7-7-2020 2:15 PM

Reinforcement learning in large, structured action spaces: A simulation study of decision support for spinal cord injury rehabilitation

Nathan Phelps, *The University of Western Ontario*

Supervisor: Lizotte, Daniel J., *The University of Western Ontario*

A thesis submitted in partial fulfillment of the requirements for the Master of Science degree in Computer Science

Abstract

Reinforcement learning (RL) has helped improve decision-making in several applications. However, applying traditional RL is challenging in some applications, such as rehabilitation of people with a spinal cord injury (SCI). Among other factors, using RL in this domain is difficult because there are many possible treatments (i.e., large action space) and few patients (i.e., limited training data). Treatments for SCIs have natural groupings, so we propose two approaches to grouping treatments so that an RL agent can learn effectively from limited data. One relies on domain knowledge of SCI rehabilitation and the other learns similarities among treatments using an embedding technique. We then use Fitted Q Iteration to train an agent that learns optimal treatments. Through a simulation study designed to reflect the properties of SCI rehabilitation, we find that both methods can help improve the treatment decisions of physiotherapists, but the approach based on domain knowledge offers better performance.

Keywords

Clustering; Large Action Space; Physiotherapy; Prior Knowledge; Representation Learning; Treatment Embedding; Word Embedding

Summary for Lay Audience

Reinforcement learning (RL) is a field of study that aims to build decision-making systems that base their decisions on the current and past state of the world, the previous actions undertaken, and the actions that may be taken in the future, and has been very successful in improving decision-making in several applications. Despite the successes of RL, applying traditional RL is challenging in some applications, such as the rehabilitation of people with a spinal cord injury (SCI). Among other factors, using RL to aid in decision-making for SCI treatment is difficult because there are many possible treatments (i.e., a large action space) and few patients (i.e., a limited training dataset). However, the treatments for SCIs have structure to them such that they can be grouped, facilitating learning about a treatment even if that treatment was not selected. In this work, we propose two approaches to grouping treatments so that an RL agent can learn effectively from limited data. One relies on domain knowledge of SCI rehabilitation and the other learns similarities among treatments using treatment embedding (inspired by word embedding). We then use Fitted Q Iteration, an iterative algorithm that estimates the value of each action in every patient state (e.g., unable to sit independently to full walking capacity), to learn which treatments are best in each state. Through a simulation study designed to reflect the properties of SCI rehabilitation, we find that both methods can be used to improve the treatment decisions of physiotherapists, but the approach based on domain knowledge offers better performance.

Acknowledgements

I would like to thank my supervisor, Dr. Dan Lizotte, for his expertise and guidance throughout this study, as well as his advice as I considered the next steps in my career. I would also like to thank Dr. Dalton Wolfe, Stephanie Marrocco, and the team of physiotherapists at Parkwood Institute for sharing their knowledge of spinal cord injury rehabilitation. Lastly, thank you to my parents for their support and for providing me with the opportunities I've had to pursue higher education.

Table of Contents

List of Tables

List of Figures

Chapter 1

1 Introduction

Reinforcement learning (RL) is a field of study that aims to build decision-making systems that base their decisions on the current and past state of the world, the previous actions undertaken, and the actions that may be taken in the future. In RL, the goal is to train a learner/decision-maker, known as the *agent*, to act intelligently (ideally, optimally) in its *environment*. The environment is everything outside of the agent with which the agent interacts. The current situation in the environment is described by its *state*. The agent interacts with its environment by selecting an *action* according to its *policy*, which is a mapping from states to actions. After selecting an action, the agent receives from the environment both a *reward* and information about the new state. A function of the rewards, known as the *return*, captures the long-term success of the agent; the return can be defined to place more or less weight on near-term versus long-term success. The expected value of this return is the quantity that the agent attempts to maximize with its actions.

RL has been very successful in improving decision-making in several applications, such as playing the game Go. In recent years, Google Deepmind's AlphaGo [1] has defeated the world's top Go players [2]. AlphaGo learned how to play through using supervised learning on games played by human experts and using RL to play against previous iterations of itself thousands of times. However, in some applications, we are not afforded the luxury of learning from such huge amounts of data. Obtaining accurate estimates of the value of each action in each situation is much more difficult with limited data to learn from. When the number of possible actions is large and actions are selected simultaneously, these issues compound and create a very difficult RL problem. The intersection of these phenomena is not uncommon, particularly in

healthcare. In this study, we focus on the rehabilitation of people with a spinal cord injury (SCI). From an RL perspective, this application can be characterized by the following criteria:

- Limited training data is available (i.e., training data is expensive)
- The action space (number of possible actions) is very large
- Actions are selected simultaneously (multi-action selection)
- The training data is obtained in advance through the decisions of an intelligent agent (the physiotherapists), but the policy followed is unknown
- The actions are structured in a manner such that they can be grouped

To our knowledge, RL has yet to be applied to physiotherapy. In general, problems with a very large action space and problems with multi-action selection are relatively unstudied in the RL field. In this study, we primarily focus on addressing the large action space through the development of methods that mitigate its negative effect. We leave it to further studies to develop methods that explicitly consider the multi-action component of this problem.

The methods we develop in this study are designed to take advantage of the structure of the actions in our application domain. We propose two methods that place the actions into groups, allowing the agent to learn about an action from data about related actions. This mitigates the impact of the large action space. In SCI rehabilitation, there is a known structure to the actions (treatments), which facilitates grouping the actions using physiotherapists' domain knowledge. However, this grouping can be a time-consuming process and requires expertise in SCI rehabilitation; people with this expertise are in short supply, so we also propose grouping the actions using a representation learning approach inspired by word embedding.

Using data to drive decision-making in physiotherapy is a new endeavour and, consequently, the data available is currently very limited. Based on the data currently available, we develop a simulator that is designed to imitate the rehabilitation process for people with an SCI. The use of a simulator also facilitates the evaluation of the methods through treating simulated patients using the decision support methods, which would not be possible using real data.

The main contributions of this thesis are 1) new methodology for RL in large action spaces with prior information and 2) a simulator for such data that mimics the properties of data obtained from SCI rehabilitation. Chapter 2 provides a detailed background of RL, representation learning, related works, and SCI rehabilitation; Chapter 3 describes the simulator we develop to imitate the rehabilitation process; Chapter 4 details the methods used for learning an optimal policy; Chapter 5 illustrates a special case of this application where all patients are assumed to return to unimpaired mobility; Chapter 6 discusses the results and possible next steps; and Chapter 7 provides the conclusions from this study.

Chapter 2

2 Background

2.1 Reinforcement Learning (RL)

This background on RL is largely based on the Tabular Solution Methods part of a book co-authored by Richard Sutton (one of the founding fathers of RL), *Reinforcement Learning: An Introduction* [3]. RL is a relatively new field of study, with modern RL dating back only to the late 1980s. RL aims to build decision-making systems that base their decisions on the current and past state of the world, the previous actions undertaken, and the actions that may be taken in the future. Along with supervised learning (e.g., regression, classification) and unsupervised learning (e.g., clustering), it is a subfield within the field of machine learning. However, the framework of an RL problem differs from that of either supervised or unsupervised learning. In a supervised learning problem, labeled training data is used to train a model so that predictions can be made on unseen data. An unsupervised learning problem involves finding structure within unlabeled training data. In RL, the goal is to train a learner/decision-maker, known as the *agent*, to interact with its *environment* in an "optimal" fashion (where "optimal" is defined using a numerical reward system). Figure 1 illustrates the general framework of an RL problem.

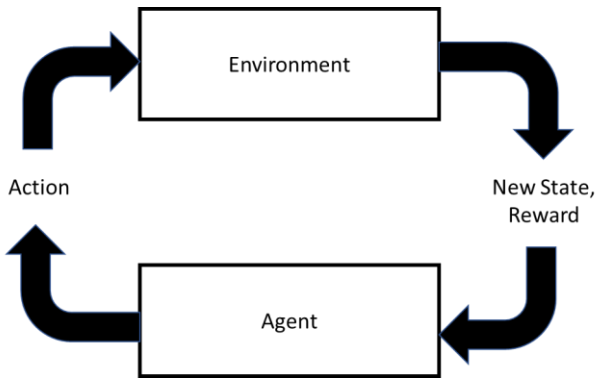

Figure 1: The general framework of a reinforcement learning problem.

In an RL framework, the environment is everything outside of the agent with which the agent interacts. The current situation in the environment is described by its *state*. For example, if the environment is a room in a house, the state might involve features such as the type of room (bedroom, living room, etc.), whether or not there are people in the room, the temperature in the room, and whether or not the lights are on. The agent interacts with its environment by selecting an *action*. Continuing with the example, the agent's candidate actions may include changing the temperature in the room or turning the lights on or off. After selecting an action, the agent receives from the environment both a *reward* and information about the new state. A function of the rewards, known as the *return*, can be constructed in order to place less value on rewards obtained further along in the future. The expected value of this return is the value that the agent attempts to maximize with its actions.

Markov decision processes (MDPs) are commonly used for modelling RL problems because they can formalize the agent-environment interaction. Consider discrete time points, t = 0, 1, 2, … At each time t, the agent receives information about the state, S_t, from the set of all possible states, S, and chooses an action, A_t, from the set of all possible actions, A. It then receives a reward, R_{t+1}, and information about the environment's new state, S_{t+1}. The transition to S_{t+1} occurs based on T, a function that defines the probability of transitioning to S_{t+1} given S_t and A_t. For each time step t, the data collected takes the following form: $(S_t, A_t, R_{t+1}, S_{t+1})$.

This process can either continue indefinitely or until an end state is reached. In cases where an end state is reached, the process is known as an *episode*.

The agent chooses its actions based on the current state according to a *policy*, which is a mapping from states to actions. Through experience, the goal is to learn a policy that maximizes

the expected return, called an *optimal policy*. Finding an optimal policy is not as simple as choosing the action that maximizes the expected immediate reward; some actions may not yield a large immediate reward, but instead cause a transition to a new state with a large expected reward (or cause a transition to a new state that will eventually lead to another state with a large expected reward). For this reason, it is necessary to have some way of assessing the value of being in each state in S. A *state value function* is a function that estimates the expected return achieved by following a given policy starting from a given state. Alternatively, a *state-action value function* can be used. For each state-action pair (s, a) in $S \times A$, the state-action value function estimates the expected return from taking action a in state s and following a given policy thereafter. State value functions must satisfy the following, known as the *Bellman equation*, which expresses the relationship between the value of a state and the value of its successor states.

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \text{ for all } s \epsilon S$$

where $\pi(a|s)$ is the probability of selecting action a while in state s under policy π, $p(s', r|s, a)$ is the probability of receiving reward r and moving to state s' given action a was chosen while in state s, γ is a discount factor between zero and one, and $v_\pi(s')$ is the expected future return given policy π is followed from state s'.

An *optimal state value function*, or *optimal state-action value function*, gives each state, or state-action pair, the largest possible expected return for any policy. The Bellman equation for the optimal state value function, v_*, is known as the *Bellman optimality equation* and can be written as follows:

$$v_*(s) = max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')] \text{ for all } s \epsilon S,$$

where $p(s', r|s, a)$ is the probability of receiving reward r and moving to state s' given action a was chosen while in state s, γ is a discount factor between zero and one, and $v_*(s')$ is the expected future return given an optimal policy is followed from state s'. In theory, if the transition and reward distributions are known, the system of $|S|$ equations with $|S|$ unknowns can be solved for v_*. Then, once the optimal state value function is known, it is relatively easy to find an optimal policy. However, in practice, iterative solution methods are typically used to approximate v_*.

Iterative policy evaluation is a method that uses an update rule to approximate v_*. *Policy improvement* refers to the method used to improve upon an existing policy in order to move towards an optimal policy (based on the current estimate of v_*). Consider a policy π that selects action a in state s. If it is better to select a different action b and follow π from that point onwards than it is to follow π from the beginning, then an improvement to π has been found, so π is adjusted to take action b in state s rather than action a. The repeated process of performing policy evaluation and policy improvement in order to gradually move towards an optimal state value function and optimal policy is known as *policy iteration*. Adjustments to these processes have been used in order to achieve faster convergence; see [3] for more information.

In the policy evaluation step, different update rules can be used. Below, a few common techniques and their update rules are discussed.

- Dynamic Programming (DP): DP is a very general algorithmic framework; in the context of RL, DP describes the method used to compute an optimal policy when we have a model of the environment (i.e., reward function and transition function). The Bellman

equation is used as its update rule. These updates are called *expected updates* because the update averages over all possible state transitions and rewards.

- Monte Carlo (MC): Unlike DP, MC methods do not use a model of the environment. They learn from sample experience recorded as an agent interacts with the environment, rather than the expected updates of DP. MC methods wait until the end of an episode before performing an update. Accordingly, the MC update rule is as follows:

$$V(S_t) = V(S_t) + \alpha[G_t - V(S_t)],$$

where $V(S_t)$ is the estimated value of state S_t, G_t is the actual return following time t, and α is a constant step-size parameter.

- Temporal-Difference (TD) Learning: TD learning combines the ideas from DP and MC methods. Like DP, updates are made before the final outcome is known, and like MC methods, TD learning uses sample experience. The following is the update rule for one-step TD learning:

$$V(S_t) = V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)],$$

where $V(S_t)$ is the estimated value of state S_t, R_{t+1} is the actual reward following time t, and α is a constant step-size parameter.

DP is known as a *model-based* policy evaluation algorithm because it makes use of the transition and reward distributions in order to estimate the value of each state, while MC and TD learning are *model-free* policy evaluation algorithms because they use recorded experience in place of a model. It should be noted that state value functions alone cannot be used to compute an optimal policy, thus the transition and reward distributions are still needed in order to compute an optimal policy under the framework shown. However, the model-free algorithms can

be adjusted to compute the optimal state-action value function instead of the optimal value function. Using the optimal state-action value function, it is trivial to determine an optimal policy.

In the "traditional" RL framework, the agent chooses actions as it learns. In other words, one of the above methods is used for policy evaluation, policy improvement is performed in order to move towards an optimal policy, actions are selected according to this policy, and the process repeats. This is called *on-policy* learning. In on-policy learning, the actions typically are not selected entirely based on the current estimate of the optimal policy. The reason for deviating from the best estimate of the optimal policy is that it is important for the agent to explore alternative actions in order to learn about them. The exploration-exploitation trade-off is a unique feature of RL compared to other forms of machine learning. In order to perform well, the agent should exploit its knowledge so that it selects the best actions. However, in order to learn which actions are the best, it must explore actions that it currently believes are suboptimal.

In some cases, the estimate of the best policy is not used at all in action selection. The training data in such cases is generated according to some other policy, known as the *behaviour policy*, and the *target policy* (i.e., the optimal policy) is learned from this data. This is called *off-policy* learning. A common off-policy technique is Q-Learning, which directly approximates the optimal state-action value function. Q-Learning's update rule is as follows:

$$Q(S_t,\, A_t) = Q(S_t,\, A_t) + \alpha[R_{t+1} + \, max_a Q(S_{t+1},\, a) \, - Q(S_t,\, A_t)] \,,$$

where $Q(S_t,\, A_t)$ is the estimated value of taking action A_t in state S_t, R_{t+1} is the reward received after taking that action, $max_a Q(S_{t+1},\, a)$ is the estimated maximum value of any action in state S_{t+1}, and α is a constant step-size parameter.

