Managing Disease Outbreaks: Current Approaches, An Artificial Intelligence Alternative, and its Performance

By

Sandya Lakkur

Dissertation

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

in

Biostatistics

August 10, 2018

Nashville, Tennessee

Approved:

Robert E. Johnson, Ph.D.

Christopher Fonnesbeck, Ph.D.

Thomas Stewart, Ph.D.

Hiba Baroud, Ph.D.

To my mom,
who always believed I could do anything.

ACKNOWLEDGEMENTS

I would like to thank my advisor Christopher Fonnesbeck for all of his patience, time, and support during my graduate experience at Vanderbilt.

I would like to thank the members of my committee Hiba Baroud, Robert Johnson, and Thomas Stewart for all of their time and input, and for making committee meeting scheduling a very non-stressful experience!

I would like to thank the EEID collaboration, who made it possible to conduct this research.

I would like to thank my mom and dad for constantly reminding me to not doubt myself, and for always believing in me. I would not have made it to this stage of my education without you both. I would also like to acknowledge my sister Sindhu, who was there for me every step of the way. And I would like to acknowledge Naveen, who was always willing to listen and provide advice.

I would like to thank my fiancé (and soon to be husband) Tim for his unwavering encouragement. I am looking forward to finally living in the same state, after spending a year apart while I finished graduate school.

Finally, I would like to thank all of my classmates and professors at Vanderbilt University who have helped me make it to this stage in my career.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CDC	Centers for Disease Control and Prevention
CI	Confidence interval
CNN	Convolutional neural network
CP	Contiguous premises
DC	Direct contact
DDQN	Double deep Q-network
DEFRA	Department for Environment, Food and Rural Affairs
DQN	Deep Q-network
FMD	Foot and mouth disease
GIS	Geographic information system
MC	Monte Carlo
MDP	Markov decision process
RGB	Red-green-blue
RL	Reinforcement learning
RS	Remote sensing
SARS	Severe acute respiratory syndrome
SEIR	Susceptible-Exposed-Infected-Recovered
SEITR	Susceptible-Exposed-Infected-Treated-Recovered
SIR	Susceptible-Infected-Recovered
SIS	Susceptible-Infected-Susceptible
USDA	United States Department of Agriculture

CHAPTER 1

INTRODUCTION

Disease outbreak management has many layers of complexity. A large component, and perhaps the most widely studied, involves modeling disease spread. After having an understanding of disease spread, one can begin to think about disease outbreak control strategies. Control of epidemics is relatively less studied in the literature compared to modeling disease outbreak spread (Khouzani et al., 2011). One layer of complexity involves defining management objectives, i.e. - what should be achieved by management. The final control strategies are affected by how the management objectives are specified (Probert et al., 2015). Once objectives are specified, the decision-maker then needs to determine if optimal management is a priority (Khouzani et al., 2011). Optimal control strategies become increasingly difficult to construct when the decision space is very large. Large decision spaces tend to arise in the disease management literature, specifically when considering resource constraints. Due to these layers of complexity, optimally managing a disease outbreak presents a challenge.

This dissertation focuses on computational methods to inform disease outbreak termination strategies. It was assumed throughout this dissertation that the disease transmission model was correct, and all of the information about the outbreak was observable. There are several concepts that need to be introduced to better understand the ideas and results presented in this dissertation. This remainder of this chapter introduces some of the core concepts discussed in later chapters, and concludes with a structural overview.

1.1 Decision Analysis

Decision analysis can be formally defined as: a quantitative method of evaluating management options using information about uncertainties (Taylor and Sit, 1998). This definition can be deconstructed into two main components: evaluating management options and uncertainty. Evaluating management options requires knowledge about the following: the objective, actions, states, and rewards. Before any management strategies can be evaluated an objective (or objectives) needs to be defined. The objective provides a metric to judge different decisions. In the context of foot-and-mouth disease (FMD), which is the disease of interest in this dissertation, suppose

the objective is to minimize the duration of the outbreak. Competing management strategies can resultantly be evaluated based on how long the outbreak persists under each respective decision. The objectives(s) also defines the environment for the decision-making process which consists of states, actions, and rewards. States refer to the measurable quantities or qualities of the environment in which a decision is made. Naturally the more states an environment has, the more complicated the decision. Actions refer to the possible management options a decision-maker can execute. As with states, the more actions a decision-maker can choose from, the more complicated the problem. Each state (or each state-action pair) is associated with a specific reward. This reward provides a summary of how desirable it is to be in a specific state (Sutton and Barto, 1998). The higher the reward, the more desirable the state. These quantities are usually defined a priori through previous knowledge. Rewards may also be expressed as costs if the objective involves some expense from the decision-maker. To better understand this framework, an example in the context of FMD is presented.

Suppose the objective is to minimize the cost to farmers of livestock lost during an outbreak. Costs will only be incurred if there is an outbreak. This objective naturally establishes the states for which a decision must be made: there is currently an outbreak, and there is currently no outbreak. Notice that in this example, there are only two states which implies that this decision is not particularly complex. Suppose that in this example, the initial state of the environment is: there is currently an outbreak. To determine whether or not the outbreak continues at the next time point, also referred to as determining the next state, an action needs to be implemented. It is important to note that an action need not be an intervention; doing nothing is also an action that causes an initial state to evolve into another state. Since FMD travels quickly between farms due to movement of livestock and other disease vectors, a common management strategy is ring-culling: killing any livestock within a certain radius of an infected farm. Suppose that this strategy is modified to accommodate vaccinations: vaccinate all susceptible livestock within the pre-determined ring and cull only the infected. This segues to the possible management actions: ring-culling, and ring-vaccination with culling only infected livestock. In this example, costs will be used instead of rewards, because the objective is to minimize cost to farmers. Recall the contribution of costs to the decision-making problem: the costs provide a summary of how undesirable it is to be in a particular state (or state-action pair). Suppose it costs \$100 to cull one livestock and \$50 to vaccinate one livestock. This means that a farmer would have to spend $\$100C$ under the culling strategy if the

outbreak continued into the next time step, where C is the number of livestock to be culled. Under the second strategy a farmer would need to spend $\$50S + \$100I$ if the outbreak continued into the next time step, where S represents the number of susceptible livestock to be vaccinated, and I represents the number of infected livestock to be culled. Figure 1.1 presents a decision tree of the possible decisions and outcomes for this example.

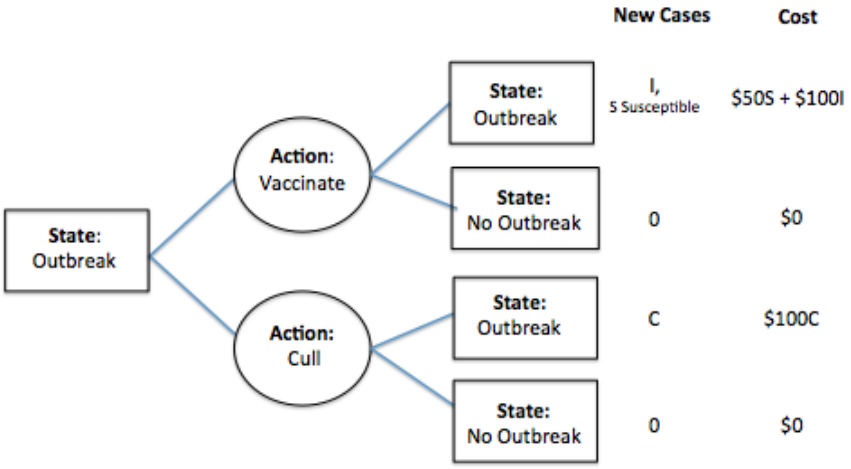

Figure 1.1: Decision tree of state, action, and new state with respective costs for FMD example.

An outbreak in this example is loosely defined as having a non-zero number of infected cases at a given time point. Thus any terminal node in the decision tree with a “no outbreak" state will have zero new cases. The costs are calculated by multiplying the number of infected or susceptible livestock with the associated culling and vaccination costs. Since the objective is to minimize costs to the farmer, the optimal action is chosen by calculating the average cost across the different branches in the tree. Notice the role of costs in this example: under both actions, it is less desirable to be in the subsequent “outbreak" state due to the higher associated costs. This motivates the need to choose an action that is more likely to result in a subsequent “no outbreak" state. This process of choosing actions that yield lower costs or higher rewards is not trivial in larger decision spaces. One approach to accommodate larger decision spaces is Q-learning, a reinforcement learning (RL) method.

1.2 Q-Learning

Reinforcement learning is a branch of artificial intelligence used to maximize reward, or minimize costs, in a complex decision environment. Q-learning is one type of RL method used in Markov decision processes (MDPs) to generate an optimal policy,

denoted as π^*. An MDP is a stochastic control process defined by states, actions, and rewards that satisfies the Markov property. This property states that a decision must only be dependent on the current environment and not by the path the decision process had to take to arrive at the current environment. This can be more formally stated as:

$$P(s_{t+1}, r_{t+1} \mid s_t, a_t, r_t, s_{t-1}, a_{t-1}, ...r_1, s_0, a_0) = P(s_{t+1}, r_{t+1} \mid s_t, a_t) \tag{1.1}$$

An optimal policy is a function that generates an optimal action given the current state, and assumes optimal actions are taken afterwards as per the Bellman optimality principle (Bellman, 1954). At time t, the MDP is in state s_t, an action a_t is chosen with guidance from the current policy, and the utility r_t is observed. The utility describes how desirable it is to be in a certain state and is generally expressed as a reward, but can also be expressed as a loss. The optimal policy cannot be constructed without estimating the expected reward function:

$$\bar{Q}^*(s_t, a_t) = Q(s, a) + \alpha[r + \gamma max_{a'} Q(s', a') - Q(s, a)] \tag{1.2}$$

where $Q(s, a)$ refers to the action-value function. Once Equation (1.2) is estimated, the optimal policy can then be constructed using the following:

$$\pi^*(s) = argmax_{a_t} \bar{Q}^*(s_t, a_t) \tag{1.3}$$

Q-learning iteratively updates $Q(s, a)$ using an "agent" that interacts with the environment. The agent is a construct that stores learned information about expected rewards from visiting a given state-action pair. The agent's goal is to choose actions that maximize expected reward, and this optimal behavior must be learned through experience. An illustration of the agent interacting with its environment is provided in Figure 1.2. The more experience the agent obtains, the better the estimates of the expected rewards. Convergence to the optimal policy using Q-learning is well-supported by the literature, and Jaakkola et al. (1994) presents this proof. The complete Q-learning algorithm is presented in Figure 1.3.

