robustness of learnt policies to temporal drift—the work here draws on data collected between 2010 and 2015, and it is important to ensure that the quality of our recommendations is invariant to any shifts in behavioural patterns in the last
five years—as well as generalizability across datasets from different health systems, such as the MIMIC database [19].

Having evaluated our policies offline under these conditions to the extent possible, we can then run comparisons with actions chosen by clinical experts post hoc; these are likely to significantly differ from actions observed in historical data, as they are unconstrained by any procedural bottlenecks. We envision an experimental design similar to that in Li et al. [68] for example, presenting clinicians with single slices of patient trajectories, encompassing all relevant patient information available up to that time. In choosing time slices that would be most informative in evaluating and honing our current optimal policy, we can draw from work by Gottesman et al. [35] on identifying influential transitions. Finally, we hope to develop within the framework of the current clinical workflow a way to operationalize these tools, either through reminders to clinicians when repletion is deemed necessary, or via a background system that presents the best route forward given an active request for repletion by care providers. While there remain a number of fundamental questions to be answered before adoption by clinicians is possible, if implemented in a scalable and sustainable way, we believe these tools can be transformative to the current healthcare system.
Chapter 7

Conclusion

In this thesis, we introduce a generalizable framework for the management of routine interventions in the care of critically ill patients. We motivate the use of reinforcement learning in the development of clinician-in-loop decision support systems and describe how we can model planning problems in the acute care setting as Markov decision processes, using clinically motivated definitions of state, action and reward function. In choosing these problems, we target an array of common diagnostic and therapeutic interventions: the management of mechanical ventilation and sedation, the ordering of laboratory tests requiring invasive procedures, and the administration of effective electrolyte repletion therapy. We explore how we can better understand objectives and biases driving current clinical behaviour with respect to these interventions, and use this where applicable to guide the intervention strategies learnt. Finally, we present various methods by which to evaluate the optimal policies learnt through offline, off-policy reinforcement learning using only past clinical histories—through qualitative assessment of produced recommendations, the application of state-of-the-art off-policy policy evaluation methods, and with comparison and analyses with domain experts—demonstrating that this approach shows promise in re-evaluating and streamlining current clinical practice.
7.1 Future Research Directions

There are several avenues to be explored in building on the methodology described in this thesis. These may be broadly encompassed by the following three prongs of work: (i) advancing representation learning for clinical time series data, (ii) developing existing batch reinforcement learning methods, to improve sample-efficiency and speed up learning from biased observational datasets with limited observability of certain regions of the state-action space, and (iii) creating a robust framework for off-policy evaluation using these observational datasets.

With respect to state and action representation, the use of Gaussian processes (GPs) in this work was restricted to either the imputation of missing values or the estimation of uncertainty in time series forecasting. However, this could in principle be extended to learning a complete state transition model for the MDP, or alternatively, to infer latent representations in the form of Gaussian process latent variable models (GPLVMs) of the physiological state of the patient. Recurrent neural networks have also been widely used both to model clinical time series [27], while other deep architectures such as auto-encoders have been attempted in learning latent state representations [103]. However, it is possible that there is a fundamental limitation in the quality of the representation we can learn given the nature of the data at our disposal, with no prior information. I believe an important direction for future investigations is in mechanisms by which we can incorporate domain knowledge more explicitly to restrict the model class we must search over, both for modelling patient dynamics and in learning a policy function.

At the opposite end of the spectrum, it may also be worthwhile to revisit the use of discrete state representations in reinforcement learning, for example through clustering methods [37] or self-organizing maps [25], and explore whether we can provide guarantees on the quality of clustering and minimize loss of information, while taking advantage of the interpretability and sample efficiency of tabular RL.
In terms of the action space in clinical decision-making, the options framework \[117\] in hierarchical reinforcement learning—where an option can be thought of as a ‘macro-action’, defined by some policy, an initiation set of states and a termination condition based on sub-goals—naturally fits into the way in which interventions in acute care are typically administered. Each option may have a different set of available actions, such as for a patient prior to mechanical ventilation, immediately following intubation, or after the initiation of weaning protocol; policies are likely to be consistent within an option, while extremely dissimilar across options.