All of the methods above are used for *online* learning. That is, the updates are performed as we continue to generate more data. In some cases, the data is generated in advance and the goal is to compute an optimal policy based on this data. This is known as *batch* learning and is common in situations where training data is very expensive to obtain. An advantage of batch learning is that its solutions are relatively stable compared to online learning [4]. One example of an off-policy batch learning algorithm is Fitted Q Iteration [5]. Its implementation is shown using the pseudocode below:

Algorithm 1 Fitted Q Iteration

Inputs:
1. dataset $D = ((S_t, A_t, R_{t+1}, S_{t+1}); t = 1, 2, ..., n)$
2. regression model M **initialized in any fashion**
Output:
1. fitted regression model M

```
1: create empty dataset T = ((S_t, A_t, Q_{t+1}); t = 1, 2, ..., n), where Q_{t+1} is the sum
   of the immediate reward and the estimated maximum value obtainable when in the
   next state
2:
3: for t = 1 To n do
4:     T.S_t ← D.S_t
5:     T.A_t ← D.A_t
6:
7: while !converged do
8:     for t = 1 To n do
9:         T.Q_{t+1} ← D.R_{t+1} + max_{a∈A}(predict(M, (D.S_{t+1}, a)))
10:    fit M using T
11:
12: return M
```

Figure 2: The Fitted Q Iteration algorithm is shown in pseudocode. The predict function (line 9) takes a model and its inputs and generates a predicted output.

Fitted Q Iteration is an iterative process whereby a new training dataset is created and a regression model is fit to the data, directly estimating the state-action value function of the optimal policy. In each iteration, the estimated state-action value function from the previous

iteration is used to create the new training dataset. The algorithm terminates when the criterion for convergence is met.

2.2 Representation Learning

In statistical modelling tasks, it is common to transform the input variables (also known as features or predictors) in some way in order to improve the model. For example, if there appears to be a quadratic relationship between an input and the output, then the second order term for that input would be incorporated into the model. However, in some cases, it may be infeasible to manually transform the input variables appropriately; the appropriate transformation may be non-trivial to identify or there may be too many input variables to consider. Representation learning, also called feature learning, is a set of techniques used to generate new features by transforming the original input. Autoencoding, principal component analysis, word embedding, and clustering are all forms of representation learning. The most relevant representation learning methods for our task are word embedding and clustering.

Word embeddings are a way of representing words in vector space in such a way that similar words should be close to each other. GloVe [6] is a popular model used to generate word embeddings using a corpus of training data. It makes use of a co-occurrence matrix, which is a representation of the number of times words occur together within a context window. A context window is the number of words we consider to the left and right of the context word. An example is shown in the table below, where the bolded word is the context word and a context window of size two is used. Words within the context window are underlined.

Example Text Document	Co-occurrence Pairs
He spiked the ball over the net.	(he, spiked); (he, the)
He **spiked** the ball over the net.	(spiked, he); (spiked, the); (spiked, ball)
He spiked **the** ball over the net.	(the, he); (the, spiked); (the, ball); (the, over)
He spiked the **ball** over the net.	(ball, spiked); (ball, the); (ball, over); (ball, net)

Table 1: An illustration of the co-occurrence pairs resulting from an example text document with a window of size two. The context word is bolded and the words within the context window are underlined.

GloVe generates word vectors by minimizing the following objective function:

$$\sum_{i,j=1}^{V} f(X_{ij})(w_i^T u_j + b_{w,i} + b_{u,j} - logX_{ij})^2,$$

where V is the size of the vocabulary (i.e., number of words), X is the co-occurrence matrix, w_i is a word vector, u_i is a context word vector, $b_{w,i}$ and $b_{u,j}$ are the biases associated with the word vector and context word vector respectively, and $f()$ is a weighting function of the following form, with x_{max} representing the chosen maximum number of co-occurrences to consider and α another chosen parameter with a default value of 0.75:

$$f(x) = \begin{cases} (x/x_{max})^{\alpha} & if\ x < x_{max} \\ 1 & otherwise \end{cases}$$

When X is symmetric, the model creates two sets of word vectors that differ only due to their random initialization. The final vector representation of the words is the sum of the two sets of word vectors.

Clustering is an unsupervised learning approach used to identify similar observations and place them into groups. There are several different clustering algorithms, including single

linkage, max linkage, and K-means. As discussed by Ben-David [7], each of these algorithms emphasizes different requirements, so the choice of clustering algorithm must be made with the application in mind. Linkage clustering begins with each observation forming its own cluster (i.e., each cluster is a singleton). In each iteration of the algorithm, a union is performed between the two clusters that are nearest to one another. For single linkage, the distance between two clusters is the *minimum* distance between any pair of observations (that are not part of the same cluster) within the two clusters. For max linkage, the distance between two clusters is the *maximum* distance between any pair of observations within the two clusters. Single linkage prioritizes placing similar points in the same cluster over ensuring that all observations within a cluster are similar, while max linkage prioritizes the opposite. Both do not place any importance on the relative size of each cluster. K-means clustering begins with K randomly generated points. Each observation is placed into the group associated with the point to which it is nearest. The mean of each group replaces the K points and each observation is placed into the group associated with the mean to which it is nearest. This process continues until none of the observations change to a new group from one iteration to the next. Unlike the linkage algorithms, K-means clustering prioritizes that each group is composed of a similar number of observations.

2.3 Related Works

Yu, Liu, and Nemati [8] provide a thorough summary of the uses of RL in healthcare. One of the most common applications is the creation of dynamic treatment regimes, which are RL policies used to aid in the treatment of chronic conditions such as cancer [9]-[25], diabetes [26]-[40], anemia [41]-[51], HIV [52]-[61], and mental illnesses [22], [62]-[87], as well as critical care [88]-[116]. Other applications of RL in healthcare include automated medical diagnosis, health resource scheduling, optimal process control, and drug discovery and development [8].

In some of these works, expert domain knowledge is used in conjunction with the RL agent in order to make treatment decisions. Sadati et al. [112] incorporate the opinions of anesthesiologists to select appropriate initial doses of anesthesia and ensure that the doses remain within a safe range. Gaweda et al. [43]-[45] also use domain knowledge in order to improve the treatment of anemia. Anemia is caused by an inability to adequately produce endogenous erythropoietin (EPO), and consequently red blood cells. Patients can be given doses of EPO in order to keep their hemoglobin levels within a healthy range. The appropriate dose varies from person to person, making RL a useful tool to aid in individualized treatment. The applicable domain knowledge in the treatment of anemia was outlined by Yu et al. [8]:

"[F]or all patients, the dose-response curve of HGB versus EPO is monotonically non-increasing. Therefore, if a patient's response is evaluated as insufficient for a particular dose at a particular state, then the physician knows that the optimal dose for that state is definitely higher than the administered one."

Incorporating this domain knowledge into model development facilitates the creation of RL models more suitable to the problem than models from traditional methods.

The problems associated with a large discrete action space have received limited consideration in the RL literature. A related, more regularly studied issue in RL is the problem of having a large state space. Commonly, data-driven feature selection and feature learning are used in such problems. For a detailed summary of incorporating feature learning in batch RL, see [117]. Although less commonly studied, there are some works that have dealt with the issue of a large action space. Pazis and Parr [118] propose a generalized value function that facilitates more efficient action selection from a large set of actions. However, their work does not address the issues involved with learning a value function in the presence of a large action space. Laber et al.

[119] study the optimal allocation of treatment in terms of space and time for treating an infectious disease. Resources are limited such that a limited number of locations, N, can be treated, so their method assigns priority scores to each location and locations are treated until the resources are depleted. Due to spatial interference, the priority score for each location is dependent on the treatment decision for the other locations. With many possible treatment locations, jointly optimizing across all possible locations is infeasible, so the authors use a greedy batch updating algorithm as described below:

1. The first n ($n \leq N$) locations for treatment are chosen assuming that no other location receives treatment.
2. The priority scores of the remaining locations are updated to account for the treatment of the chosen locations.
3. Based on the priority scores computed in Step 2, another batch of n (or $N - n$ in the case $2n > N$) locations are chosen for treatment.
4. Steps 2 and 3 are repeated until N locations are chosen for treatment.

Dulac-Arnold, Evans, et al. [120] propose an actor-critic approach that uses a method similar to word embedding, where all of the actions are mapped to vector space. Given a state, the actor chooses some proto action (i.e., some value in vector space). It must be noted that this proto action is unlikely to be one of the candidate actions. Based on Euclidean distance, the k actions nearest the proto action are kept as candidates for action selection. From these k actions, the critic then chooses the action with the highest state-action value. The training of both the actor and the critic is performed in the continuous action space with multi-layer neural networks using Deep Deterministic Policy Gradient [121].

Some past work in RL has involved hierarchies of actions [122]-[124]. Dietterich et al. [125] propose a model-based action refinement method whereby the models of the transition and reward distributions for an action in a given state are smoothed by other related actions (i.e., actions within the same group according to the part of the hierarchy one level up). They demonstrated that this smoothing across actions can dramatically reduce the amount of training data needed in order to learn an effective policy. Learning a policy according to grouped actions, as we propose in this work, is a specific case of this smoothing.

2.4 Spinal Cord Injury Rehabilitation

In 2012, Noonan et al. [126] estimated that 85 556 people in Canada were living with an SCI. Depending on the severity of the injury, an SCI can cause loss of various voluntary muscle movements and sensation, complete paralysis of limbs [127], and loss of autonomic functions such as bowel or bladder control [128]. Treatment for SCI includes physiotherapy such as sitting, standing, and balance training, strength activities, gait re-training, the use of orthoses/braces, function stimulation, patterned electrical stimulation, and walking. The goal of SCI rehabilitation is generally to regain functions that are common in everyday life, such as balance and walking [127], but may also include patient-specific goals such as being able to play a favourite sport again.

The Lawson Health Research Institute (LHRI) is the research institute of London Health Sciences Centre and St. Joseph's Health Care London in London, Ontario. LHRI's Research 2 Practice (R2P) team's goal is to optimize treatment for patients with either an SCI or acquired brain injury. This team has been collecting data that captures the experience of an SCI patient in their care through their Health Sciences Research Ethics Board (HSREB) approved project (HSREB #108848). The following section describes at a high level the process undertaken in the

treatment of an SCI inpatient at Parkwood Institute (PI) (part of the LHRI), with a focus on the data that they have been collecting.

Upon admission, the patient is assessed and assigned to one of 12 states, ranging from being unable to sit independently to full walking capacity, according to the SCI Standing and Walking Assessment Tool (SWAT) [129]. In general, patients that have more serious injuries are provided more rehabilitation and longer inpatient hospital stays. Physiotherapists select treatments from a myriad of candidate treatments and, on a weekly basis, document the treatments that were performed. These treatments have similarities, facilitating their placement into groups. Figure 3 shows the treatment classification model developed by the Parkwood Rehabiliation Innovations in Mobility Enhancement (PRIME) project team. They have grouped treatments together based on the properties of the treatments such as the treatment's purpose (impairment management, priming, task specific, functional), the patient's orientation (lying, sitting, kneeling, standing), the level of movement involved (static, dynamic), and the patient's level of independence (assisted, independent). It should be noted that a treatment may be placed in more than one group; depending on the situation, some treatments can be used to serve different purposes. However, the methods we propose in this thesis place each individual treatment into exactly one group.

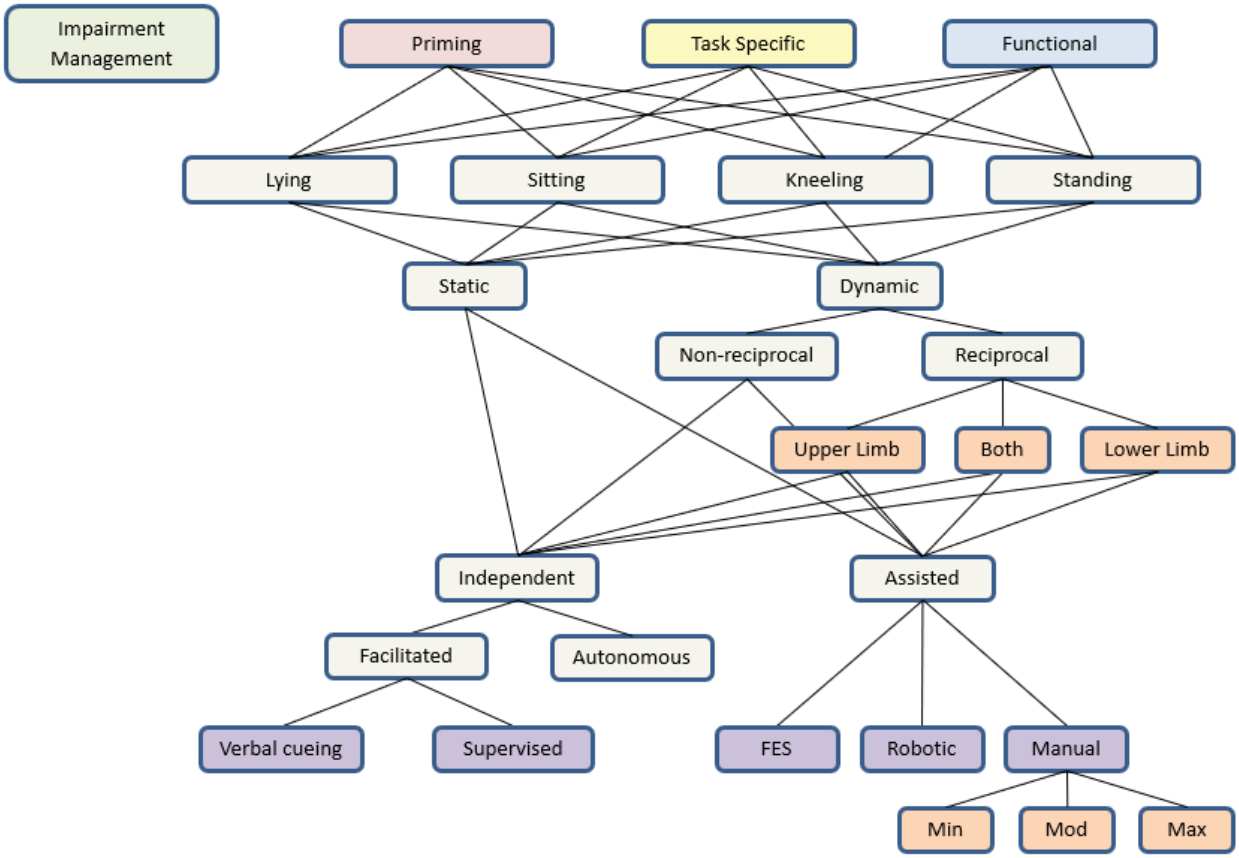

Figure 3: An illustration of the system developed by the Parkwood Rehabilitation Innovations in Mobility Enhancement (PRIME) project team to classify individual physiotherapy treatments into groups. Functional Electronic Stimulation is represented by “FES”.

Chapter 3

3. Simulator

In order to facilitate the development of reinforcement learning (RL) methods for spinal cord injury (SCI) treatment selection, we create a simulator to imitate the rehabilitation process for a patient. This simulator was constructed from a combination of existing SCI data and extensive input from physiotherapists[1] from the Research 2 Practice (R2P) team, Parkwood Institute (PI), Lawson Health Research Institute (LHRI). Although existing data was used to guide parameter choices for the simulator, the focus when creating the simulator was placed on the properties of SCI rehabilitation. Thus, while the simulator generally reflects the internal characteristics of SCI rehabilitation, it is not necessarily an accurate representation of SCI rehabilitation in its entirety.