One of the most popular ways to illustrate Q-learning, and other RL algorithms, is to use a grid world example. To better understand all of the mechanics of Q-learning, the cliff walking grid world example from (Sutton and Barto, 1998) will be discussed. Figure 1.4 represents the grid world of interest. S is state the agent will start at the beginning of each episode, and G is the goal state. The agent can only choose from the following actions: up, down, left, and right. The agent also receives a reward of -1

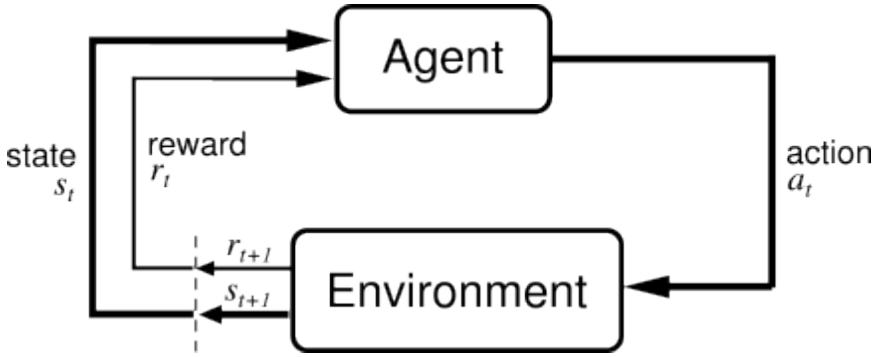

Figure 1.2: Agent interacting with its environment. The agent executes and action, observes the new state and reward, and uses this new information to choose a new action from the next state (Sutton and Barto, 1998).

Initialize all values in $Q(s, a)$
For each episode
    Observe initial s
    For each epoch
        Choose a using $Q(s, a)$
        Observe r, s'
        $Q(s, a) = Q(s, a) + \alpha[r + \gamma max_{a'} Q(s', a') - Q(s, a)]$ (Q-update)
        $s = s'$

Figure 1.3: The Q-learning algorithm for approximating the expected reward, and thus arriving at an optimal policy based on those rewards

for every action, except for the action that results in the goal state. This encourages choosing actions that finishes the task as quickly as possible. Choosing an action that results in the agent moving to the cliff results in a -100 reward. Notice the large difference in the two rewards, -100 and -1. This difference creates a prioritization in action choice: actions that take the agent to the cliff are much less favored than any other action. If the rewards were equal, the optimal policy would still be attained, but more training would be required.

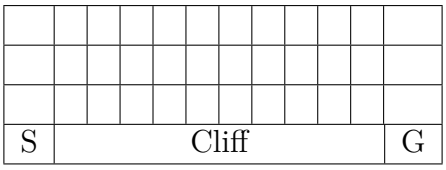

Figure 1.4: Cliff walking grid world from Sutton and Barto (1998). S represents the starting state, and G represents the goal state.

Using the Q-learning algorithm presented in Figure 1.3, 500 episodes of training was implemented. The results of the trajectory of training are presented in Figure

1.5. Notice that in the beginning of training the agent is not sure at all of how to act, taking nearly 450 steps to reach the goal state. However, through iterations of episodes, the agent begins to act optimally, reaching the goal state in about 13 steps almost every iteration.

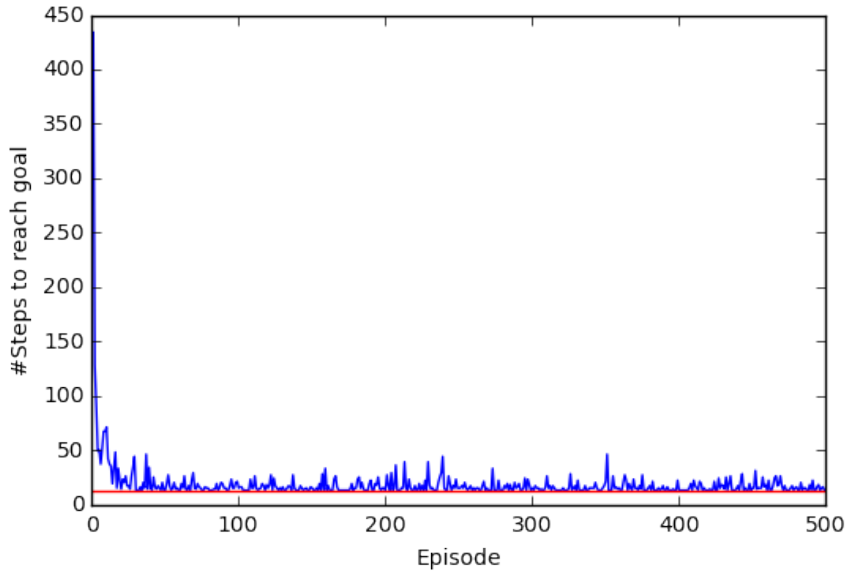

Figure 1.5: Trajectory of training, represented by number of steps to complete episode. Red line represents the minimum number of actions needed to reach the goal state from the start state, 13.

One advantage of Q-learning is that it is model-free. In other words the agent does not need to know how the decision environment will change based on an action, i.e. - know the transition probabilities $P(s_{t+1} \mid s_t, a_t)$. Instead, the algorithm uses a Q-update, presented in Figure 1.3, which combines previous experience with current experience to construct a new reward for a state-action pair (Dayan and Niv, 2008). Notice there is no transition probability in the Q-update. While Q- learning has the advantage of being model-free, it is still experiences computational storage problems in decision spaces where the state space is large. To apply Q-learning to such a decision space, a function approximator such as neural networks is required.

1.3 Convolutional Neural Networks

Convolutional neural networks are a sub-family of multilayer perceptrons that allow for local feature extraction (refer to chapter 2 for an introduction to neural networks). Multilayer perceptrons were originally inspired from the biological mechanics within the brain, however this process neglected the visual cortex's sensitivity to small sub-regions of the visual field, called receptive fields (Hubel and Wiesel, 1968).

The visual cortex decomposes an image into a collection of receptive fields, and this decomposition is naturally suited to capture spatial correlation structures inherent in an image. To allow for this same local, spatial correlation in a multidimensional input, convolutional neural networks include both locally connected and fully connected layers in its architecture. Figure 1.6 provides an illustration of a traditional fully connected network. Each node in one layer is connected to every node in the succeeding layer.

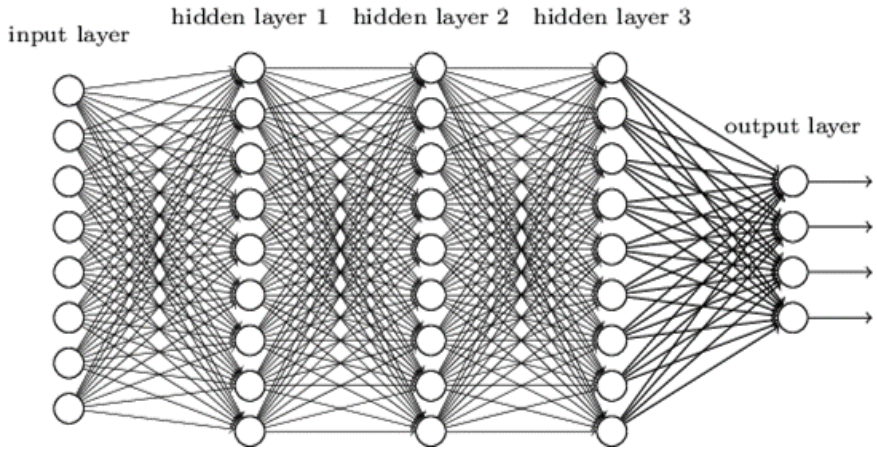

Figure 1.6: Example of a fully connected neural network with three hidden layers, from MathWorks (2018)

In a convolutional neural network, the input layer consists of multiple receptive fields, otherwise known as patches, or a neighborhood of nodes. In Figure 1.7, the patch in the input layer is denoted by the gray square. The image in the input layer is convolved with a learned weight matrix, also known as a filter, to generate a single activation map, represented by a white square in the first convolutional layer. Each activation map requires a different filter. In Figure 1.7 there are four activation maps in the first convolutional layer, thus four filters, and four different convolutions with the initial image, are required. The number of activation maps in a convolutional layer corresponds to the number of desired features from the preceding layer. Suppose the input image has dimension $r \times c$, and P features are desired for the first convolutional layer. The following equation can be used to construct each of the p activation maps:

$$Z_p = \sigma(W_p * X), p = 1, ..., P \tag{1.4}$$

where $\sigma()$ is the activation function, which is traditionally nonlinear, and the filter W_p is convolved with image X. 'Activation map' here refers to the matrix generated by performing the transformation of the filter convolved with the image. The dimensions

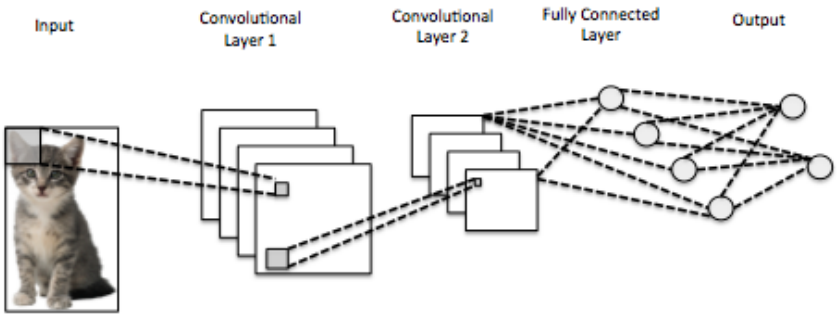

Figure 1.7: Example of a convolutional neural network

of the filter must be smaller than those of the initial input, call this $m \times m$ where $m < r, c$. The primary goal of convolutional layers is to extract features, via activation maps. To obtain more complex relationships between these learned features, it is common to include fully connected layers before the output layer, also illustrated in Figure 1.7. This would require the P activation maps to first be flattened to a single column vector, call this c of dimension, $(P \times m \times m) \times 1$. The following final transformation can then performed to estimate the output:

$$\hat{y} = g(W_f c) \tag{1.5}$$

If the domain of the output is infinite, $g()$ can be a linear function. W_f represents the weight matrix associated with the fully connected layer.