Reward design is central to efficient learning in RL and remains an incredibly challenging problem. While a number of heuristics and preliminary approaches to systematic reward design are presented in this thesis, work is still needed in designing mechanisms to, for example, explicitly learn the prioritization of frequent reward signals against sparse feedback, as well as immediate versus long-term objectives in problems with extended horizons. One possible approach to tackling these questions is by modelling discount factors. The degree of long-term impact can vary according to the reward objectives we consider. For example, while the negative impact of sudden spikes in pain levels or transient physiological instability may be relatively brief, the need for reintubation or the onset of organ failure can have a much more prolonged impact on patient outcome. This motivates the optimization of different objectives with tailored discount functions or, equivalently over different treatment horizons \[22\]. Improving our understanding of current clinical practice is also incredibly important. A well-calibrated model of clinical reasoning in diagnostic and therapeutic decisions can be used to bootstrap the learning of optimal policies. In doing so, we can account for how this reasoning varies across individuals and is subject to procedural limitations, and design incentives to shift behaviour towards more efficient care.
Finally, robust off-policy evaluation is a fundamental roadblock in use of reinforcement learning in practice and continues to be an important, active area of research. While approaches to evaluation in this work are restricted to model-free methods, this inherently limits the quality of both the learnt policies and of off-policy evaluation in data-poor settings. Leveraging recent work that looks to enable robust evaluation in continuous, high-dimensional environments [64], and drawing on progress in modelling transition dynamics in clinical data to aid the development of model-based or hybrid OPE methods, that allow for deeper analysis of policy performance offline, is necessary in engendering confidence in these methods and facilitating the next steps towards implementation in practice.

7.1.1 Translation to Clinical Practice

Beyond the modelling questions inherent to machine learning algorithms for clinical decision support, there are several hurdles to be overcome before their adoption in standard clinical practice [109, 127]. In this thesis, we touch upon the importance of careful consideration of the choice of problem, along with endorsement by relevant organizational stakeholders. Once a useful solution had been developed using retrospective data as proof of concept, a necessary next step is to clearly quantify the estimated value addition of the tool—in terms of clinical outcome as well as cost and time saved—in order to justify the launch of prospective studies [110]. These prospective studies typically involve ‘silent’ implementation of the decision support tool, evaluated by clinicians post hoc, rather than immediately influencing patient care [110]. This is followed by peer-reviewed randomized control trials evaluating the statistical validity of estimated benefits, and ensuring that the tool is providing novel, substantive insights rather than simply fitting to confounders. Additionally, it is crucial to verify that the recommendations provided are consistent in accuracy and utility when accounting for sociocultural factors in the healthcare delivery en-
vironment (such as clinician expertise, attitudes, and existing care patterns) and to
determine how the added benefit of the system may be influenced by characteristics
of the patient population [11]. This requires exploring the quality of learnt policies for
minority subgroups, and testing their robustness to dataset shift over time.

Even where the performance of the system is acceptable, a thoughtful approach
to diffusion of the technology and the reporting of recommendations is necessary for
sustained adoption by clinicians [53]. This includes providing some transparency and
explainability in output policies, both to foster trust in the machine learning system,
and to ensure continued confidence between patients and physicians [90]. Incorporat-
ing factors such as compliance, efficacy, and constraints in personnel or equipment,
as well as providing a degree of flexibility in recommendations that accounts for the
experience or expertise of the clinician, can help with this. It is also critical that the
necessary logistical infrastructure is in place to implement the recommended poli-
cies, through well-integrated EHR, pathology and prescribing systems for example,
and that clear regulation is in place regarding where responsibility lies for the de-
cisions made. This is essential in countering the legal and economic incentives that
perpetuate current modes of practice [95].

Finally, strategies are needed for the continual monitoring and maintenance of
these systems. It is important to be able to model any downstream effects of policies:
the potential impact of interventions on immediate as well as long-term patient health,
and whether these interventions may cause a shift in pressures to other stages of the
healthcare pipeline. For instance, policies that push for increased testing can in
turn increase the rate of false positives, and cause heightened demand for certain
unnecessary treatments. Policies favouring prolonged life support may overburden
the ICU, or palliative care facilities, while adding limited value in terms of quality
of life. It is also possible that recommendations reinforce biases that already exist
in the system, as these tools are used in both intended and unintended ways. These
are questions that are just beginning to arise in other fields, such as insurance or loan approval systems [11, 69], but are still under-explored in the healthcare context. Understanding how learning systems can be adapted to account for these issues, how frequently systems should be adapted—continual updates may be subject to drift, and are difficult to validate—and how these updates can incorporate changes in the clinical landscape (from evolving definitions of disease progression to the inclusion of new procedures or therapeutics) poses a significant challenge to future work.

Ultimately, data-driven decision support systems have the potential to be enormously impactful to patient management in critical care. Recent events have drawn sharp focus to the precarious state of current health infrastructure and the pressures faced by healthcare workers; this is true on a global scale. Building robust data-driven systems with careful consideration is one way in which we can ease these pressures, streamline care, and help ensure that we are better prepared to tackle future crises.
Bibliography


ICUMedical Inc. Reducing the risk of iatrogenic anemia and catheter-related bloodstream infections using closed blood sampling. 2015.