There are two main benefits of using a simulator in this study. First, although it is anticipated that SCI treatment data will accrue through the R2P team's project, the amount of real data currently available is limited, so much so that it may not be reasonable to expect to be able to reliably train an agent. Secondly, the use of a simulator facilitates evaluation of the methods in a way that would not be possible with real data; the same set of simulated patients can be treated using different treatment selection processes and the results from each process can be compared to one another. The simulator is implemented in R version 3.5.1 [130].

3.1 Environment

In the case of SCI treatment, the environment is the patient. The states reflect the current health status as relevant for treatment and the actions correspond to treatment choices.

[1] We thank Dr. Dalton Wolfe, Stephanie Marrocco, Stephanie Cornell, Melissa Fielding, Deena Lala, Patrick Stapleton, Bonnie Chapman, Heather Askes, Rozhan Momen, and the rest of the physiotherapists from the spinal cord injury and acquired brain injury rehabilitation programs at Parkwood Institute.

3.1.1 States

The state that each patient is in is defined by two components, their stage and the number of weeks remaining in their rehabilitation.

3.1.1.1 Stages

There are 12 stages ranging from zero, the most impaired, to eleven, a terminal state indicating unimpaired mobility (i.e., no standing or walking impairment). These stages are chosen to represent the 12 stages of the SCI Standing and Walking Assessment Tool (SWAT) [129].

3.1.1.2 Number of Weeks Remaining

Each patient has a limited number of weeks of treatment. Based on their initial stage, they are afforded a maximum number of treatment weeks according to the following formula:

$$maximum\ number\ of\ weeks = floor[1.25(11 - stage)]$$

This formula was chosen to reflect the grouping methodologies used by hospitals and other health care facilities [131] and to facilitate a mean length of inpatient stay that approximates the mean stay in the data obtained from PI. At PI, they use the Rehabilitation Patient Grouping Methodology. After each week of treatment, a patient's remaining number of weeks is decremented by one. It should be noted that the formula we use in the simulator is a simplification of the actual methodologies used in practice; however it could easily be substituted with the formula used in a particular health care system/setting.

3.1.2 Treatments

In clinical practice, there are a myriad of potential treatments (from 10s to 100s depending on how treatments are classified), but they can be grouped in a meaningful way in terms of their

appropriateness for patients with different health statuses (for example, by using the classification scheme outlined in Figure 3). In the simulator, we chose to have 110 treatments numbered 1 to 110, each placed into one of 11 groups. The first group is composed of treatments 1 to 10, the second group is composed of treatments 11 to 20, and so on. For each stage each group is assigned a ranking (lower is better), indicating how useful treatments from these groups are expected to be if applied to a patient in that stage. As an example, for Stage Four patients, treatments 41 to 50 (the fifth group) are given a ranking of one, treatments 31 to 40 and 51 to 60 (groups four and six, respectively), are given a ranking of two, and the remaining treatments are given a ranking of three. Table 1 shows the ranking of each treatment group for Stage Four. Using this approach, each stage has 10 actions expected to be the best, 20 actions expected to be second to these top 10 actions, and another 80 actions expected to be inferior to the top 30. In order to maintain this distribution of actions for the edge stages (zero and 10, since 11 is terminal), the two groups nearest the group ranked first are both given a ranking of two.

Group	1	2	3	4	5	6	7	8	9	10	11
Ranking	3	3	3	2	1	2	3	3	3	3	3

Table 2: This table outlines the ranking (lower is better) of each treatment group for a patient in Stage Four.

In order to simulate the effectiveness of treatments, it is necessary to have some method for differentiating one treatment from another. It is unreasonable to assume that every treatment from each group is equally useful for each stage. For this reason, we use the idea of a *treatment benefit* to provide a numerical value indicating the usefulness of each treatment in each stage. In practice, these treatment benefits are unobservable, so they are used only to simulate experience and are not involved in the process of learning an improved policy.

3.1.2.1 Actual Treatment Benefits

Each treatment has 11 associated treatment benefits, one for each non-terminal stage. The treatment benefits are generated from a normal distribution with mean and standard deviation set based on the treatment's ranking. The table below shows the mean and standard deviation associated with each treatment ranking. The standard deviations increase with treatment ranking because we assume that treatment benefits vary more for treatments that are not specifically intended for the given stage. The actual treatment benefits are randomly generated once, at the beginning of the simulation, and are constant for the entire simulation.

Treatment Ranking	Mean	Standard Deviation
1	7.0	0.75
2	5.5	1.25
3	4.0	1.50

Table 3: The actual treatment benefits are simulated from a normal distribution using the parameters associated with their treatment ranking. This table shows the parameters associated with each rank.

3.1.2.2 Perceived Treatment Benefits

In order to simulate the presence of domain knowledge about different treatments, we introduce the idea of a perceived treatment benefit. Perceived treatment benefits represent a physiotherapist's opinion on the benefit of each treatment for each stage. Since physiotherapists are knowledgeable about these treatments, the perceived treatment benefits are designed to be correlated with the actual treatment benefits. They are generated from a conditional normal distribution, conditioned on the actual treatment benefits. The correlation and unconditional mean and standard deviation for each treatment ranking are shown in Table 4. With ρ

representing the correlation and ATB representing the actual treatment benefit, the conditional mean and standard deviation are computed as follows:

$$Conditional\ Mean = Unconditional\ Mean + \rho(Unconditional\ Standard\ Deviation)/(Standard\ Deviation_{ATB})(ATB - Mean_{ATB})$$

$$Conditional\ Standard\ Deviation = \sqrt{(1-\rho^2)(Unconditional\ Standard\ Deviation)}$$

Like with the actual treatment benefits, the perceived treatment benefits vary more for treatments that are not specifically intended for the given stage. Unlike the actual treatment benefits, perceived treatment benefits are generated for each iteration of the simulation, representing differences in individual physiotherapists' opinions.

Treatment Ranking	Correlation	Unconditional Mean	Unconditional Standard Deviation
1	0.8	7.0	1.00
2	0.7	5.5	1.50
3	0.5	4.0	1.75

Table 4: The perceived treatment benefits are simulated from a conditional normal distribution based on the actual treatment benefits. The correlation between the perceived treatment benefits and the actual treatment benefits and the unconditional mean and standard deviation for each treatment ranking are shown in this table.

The scatter plot in Figure 4 shows the actual treatment benefits in Stage Zero and the associated perceived treatment benefits that a physiotherapist may have. This plot clearly shows that the treatments with a larger rank exhibit a much weaker correlation between actual and perceived treatment benefits.

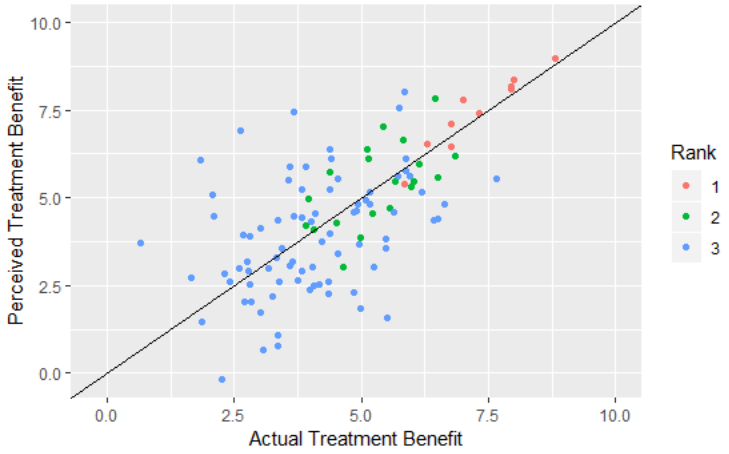

Figure 4: A scatter plot of the perceived treatment benefit versus actual treatment benefit for a patient in Stage Zero, categorized by their treatment ranking.

3.1.2.3 Aggregate Treatment Benefit

For a given treatment period, a physiotherapist selects multiple treatments to use. The aggregate treatment benefit refers to the sum of the individual treatment benefits of the selected treatments.

3.1.3 Transition Function

From a stage *x*, there are only two possible next stages, *x* and *x+1*. The probability of a transition from *x* to *x+1* is conditioned only on the aggregate treatment benefit of the selected treatments. The probability, p, of a transition to the next stage is computed using the formula shown below and is visualized in Figure 5:

$$p = \frac{2}{3}\left[\frac{1}{1 + e^{-(0.2339304 - 13.49396(aggregate\ treatment\ benefit))}}\right]$$

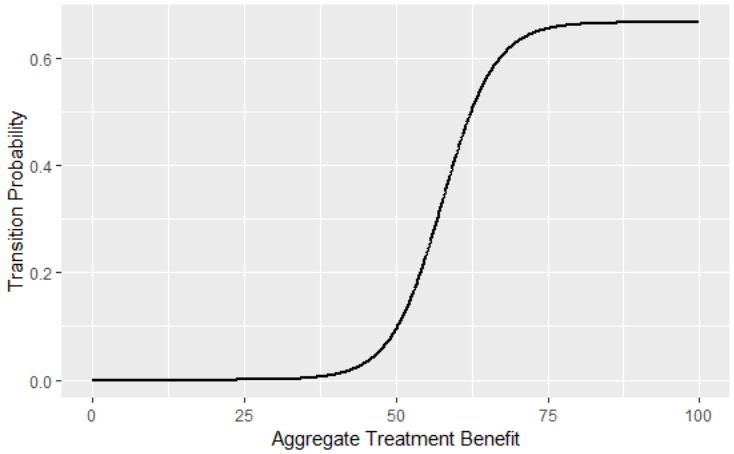

Figure 5: The probability of a patient's transition to the next stage as a function of their aggregate treatment benefit.

This transition function is designed to meet three criteria:

- In accordance with the data received from the R2P team, PI, LHRI, the mean probability of transition using the physiotherapists' treatment selections should be in the range of 0.25-0.30. Note that this range is a coarse estimate, as it is based on the cumulative improvement in stage from admission to discharge for all patients divided by the cumulative number of weeks of treatment. Under simulated physiotherapist treatment selection, the mean transition probability is approximately 0.253.
- The transition probability should be very small, but non-zero, when treatments are selected uniformly randomly. Uniform random treatment selection results in a mean transition probability of approximately 0.008.
- Regardless of the treatments chosen, the transition probability should never approach 1 because the body requires time to heal. With this transition function, the transition probability cannot exceed 0.667.

For the first two criteria, it must be noted that transition probabilities are dependent on the set of actual treatment benefits and that these mean transition probabilities have been computed across these sets. Thus, the mean transition probabilities under a single realization of a set of actual treatment benefits (which is what is used in this study) are not necessarily the values stated.

3.2 Generating Experience Data

The experience data is generated using episodes. An episode is a patient's entire treatment period.

3.2.1 Episode Initialization

An episode begins with the random selection of a non-terminal stage. Based on the data obtained from the R2P team, PI, LHRI, the initial stage is sampled from a distribution where Stage Zero is selected with a probability of $\frac{2}{7}$. The remaining ten non-terminal stages each account for $\frac{1}{14}$ of the distribution's probability mass.

3.2.2 Treatment Selection

A set of perceived treatment benefits is generated for the given stage. From this, a treatment plan is created, representing a course of treatment that a physiotherapist might plan for a week. The treatment plan is composed of the eight treatments with the highest perceived treatment benefit.

3.2.3 Episode Termination

An episode terminates when a patient either reaches the final (fully healthy stage) or after their allocated number of weeks has elapsed. The figure below shows six possible patient trajectories through state space. An episode ends upon reaching either the horizontal (out of time) or vertical (unimpaired mobility) dashed line.

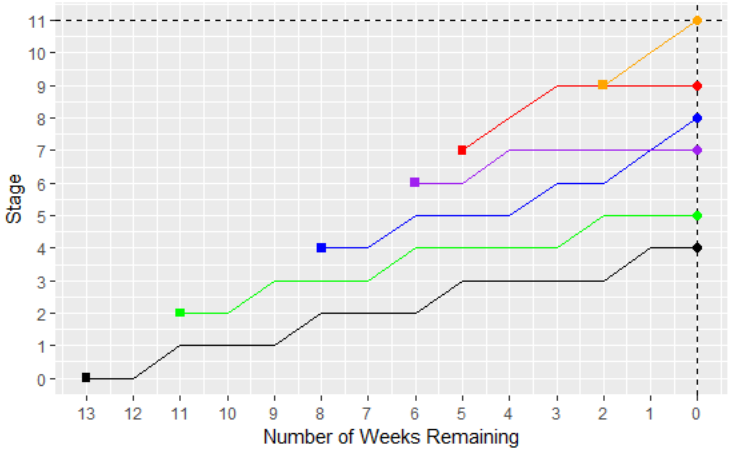

Figure 6: Possible trajectories through the state space for six simulated patients. The start and end of the episodes are represented by a square and circle respectively.

Chapter 4

4. Learning an Optimal Policy

In this study, we focus on what we believe to be the characteristic of spinal cord injury (SCI) rehabilitation that most dramatically limits the effectiveness of reinforcement learning (RL) as a decision support tool: the very large action space. We use three different methods designed to learn an optimal policy, with each one handling the action space in a different manner. However, all three methods share the same reward function and return.

4.1 Reward Function and Return

A reward of zero is given for each step until the final week of treatment. At this point, the reward is the final stage reached by the patient. A discount factor of one (i.e., undiscounted) is used, so the return is simply the reward given at the end of the treatment period.

4.2 Baseline Approach

First, we attempt to learn an optimal policy using an approach that does not group the actions.

4.2.1 Defining State-Action Pairs

One baseline approach would be to consider each unique joint action as a different action, and not to generalize across these. However, this would induce an action space of size $\binom{110}{8}$, which is larger than 400 billion, so a plausibly-sized dataset would have many unobserved joint actions and possibly zero joint actions with more than one observation. Hence, this naïve approach would make learning and generalization impossible.

We therefore provide an alternative baseline learning method for comparison that assumes that the impact of any given action choice is unrelated to any other action choices made

at the same timepoint. To be clear, this assumption is not plausible, but represents the simplest possible way of handling the multi-action component of the problem. The original data has eight actions in each row of data. Based on the aforementioned assumption, each row of data is mapped to eight rows, so that each action is represented by a single row and the stage, number of weeks remaining, reward, and next stage are the same for all eight rows. An example is shown below:

Original Data:

Stage	No. of Weeks Remaining	Actions	Reward	Next Stage
3	2	24, 26, 33, 34, 38, 39, 42, 50	0	3

Adjusted Data:

Stage	No. of Weeks Remaining	Action	Reward	Next Stage
3	2	24	0	3
3	2	26	0	3
3	2	33	0	3
3	2	34	0	3
3	2	38	0	3
3	2	39	0	3
3	2	42	0	3
3	2	50	0	3

Table 5: Table 5a (top) shows a possible row from the simulated data in its original format. Table 5b (bottom) shows the transformation of that row into eight rows in the adjusted dataset.