As with traditional neural networks, each weight matrix is "learned" through iterations of training. The process begins with a random initialization of weight values and then proceeds through forward propagation This process feeds the neural network weights through the equations in (1.4) and (1.5) and predicts the outcome. The prediction is then compared to the true outcome, and the error is calculated through a loss function, typically squared error:

$$L = \sum_{i=1}^{K} (\hat{y}_i - y_i)^2 \tag{1.6}$$

where K represents the total number of nodes in the output layer. Once forward propagation is complete, back-propagation is implemented to readjust the weights based on the results of forward propagation. Back-propagation consists of two steps: 1) computing partial derivatives of the loss function, and 2) adjusting the coefficients via a gradient descent update. The following partial derivatives are taken with respect

to the two types of coefficients using the chain rule:

$$\begin{aligned} \frac{\delta L}{\delta W_p} &= \frac{\delta L}{\delta Z_p}\frac{\delta Z_p}{\delta W_p}, \\ \frac{\delta L}{\delta W_f} &= \frac{\delta L}{\delta \hat{y}}\frac{\delta \hat{y}}{\delta W_f} \end{aligned} \tag{1.7}$$

Once the partial derivatives are calculated, the $(r+1)$st update is performed through gradient descent using the following equations:

$$\begin{aligned} W_p^{(r+1)} &= W_p^{(r)} - \gamma\frac{\delta L}{\delta W_p^{(r)}}, \\ W_f^{(r+1)} &= W_f^{(r)} - \gamma\frac{\delta L}{\delta W_f^{(r)}} \end{aligned} \tag{1.8}$$

The number of iterations, r, and the learning rate (the degree to which new information contributed to the estimation of the weights), γ are specified by the user *a priori*. Because convolutional neural networks have yielded strong performance with high dimensional image data, it is well suited for decision spaces where the state space is large (Guo et al., 2015).

1.4 Structural Overview

The remainder of this dissertation is divided into three main components. Chapter 2 reviews current computational methods that act as decision support tools to guide outbreak management, and methods that construct optimal policies to terminate an outbreak. Each method has associated strengths and weaknesses, which are also discussed. However, none of these methods are able to scale to complex decision spaces. Chapter 3 introduces a modified version of deep Q-networks (DQN) to optimally manage a disease outbreak. The decision analysis framework is used in this chapter to address structured decision-making, via defining objectives, utility, and alternative actions. DQN has recently been successful in areas such as video gaming and robotics, however, has never been applied to disease management. Chapter 4 evaluates the performance of DQN and compares the outcomes to widely used, simpler methods. In addition, Chapter 4 discusses how deep learning methods are associated with a large degree of tuning and computational run time. These limitations need to be considered when selecting a computational method for disease management. The dissertation concludes with Chapter 5, which reviews the main results of the dissertation, and highlights future areas of work.

CHAPTER 2

A REVIEW OF COMMON COMPUTATIONAL APPROACHES TO MANAGE DISEASE OUTBREAKS

2.1 Introduction

Disease outbreak management is a severely understudied field. While many advancements have been made with respect to modeling disease outbreak spread, less is known about management and control (Khouzani et al., 2011). Though there is a variety of candidate interventions for outbreak management including quarantine, educational interventions, vaccination programs, the challenge remains in determining the optimal order in which individuals or groups of individuals should receive an intervention.

This ordering is essential because disease outbreak management is constrained by limited resources. The 2001 United Kingdom foot-and-mouth disease (FMD) epidemic illustrated the consequences of not accounting for resource constraints. During the 2001 epidemic, the disease had already spread to multiple farms before the first case was detected (Gibbens et al., 2001). This resulted in an immediate shortage of veterinarians for identifying infected animals, and infected animals not being slaughtered quickly enough (Eales et al., 2002). Ultimately, over six million cattle were slaughtered by the end of the outbreak. Prioritization schemes could have provided guidance as to where veterinarians should be sent to first, or which infected farms should receive an intervention first. This particular epidemic highlights the need for prioritization schemes due to resource constraints.

Some disease systems currently have suggestions for prioritization schemes. For example, in sub-Saharan Africa prioritization of HIV/AIDS interventions is based on demographics, i.e. - pregnant women, children, the sickest individuals, etc. (Cheek, 2010; Wilson and Blower, 2005). This recommendation targets high risk individuals, thereby addressing how resources should be allocated. Management uncertainity can be added to this problem: would targeting these high risk individuals generate better outcomes compared to targeting another group of individuals? We would need to define what it means for one intervention to be better than another, and provide a measurable quantity that validated the superiority of one intervention over another. A formal framework would be required to fully explore the question.

Many current disease management strategies are informed by computational methods. The literature lacks a comprehensive review of these computational approaches

applied to disease outbreak management. Through such a review, researchers and policy makers would be able to better identify which approach is most suitable for their needs. They may also learn that their current practices do not achieve optimal outcomes, and that other approaches may better achieve their objectives. In this chapter, we review current computational approaches for the support of disease outbreak management. The studies referenced in this chapter apply a specific computational method to a particular disease system. However, each computational method can be modified to accommodate management in any disease system. This review examines the following computational approaches: disease surveillance methods, dynamic programming, Monte Carlo approaches, genetic algorithms, and network-based methods.

2.2 Disease Surveillance Methods as a Decision Support Tool

Disease surveillance has played an important role in preventing and controlling the spread of disease (Liao et al., 2017). This method identifies groups of individuals with high risk of transmission which can provide markers of detecting a prospective outbreak. Once these individuals have been identified, policy makers can then decide how to proceed in a pre-emptive management process. Disease surveillance is based on risk modeling, which can be performed using a variety of methods (Unkel et al., 2011). This section will focus on two forms of risk models: regression and neural networks.

2.2.1 Regression Models

Regression models are generally used if a primary goal of management is to understand the relationship between the outcome of interest and model predictors. In the context of disease surveillance, regression models can inform which population characteristics are associated with disease outbreak occurrence, and provides measurable quantities of association. Understanding this predictor-outcome relationship can ultimately inform prioritization schemes. Before considering extensions of regression modeling to prioritization schemes, we will first develop a methodological understanding of regression.

Depending on the data at hand, a regression model can be specified in many ways. In prospective outbreak detection the outcome of interest is usually binary, the occurrence of an outbreak (yes/no) (Liao et al., 2017). The probability of a binary outcome can be modeled using the logistic function:

$$P(\text{There will be an outbreak}) = \frac{e^{\sum_{i=0}^{N} \beta_i x_i}}{1 + e^{\sum_{i=0}^{N} \beta_i x_i}} \tag{2.1}$$

The covariates in the model, represented by $x_1, ...x_N$, are thought to be associated with the outcome, and $\beta_1, ...\beta_N$ are coefficients associated with the respective covariates. Generally the inclusion of covariates in a model is informed through prior scientific knowledge, or were previously found to be associated with the outcome using some form of univariate analysis.

Model fit is also an important consideration. If the model is underfitting the training data, the data used to construct the model, then there may not be enough information from the covariates to capture a relationship between the covariates and the outcome. If a model is overfitting the data then the model performs extremely well in predicting the outcome in the training data, but not well in a different dataset, i.e. - the testing data or future data. Feature selection (eliminating some covariates), and increasing the amount of regularization (constraints on model coefficients (β_i)) are common solutions for overfitting. The functional form of the covariates also need to be specified *a priori*. These are also generally informed by domain knowledge. Model specification is one of of the more challenging aspects of regression modeling and is still under study.

As previously stated, one advantage of regression modeling is inference i.e. - understanding the relationship between model predictors and the outcome. Consider the prospective outbreak detection model in (2.1) and let $p = P(\text{There will be an outbreak})$. Using a logit transform we can rewrite the logistic regression model:

$$log(\frac{p}{1-p}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... \tag{2.2}$$

Equation (2.2) models the log odds of the occurrence of an outbreak as a function of covariates. We will use the results from Leung and Tran (1999) to better understand how inferences can be made under logistic regression. They constructed a logistic regression model to predict the occurrence of an aquaculture disease outbreak in shrimp farms given certain covariates thought to be associated with the outcome.

Figure 2.1 illustrates the results of the logistic regression. The $exp(\beta)$ column presents odds ratios which can be used to make inferences. For example silt deposit is a binary covariate in the model (yes/no), with $exp(\beta)$ equal to 2.8975. This means that the odds are 2.8975 greater of an outbreak occurring when there is silt deposit compared to the odds of an outbreak occurring when there is no silt deposit, con-

Table 1
Results of the logistic regression model[a]

Variables[b]	Estimates of β	Standard error	P-value	Exp(β)	Probability
POLYCULTURE	−1.0961	0.3055	0.0003	0.3342	0.250
DRY POND	−1.0393	0.3957	0.0086	0.3537	0.261
I/D-CANAL	1.3984	0.3560	0.0001	4.0486	0.802
WATER-SOURCE:			0.0002		
—Estuary/River	−2.1598	0.5288	0.0000	0.1154	0.103
—Direct-from-sea	0.0606	0.6094	0.9208	1.0625	0.515
—Canal-from-sea	−0.0479	0.3884	0.9018	0.9532	0.488
—Other	−1.3885	0.5887	0.0184	0.2495	0.200
SITE-SELECTION	−1.6936	0.4026	0.0000	0.1839	0.155
SILT-DEPOSIT	1.0638	0.3001	0.0004	2.8975	0.743
Constant	1.1428	0.5079	0.0244		

[a]Model $\chi^2 = 204.42$.
[b]Variables: — POLYCULTURE: yes = 1, 0 otherwise; — DRY POND: yes = 1, 0 otherwise; — I/D-CANAL: water discharge into intake/drainage canal; yes = 1, 0 otherwise; — WATER-SOURCE: the main salt/brackish water source. The effect of the four categories in the table are compared to the category of 'Saltwater creek'. — SITE-SELECTION: site selection to avoid impacts of other users; yes = 1, 0 otherwise; — SILT-DEPOSIT: deposit silt on-farm; yes = 1, 0 otherwise.