4.2.2 Estimating the Optimal State-Action Value Function

We use the treatment experiences of 1000 simulated patients in order to estimate the optimal state-action value function. The Research 2 Practice (R2P) team, Parkwood Institute (PI), Lawson Health Research Institute (LHRI) has collected data from the treatment of approximately 90 patients over the last few years, so we consider 1000 patients a reasonable upper bound on the number of patients from which treatment data can be collected before we require that the agent is able to learn something meaningful. Since the training data is obtained in advance through following a less than optimal policy, we use an off-policy batch learning algorithm, Fitted Q Iteration, to estimate the optimal state-action value function. Our requirement for convergence is that none of the coefficients of the following regression model change by more than 0.0001 from one iteration to the next:

$$
\begin{aligned}
&Q(s, a, No.\,of\ Weeks\ Remaining) \\
&\quad = \beta_0 + \beta_1(No.\,of\ Weeks\ Remaining) \\
&\quad + \sum_{action} \beta_{action} I(action = a) + \sum_{stage} \beta_{stage} I(stage = s) \\
&\quad + \sum_{action,stage} \beta_{action,stage} I(action = a) I(stage = s)
\end{aligned}
$$

4.2.3 Results

We find that the estimated state-action values for the optimal policy under this methodology are far too optimistic. For an incoming patient beginning at Stage Zero, the minimum state-action value is 7.98, despite the fact that the median Stage Zero patient from the training data reaches Stage Three and no Stage Zero patient reaches Stage Eight. The estimates of the optimal state-action value function for Stage 10, the last non-terminal stage, have a very high variance because

some actions are selected very few times at this stage, even after 1000 episodes. Consequently, some state-action value estimates for actions in Stage 10 are very high. In turn, this results in other actions also having an optimistic state-action value estimate in Stage 10 because the agent believes that, even if the patient does not reach unimpaired mobility immediately, it will be able to select actions next week that will help the patient reach unimpaired mobility. This optimism percolates to the state-action value estimates in Stage Nine, then Stage Eight, and so on, all the way down to Stage Zero. As a result, the agent trained using individual actions is not useful.

4.3 Using Groups of Actions

As discussed in Section 4.2.3, the state-action value estimates from the baseline approach have a large variance because the number of samples is small. As in regularized regression like lasso regression [132] and ridge regression [133], it can be beneficial to use a model with more bias in order to reduce the variance of the parameter estimates. With this in mind, we propose grouping the actions since it is known that there is some structure to the actions in SCI rehabilitation. With an infinite amount of data, models using grouped data would be less effective than the traditional approach because of their relatively high bias. However, the reduced variance in the models with grouped actions should allow the agent to learn faster, facilitating its ability to learn something meaningful from the limited amount of data available.

We present two approaches to grouping actions that make use of knowing the actions are structured in some manner. The first approach requires an explicit mapping of actions to groups. In other words, the domain knowledge must be so extensive that the structure of the actions is completely known. The second approach requires less domain knowledge; under the assumption that actions are selected intelligently, the action selections in the data can provide guidance with regards to the grouping of the actions. Similar actions (i.e., actions that should belong to the

same group) should be used in similar situations. In this case, explicit domain knowledge regarding the grouping of actions is not needed.

4.3.1 Domain Knowledge Based Grouping (DKBG)

Using their knowledge of SCI rehabilitation, physiotherapists can group treatments together based on characteristics such as orientation (lying, sitting, standing, etc.), level of movement (static vs. dynamic), purpose (priming, impairment management, etc.), and level of independence (assisted vs. independent). To reflect this idea in the simulator, the actions are placed into 11 groups of 10 actions. We represent the physiotherapists grouping SCI treatments using their knowledge by using the known 11 groups of actions. For the remainder of this paper, we will refer to the agent that groups actions in this way as the DKBG agent.

4.3.2 Treatment Embedding Based Grouping (TEBG)

Both time and domain expertise are needed in order to be able to formulate an action grouping system as described in Section 4.3.1. It is possible that it may be infeasible to create such a system. For this reason, we suggest an alternative approach that groups the actions using representation learning. Since the training data is generated through an intelligent behaviour policy (the physiotherapists' decisions), we can assume that the treatments that are commonly selected simultaneously are similar to one another.

Inspired by word embedding, we propose mapping each of the individual treatments to a point in vector space (i.e., treatment embedding). The training data obtained from SCI rehabilitation is structured in a similar way to a corpus used to generate word vectors; each weekly treatment plan is analogous to a text document and the treatments themselves are analogous to the words. Using the package text2vec [134], we generate treatment vectors by

implementing the GloVe [6] modeling framework using 50 epochs and an *x_max* argument (discussed in Section 2.2) of 10. In order to create the co-occurrence matrix needed to train the model, we use an infinite context window since the ordering of treatments within the same treatment plan is arbitrary. An example of some of the co-occurrence pairs generated from a treatment plan are shown below. The bolded treatment is the context treatment and the treatments within the context window (all other treatments) are underlined. Note that a treatment plan is a set, so the order of the treatments is unimportant.

Example Treatment Plan	**Co-occurrence Pairs**
24, 26, 33, 34, 38, 39, 42, 50	(24, 26); (24, 33); (24, 34); (24, 38); (24, 39); (24, 42); (24, 50)
24, **26**, 33, 34, 38, 39, 42, 50	(26, 24); (26, 33); (26, 34); (26, 38); (26, 39); (26, 42); (26, 50)
24, 26, **33**, 34, 38, 39, 42, 50	(33, 24); (33, 26); (33, 34); (33, 38); (33, 39); (33, 42); (33, 50)
24, 26, 33, **34**, 38, 39, 42, 50	(34, 24); (34, 26); (34, 33); (34, 38); (34, 39); (34, 42); (34, 50)

Table 6: An illustration of some of the co-occurrence pairs resulting from an example weekly treatment plan. The context treatment is bolded and treatments within the context window are underlined.

After each treatment is represented as a vector, the treatments can be grouped using a clustering algorithm. In the simulator, each of the action groups is composed of 10 treatments. Since K-means clustering is sensitive to imbalance in the number of treatments in each group, we choose to use this approach. The treatments are grouped into 11 groups (the same number of

groups as created using domain knowledge) using the kmeans function [135] with the number of initial configurations and the maximum number of iterations both set to 1000. For the remainder of this paper, we will refer to the agent that groups actions in this way as the TEBG agent.

4.3.3 Defining State-Action Pairs with Grouped Actions

In order to facilitate an agent learning from the grouped data, the original data needs to be converted to data that indicates the action groups selected in a treatment plan, rather than the individual treatments themselves. For each treatment selected, its action group is stored in the data instead. Even after using action groups rather than the individual treatments, there are still thousands of unique joint actions. Consequently, as done in the baseline approach, we assume that the impact of any given action group is unrelated to any other action group choices made at the same timepoint. An example illustrating a mapping of the original data to the grouped data used to train the agent is shown in Table 7.

Original Data:

Stage	No. of Weeks Remaining	Actions	Reward	Next Stage
3	2	24, 26, 33, 34, 38, 39, 42, 50	0	3

Mapping of Action to Action Group:

Action	Action Group
24	3
26	3
33	4
34	4
38	4
39	4
42	5
50	5

Grouped Data:

Stage	No. of Weeks Remaining	Action Group	Reward	Next Stage
3	2	3	0	3
3	2	3	0	3
3	2	4	0	3
3	2	4	0	3
3	2	4	0	3
3	2	4	0	3
3	2	5	0	3
3	2	5	0	3

Table 7: Table 7a (top) shows a possible row from the simulate data in its original format, Table 7b (middle) shows the mapping of each applicable action to its action group, and Table 7c (bottom) shows the transformation of the original row into eight rows in the adjusted dataset using the grouping of actions.

4.3.4 Estimating the Optimal State-Action Value Function

Like with the baseline approach, we use Fitted Q Iteration to estimate the optimal state-action value function. Again, our requirement for convergence is that none of the coefficients of the regression model change by more than 0.0001 from one iteration to the next. The model is shown below:

$$Q(s, a_g, No.\,of\ Weeks\ Remaining)$$
$$= \beta_0 + \beta_1(No.\,of\ Weeks\ Remaining)$$
$$+ \sum_{actionGroup} \beta_{actionGroup} I(actionGroup = a_g) + \sum_{stage} \beta_{stage} I(stage = s)$$
$$+ \sum_{actionGroup,stage} \beta_{actionGroup,stage} I(actionGroup = a_g) I(stage = s)$$

4.3.5 Testing the Agents

We simulate the rehabilitation process for 1000 patients, choosing their treatments using various selection processes. In order to facilitate meaningful evaluation of the agents' decision-making and their effectiveness in improving the treatment selection for people with an SCI, we select treatments using the four policy definitions below:

1. π_{PT}: Treatments are selected according to their perceived treatment benefit. This treatment selection process is identical to the treatment selection process used to generate the training data and is intended to reflect the approach taken by physiotherapists.
2. π_{Agent}: Treatments are selected using the agent's estimated state-action values. However, the actions represented by this model are the grouped actions, not individual treatments. For this reason, rather than directly using the state-action values to inform treatment

selection, we use an alternate value that is the product of two entities. The first entity is the proportion of selections for each action in the given stage relative to the number of selections for its action group in the given stage. The second entity is the standardized value of each action's action group, computed as follows:

$$\left[Q(s,a_g) - mean_{a_g}(Q(s,a_g))\right] / max_{a_g}\left[Q(s,a_g) - mean_{a_g}(Q(s,a_g))\right]$$

For brevity, let s represent the entire state (i.e., both the stage and the number of weeks remaining). a_g represents an action group. We call the product of these two entities the standardized state-action value contribution (SSAVC) and select the treatments according to this product.

3. π_{Mixed}: Treatments are selected through a combination of the first and second policy definitions, represented by the following expression:

$$perceived\ treatment\ benefit + weight * SSAVC,$$

where $weight$ is a parameter used to alter the value placed on the agent's opinion. We consider integer weights from one to 20.

4. $\pi_{Optimal}$: Treatments are selected according to their actual treatment benefit, which always results in the highest probability of transitioning to the next stage and hence maximizes the return. In practice, choosing treatments in this manner is not possible, but using this process facilitates comparing our other treatment selection processes to the optimal process.

4.3.6 Varying the Training Data

For π_{Agent}, the agents are trained using 1000 patients of training data. Again, we consider 1000 patients of training data a reasonable upper bound on the number of patients because the R2P team, PI, LHRI has approximately 90 patients of data from the few years they have been collecting data.

However, since the process of obtaining training data is ongoing, it is also of interest to see the impact of the size of the training dataset on the usefulness of the treatment selection processes. This is of particular interest for π_{Mixed}, which is how we expect a decision support system would be used in practice. Changing the size of the training set using this policy definition facilitates examining the change in the weight that should be given to the agent as the amount of training data increases. We use training set sizes of 100 patients up to 1000, using increments of 100. When increasing the size of the training set, we do not create an entirely new dataset; instead, we add 100 patients to the previous training set.

In order to assess the variability of these treatment selection processes, we run this entire simulation process 100 times using SHARCNET[2], changing only the training data each time.

4.3.7 Results

Figure 7 shows violin plots for the average return using π_{Agent} with the DKBG agent and the TEBG agent. The black dots represent the mean average return and the horizontal line shows the average return under π_{PT}. For both agents, the distributions are approximately symmetric, so

[2] This research was enabled in part by support provided by Compute Ontario (www.computeontario.ca) and Compute Canada (www.computecanada.ca).

their medians are very similar to their means. On average, the DKBG agent outperforms the physiotherapists and the TEBG agent performs only slightly worse than the physiotherapists.

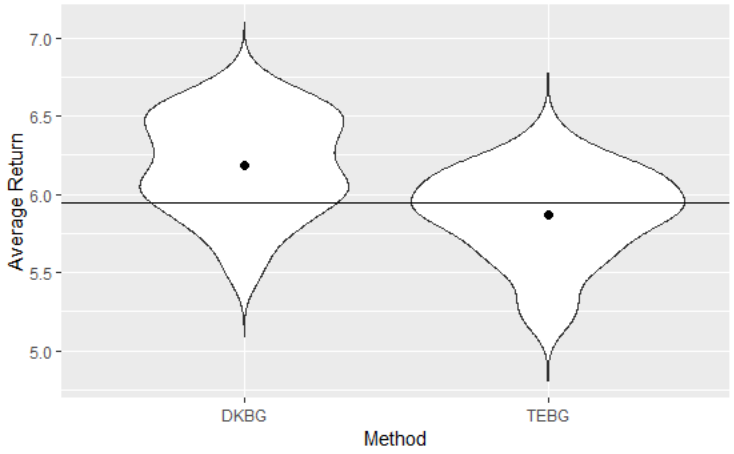

Figure 7: Violin plots of the average return under π_{Agent} using the domain knowledge based grouping (DKBG) agent and the treatment embedding based grouping (TEBG) agent. The horizontal line represents the average return under π_{PT} and the black dots represent the mean average return.

Since the simulator is designed such that a patient can only improve by one stage at a time, the transition probabilities provide another way of assessing the effectiveness of the methods. For each of the 100 simulations, the transition probability in each stage is calculated for the DKBG and TEBG agents' treatment selections. Table 8 shows the average of these transition probabilities for each stage, using a remaining number of weeks of treatment that corresponds to a patient that just began their treatment[3]. Except for Stage Zero and Stage 10, the DKBG agent outperforms the TEBG agent. It's possible that the grouping procedure used in the TEBG approach performs particularly well in the edge stages, but further investigation is needed to determine if this is the cause of the TEBG agent's relatively strong performance for these stages.

[3] It should be noted that the number of weeks remaining has no impact on the transition probability in each stage. Nonetheless, a value must be chosen in order to compute a probability.

The performance of both agents deteriorates as the stage increases, which is expected because there is less training data for these stages. For comparison, the mean transition probability for each stage under π_{PT} is also shown. Note that the physiotherapist treatment selections are designed such that the mean transition probability under π_{PT} is 0.253 for every stage, so the observed differences are due to randomness. The agents both outperform the physiotherapists in general for the earlier stages, but perform worse in the later stages.

Stage	**Mean Transition Probability**		
	Physiotherapist	DKBG	TEBG
0	0.286	0.266	0.340
1	0.220	0.326	0.261
2	0.264	0.377	0.375
3	0.244	0.380	0.265
4	0.246	0.290	0.252
5	0.278	0.300	0.216
6	0.121	0.201	0.148
7	0.341	0.244	0.145
8	0.336	0.233	0.231
9	0.183	0.164	0.118
10	0.203	0.121	0.160

Table 8: Mean transition probabilities for each stage under π_{PT} (physiotherapist) and π_{Agent}, using the domain knowledge based grouping (DKBG) and treatment embedding based grouping (TEBG) agents.

The two surface plots shown in Figure 8 illustrate the mean average return for π_{Mixed}. The left and right surfaces are created using the DKBG and TEBG agents, respectively. These surfaces show the relationship between the weight given to the agent, the number of training episodes, and the average return. The values of these features have been standardized, so only the shape of the surfaces should be considered (i.e., individual points on the left and right surfaces are incomparable). To illustrate this point, at 100 training episodes and a weight of one, the

visualizations make it look like using the DKBG agent is inferior to using the TEBG agent, but this is not the case; their average returns are 6.034 and 6.023, respectively.

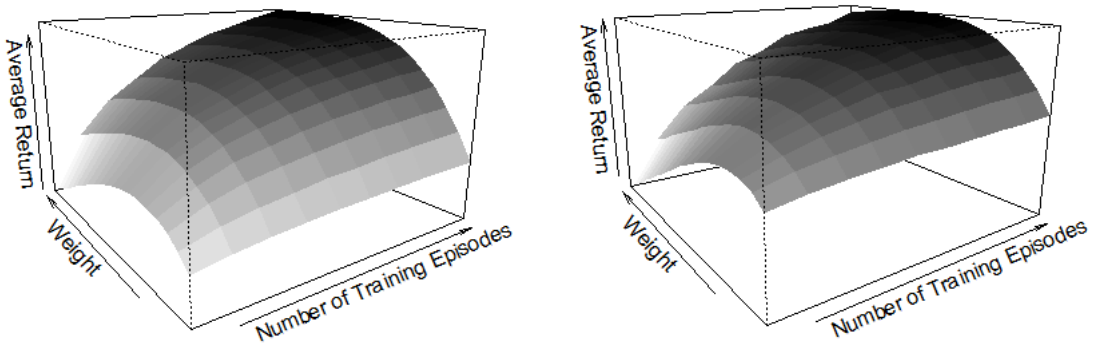

Figure 8: Surface plots of the mean average return obtained under π_{Mixed} with the domain knowledge based grouping (DKBG) (left) agent and treatment embedding based grouping (TEBG) (right) as a function of the number of training episodes used for training and the weight given to the agent. Note that these plots are standardized so they cannot be compared to one another.

For relatively few training episodes, both surfaces have an inverted U-shape, where medium-sized weights perform the best. At 100 training episodes and a weight of 20, using the DKBG and TEBG agents leads to average returns of 5.796 and 5.538, respectively; both agents negatively impact performance in this case, as the physiotherapists alone achieved an average return of 5.949. In at least one case, the fit of the regression model was rank-deficient. Although this is not particularly troubling when we are interested in prediction (as opposed to inference), it does indicate that the amount of data is insufficient. As the number of training episodes increases, both the optimal weight given to the agents and the mean average return increase as well. The maximum mean return using the DKBG and TEBG agent are 6.876 and 6.608 respectively, achieved with weights of 15 and 12 respectively. This indicates that the inverted U-shape shown for relatively few training episodes is still present with 1000 training episodes,

albeit less clearly. In both cases, the maximum mean average return obtained is much larger than the mean average return the agents achieve on their own, shown in Figure 7.

In Figure 9, a cross-section from each of the two surfaces in Figure 8 is shown. Unlike in Figure 8, the two selection processes, one using the DKBG agent and the other using the TEBG agent, can be directly compared. The cross-section shown is the part of the surfaces where the weight is 11. This weight is chosen to strike a balance between reasonable effectiveness with relatively few training episodes and maximizing the use of the agent with relatively many training episodes. The darker shaded areas represent 95% confidence intervals and the lighter shaded areas are the areas within the fifth and 95th quantiles of the simulation results. For reference, the average return achieved under π_{PT} and $\pi_{Optimal}$ are shown. Clearly, the process using the DKBG agent performs better than the approach using the TEBG agent, although both can be used to improve treatment selection. After 1000 training episodes, the DKBG and TEBG approaches reach mean average returns of 6.853 and 6.608 respectively, which corresponds to eliminating 54.1% and 39.4% of the gap between optimal treatment selection and physiotherapist treatment selection.

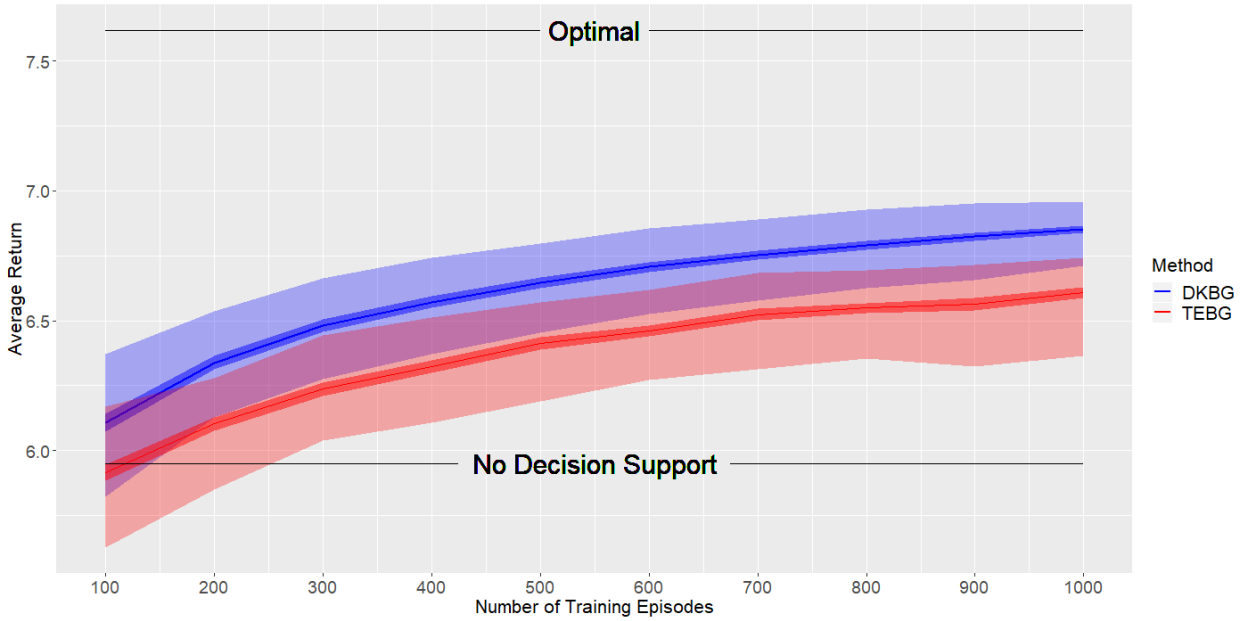

Figure 9: A plot of the mean average return under π_{Mixed} (with a weight of 11) versus the number of episodes used to train the agent for both the domain knowledge based grouping (DKBG) and treatment embedding based grouping (TEBG) agents. The darker shaded areas represent 95% confidence intervals and the lighter shaded areas are the areas within the fifth and 95th quantiles of the simulation results. The horizontal lines show the average return under π_{PT} (no decision support) and $\pi_{Optimal}$.

Chapter 5

5. A Special Case: All Patients Reach Unimpaired Mobility

In general, it is not reasonable to assume that all people with a spinal cord injury (SCI) will return to a state of unimpaired mobility (i.e., full walking capacity). However, there may be some subset of the population where this is a reasonable assumption. If we consider only this subset of the population, then the reward function and return outlined in Section 4.1 will not help the reinforcement learning (RL) agent learn, as the return would be the same after every episode. Instead, negative rewards can be used to encourage faster recovery. With a discount factor of one (i.e., undiscounted), we use a reward of -0.25 for each non-terminal step and a reward of 11 upon reaching unimpaired mobility. In addition to the aforementioned changes to the reward function, a few other changes are made to the process outlined in Chapter 3 and Chapter 4:

- The maximum number of weeks of treatment is not used (in order to facilitate all patients reaching unimpaired mobility).
- The number of training episodes begins at 25 and increases up to 250 with increments of 25. Under the simulated physiotherapists' treatment, the average training episode with all patients reaching unimpaired mobility is roughly four times as long as the average training episode with a maximum number of weeks of treatment.
- The convergence requirement for the Fitted Q Iteration algorithm is that none of the coefficients of the regression model change by more than 0.001 from one iteration to the next (as opposed to 0.0001).

In Figure 10, kernel density estimates are shown for the average return under π_{Agent} using both the domain knowledge based grouping (DKBG) and treatment embedding based

grouping (TEBG) agents. Both methods have very long left tails, indicating that there are cases where the methods perform very poorly. These long left tails are not present for either method in the results shown in Section 4.6.7, but there is an important difference between the two cases; the results shown in Section 4.6.7 are bounded below by 3.93, the mean of the stage initialization distribution, while these results do not have a lower boundary.

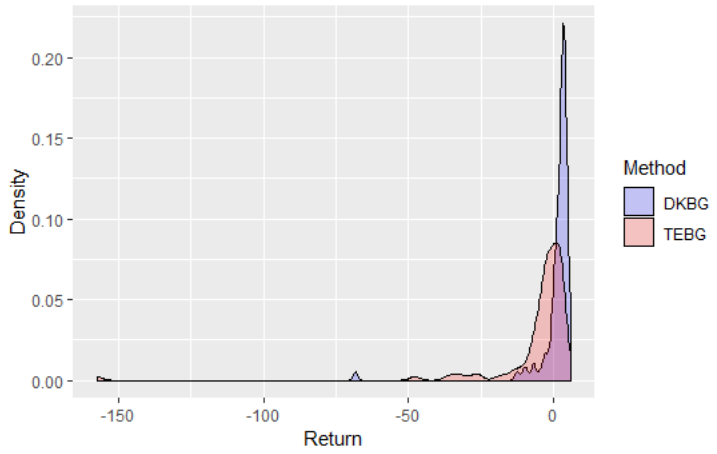

Figure 10: Kernel density estimates of the average return under π_{Agent} with the domain knowledge based grouping (DKBG) and treatment embedding based grouping (TEBG) agents when patients are treated until they reach a fully healthy state.

In the violin plots shown in Figure 11, the worst 10 cases have been removed for both methods. The black and red points represent the mean and median respectively of each method prior to removing the worst 10 cases. The horizontal line represents the average return under π_{PT}. The DKBG agent outperforms the TEBG agent, but both are outperformed by the physiotherapists. Even after the removal of the worst 10 cases, the TEBG agent still exhibits a very long left tail.

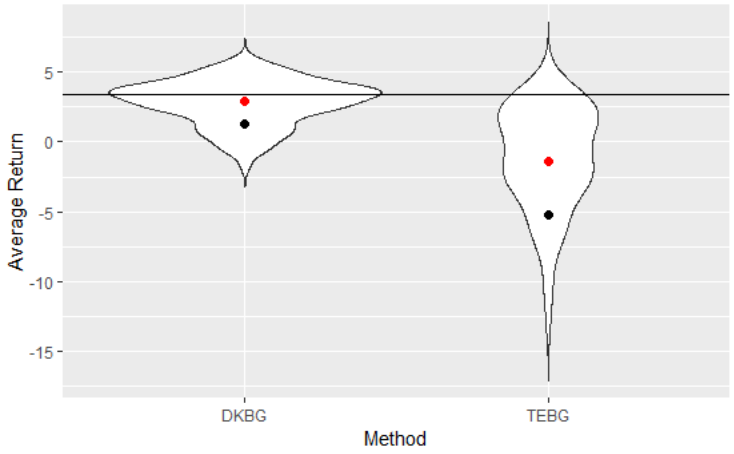

Figure 11: Violin plots of the average return under π_{Agent} for the domain knowledge based grouping (DKBG) agent and the treatment embedding based grouping (TEBG) agent when patients are treated until they reach a fully healthy state. The horizontal line represents the average return under π_{PT} and the black dots represent the mean average return. The 10 worst cased have been removed for both methods.

The methods can be assessed by examining the transition probabilities under their treatment selection. Table 9 shows the average transition probability for each stage under both treatment selection processes, computed by averaging across the individual transition probabilities from each of the 100 simulations. Unlike in the original framework, both agents perform better as the stage increases. Since every patient eventually reaches unimpaired mobility, the agents have more data to learn from in the later stages than the earlier stages, so this is an intuitive result. Like in Table 8, the mean transition probability for each stage under π_{PT} is also shown. Of the eleven stages, the DKBG and TEBG agents outperform the physiotherapists in six and four stages respectively. The DKBG agent's overall performance is dramatically hurt by its performance in Stage Zero.

Stage	Mean Transition Probability		
	Physiotherapist	DKBG	TEBG
0	0.286	0.109	0.141
1	0.220	0.219	0.202
2	0.264	0.289	0.271
3	0.244	0.293	0.193
4	0.246	0.257	0.206
5	0.278	0.274	0.185
6	0.121	0.233	0.173
7	0.341	0.292	0.219
8	0.336	0.287	0.235
9	0.183	0.303	0.224
10	0.203	0.321	0.279

Table 9: Mean transition probabilities for each stage under π_{PT} (physiotherapist) and π_{Agent}, using the domain knowledge based grouping (DKBG) and treatment embedding based grouping (TEBG) agents, when patients are treated until they reach a fully healthy state.

Figure 12 shows the mean average return under π_{Mixed}. The DKBG and TEBG agents are shown on the left and right respectively. These surfaces illustrate the relationship between the weight given to the agent, the number of training episodes, and the average return. Only the shape of these surfaces can be compared, as the values of the features have been standardized. For relatively few training episodes, both surfaces have an inverted U-shape, but the TEBG agent has a much steeper decline in performance as the weight given to the agent increases. With few training episodes, at least one model fit was rank-deficient. As mentioned in Section 4.3.7, even though we are interested in prediction, not inference, this is still troubling because it indicates that the training dataset is not adequately large. As the number of training episodes increases, both the optimal weight given to the agents and the mean average return increase as well.

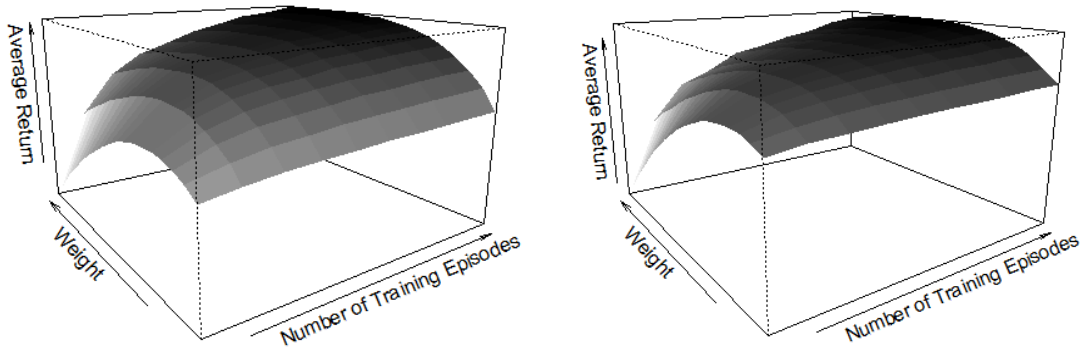

Figure 12: Surface plots of the mean average return obtained using π_{Mixed} with the domain knowledge based grouping (DKBG) (left) agent and treatment embedding based grouping (TEBG) (right) as a function of the number of training episodes used for training and the weight given to the agent. Note that these plots are standardized so they cannot be compared to one another.

A cross-section from each of the surfaces in Figure 12 is shown in Figure 13. The cross-section is the part of the surface where the weight is eight, chosen to balance the performance of the methods with relatively few and relatively many training episodes. The darker shaded areas represent 95% confidence intervals and the lighter shaded areas are the areas within the fifth and 95th quantiles of the simulation results. For reference, the average return achieved under π_{PT} and $\pi_{Optimal}$ are shown. Using the DKBG clearly results in better outcomes compared to using the TEBG agent, but both provide improvement upon the baseline treatment selection. After 250 training episodes, the DKBG and TEBG procedures obtain mean average returns of 5.887 and 5.379 respectively, which corresponds to making up for 66.0% and 52.7% of the gap between optimal treatment selection and physiotherapist treatment selection. Despite some models performing extremely poorly (shown by the extremely long left tails in Figure 10), both methods can be used in conjunction with the knowledge of physiotherapists to improve treatment selection.