Table 2.1: Logistic regression results for predicting occurrence of disease outbreak in shrimp farm, from Leung and Tran (1999). The left column includes the covariates in the model, the description of each covariate is provided below the table of results, and the $exp(\beta)$ column presents the odds ratios of the occurrence of an outbreak conditional on covariate values.

ditional on the other covariates being fixed. This suggests that if there are resource constraints, farms with silt deposits should be prioritized to receive an intervention over farms with no silt deposits because they have a higher odds of experiencing an outbreak. If a researcher or policy maker believed that there was a multiplicative effect between water discharge and silt deposit, an interaction term could be included in the model. This interaction term would present the odds of an outbreak when there is water discharge compared to no water discharge given silt deposits exist, compared to the odds of when there is water discharge compared to no water discharge given silt deposits exist, assuming all other covariates are fixed. If this odds ratio is much larger than one, then resources should be prioritized to farms with both silt deposits and water discharge. These rules of thumb regarding which groups of individuals, or in this case which types of farms, spur disease outbreaks can help policy makers prioritize who should receive an intervention first under resource constraints.

Regression models are not just limited to binary outcomes. Linear regression can be used for outcomes that are continuous; for example predicting the area of an outbreak or predicting the number of infections based on certain covariates (Serfling, 1963; Stroup et al., 1989; Costagliola et al., 1991; Simonsen et al., 1997; Jackson

et al., 2007). Regression methods also accommodate many data sources. In most of the previous referenced studies, demographic, survey or lab data was collected from large-scale studies conducted by government entities such as the CDC. However, regression models have also been paired with passive reporting sources to predict the occurrence of an outbreak (Eysenbach, 1970; Cooper et al., 2003; Hogan et al., 2003; Polgreen et al., 2008; Althouse et al., 2011). Some examples of passive reporting sources are internet searches for a specific keyword, or phone calls to a health hotline.

In regression models, covariates can be included that implicitly describe spatial orientation, e.g. - distance to nearest infected site, average distance to infected sites, etc. The literature refers to spatial model inferences as "disease mapping", or "predictive mapping" and has been widely explored in the disease surveillance (Thomson et al., 1999; Best et al., 2005; Raso et al., 2005; Eisen and Eisen, 2008; Mondini and Chiaravalloti-Neto, 2008). The inclusion of an explicit spatial structure allows for another layer of complexity in regression models, specifying correlation between sites. It is reasonable to assume that sites closer to each other are more related than sites farther away. A kernel is a function which specifies the degree to which closeness is associated with outcomes that are similar. Some common kernels to define spatial correlation structure are: linear, exponential, gaussian, spherical, and rational. Gaussian, exponential, and spherical structures, for example, are symmetric and are thus generally used if sites are correlated regardless of their physical directions from one another. As an example, Clements et al. (2006) constructed a binomial risk model to predict the number of infected children at particular schools in Tanzania. An exponential spatial correlation structure was defined: $f(d_{ij}; \phi, \kappa) = e^{-(\phi d_{ij})^{\kappa}}$. The parameter ϕ represents the rate of decline of correlation as distance increases, and κ represents the degree of spatial smoothing. It is important to note that spatial structure does not need to be used in a regression model if spatial correlation is not needed to calculate risk. However, failure to account for spatial correlation in the data when it is important can potentially lead to inflated significance of regression coefficients and overestimation of the precision of predictions (Clements et al., 2006).

Risk models also appear in the remote sensing (RS) literature. RS is defined as the science of obtaining information about an area or phenomenon via a remote device, usually a satellite, for improved surveillance and monitoring (Kumar, 2009). In RS, risk model inferences have been integrated with RS tools such as geographic information system (GIS) software or Google Earth to provide map-based outputs for policy-makers regarding where interventions should be administered (Moloney et al., 1998; Brooker and Michael, 2000; Elnaiem et al., 2003; Rogers and Randolph,

2003; Chansang and Kittayapong, 2007). Eisen and Lozano-Fuentes (2009) made a distinction between presenting risk model results and presenting those results with a disease map, "Consider first the statement 'dengue cases were concentrated to the northern part of the city.' This provides a general idea of where disease cases were most common but does not necessarily provoke further interest. By complementing the statement 'dengue cases were concentrated to the northern part of the city' with a map showing case locations ...capture the imagination of the audience."

Other extensions of using regression models for outbreak detection have also been explored. For example, time-series regression models have been constructed to predict outbreaks accounting for temporal correlation (Toubiana and Flahault, 1998; Wagner et al., 2001; Tsui et al., 2002; Magruder, 2003; Suyama et al., 2003). Less studied are "thresholding" methods, based on the results of a regression model. If the expected number of infected cases exceeded a pre-specified threshold by time t, then decision-makers would be alerted to begin implementing an intervention (Farrington, 1996).

We have discussed how regression modeling heavily depends on domain knowledge from the researcher, by means of including correct covariates, and potential spatial structure. Assuming that the model fits the data well, valid inferences can be made. The inferences can ultimately provide rules of thumb regarding how to prioritize which individuals receive treatment, and RS tools have previously been used to assist in visualizing these prioritization schemes. There are many free sources of software that quickly fit regression models, such as `glm` in R and `statsmodels` in Python, making them universally accessible to researchers.

2.2.2 Neural Networks

As an alternative to traditional statistical methods, like regression modeling, machine learning provides the option to fit extremely flexible models to the data. Machine learning is a rapidly growing field and has made many advancements in image recognition and robotics. One of the most popular machine learning methods is neural networks, sometimes referred to as multilayer perceptrons (Wang et al., 2017). We will first develop a mathematical foundation of neural networks and discuss some example applications to disease surveillance.

Figure 2.1 illustrates a K-class classification neural network with K nodes in the output layer. Neural networks are composed of nodes, which refer to the individual circles in Figure 2.1. There are three types of layers: input (specified by the blue neurons), hidden (red neurons), and the output (yellow neurons). Usually, all of the

nodes in each layer are connected via connection weights, the individual black lines.

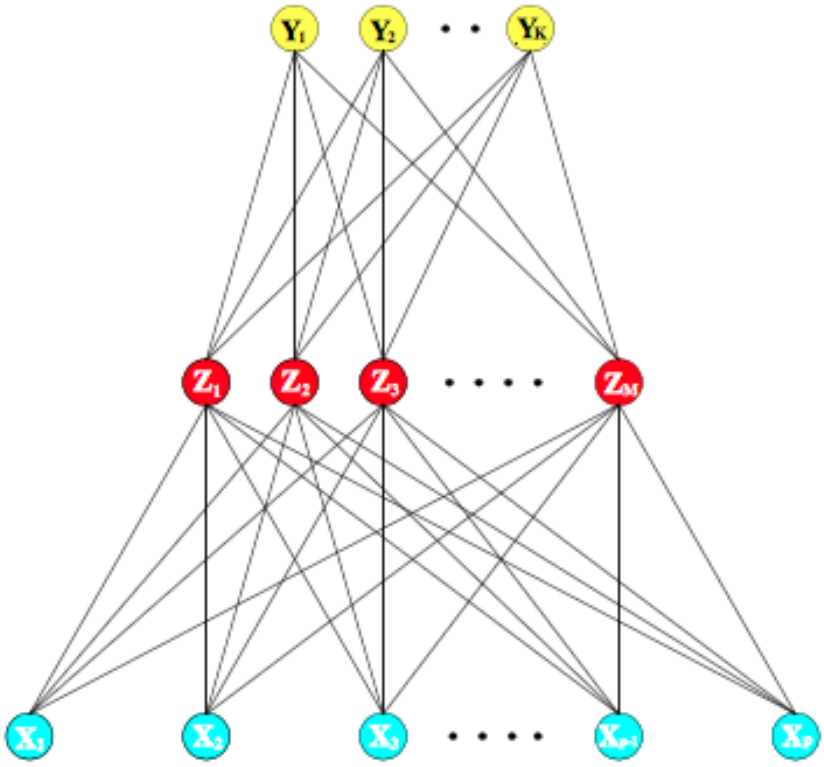

Figure 2.1: Sketch of a neural network, from Hastie et al. (2005).

The output layer can have multiple predicted values (multiple nodes) if the outcome is categorical, which usually suggests a classification problem (as in Figure 2.1), while if there is only one predicted value this suggests a regression problem. Neural networks are considered two-stage models because the hidden layer(s) Z is/are created from a non-linear transformation, via an activation function of the linear combinations of covariates X and the output layer transforms the linear combination of the values T from the hidden layer(s) (Hastie et al., 2005). These relationships can be further expressed through the following chain of equations:

$$\begin{aligned} Z_m &= \sigma(\alpha_{0m} + \alpha_m^T X), m = 1, ..., M \\ T_k &= \beta_{0k} + \beta_k^T Z, k = 1, ..., K \\ f_k(X) &= g_k(T), k = 1, ...K, \end{aligned} \tag{2.3}$$

where $Z = (Z_1, Z_2, ...Z_M)$, and $T = (T_1, T_2, ..., T_K)$. The final equation in 2.3 generates predictions for the outcome $Y_1, ..., Y_K$ as previously expressed in Figure 2.1. Notice that the first equation in (2.3) is the non-linear transformation of inputs,

and the last equation is the transformation of the linear combination of the nodes of the final hidden layer. The output function $g_K(T)$ does not necessarily have to be nonlinear, particularly if the output is continuous. The coefficients in the model $(\alpha_{0m}, ...\alpha_m, \beta_{0k}, ...\beta_k)$ are represented by the connection weights in Figure 2.1. These coefficients are "learned'" through iterations of training. The coefficient-fitting process begins with a random initialization and then proceeds through a process called forward propagation. This process feeds the neural network weights through the equations in (2.3) and predicts the outcome. This prediction is then compared to the true outcome, and an error is calculated through a loss function. One example is squared error:

$$R(\theta) = \sum_{k=1}^{K} \sum_{i=1}^{N} (y_{ik} - f_k(x_i))^2 \tag{2.4}$$

The θ in Equation (2.4) refers to the collection of weights: $(\alpha_{0m}, ...\alpha_m, \beta_0 k, ...\beta_k)$. Once forward propagation is complete, back-propagation is implemented to readjust the coefficients based on the results of the forward propagation. Back-propagation consists of two steps: 1) computing partial derivatives of the loss function, and 2) adjusting the coefficients via a gradient descent update. The following partial derivatives are taken with respect to the two types of coefficients:

$$\begin{aligned} \frac{\delta R_i}{\delta \beta_{km}} &= -2(y_{ik} - f_k(x_i)) g_k'(\beta_k^T z_i) z_{mi}, \\ \frac{\delta R_i}{\delta \alpha_{ml}} &= -\sum_{k=1}^{K} 2(y_{ik} - f_k(x_i)) g_k'(\beta_k^T z_i) \beta_{km} \sigma'(\alpha_m^T x_i) x_{il} \end{aligned} \tag{2.5}$$

Once the partial derivatives are calculated, the $(r+1)$st coefficients' update is performed through gradient descent using the following equations:

$$\begin{aligned} \beta_{km}^{(r+1)} &= \beta_{km}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\delta R_i}{\delta \beta_{km}^{(r)}} \\ \alpha_{ml}^{(r+1)} &= \alpha_{ml}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\delta R_i}{\delta \alpha_{ml}^{(r)}} \end{aligned} \tag{2.6}$$

The number of iterations, r, and the learning rate (the degree to which new information contributes to the estimation of the coefficient), γ are specified by the user *a priori*. There are many neural network parameters to tune along with r and γ: number of nodes per hidden layer, number of hidden layers, type of activation function, number of epochs (the amount of forward and backward passes of all the training samples in one iteration). Thus finding the combination of parameters to achieve

optimal performance can be a source of computational complexity.

More recently, neural networks have been incorporated into disease surveillance (Husin et al., 2008; Zhang et al., 2012; Luma et al., 2014; Zhang et al., 2014; Wang and Deng, 2016). Referring back to the shrimp farm example referenced in Leung and Tran (1999), a neural network was used for prospective outbreak detection. The input layers consisted of 68 covariates describing farm site selection and design and farm management practices. The output layer consisted of two nodes: presence or absence of disease. Neural networks have also been used in the RS literature (Curran et al., 2000; Kiang et al., 2006; Wang et al., 2008; Akil and Ahmad, 2016; Muharam et al., 2017). As an example, Wang et al. (2017) constructed a neural network to predict the number of human brucellosis cases at the county level in Inner Mongolia, China. The input layer consisted of ten covariates and the output layer contained one node: log-transformed number of human brucellosis cases at the county level. The estimates from the neural network were then integrated with GIS software to construct map-based outputs.

2.2.3 Inference, Prediction, and Decision Support

Neural networks and regression modeling both address the same goal: prospective outbreak detection. However, the choice of which to use depends on preferences of inference or prediction. The main advantage to regression modeling is the ease of interpreting coefficients, as previously demonstrated with odds ratios in logistic regression. Inferences from the regression model can then inform prioritization schemes. The main advantage of neural networks is flexible modeling, by means of layers of non-linear transformations and series of interaction terms (Tu, 1999). A logistic regression model uses one non-linear transformation of a linear combination of data, but neural networks perform multiple non-linear transformations. Something to consider before using neural networks for disease surveillance, is that a large amount of data are required for training. For example Yu et al. (2014) used a neural network to predict incidence of hand-foot-mouth disease in China. Four years of data, over 90,000 cases, were used to train the neural network. Less data is required to fit a regression model.

If it is suspected that there is a complex relationship between the covariates and the outcome, then a neural network may achieve better performance. A complicated risk model is not always necessary to achieve superior performance. In Leung and Tran (1999) model accuracy was improved by only 8.13% when a neural network was

used compared to logistic regression. While model coefficients can be extracted from the neural network, they do not yield the same interpretations as odds ratios. A policy-maker would be able to generate predictions of prospective outbreaks, but not rules of thumb for how to prioritize which individuals receive an intervention.

Disease surveillance methods are a decision support tool. General rules of thumb for which groups of individuals should be prioritized for an intervention can easily be extracted from the results of regression models. These individuals were identified due to their increased probability of causing an outbreak to occur. In cases where more flexible models better fit the data, neural networks can improve predictive performance.

Regression models and neural networks do not inform policy makers about optimal decisions. It is unclear if administering an intervention to one group of individuals results in a smaller cost or a smaller outbreak duration than administering an intervention to another group of individuals. Furthermore, it is unclear if a prioritization scheme informed from a regression model is the best prioritization scheme. Decision analysis would need to be incorporated to address this concern. If a policy-maker attempts to be proactive in managing a potential disease outbreak, then disease surveillance methods are sufficient. Often times however, policy makers are interested in achieving some objective as an outbreak is occurring.

2.3 Optimal Control Theory

2.3.1 Decision Analysis

If it is of interest to construct the best prioritization scheme, or the best general management strategy, decision analysis could provide a tool to compare the management strategies. Decision analysis can be formally defined as a quantitative method of evaluating management options using information about uncertainties (Peterman and Anderson, 2012). Evaluating management options requires knowledge about the following ideas: the objective, actions, states, and utility. The objective provides a metric to evaluate different decisions, for example minimizing cost or minimizing the duration of an outbreak. States refer to the measurable quantities or qualities of the environment in which a decision is made. One common example of a state in disease management is the number of infected individuals. Actions, or controls, refer to the possible management options a decision maker can execute. Each state (or each state-action pair) is associated with a specific utility, which describes how desirable it is to be in a specific state. The primary goal in decision analysis is to maximize

utility according to the specified objective. This component was missing from disease surveillance. Inferences from regression models could guide prioritization of which individuals should receive an intervention, but there is no measure of utility associated with a particular control. Thus, one would not be able to conclude that one control generated better outcomes, i.e. - a higher utility, than another control. The optimal control yields the highest average utility, defined by the following equation:

$$\bar{R}(a) = \sum_{s} P(s' \mid s, a) R(s, a) \tag{2.7}$$

The term R or $R(s, a)$ in Equation (2.7) denotes a reward, which is one form of utility. They are also specified *a priori* by the decision-maker. $P(s' \mid s, a)$ represents a transition probability, the probability that the subsequent state s' is observed given control a is implemented. Synthesizing the components of Equation (2.7) yields the following interpretation: the average utility of a control a is the sum of the utilities of the resulting states weighted by the states' transition probabilities. Compartment models are required to define transition probabilities.

2.3.2 Compartment Models

In the context of disease dynamics, compartment models describe the rates at which individuals move among disease states. Compartment models are not optimal control models, but are defined *a priori* to simulate outbreaks and construct optimal controls for some computational methods. These models are usually expressed as a collection of deterministic differential equations. One such compartment model is SIR, represented using the following collection of differential equations:

$$\begin{aligned} \frac{\delta s}{\delta t} &= -\beta \frac{I(t)S(t)}{N} \\ \frac{\delta i}{\delta t} &= \beta \frac{I(t)S(t)}{N} - \mu I(t) \\ \frac{\delta r}{\delta t} &= \mu I(t) \end{aligned} \tag{2.8}$$

The quantities $S(t)$, $I(t)$, and $R(t)$ in (2.8) refer to the number of susceptible individuals, infected individuals, and recovered individuals at time t, respectively. The first equation refers to the rate at which susceptible individuals are removed from the population, where β represents a rate of infection. The second equation represents the difference between the rate at which susceptible individuals are removed from the population and the rate at which infected individuals become recovered, where

μ is the rate of recovery. The third equation refers to the rate at which infected individuals become recovered. The appropriate type of compartment model depends on the characteristics and dynamics of the disease. For example measles, mumps and rubella traditionally follow an SIR model, whereas influenza traditionally follows an SIS model - a susceptible individual becomes infected, then susceptible again (Keeling and Rohani, 2007).

While deterministic compartment models are traditionally used for optimal control theory, the components can also be expressed as stochastic entities. In a study by Yaesoubi and Cohen (2011*a*), a stochastic compartment model was used to describe the dynamics of an influenza outbreak in a British boarding school. Following the decision analysis framework, the state was defined by two parameters: $X_S(t)$, the number of susceptible individuals at time t and $X_I(t)$ the number of infected individuals at time t. There were two types of interventions: number of boys to vaccinate (0 to 763), and whether or not a transmission-reducing intervention (e.g. school closure, hygienic interventions, isolation, etc.) should be implemented. Thus any one action would consist of a combination of a discrete value, number of kids to vaccinate, and a binary value, implement or do not implement the transmission-reducing intervention. The primary goal of this study was to find the optimal combination of interventions. The reward was quantified by a combination of: the policy maker's willingness to pay for the intervention, the cost of the intervention, and the loss in health due to infections (quantified by quality-adjusted life years). A stochastic compartment model was used to generate the number of new infections. It followed a binomial distribution with the probability of a susceptible individual becoming infected defined as:

$$P_{I(t)}(i \mid X_S(t), X_I(t)) = \begin{cases} \binom{X_S(t)}{i} \phi(t)^i (1-\phi(t))^{X_S(t)-i} & 0 \leq i \leq X_s(t) \\ 0 & otherwise \end{cases} \tag{2.9}$$

where $\phi(t)$ represented the overall probability that a susceptible person became infected. The transition probabilities were calculated using Equation (2.9):

$$P(s' \mid s, a) = \begin{cases} P_{I(t)}(x_I \mid X_S(t), X_I(t)) & x_S + x_I = X_S(t) \\ 0 & otherwise \end{cases} \tag{2.10}$$

Notice that the compartment model was directly related to calculating transition probabilities.