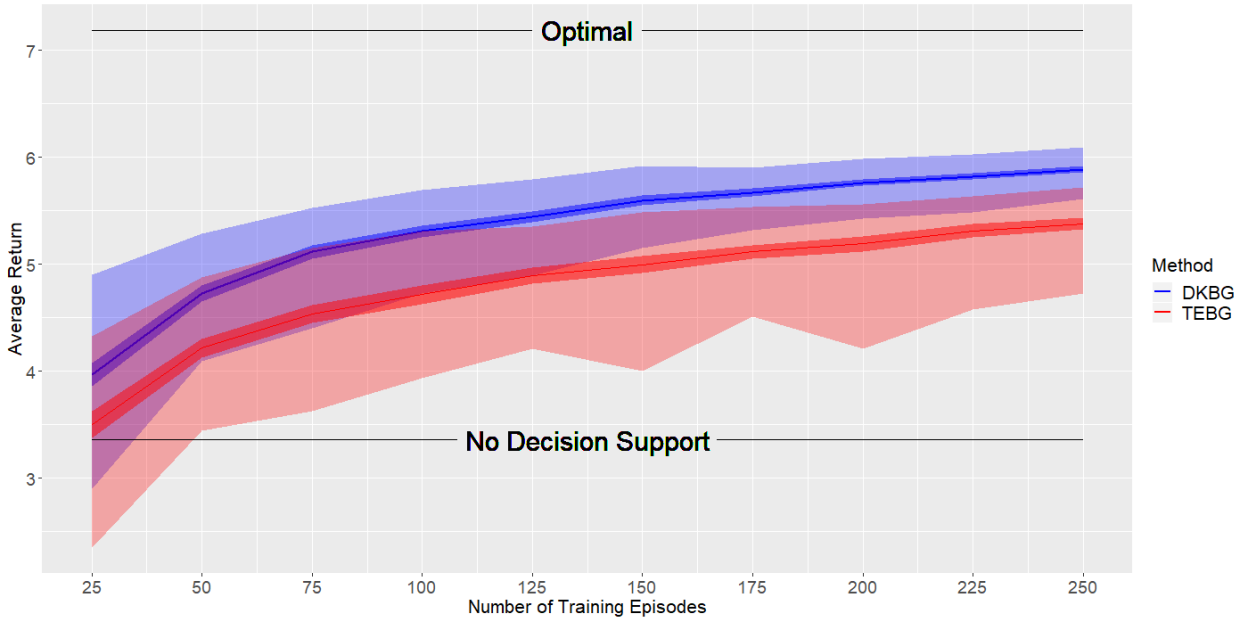

Figure 13: A plot of the mean average return using π_{Mixed} (with a weight of eight) versus the number of episodes used to train the agent for both the domain knowledge based grouping (DKBG) and treatment embedding based grouping (TEBG) agents. The darker shaded areas represent 95% confidence intervals and the lighter shaded areas are the areas within the fifth and 95th quantiles of the simulation results. The horizontal lines show the average return using π_{PT} (no decision support) and $\pi_{Optimal}$.

Chapter 6

6. Discussion

6.1 Interpreting the Results

The very large action space associated with spinal cord injury (SCI) rehabilitation poses a challenge for the effective training of a reinforcement learning (RL) agent, especially given the limited training data available for SCI treatment. As discussed in Section 4.2.3, training an agent that considers each action independently is ineffective, as the variance of the state-action value estimates are too high (resulting in optimistic state-action value estimates) even after 1000 training episodes. However, the results shown in Section 4.3.7 and Chapter 5 show that it is possible for an RL agent to learn something meaningful in this domain by grouping the actions. The results indicate that both agents we train using grouped data, the domain knowledge based grouping (DKBG) agent and the treatment embedding based grouping (TEBG) agent, can be used to augment SCI treatment selection. The DKBG agent clearly outperforms the TEBG agent on average in all cases. Thus, if possible, using domain knowledge to group the actions seems preferable to grouping the actions using representation learning. If this is not practical, the TEBG agent can still be used to improve treatment selection, albeit to a lesser degree.

For both agents, using the agent to independently select treatments (π_{Agent}) is clearly less effective than combining the agent with the domain knowledge of the simulated physiotherapists (π_{Mixed}). In practice, it is expected that the system would be used in this way (i.e., a combination of the knowledge of the physiotherapists and the agent, albeit in a less explicit, numerical format), both because it has been shown to be the most effective in this work and because individual patients will have specific needs or limitations that are not incorporated into their state. For example, if a patient has a broken foot, a standing exercise may not be possible, even

though it may be the optimal treatment for this patient given the state of their SCI. The physiotherapist would know of this patient's limitation, but in its current state, the RL agent would not.

The results in Chapter 5 indicate that the grouping approaches that we propose may be applicable to other forms of rehabilitation as well. In some situations, for example the case of a professional athlete with an injury, it is known that the person will make a full recovery and the goal is simply to minimize the amount of time until that recovery is reached. Sprained or torn ligaments are a common injury for professional athletes that are treated using physiotherapy and, like SCI rehabilitation, may also have a large action space. Another possible application is the rehabilitation of people with an addiction to drugs and/or alcohol.

6.2 Limitations in Interpreting the Results

RL methods are typically used in situations where some actions may be beneficial in the long term, but not the short term. For example, a situation could arise where choosing action a results in an immediate reward (and no further future rewards), but choosing action b results in a larger, delayed reward (through transitioning to a new state where this reward is obtainable). With a discount factor of one, action b is preferable even though it does not yield an immediate reward. RL approaches are designed to account for this delayed reward in order to maximize the long term return. However, in our case, the simulator is structured in such a way that maximizing the long term return is achieved simply by selecting the treatments that maximize the probability of transitioning to the next stage; actions cannot contribute to a good long term outcome without causing a good short term outcome. We assume that in SCI rehabilitation this is actually not the case, but using this simplified framework facilitates evaluating the methods on a stage-by-stage basis, as shown in Table 8 and Table 9. Although the simulator is set up in such a way that

actions that contribute to a good long term outcome also cause a good short term outcome, the RL techniques used to learn from the data do not make use of this information. Thus, setting up the simulator in this way facilitates a useful way of assessing the techniques without giving the techniques an unrealistic advantage.

The RL techniques do not have a known unrealistic advantage in the simulation relative to practice, but the improvement in patient outcomes as a result of using decision support may be different in practice from what we observe in the simulation. Although the simulator is designed to reflect SCI rehabilitation, we focus more on reflecting the *properties* of SCI rehabilitation rather than the *parameters*. The true actual treatment benefits of each treatment are unknown in practice, as is the correlation between the physiotherapists' perceived treatment benefits to the actual treatment benefits and the transition function. All of these impact the improvement in patient outcome that we might see in practice. However, based on the results of using decision support for simulated patients, it is reasonable to conclude that these techniques can be used to improve treatment selection for SCI rehabilitation; the magnitude of this improvement in practice just cannot be estimated using the simulation results.

6.3 Possible Adjustments to the RL Methods

The RL methods that we have developed in this work may need to be adjusted slightly for use in practice. For the TEBG agent, K-means clustering is used to create the groups of actions. Recall that K-means clustering is chosen because all of the action groups in the simulator are the same size and K-means clustering is sensitive to imbalance in the group sizes. However, in practice, the number of actions in each group may vary considerably, so K-means clustering may not be an appropriate choice of clustering algorithm; single linkage clustering, max linkage

clustering, or some other clustering algorithm may be a better choice; see [7] for a discussion on the suitability (or lack of suitability) of these clustering algorithms for various tasks.

There also may be interactions between the actions, which could be present in various forms. It is possible that using a pair of actions simultaneously (in the same treatment plan) may provide more benefit to the patient than the sum of the benefits that the actions provide the patient individually. The opposite effect could also be possible if treatments provide similar types of benefits (i.e., using both treatments is redundant). Interactions between action groups can be incorporated into the model, but this would require adjusting the data in a different way than we have in this study. Also, when selecting actions, it then may be necessary to use a batch updating algorithm akin to the one used by Laber et al. [119]. Another form of interaction that may be present is that some actions may act as primers for other treatments. For example, selecting action a in one week may transition the patient from their current state to another state (without transitioning to a new stage) where selecting action b now provides a great benefit. In this case, the interaction between patients can be represented as part of the state through keeping track of the patient's recent treatment history (e.g., the treatments from the previous week). In order to keep the state space a manageable size, it may be necessary to use a coarse representation of treatment history such as the most common treatment group from the previous week.

Another aspect that could be improved upon is the reward function, which is based on the SCI Standing and Walking Assessment Tool (SWAT) [129]. Musselman et al. [136] studied the validity and use of SWAT, finding that physiotherapists felt that the ordering of the SWAT stages are appropriate, but that the spacing between the stages is inconsistent. Since we assess the methods based on the average return, our assessment of the methods' effectiveness is skewed by the unequal spacing between stages.

6.4 Bayesian Approach

Based on the results, it is clear that an appropriate combination of the physiotherapists' knowledge and an RL agent is more effective than either decision-maker acting independently. However, the appropriateness of the combination is dependent on the weight given to the agent, especially when there are relatively few (e.g., 100) training episodes. The surface plots in Figure 8 and Figure 12 indicate that the weight given to the agents should increase as the agents have more training data to learn from. However, Table 8 and Table 9 show that the performance of the agents is based on the amount of training data available for each state, so the weight given to an agent should be based on a more refined scale than the number of training episodes; it should be based on the number of training observations in the current stage. In practice, assigning a weight to an agent in this manner is non-trivial. Using confidence intervals for the state-action values would provide a way of determining the weight to give to the agent, since tight confidence intervals indicate that the agent is confident in its assessment of that state-action value and thus indicate that we should give a large weight to the agent's opinion. However, Laber et al. [137] show that standard bootstrapping, the typical approach for computing confidence intervals for a value with an unknown distribution, does not yield a reasonable confidence interval for our situation (even ignoring the grouping aspect, which further complicates the problem).

Another interesting aspect to consider for use in practice is balancing the exploration-exploitation trade-off. Although the agent will already be trained using the original dataset generated using only the physiotherapists' treatment selections, the agent can (and should be) continually trained as more data becomes available. The exploration aspect of this problem is unique due to the selection of multiple actions in each time step. The probability of transitioning to the next stage is dependent on all of the actions selected in each week; thus, for an action a in

stage s, the estimated state-action value, $Q(s,a)$, is dependent on the actions selected with a while in s. For this reason, the exploration component may include selecting a subset of actions while consciously choosing not to use another subset of actions.

Bayesian reinforcement learning (BRL) is an RL approach that has two distinct, relevant advantages over other forms of RL; for a detailed review, see [138]. BRL provides a natural way to both incorporate prior knowledge and optimize the exploration-exploitation trade-off. For these reasons, BRL seems like an excellent fit as a decision support tool for SCI treatment selection. However, there are practical limitations to its use in this domain.

BRL incorporates prior knowledge in different ways, depending on whether the algorithm used is a model-based or model-free algorithm. In either case, BRL works by starting with a prior distribution over some entity and updating it based on the experience in order to obtain a posterior distribution, which then becomes the new prior in the next iteration of the algorithm. For model-based and model-free BRL algorithms, we start with a prior distribution of the transition function and state-action value function respectively and update our beliefs as we experience more data. These prior distributions would have to be based on the knowledge of the physiotherapists, but knowing these prior distributions and knowing which treatments are best for each stage of SCI are two very different things; the physiotherapists may not be able to provide reasonable prior distributions. BRL methods can still work with inaccurate initializations of the prior distribution, but this will slow down learning and, due to the limited amount of training data, learning quickly is an important requirement that must be satisfied by the RL method(s) used.

Although BRL seems like an intuitive choice for a problem such as this one that incorporates prior knowledge, it is not clear that BRL is an appropriate choice at this time given the current

states of the domain knowledge in SCI rehabilitation. However, this may be a promising approach for future decision support efforts in SCI rehabilitation. In addition, in practice it may be appropriate to take an implicit Bayesian approach when using the decision support proposed in this work. Bain [139] outlines arguments both for and against the idea that humans think in a Bayesian manner. Although it is by no means proven that humans think in a Bayesian manner, the fact that it is a debated topic indicates that people can, to some degree, think in a manner that seems to be approximately Bayesian. In practice, it may be most appropriate to initially use the decision support system in cases with clinical equipoise (i.e., when the physiotherapists are uncertain of which treatments are most suitable). In such cases, the agent could choose between the candidate treatments considered by the physiotherapist, which implicitly corresponds to applying a very small weight to the agent's opinions. Over time, the physiotherapists could increasingly consider the opinion of the decision support system (i.e., increasing the weight applied to the agent). Intuitively, physiotherapists should behave in this way naturally; as they observe that the treatment selections of the decision support system are effective, they should trust its opinions more and more.

6.5 Working in a Continuous Action Space

In this study, we have used grouped actions to facilitate learning from limited data in a domain with a large action space. An alternative approach would be to work in a continuous action space. Rather than grouping the treatments after creating the treatment embeddings, we could work in the continuous action space, learning the state-action values and performing action selection in a fashion similar to the one proposed by Dulac-Arnold, Evans, et al. [120].

A second approach that uses the continuous action space may address the multi-action component of this problem. In this approach, the treatment groups would be formed exactly as

done in this work. From there, each row of the original data (which has multiple treatments/actions) is mapped to a new format that represents how many times an action from each group was chosen. In vector space, each group can be represented by a point, the centroid of the vector representations of the actions in the group. Using a weighted function (based on the number of times each group was selected) of these centroids, each joint action (now based on action groups) can be mapped to a point in vector space. Then, an approach similar to the one proposed by Dulac-Arnold, Evans, et al. [120] can be used to learn about these joint actions in the continuous space. However, this approach may require more data than the approaches outlined in this work in order to be successful. In addition, this approach would not select individual treatments, so it may need to be combined with another treatment selection methodology (possibly the one used in this work).

Chapter 7

7. Conclusion

The main contributions of this thesis are 1) new methodology for reinforcement learning (RL) in large action spaces with prior information and 2) a simulator that mimics the properties of data obtained from spinal cord injury (SCI) rehabilitation. We have proposed two methods that group the actions using prior information. Domain knowledge based grouping (DKBG) explicitly uses domain knowledge to group the actions, while treatment embedding based grouping (TEBG) makes use of the intelligent treatment (action) selections made by the physiotherapists when treating a patient, grouping the treatments based on the frequency with which treatments are used together. We train two RL agents, each using one of the grouping approaches, and we find that when used in conjunction with the prior knowledge of the physiotherapists, both are able to augment treatment selection for simulated SCI patients. Although both techniques improved treatment selection, the DKBG agent consistently outperformed the TEBG agent. However, this approach requires much more time and domain knowledge than the TEBG approach.

This work is only a first step in using RL for physiotherapy, and in particular SCI rehabilitation. Developing techniques that more elegantly handle the multi-action selection component of this problem may lead to further improvements in performance. In the coming years, the data obtained from the study at Parkwood Institute (PI) should be used to develop a decision support system that can be used in practice. This will come with its own challenges not present in this study; physiotherapy data is notoriously messy and incomplete and consequently will require substantial preprocessing before it can be used. Once this decision support system is developed, a considerable challenge will be weighing the knowledge of the physiotherapists and the agent in an optimal (or near optimal) manner for treatment selection. Of particular interest is

the element of exploration, since RL agents continue to learn through exploring actions currently believed to be non-optimal through incorporating an element of randomness in their action selection. However, in the domain of SCI rehabilitation, it may be costly to use certain treatments in various stages and possibly an intolerable risk in some cases, thus a random component that facilitates selection of any treatment may be unacceptable. Formalizing the balance between the desired treatment choices of the physiotherapists and the agent (especially when it comes to exploring treatments) in a safe, effective manner will require further collaboration between physiotherapists and computer scientists.

References

[1] Silver, D., Huang, A., Maddison, C. *et al.* Mastering the game of Go with deep neural networks and tree search. *Nature* **529,** 484–489 (2016). https://doi.org/10.1038/nature16961

[2] Reynolds, Matt. (2017, May 23). DeepMind's AI beats world's best Go player in latest face-off. https://www.newscientist.com/article/2132086-deepminds-ai-beats-worlds-best-go-player-in-latest-face-off/#ixzz6GJ6xuLeC

[3] Sutton, R. S., Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. The MIT Press.