Many previous studies have leveraged another component of the compartment model, the basic reproductive number, to make general rules of thumb for manage-

ment (Driessche and Watmough, 2001; Ferguson et al., 2005; White and Pagano, 2008; Boëlle et al., 2009; White et al., 2009). The basic reproductive number, denoted by R_0 describes the average number of secondary infections caused by a single infected individual in a completely susceptible population (Miller Neilan et al., 2010). If $R_0 > 1$ then the disease is considered an epidemic. In a study conducted by Ferguson et al. (2001) different culling scenarios to manage the United Kingdom foot and mouth disease outbreak were explored. It was found that a small reduction in slaughter times results in a disproportionate reduction in R_0, suggesting that rapid slaughter achieves $R_0 < 1$. In another study, control measures and general rules of thumb were informed by R_0 to manage a severe acute respiratory syndrome (SARS) outbreak in Singapore (Lipsitch et al., 2003). This study found that due to low values of R_0 a combination of shortening the time from symptom onset to isolation and contact tracing can be effective in containing SARS. In addition, this study found that if SARS spread over a long period, with $R_0 > 1$ the quarantine would impose a large burden on the population with individuals needing to be quarantined multiple times over the course of the outbreak. Thus an aspect of the compartment model, R_0 can help provide general rules of thumb and lessons learned that can be applied to future management. Disease management using the compartment model directly, has been explored in other capacities as well. For example, compartment models have previously been combined with classical optimization techniques to explore optimal control methods, due to the fact that compartment models are generally expressed as a system of differential equations (Castilho, 2006; Barrett and Hoel, 2007; Yan and Zou, 2007; Blayneh et al., 2009; Okosun et al., 2011).

When the decision space has a relatively small number of actions and states, the optimal control can be found through exhaustive search. This is not the case with the example presented in Yaesoubi and Cohen (2011*a*), thus the state and action space was further discretized. In general, challenges arise when controls are continuous, or if there are many controls, or many states.

2.3.3 Dynamic Programming

The decision analysis framework is designed to allow policy-makers to optimize an average utility and achieve a given objective. As previously mentioned, optimizing the average utility is not always straightforward - particularly in complex decision spaces. Dynamic programming incorporates the decision analysis framework, and is based upon the Bellman optimality principle, which states that an optimal policy has

the property that whatever the initial state and decisions are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decisions (Bellman, 1954). In other words, sub-sequences of an optimal policy are also optimal. A policy provides a mapping from a current state to an action:

$$\pi(s) = argmax_a \bar{R}(a) \tag{2.11}$$

It was previously discussed that optimal actions were chosen based on the largest utility. An optimal policy fits this definition as well, but also assumes that after the optimal action at time t is taken, optimal actions continue to be taken.

Dynamic programming depends on the Bellman optimality equation, presented in Equation 2.12. Because the Bellman optimality equation requires the immediate utility from the current state and the calculation of the average utility of the next state, dynamic programming is inherently a backwards iterative process (Yaesoubi and Cohen, 2011*a*). The intuition to using a backwards iterative process is to ensure that for any given time point, the selected action generates the highest expected utility by the end of management. If forward iteration is used, an unknown expected value will be in the formula, indicated by equation 2.12. That is, assuming we know the expected utility for time T, the action we choose at time $T-1$ will maximize the utility for the current time step and the future time step.

$$\bar{Q}^*(s_t, a_t) = max_{a_t}[\sum_i P(H_i)[R(a_t \mid s_t) + \sum_{s_t+1} P(s_{t+1} \mid s_t, a_t) Q_i(s_{t+1}, a_{t+1})]] \tag{2.12}$$

The term $Q(s_t, a_t)$ in Equation 2.12 is the action-value function, a function of expected reward. The term, $P(H_i)$ represents a policy maker's belief that a pre-defined set of criteria defining the decision environment are true. For example, the policy maker may have a 40% belief that sheep are more susceptible to FMD than cows. Based on these beliefs the measures of uncertainty can be incorporated into the Bellman optimality equation, $P(H_1) = .40$, and $P(H_2) = .60$, and the expected utility can be calculated incorporating that uncertainty. Using the Bellman optimality principle, and the previously discussed intuition to backwards iteration, the optimal action determined at time $T-1$ constitutes part of the optimal policy. This process can be repeated again for time $T-2$, $T-3, ..., t$, where t is the desired time point. The complete collection of actions generates the optimal policy. The construction can be written by the following:

$$
\begin{aligned}
&t = T : \bar{Q}(a_T, s_T) \\
&t = T-1 : \bar{Q}(a_{T-1}, s_{T-1} \mid a_T, s_T) \\
&\vdots \\
&t = T : \bar{Q}(a_t, s_t \mid a_{t+1}, s_{t+1})
\end{aligned}
\tag{2.13}
$$

Since the dynamic programming framework starts at the terminal point, $\bar{Q}(s_T, a_T)$ needs to be defined. Fortunately, the terminal action-value quantity can be chosen arbitrarily because the action value function will be updated iteratively, and converge to the true action value function in the limit ($Q(s, a) = 0$ for all s is one common example of arbitrary assignment). Most of the time, however, $\bar{Q}(s_T, a_T)$ is chosen informatively to improve convergence.

Dynamic programming has been previously studied in applications to disease management (Clancy and Green, 2007; Yaesoubi and Cohen, 2011*b*; Deng et al., 2013; Ludkovski and Niemi, 2013; Zhu et al., 2016). In the wildlife management literature, a stochastic implementation of dynamic programming tends to be used (Shoemaker, 1981; Shea, 1998; Baxter et al., 2007; Valetta, 2007; Marescot et al., 2013). Yaesoubi and Cohen (2011*a*) implemented a dynamic programming approach with a relatively small decision space: two state parameters, and two action parameters. As the number of states increase, the dynamic programming algorithm begins to lose efficiency. This phenomenon refers to the curse of dimensionality: the number of operations needed to arrive at an optimal decision tends to grow exponentially with the number of state variables. It is not always possible to generate exact solutions when the state or action space is continuous, thus a common approach to generate approximate solutions is to discretize the decision space. However by employing discretization, the dynamic programming approach may not yield an exact optimal solution. Compared to an exhaustive search approach, dynamic programming is exponentially faster because direct search would require manual examination of each policy to ensure optimality (Sutton and Barto, 1998). As the decision space becomes more complex it becomes harder to present and visualize results. Expected rewards are usually stored as a "look-up" table with states as rows and actions as columns, with each cell populated by the expected reward for the corresponding state-action pair. However, this table can present computational storage problems when there are many states (Ge et al., 2010).

2.4 Heuristics for Disease Outbreak Management

Heuristics are an alternative to optimal control methods. As previously discussed, optimal control methods present computational storage and computational expense challenges in larger decision spaces. Heuristics are more easily derived in comparison to optimal control methods, however they do not guarantee an optimal policy will be constructed. If it is sufficient to generate a policy that is close to optimal, and computational time is more valuable, then heuristics may be the better approach for the decision-maker. The next two sections discuss two heuristics that are commonly used in the disease outbreak management literature.

2.4.1 Monte Carlo Methods

Various heuristics for calculating expected rewards have been investigated in applications to disease management. Monte Carlo (MC) methods are a collection of algorithms that use repeated sampling to generate estimates (Eckhardt, 1987). This method is popular in many fields due to its ease of implementation and ability to capture the randomness inherent in real-life systems (Kroese et al., 2014). Through repeated simulation, a researcher can also quantify uncertainty of a particular control on expected utilities. This method also allows for many competing interventions to be quickly and easily compared. For example, (Merl et al., 2009) was interested in managing an influenza outbreak in a British boarding school. Using a decision analysis framework, the objective for this study was to minimize the cost of the intervention. The state was specified by the number of susceptible and infected individuals, and the control consisted of a combination of: the fraction of susceptible individuals to vaccinate, and the number of susceptible individuals that had to be left in other to terminate the campaign. The utility was defined by a combination of the cost units per dose of vaccine and the cost of treating an infected individual. There were 100 competing control strategies: ten ranges of proportion of individuals to vaccinate, and ten ranges of the number of susceptible individuals to stop the intervention. Thus 100 types of simulations of the disease outbreak were conducted using different control combinations and the intervention that produced the lowest average cost was defined as the optimal intervention.

Figure 2.2 illustrates the cost surface associated with the 100 types of simulations. In this example Monte Carlo was used as an exhaustive search method to choose the best control, vaccinating 30% of susceptible individuals at each each time step until the number of susceptible individuals falls below 150. In another study the 2001

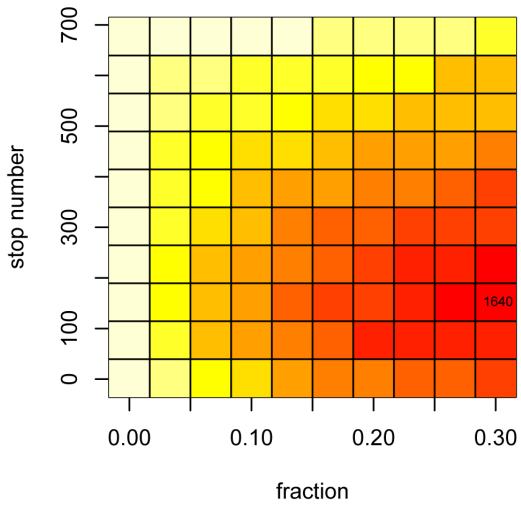

Figure 2.2: Cost surface generated from simulation of different combinations of control (Merl et al., 2009)

United Kingdom foot and mouth disease outbreak was managed using a vaccination annulus, defined by an inner and outer ring (Tildesley et al., 2006). Once a combination of inner and outer radii were chosen then vaccination priority was given to farms within the annulus that were farthest away from the infected premises. The choice of combination of inner and outer radius distances were chosen via Monte Carlo simulation. There were fifty values of inner radii and outer radii that were explored.