[4] Lange S., Gabel T., Riedmiller M. (2012) Batch Reinforcement Learning. In: Wiering M., van Otterlo M. (eds) Reinforcement Learning. Adaptation, Learning, and Optimization, vol 12. Springer, Berlin, Heidelberg

[5] Ernst, D., Geurts, P., and Wehenkel, L. (2005). Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, *6*(Apr), 503-556.

[6] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. (2014). GloVe: Global Vectors for Word Representation.

[7] Ben-David, S. (2018, April). Clustering-what both theoreticians and practitioners are doing wrong. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[8] Yu, C., Liu, J., and Nemati, S. (2019). Reinforcement learning in healthcare: a survey. *arXiv preprint arXiv:1908.08796*.

[9] I. Ahn and J. Park, "Drug scheduling of cancer chemotherapy based on natural actor-critic approach," BioSystems, vol. 106, no. 2-3, pp. 121–129, 2011.

[10] R. Akrour, M. Schoenauer, and M. Sebag, “April: Active preference learning-based reinforcement learning,” in Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, 2012, pp. 116–131.

[11] R. Busa-Fekete, B. Szörényi, P. Weng, W. Cheng, and E. Hüllermeier, ¨ “Preference-based reinforcement learning: evolutionary direct policy search using a preference-based racing algorithm,” Machine Learning, vol. 97, no. 3, pp. 327–351, 2014.

[12] W. Cheng, J. Furnkranz, E. Hüllermeier, and S.-H. Park, “Preference- ¨ based policy iteration: Leveraging preference learning for reinforcement learning,” in Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, 2011, pp. 312– 327.

[13] J. Furnkranz, E. Hüllermeier, W. Cheng, and S.-H. Park, “Preference- ¨ based reinforcement learning: a formal framework and a policy iteration algorithm,” Machine Learning, vol. 89, no. 1-2, pp. 123–156, 2012.

[14] Y. Goldberg and M. R. Kosorok, “Q-learning with censored data,” Annals of Statistics, vol. 40, no. 1, p. 529, 2012.

[15] A. Hassani et al., “Reinforcement learning based control of tumor growth with chemotherapy,” in 2010 International Conference on System Science and Engineering (ICSSE). IEEE, 2010, pp. 185–189.

[16] K. Humphrey, “Using reinforcement learning to personalize dosing strategies in a simulated cancer trial with high dimensional data,” MS Thesis, The University of Arizona Repository, 2017.

[17] A. Jalalimanesh, H. S. Haghighi, A. Ahmadi, and M. Soltani, “Simulation-based optimization of radiotherapy: Agent-based modeling and reinforcement learning,” Mathematics and Computers in Simulation, vol. 133, pp. 235–248, 2017.

[18] A. Jalalimanesh, H. S. Haghighi, A. Ahmadi, H. Hejazian, and M. Soltani, “Multi-objective optimization of radiotherapy: distributed q-learning and agent-based simulation,” Journal of Experimental & Theoretical Artificial Intelligence, pp. 1–16, 2017.

[19] R. Padmanabhan, N. Meskin, and W. M. Haddad, “Reinforcement learning-based control of drug dosing for cancer chemotherapy treatment,” Mathematical biosciences, vol. 293, pp. 11–20, 2017.

[20] Y. M. Soliman, “Personalized medical treatments using novel reinforcement learning algorithms,” arXiv preprint arXiv:1406.3922, 2014.

[21] H. H. Tseng, Y. Luo, S. Cui, J. T. Chien, R. K. Ten Haken, and I. E. Naqa, “Deep reinforcement learning for automated radiation adaptation in lung cancer,” Medical Physics, vol. 44, no. 12, pp. 6690–6705, 2017.

[22] R. Vincent, “Reinforcement learning in models of adaptive medical treatment strategies,” Ph.D. dissertation, McGill University Libraries, 2014.

[23] G. Yauney and P. Shah, “Reinforcement learning with action-derived rewards for chemotherapy and clinical trial dosing regimen selection,” in Machine Learning for Healthcare Conference, 2018, pp. 161–226.

[24] Y. Zhao, M. R. Kosorok, and D. Zeng, “Reinforcement learning design for cancer clinical trials,” Statistics in Medicine, vol. 28, no. 26, pp. 3294–3315, 2009.

[25] Y. Zhao, D. Zeng, M. A. Socinski, and M. R. Kosorok, “Reinforcement learning strategies for clinical trials in nonsmall cell lung cancer,” Biometrics, vol. 67, no. 4, pp. 1422–1433, 2011.

[26] H. Asoh, M. Shiro, S. Akaho, T. Kamishima, K. Hashida, E. Aramaki, and T. Kohro, “Modeling medical records of diabetes using markov decision processes,” in Proceedings of ICML2013 Workshop on Role of Machine Learning in Transforming Healthcare, 2013.

[27] H. Asoh, M. S. S. Akaho, T. Kamishima, K. Hasida, E. Aramaki, and T. Kohro, “An application of inverse reinforcement learning to medical records of diabetes treatment,” in ECMLPKDD2013 Workshop on Reinforcement Learning with Generalized Feedback, 2013.

[28] M. K. Bothe, L. Dickens, K. Reichel, A. Tellmann, B. Ellger, M. Westphal, and A. A. Faisal, “The use of reinforcement learning algorithms to meet the challenges of an artificial pancreas,” Expert Review of Medical Devices, vol. 10, no. 5, pp. 661–673, 2013.

[29] E. Daskalaki, L. Scarnato, P. Diem, and S. G. Mougiakakou, “Preliminary results of a novel approach for glucose regulation using an actor-critic learning based controller,” UKACC International Conference on Control, 2010.

[30] E. Daskalaki, P. Diem, and S. G. Mougiakakou, “An actor–critic based controller for glucose regulation in type 1 diabetes,” Computer Methods and Programs in Biomedicine, vol. 109, no. 2, pp. 116–125, 2013.

[31] E. Daskalaki, P. Diem, and S. G. Mougiakakou,, “Personalized tuning of a reinforcement learning control algorithm for glucose regulation,” in 2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). IEEE, 2013, pp. 3487–3490.

[32] E. Daskalaki, P. Diem, and S. G. Mougiakakou,, “Model-free machine learning in biomedicine: Feasibility study in type 1 diabetes,” PloS One, vol. 11, no. 7, p. e0158722, 2016.

[33] Luckett, D. J., Laber, E. B., Kahkoska, A. R., Maahs, D. M., Mayer-Davis, E., & Kosorok, M. R. (2019). Estimating dynamic treatment regimes in mobile health using v-learning. *Journal of the American Statistical Association*, *115*(530), 692-706.

[34] P. D. Ngo, S. Wei, A. Holubova, J. Muzik, and F. Godtliebsen, “Reinforcement-learning optimal control for type-1 diabetes,” in 2018 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI). IEEE, 2018, pp. 333–336.

[35] Ngo, P. D., Wei, S., Holubová, A., Muzik, J., and Godtliebsen, F. (2018). Control of blood glucose for type-1 diabetes by using reinforcement learning with feedforward algorithm. *Computational and mathematical methods in medicine*, *2018*.

[36] A. Noori, M. A. Sadrnia et al., “Glucose level control using temporal difference methods,” in 2017 Iranian Conference on Electrical Engineering (ICEE). IEEE, 2017, pp. 895–900.

[37] M. De Paula, L. O. Avila, and E. C. Martinez, “Controlling blood glucose variability under uncertainty using reinforcement learning and gaussian processes,” Applied Soft Computing, vol. 35, pp. 310–332, 2015.

[38] M. De Paula, G. G. Acosta, and E. C. Martinez, “On-line policy learning and adaptation for real-time personalization of an artificial pancreas,” Expert Systems with Applications, vol. 42, no. 4, pp. 2234– 2255, 2015.

[39] Sun, Q., Jankovic, M. V., Budzinski, J., Moore, B., Diem, P., Stettler, C., & Mougiakakou, S. G. (2018). A dual mode adaptive basal-bolus advisor based on reinforcement learning. *IEEE journal of biomedical and health informatics*, *23*(6), 2633-2641.

[40] S. Yasini, M. B. Naghibi Sistani, and A. Karimpour, “Agent-based simulation for blood glucose,” International Journal of Applied Science, Engineering and Technology, vol. 5, pp. 89–95, 2009.

[41] P. Escandell-Montero, J. M. Martinez-Martinez, J. D. Martin-Guerrero, E. Soria-Olivas, J. Vila-Frances, and R. Magdalena-Benedito, “Adaptive ´ treatment of anemia on hemodialysis patients: A reinforcement learning approach,” in CIDM2011. IEEE, 2011, pp. 44–49.

[42] P. Escandell-Montero, M. Chermisi, J. M. Martinez-Martinez, J. Gomez-Sanchis, C. Barbieri, E. Soria-Olivas, F. Mari, J. VilaFrances, A. Stopper, E. Gatti et al., “Optimization of anemia treatment in hemodialysis patients via reinforcement learning,” Artificial Intelligence in Medicine, vol. 62, no. 1, pp. 47–60, 2014.

[43] A. E. Gaweda, M. K. Muezzinoglu, G. R. Aronoff, A. A. Jacobs, J. M. Zurada, and M. E. Brier, “Incorporating prior knowledge into q-learning for drug delivery individualization,” in Fourth International Conference on Machine Learning and Applications. IEEE, 2005.

[44] A. E. Gaweda, M. K. Muezzinoglu, G. R. Aronoff, A. A. Jacobs, J. M. Zurada, and M. E. Brier, “Using clinical information in goal-oriented learning,” IEEE Engineering in Medicine and Biology Magazine, vol. 26, no. 2, p. 27, 2007.

[45] A. E. Gaweda, “Improving management of anemia in end stage renal disease using reinforcement learning,” in IJCNN 2009. IEEE, 2009, pp. 953–958.

[46] A. E. Gaweda, M. K. Muezzinoglu, G. R. Aronoff, A. A. Jacobs, J. M. Zurada, and M. E. Brier, “Reinforcement learning approach to individualization of chronic pharmacotherapy,” in IJCNN’05, vol. 5. IEEE, 2005, pp. 3290–3295.

[47] A. E. Gaweda, M. K. Muezzinoglu, A. A. Jacobs, G. R. Aronoff, and M. E. Brier, “Model predictive control with reinforcement learning for drug delivery in renal anemia management,” in IEEE EMBS’06. IEEE, 2006, pp. 5177–5180.

[48] A. E. Gaweda, M. K. Muezzinoglu, G. R. Aronoff, A. A. Jacobs, J. M. Zurada, and M. E. Brier, “Individualization of pharmacological anemia management using reinforcement learning,” Neural Networks, vol. 18, no. 5-6, pp. 826–834, 2005.

[49] J. M. Malof and A. E. Gaweda, “Optimizing drug therapy with reinforcement learning: The case of anemia management,” in Neural Networks (IJCNN), The 2011 International Joint Conference on. IEEE, 2011, pp. 2088–2092.

[50] J. D. Martin-Guerrero, F. Gomez, E. Soria-Olivas, J. Schmidhuber, M. Climente-Marti, and N. V. Jimenez-Torres, “A reinforcement learn- ´ ing approach for individualizing erythropoietin dosages in hemodialysis patients,” Expert Systems with Applications, vol. 36, no. 6, pp. 9737– 9742, 2009.

[51] J. D. Martin-Guerrero, E. Soria-Olivas, M. Martinez-Sober, M. Climente-Marti, T. De Diego-Santos, and N. V. Jimenez-Torres, “Validation of a reinforcement learning policy for dosage optimization of erythropoietin,” in Australasian Joint Conference on Artificial Intelligence. Springer, 2007, pp. 732–738.

[52] B. M. Adams, H. T. Banks, H.-D. Kwon, and H. T. Tran, “Dynamic multidrug therapies for hiv: Optimal and sti control approaches,” Mathematical Biosciences and Engineering, vol. 1, no. 2, pp. 223–241, 2004.

[53] D. Ernst, G.-B. Stan, J. Goncalves, and L. Wehenkel, “Clinical data based optimal sti strategies for hiv: a reinforcement learning approach,” in 45th IEEE Conference on Decision and Control. IEEE, 2006, pp. 667–672.

[54] K. Kawaguchi, “Bounded optimal exploration in mdp.” in AAAI, 2016, pp. 1758–1764.

[55] T. W. Killian, S. Daulton, G. Konidaris, and F. Doshi-Velez, “Robust and efficient transfer learning with hidden parameter markov decision processes,” in Advances in Neural Information Processing Systems, 2017, pp. 6250–6261.

[56] V. N. Marivate, J. Chemali, E. Brunskill, and M. L. Littman, “Quantifying uncertainty in batch personalized sequential decision making.” in AAAI Workshop: Modern Artificial Intelligence for Health Analytics, 2014

[57] S. Parbhoo, “A reinforcement learning design for hiv clinical trials,” Ph.D. dissertation, 2014.

[58] S. Parbhoo, J. Bogojeska, M. Zazzi, V. Roth, and F. Doshi-Velez, “Combining kernel and model based learning for hiv therapy selection,” AMIA Summits on Translational Science Proceedings, vol. 2017, p. 239, 2017.

[59] J. Pazis and R. Parr, “PAC optimal exploration in continuous space markov decision processes.” in AAAI, 2013.

[60] Yao, J., Killian, T., Konidaris, G., & Doshi-Velez, F. (2018). Direct policy transfer via hidden parameter markov decision processes. In *LLARLA Workshop, FAIM* (Vol. 2018).

[61] C. Yu, Y. Dong, J. Liu, and G. Ren, “Incorporating causal factors into reinforcement learning for dynamic treatment regimes in hiv,” BMC medical informatics and decision making, vol. 19, no. 2, p. 60, 2019.

[62] K. Bush and J. Pineau, “Manifold embeddings for model-based reinforcement learning under partial observability,” in Advances in Neural Information Processing Systems, 2009, pp. 189–197.

[63] Butler, E. L., Laber, E. B., Davis, S. M., & Kosorok, M. R. (2018). Incorporating patient preferences into estimation of optimal individualized treatment rules. *Biometrics*, *74*(1), 18-26.

[64] B. Chakraborty, E. B. Laber, and Y. Zhao, “Inference for optimal dynamic treatment regimes using an adaptive m-out-of-n bootstrap scheme,” Biometrics, vol. 69, no. 3, pp. 714–723, 2013.

[65] B. Chakraborty, S. Murphy, and V. Strecher, “Inference for non-regular parameters in optimal dynamic treatment regimes,” Statistical Methods in Medical Research, vol. 19, no. 3, pp. 317–343, 2010.

[66] B. Chakraborty, V. Strecher, and S. Murphy, “Bias correction and confidence intervals for fitted q-iteration,” in Workshop on Model Uncertainty and Risk in Reinforcement Learning, NIPS, Whistler, Canada. Citeseer, 2008.

[67] B. Chakraborty and E. E. M. Moodie, Statistical Reinforcement Learning. Springer New York, 2013.

[68] K. Deng, R. Greiner, and S. Murphy, “Budgeted learning for developing personalized treatment,” in ICMLA2014. IEEE, 2014, pp. 7–14.

[69] A. Ertefaie, S. Shortreed, and B. Chakraborty, “Q-learning residual analysis: application to the effectiveness of sequences of antipsychotic medications for patients with schizophrenia,” Statistics in Medicine, vol. 35, no. 13, pp. 2221–2234, 2016.

[70] A. Guez, “Adaptive control of epileptic seizures using reinforcement learning,” Ph.D. dissertation, McGill University Library, 2010.