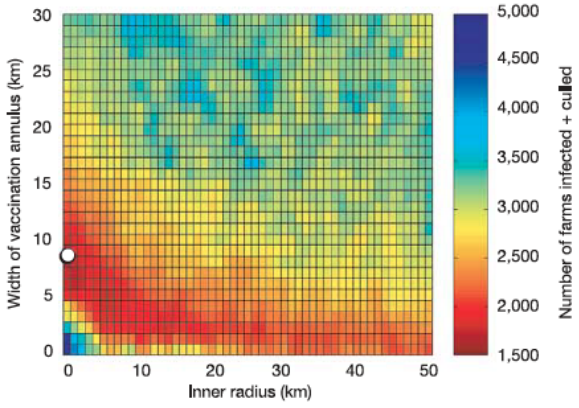

Figure 2.3: Cost surface generated from simulation of different combinations of control Tildesley et al. (2006)

The combination chosen achieved the smallest number of total infected and culled farms as denoted by the white dot in Figure 2.3.

In the examples discussed so far, Monte Carlo methods used exhaustive search to estimate expected utilities, and thus generate an action. This implementation of

Monte Carlo as an exhaustive search tool has been widely used to evaluate management strategies (Keeling et al., 2001; Morris et al., 2001; Keeling et al., 2003; Tildesley et al., 2009, 2010). Monte Carlo methods can also be used to generate a policy based on the calculated approximate rewards, as an alternative to dynamic programming. Policy construction via Monte Carlo has not been explored in managing disease outbreaks. To construct a policy using Monte Carlo, states and actions would need to be visited an infinite number of times. Expected utility would then be calculated by taking the average of all of the rewards from a given state-action pair.

Each simulation of the outbreak is defined as an "episode." Convergence to the optimal policy is guaranteed with an infinite number of episodes since the changes to the action-value function decrease over time, but a large number of episodes will approximate the optimal policy (Sutton and Barto, 1998). Expected rewards are updated based on the results, the new state and the observed reward, of each episode. A larger number of episodes results in more opportunities to visit each state-action pair, s, a, reducing MC error. Just as with dynamic programming, Monte Carlo methods are also limited in complex decision spaces. For example, if there are many states and actions it may be computationally impossible to generate and store expected rewards. It may also be difficult for each state-action pair to be visited during simulations. However, this method provides a straightforward implementation of "learning from experience" to estimate expected rewards and thus approximate the optimal policy based on those rewards.

2.4.2 Genetic Algorithms

Genetic algorithms are another family of heuristics for optimal control that has been explored in disease outbreak management. They are biologically motivated, simulated systems based on natural selection and genetic recombination to achieve the 'fittest' solutions (Baluja and Caruana, 1995). The main components of a genetic algorithm are: a fitness function, a population of chromosomes, a selection operator to choose which chromosomes reproduce, a crossover function to produce the next generation of chromosomes, and mutation of chromosomes in the new generation. To better understand these components, they will be discussed in the context of disease management under limited resources.

Suppose there is an impending outbreak and it has been identified that n cities should receive the first wave of intervention. However, there are only enough resources to intervene at one city per day. The goal of management is to choose the sequence

in which the n cities should receive an intervention such that the outbreak has the shortest duration. When n is small, the sequence of interventions can be determined through exhaustive search. However, the number of computations needed to arrive at the optimal solution increases on a factorial scale as the number of cities increases. Because genetic algorithms do not evaluate each control, they are a faster alternative to exhaustive search.

To simplify this management problem, suppose that there are five cities that were identified a priori to receive an intervention in the first wave of management. To further simplify this example, we will also assume that the outbreak spreads the same way in every iteration. Thus we attempt to find the optimal sequence of actions for a given outbreak spread. One example of a chromosome for this scenario is: $\{5, 2, 1, 4, 3\}$, where each element in the chromosome, referred to as a gene, represents one of the five cities. A chromosome can be thought of as a control in the decision analysis framework. The goal is to find the chromosome that achieves optimal fitness. In this example, the fitness could be calculated through a function of duration of the epidemic, since the goal is to minimize duration of the outbreak. Fitness in the genetic algorithm is equivalent to expected utility in the decision analysis framework, and also must be specified by the user. The genetic algorithm process begins with a random set of chromosomes, this is the initial population. In this case there are 120 combinations of the five cities (5!), so the initial population can consist of some subset, let's say 20, of these 120 chromosomes. The fitness, a function of outbreak duration, is then calculated for each of the 20 chromosomes. A subset of the 20 chromosomes are chosen to reproduce via a selection operator, let's say 10 are chosen to reproduce. The most common selection operator chooses chromosomes proportional to their fitness, F:

$$P(\text{chromosome}) = \frac{F(\text{chromosome})}{\sum_{i=1}^{N} F(\text{chromosome}_i)} \tag{2.14}$$

The chromosomes chosen via the selection operator, referred to as parent chromosomes, will ideally pass on their favorable characteristics to the next generation. The most common processes of creating diversity in the new chromosomes are: crossover and mutation. Crossover takes a sub-sequence of each parent chromosome after a particular gene, and switches them. For example, suppose the two parent chromosomes are: $\{4, 1, 5, 3, 2\}$, $\{3, 4, 2, 1, 5\}$, and crossover occurs after the second gene. The two offspring are shown in the right-most column of Table 2.2. In this example there is an issue with the new offspring, they have repeated genes. "Cycle" crossover can

Parent Chromosome	Crossover	Offspring
4 1 5 3 2	4 1 | 5 3 2	4 1 2 1 5
3 4 2 1 5	3 4 | 2 1 5	3 4 5 3 2

Table 2.2: Example of Crossover with Issues

be implemented to ensure that the two new offspring each contain five unique genes (Haupt and Haupt, 1998). The process begins with choosing a single gene in the

Parent Chromosome	Offspring (step 1)	Offspring (step 2)
4 1 |5| 3 2	4 1 **2** 3 |2|	4 1 2 3 5
3 4 |2| 1 5	3 4 **5** 1 |5|	3 4 5 1 2

Table 2.3: Example of "Cycle" Crossover

two parent chromosomes and switching them. In Table 2.3, the first switch occurs at the second gene. The first offspring now has two 2's, thus a second switch occurs at the fifth gene, yielding the two offspring in the right-most column. Notice that each of these offspring now have five unique genes, thus ending the cycle crossover process. Mutation provides another source of genetic diversity: two genes are randomly switched in a chromosome of the new generation. Search algorithms for a decision space must balance exploitation and exploration. Crossover exploits previous chromosomes by passing on a subsequence to the next generation. Mutations allow the decision space to be further explored by generating chromosomes that may not have been considered through crossover. A mutation usually occurs with a small probability to allow iterations of exploitation within the new decision subspace generated by the mutation. This process of: calculating fitness for the chromosomes, choosing parent genes, implementing crossover, and implementing mutation is repeated many times until convergence in fitness is reached.

Genetic algorithms have previously been investigated in generating optimal disease control strategies (de Souza and T., 1993; Caetano and Yoneyama, 2001; Yan and Zou, 2008; Liu and Zhao, 2009; Wang et al., 2009). For example, Patel et al. (2005) used genetic algorithms to obtain competitive vaccination strategies to combat influenza epidemics using two historical outbreaks as examples: the Asian pandemic in 1957 and the the Hong Kong pandemic in 1968. There were five age groups that were considered in this study: pre-school, school, young adults, middle adults, and old adults. The goal was to determine, constrained by a limited amount of vaccine, which age groups

to target. The chromosome consisted of five genes, five proportions representing the fraction of individuals to vaccinate for each age group. Fitness was defined as a loss function describing a weighted proportion of vaccinated and unvaccinated individuals in an age group that ultimately became sick over the course of the epidemic; thus lower fitness values were preferred. This method was shown to yield better results than random vaccination.

Genetic algorithms have also been implemented in disease surveillance (Peterson et al., 2004; Levine et al., 2004; Peterson et al., 2005; Blackburn et al., 2007; Duczmal et al., 2007). For example, Husin et al. (2012) used a combination of genetic algorithms and neural networks to predict the number of dengue cases in Malaysia over time. A genetic algorithm was used to determine optimal weights in the neural network, and the neural network predicted disease occurrence. This hybrid model yielded better performance compared to standalone models.

Genetic algorithms are heuristics for optimal control. Unlike dynamic programming, it is possible for genetic algorithms to be trapped in local minima or maxima. This can be addressed by increasing the amount of mutation, increasing the number of chromosomes in each generation, changing the initial population, or implementing additional constraints in the fitness function (Haupt, 1995) . For this reason, some degree of tuning may be required to optimize genetic algorithm performance and speed. Even after tuning, genetic algorithms may not yield an optimal action. Genetic algorithms do not assume the Markov property, as with dynamic programming and some Monte Carlo methods. There has been some exploration in using genetic algorithms to generate policies, but this has not been implemented in disease outbreak management (Chin and Jafari, 1998).

2.5 Network-Based Approaches

One assumption of methods that use compartment models is that the population is fully mixed, meaning every individual has an equal chance of spreading the disease to every other individual (Meyers et al., 2004). If a researcher is unable to make this assumption then a contact structure can be created via a graphical model, known as network analysis. A network is usually expressed through three components: a set of vertices, or nodes (V), a set of edges (E), and weights for each edge (J. Newman, 2006).

Figure 2.4 illustrates a network with seven vertices, $V = \{A, B, C, D, E, F, G\}$ and nine edges, $E = \{AB, AC, AE, BC, BD, CD, CE, DE, EF\}$. By convention, because

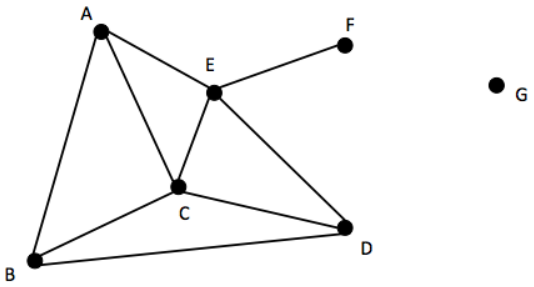

Figure 2.4: Example of a small network with seven vertices (nodes) and nine edges.

no weights are denoted, it can be assumed that each edge is equally weighted. Weights represent the extent of connection between two vertices. For example, suppose that each vertex in the network represented an individual, and an edge was drawn between two individuals if they got into physical contact. A weight representing the number of times the two individuals got into contact with each other could be assigned to each edge. A network is connected if each node can reach every other node by a series of edges. The network in Figure 2.4 is not connected because vertex G is not connected to any of the other vertices.