[71] A. Guez, R. D. Vincent, M. Avoli, and J. Pineau, “Adaptive treatment of epilepsy via batch-mode reinforcement learning.” in AAAI, 2008, pp. 1671–1678.

[72] E. B. Laber, K. A. Linn, and L. A. Stefanski, “Interactive model building for q-learning,” Biometrika, vol. 101, no. 4, pp. 831–847, 2014.

[73] E. B. Laber, D. J. Lizotte, and B. Ferguson, “Set-valued dynamic treatment regimes for competing outcomes,” Biometrics, vol. 70, no. 1, pp. 53–61, 2014.

[74] D. J. Lizotte, M. Bowling, and S. A. Murphy, “Linear fitted-q iteration with multiple reward functions,” Journal of Machine Learning Research, vol. 13, no. Nov, pp. 3253–3295, 2012.

[75] D. J. Lizotte and E. B. Laber, “Multi-objective markov decision processes for data-driven decision support,” The Journal of Machine Learning Research, vol. 17, no. 1, pp. 7378–7405, 2016.

[76] S. A. Murphy, “Optimal dynamic treatment regimes,” Journal of the Royal Statistical Society: Series B (Statistical Methodology), vol. 65, no. 2, pp. 331–355, 2003.

[77] S. A. Murphy, Y. Deng, E. B. Laber, H. R. Maei, R. S. Sutton, and K. Witkiewitz, “A batch, off-policy, actor-critic algorithm for optimizing the average reward,” arXiv preprint arXiv:1607.05047, 2016.

[78] V. Nagaraj, A. Lamperski, and T. I. Netoff, “Seizure control in a computational model using a reinforcement learning stimulation paradigm,” International Journal of Neural Systems, vol. 27, no. 07, p. 1750012, 2017.

[79] D. Ormoneit and S. Sen, “Kernel-based reinforcement learning,” ´ Machine learning, vol. 49, no. 2-3, pp. 161–178, 2002.

[80] G. Panuccio, A. Guez, R. Vincent, M. Avoli, and J. Pineau, “Adaptive control of epileptiform excitability in an in vitro model of limbic seizures,” Experimental Neurology, vol. 241, pp. 179–183, 2013.

[81] J. Pineau, A. Guez, R. Vincent, G. Panuccio, and M. Avoli, “Treating epilepsy via adaptive neurostimulation: a reinforcement learning approach,” International Journal of Neural Systems, vol. 19, no. 04, pp. 227–240, 2009.

[82] J. Pineau, M. G. Bellemare, A. J. Rush, A. Ghizaru, and S. A. Murphy, “Constructing evidence-based treatment strategies using methods from computer science,” Drug & Alcohol Dependence, vol. 88, pp. S52–S60, 2007.

[83] P. J. Schulte, A. A. Tsiatis, E. B. Laber, and M. Davidian, “Q-and alearning methods for estimating optimal dynamic treatment regimes,” Statistical science: a review journal of the Institute of Mathematical Statistics, vol. 29, no. 4, p. 640, 2014.

[84] S. M. Shortreed, E. Laber, D. J. Lizotte, T. S. Stroup, J. Pineau, and S. A. Murphy, “Informing sequential clinical decision-making through reinforcement learning: an empirical study,” Machine Learning, vol. 84, no. 1-2, pp. 109–136, 2011.

[85] R. Song, W. Wang, D. Zeng, and M. R. Kosorok, “Penalized q-learning for dynamic treatment regimens,” Statistica Sinica, vol. 25, no. 3, p. 901, 2015.

[86] Y. Tao, L. Wang, D. Almirall et al., “Tree-based reinforcement learning for estimating optimal dynamic treatment regimes,” The Annals of Applied Statistics, vol. 12, no. 3, pp. 1914–1938, 2018.

[87] K. A. Linn, E. B. Laber, and L. A. Stefanski, “Interactive q-learning for quantiles,” Journal of the American Statistical Association, vol. 112, no. 518, pp. 638–649, 2017.

[88] E. C. Borera, B. L. Moore, A. G. Doufas, and L. D. Pyeatt, “An adaptive neural network filter for improved patient state estimation in closedloop anesthesia control,” in IEEE ICTAI’11. IEEE, 2011, pp. 41–46.

[89] C.-H. Chang, M. Mai, and A. Goldenberg, “Dynamic measurement scheduling for adverse event forecasting using deep rl,” arXiv preprint arXiv:1812.00268, 2018.

[90] L.-F. Cheng, N. Prasad, and B. E. Engelhardt, “An optimal policy for patient laboratory tests in intensive care units,” arXiv preprint arXiv:1808.04679, 2018.

[91] Futoma, J., Lin, A., Sendak, M., Bedoya, A., Clement, M., O’Brien, C., & Heller, K. Learning to treat sepsis with multi-output gaussian process deep recurrent q-networks, 2018. In *URL https://openreview. net/forum*.

[92] P. Humbert, J. Audiffren, C. Dubost, and L. Oudre, “Learning from an expert.”, In “30th Conference on Neural Information Processing Systems (NIPS)”, 2016.

[93] A. Jagannatha, P. Thomas, and H. Yu, “Towards high confidence offpolicy reinforcement learning for clinical applications.”, CausalML Workshop, ICML, 2018.

[94] M. Komorowski, A. Gordon, L. Celi, and A. Faisal, “A markov decision process to suggest optimal treatment of severe infections in intensive care,” in Neural Information Processing Systems Workshop on Machine Learning for Health, 2016.

[95] E. F. Krakow, M. Hemmer, T. Wang, B. Logan, M. Arora, S. Spellman, D. Couriel, A. Alousi, J. Pidala, M. Last et al., “Tools for the precision medicine era: How to develop highly personalized treatment recommendations from cohort and registry data using q-learning,” American journal of epidemiology, vol. 186, no. 2, pp. 160–172, 2017.

[96] L. Li, M. Komorowski, and A. A. Faisal, “The actor search tree critic (astc) for off-policy pomdp learning in medical decision making,” arXiv preprint arXiv:1805.11548, 2018.

[97] R. Lin, M. D. Stanley, M. M. Ghassemi, and S. Nemati, “A deep deterministic policy gradient approach to medication dosing and surveillance in the ICU,” in IEEE EMBC’18. IEEE, 2018, pp. 4927–4931.

[98] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, “A survey of deep neural network architectures and their applications,” Neurocomputing, vol. 234, pp. 11–26, 2017.

[99] C. Lowery and A. A. Faisal, “Towards efficient, personalized anesthesia using continuous reinforcement learning for propofol infusion control,” in IEEE/EMBS NER’13. IEEE, 2013, pp. 1414–1417.

[100] B. L. Moore, E. D. Sinzinger, T. M. Quasny, and L. D. Pyeatt, “Intelligent control of closed-loop sedation in simulated icu patients.” in FLAIRS Conference, 2004, pp. 109–114.

[101] B. L. Moore, A. G. Doufas, and L. D. Pyeatt, “Reinforcement learning: a novel method for optimal control of propofol-induced hypnosis,” Anesthesia & Analgesia, vol. 112, no. 2, pp. 360–367, 2011.

[102] B. L. Moore, T. M. Quasny, and A. G. Doufas, “Reinforcement learning versus proportional–integral–derivative control of hypnosis in a simulated intraoperative patient,” Anesthesia & Analgesia, vol. 112, no. 2, pp. 350–359, 2011.

[103] B. L. Moore, L. D. Pyeatt, V. Kulkarni, P. Panousis, K. Padrez, and A. G. Doufas, “Reinforcement learning for closed-loop propofol anesthesia: a study in human volunteers,” The Journal of Machine Learning Research, vol. 15, no. 1, pp. 655–696, 2014.

[104] B. L. Moore, P. Panousis, V. Kulkarni, L. D. Pyeatt, and A. G. Doufas, “Reinforcement learning for closed-loop propofol anesthesia: A human volunteer study.” in IAAI, 2010.

[105] S. Nemati, M. M. Ghassemi, and G. D. Clifford, “Optimal medication dosing from suboptimal clinical examples: A deep reinforcement learning approach,” in IEEE 38th Annual International Conference of the Engineering in Medicine and Biology Society. IEEE, 2016, pp. 2978–2981.

[106] R. Padmanabhan, N. Meskin, and W. M. Haddad, “Closed-loop control of anesthesia and mean arterial pressure using reinforcement learning,” Biomedical Signal Processing and Control, vol. 22, pp. 54–64, 2015.

[107] X. Peng, Y. Ding, D. Wihl, O. Gottesman, M. Komorowski, L.-w. H. Lehman, A. Ross, A. Faisal, and F. Doshi-Velez, “Improving sepsis treatment strategies by combining deep and kernel-based reinforcement learning,” arXiv preprint arXiv:1901.04670, 2019.

[108] B. K. Petersen, J. Yang, W. S. Grathwohl, C. Cockrell, C. Santiago, G. An, and D. M. Faissol, “Precision medicine as a control problem: Using simulation and deep reinforcement learning to discover adaptive, personalized multi-cytokine therapy for sepsis,” arXiv preprint arXiv:1802.10440, 2018.

[109] N. Prasad, L.-F. Cheng, C. Chivers, M. Draugelis, and B. E. Engelhardt, “A reinforcement learning approach to weaning of mechanical ventilation in intensive care units,” arXiv preprint arXiv:1704.06300, 2017

[110] A. Raghu, M. Komorowski, I. Ahmed, L. Celi, P. Szolovits, and M. Ghassemi, “Deep reinforcement learning for sepsis treatment,” arXiv preprint arXiv:1711.09602, 2017.

[111] A. Raghu, M. Komorowski, L. A. Celi, P. Szolovits, and M. Ghassemi, “Continuous state-space models for optimal sepsis treatment: a deep reinforcement learning approach,” in Machine Learning for Healthcare Conference, 2017, pp. 147–163.

[112] N. Sadati, A. Aflaki, and M. Jahed, “Multivariable anesthesia control using reinforcement learning,” in IEEE SMC’06, vol. 6. IEEE, 2006, pp. 4563–4568.

[113] E. D. Sinzinger and B. Moore, “Sedation of simulated ICU patients using reinforcement learning based control,” International Journal on Artificial Intelligence Tools, vol. 14, no. 01n02, pp. 137–156, 2005.

[114] C. P. Utomo, X. Li, and W. Chen, “Treatment recommendation in critical care: A scalable and interpretable approach in partially observable health states,” 2018.

[115] L. Wang, W. Zhang, X. He, and H. Zha, “Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation,” in Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. ACM, 2018, pp. 2447–2456.

[116] W.-H. Weng, M. Gao, Z. He, S. Yan, and P. Szolovits, “Representation and reinforcement learning for personalized glycemic control in septic patients,” arXiv preprint arXiv:1712.00654, 2017.

[117] Liu, D. R., Li, H. L., and Wang, D. (2015). Feature selection and feature learning for high-dimensional batch reinforcement learning: A survey. *International Journal of Automation and Computing*, *12*(3), 229-242.

[118] Pazis, J. and Parr, R. (2011). Generalized value functions for large action sets. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)* (pp. 1185-1192).

[119] Laber, E. B., Meyer, N. J., Reich, B. J., Pacifici, K., Collazo, J. A., and Drake, J. M. (2018). Optimal treatment allocations in space and time for on-line control of an emerging infectious disease. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, *67*(4), 743-789.

[120] Dulac-Arnold, G., Evans, R., van Hasselt, H., Sunehag, P., Lillicrap, T., Hunt, J., ... and Coppin, B. (2015). Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*.

[121] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., ... and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

[122] Dietterich, T. G. (2000). Hierarchical reinforcement learning with the MAXQ value function decomposition. Journal of Artificial Intelligence Research, 13, 227–303.

[123] Parr, R. and Russell, S. (1998). Reinforcement learning with hierarchies of machines. NIPS-98 (pp. 1043–1049). Cambridge, MA: MIT Press.

[124] Sutton, R. S., Precup, D., and Singh, S. (1999). Between MDPs and Semi-MDPs: A framework for temporal abstraction in reinforcement learning. Artificial Intelligence, 112, 181–211.

[125] Dietterich, T. G., Busquets, D., De Màntaras, R. L., and Sierra, C. (2002, July). Action Refinement in Reinforcement Learning by Probability Smoothing. In *ICML* (pp. 107-114).

[126] Noonan, V. K., Fingas, M., Farry, A., Baxter, D., Singh, A., Fehlings, M. G., and Dvorak, M. F. (2012). Incidence and prevalence of spinal cord injury in Canada: a national perspective. *Neuroepidemiology*, *38*(4), 219-226.

[127] Lam, T., Wolfe, D. L., Eng, J. J., and Domingo, A. (2010). Lower limb rehabilitation following spinal cord injury. *Spinal Cord Injury Rehabilitation Evidence. Version*, *5*, 1-74.

[128] Krassioukov, A. (2009). Autonomic function following cervical spinal cord injury. *Respiratory physiology & neurobiology*, *169*(2), 157-164.

[129] Verrier M, Gagnon D, Musselman K. (2014) Toolkit for SCI Standing and Walking Assessment. Available from http://sci2. rickhanseninstitute.org/standing-walking/rhscir-toolkit/sci2- standingandwalking-toolkit-online-html. Note: Site moving to https://scireproject.com/outcome-measures/outcome-measure-tool/standing-walking-toolkit/

[130] R Core Team. (2017). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. https://www.R -project.org/.

[131] Canadian Institute for Health Information. *Case-mix grouping methodologies help health care facilities plan and manage their services.* https://www.cihi.ca/en/submit-data-and-view-standards/methodologies-and-decision-support-tools/case-mix.

[132] Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, *58*(1), 267-288.

[133] Hoerl, A. E. and Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, *12*(1), 55-67.

[134] Selivanov, D. (2016). text2vec. http://text2vec.org/

[135] RDocumentation. *kmeans.* https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/kmeans.

[136] Musselman, K. E., Lemay, J. F., Walden, K., Harris, A., Gagnon, D. H., and Verrier, M. C. (2019). The standing and walking assessment tool for individuals with spinal cord injury: A qualitative study of validity and clinical use. *The Journal of Spinal Cord Medicine*, *42*(sup1), 108-118.

[137] Laber, E. B., Lizotte, D. J., Qian, M., Pelham, W. E., and Murphy, S. A. (2014). Dynamic treatment regimes: Technical challenges and applications. *Electronic journal of statistics*, *8*(1), 1225.

[138] Ghavamzadeh, M., Mannor, S., Pineau, J., and Tamar, A. (2016). Bayesian reinforcement learning: A survey. *arXiv preprint arXiv:1609.04436*.

[139] Bain, R. (2016). Are our brains Bayesian?. *Significance*, *13*(4), 14-19.

Curriculum Vitae

Name:	Nathan Phelps
Post-secondary Education and Degrees:	University of Western Ontario London, Ontario, Canada 2014-2018 BSc
	University of Western Ontario London, Ontario, Canada 2018-2020 MSc
Honors and Awards:	Continuing Admission Scholarship 2014
	Ross and Jean Clark Continuing Scholarship 2015
	London Life Actuarial Career Scholarship, Three Year Continuing 2015
	Dean's Honor List 2015-2018
	NSERC Undergraduate Student Research Award 2018
	Alexander Graham Bell Canada Graduate Scholarship 2018
	John A. Mereu Book Prize 2018
	Gold Medal – Major in Actuarial Science 2018
	Gold Medal – Major in Data Science 2018
	Academic All-Canadian 2016-2019
Related Work Experience	Teaching Assistant University of Western Ontario 2018-2020

Research Assistant
University of Western Ontario
2018-2019, 2020