Networks used in contact tracing and social networks tend to have many nodes and edges. Both of these types of networks are commonly used in disease surveillance. Contact tracing is a retrospective effort, one can determine how an initial case spread the disease (Kretzschmar et al., 1996; Eames and Keeling, 2003; Fraser et al., 2004; Lloyd-Smith et al., 2005; Kiss et al., 2006). This information can be used to target particular individuals, modes of transportation, or particular locations for future outbreaks. Since both of these types of networks tend to be quite large it is impractical to answer questions such as, which node would most affect the network if removed, since there would be too many nodes to feasibly explore. Instead, one could ask what proportion of vertices need to be removed to prevent a large-scale outbreak (J. Newman, 2006). This type of question targets characteristics of the network such as clustering, node degrees, and resilience. Clustering describes the connectedness of nodes. For example, if an infected individual came into contact with a susceptible individual, and the susceptible individual frequently contacts individuals at a community center, then there is a high probability that the individuals in the community center will become infected. The infected individual, susceptible individual, and the individuals at the community center are all connected in a cluster. By understanding this pathway, policy-makers would be encouraged to target individuals within a cluster that are connected to a large number susceptible individuals. The degree of

a node refers to the number of edges that is connected to that particular node, and individuals with a higher degree would ideally be targeted first for an intervention because they highly affect the network. Resilience refers to how the network responds if a particular node is removed. Researchers may find that if certain individuals were to become infected, then the rate of spread of the disease may increase. Targeting these individuals would potentially prevent an even more wide scale outbreak.

Policy-makers can change the structure of a contact network through targeted interventions (Edmunds et al., 1997; Meyers et al., 2004; Eames, 2008; Salathe and Jones, 2010; Preciado et al., 2013). As an example, Salathe and Jones (2010) traced disease spread using friendship data collected from Facebook. Nodes in the network were ranked using measures of centrality. Random walk centrality, for example, determines the amount of times a node was passed in a random walk. Stochastic algorithms were also explored in this study, specifically determining which neighboring nodes should be vaccinated when a particular node was infected. This concept of identifying "bridging" nodes is one advantage of network-based approaches. This study found that targeting individuals that bridge communities was more effective than targeting highly connected individuals. In another study, a social network was constructed based on the question, "who are your playmates?" to monitor the spread of influenza (Cauchemez et al., 2011). This study suggested that boys were more likely to transmit the disease to other boys than to girls. This study also found that there were waves of transmission between schools and households. Like with regression models, network-based approaches help construct general rules for management and disease spread.

One shortcoming of this method is that full contact networks relevant to the spread of infectious diseases are generally not known. For example, in the United States the structure of regional farm and livestock movement networks are not known, including missing data about farm size, location, and animal movements. (Craft, 2015). This method can be used as a decision support tool, as with other disease surveillance methods, however this method has no measure of utility. One extension to network analysis that may allow for exploration in optimal control is agent-based modeling. Agent-based modeling simulates interactions between individuals, or agents, in a community using an assumed network structure (Shiflet and Shiflet, 2014). This approach has been implemented in investigations of modeling disease spread assuming certain rules, and may have a future in exploring optimal disease outbreak control methods (Eubank et al., 2004; Situngkir, 2004; Rahmandad and Sterman, 2008; Perez and Dragicevic, 2009; Ajelli et al., 2010)

2.6 Discussion

This chapter reviewed various computational methods for managing disease outbreaks. Each present their own strengths and weaknesses as outlined in Table 2.4. The primary split between the previously discussed methods is the goal of constructing optimal strategies. Many computational approaches have been investigated for optimal disease outbreak management (Dasaklis et al., 2012). This review focused on three methods: dynamic programming, Monte Carlo methods, and genetic algorithms. Dynamic programming (and exhaustive search) is the only method discussed that has the ability to achieve optimal policies in small, finite spaces, and Monte Carlo methods and genetic algorithms are heuristics for generating optimal controls. This brings into question whether or not it is worth searching for optimal solutions. Yan and Zou (2007) were interested in finding the optimal combination of rate of quarantine and rate of isolation to manage the SARS outbreak in Beijing, 2002. Two approaches were compared: classical optimization and genetic algorithms. Both approaches yielded virtually the same number of infected individuals and costs. In addition the controls generated from the genetic algorithm were much simpler to implement from a policy standpoint; the rates of quarantine and isolation decrease in a stepwise manner however the classical optimization approach required the rates to change every day. Thus it may be easier and more practical to use heuristics in generating a decision. However, it is important to note that the construction of optimal policies have not been investigated in genetic algorithms or Monte Carlo in the context of disease outbreak management.

After presenting some of the strengths and weaknesses of each method it is natural to ask which method is the best. The answer to this question depends on the policy makers' and researchers' goals. The flowchart in Figure 2.5 may provide some insight to choose a method. The primary distinguishing characteristic is whether or not the policy maker is interested in pre-emptive management. This separates out many of the disease surveillance methods, and other methods for constructing rules of thumb for prioritization schemes. Fitting a model is another separating characteristic; a researcher may be interested in generating quantifiable relationships between covariates and the outcome which would suggest the use of regression or neural networks.

One limitation of all the presented methods is the ability to generate optimal controls in a complex decision space. Dynamic programming suffers from the curse of dimensionality in large decision spaces and storage limitations with large look-up tables. Monte Carlo methods and genetic algorithms tend to be trapped in local minima or not explore enough of the decision space. Complexity in disease outbreak

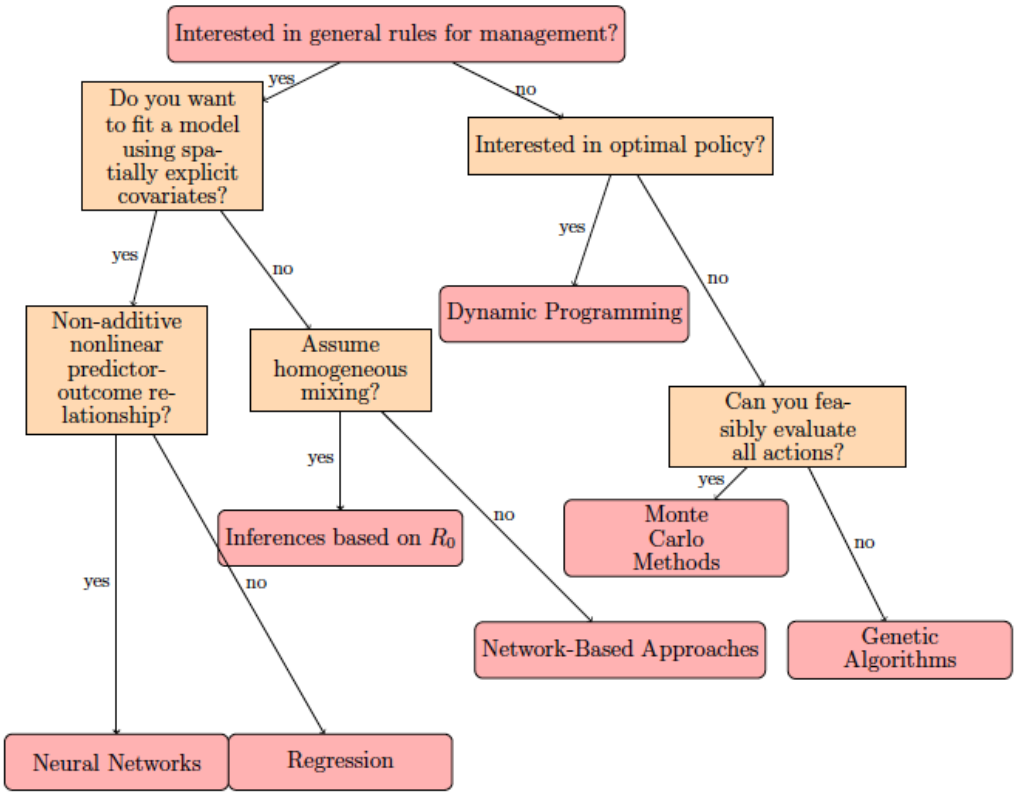

Figure 2.5: Flowchart for use of method

Method	Strengths	Weaknesses
Disease Surveillance Methods	- Inferences from regression models can aid prioritization schemes - Easy to implement - Model flexibility with neural networks	- Model specification - Parameter tuning for neural networks - Decision support tool, does not generate optimal controls - Used for prospective outbreak detection, not during the outbreak itself
Dynamic Programming	- Generates optimal policy	- Computational and storage issues as decision space becomes more complex
Monte Carlo Methods	- Understand uncertainty of control measure - Easy to implement for small decision spaces	- Computational and storage issues as decision space becomes more complex - Heuristic - Has not been explored in generating optimal polices in context of outbreak management
Genetic Algorithms	- Can be paired with other methods to improve performance - No need to explore every control strategy	- Computational and storage issues as decision space becomes more complex - Heuristic, can be trapped in local optima - Has not be explored in generating optimal policies in context of outbreak management - Parameter tuning may be required to improve performance/speed
Network-Based Approaches	- Does not assume population is homogeneously mixed - Provides retrospective approach to trace progression of disease	- Decision support tool, does not generate optimal controls - Contact networks generally not known

Table 2.4: Strengths and Weaknesses of Each Method

management arises when constructing an optimal policy under resource constraints. Due to the limitations of the aforementioned methods, a different approach would be required for management in this more complex scenario. Deep Q-networks (DQN) have previously shown strong performance in complex decision spaces. This method has not yet been explored in disease outbreak management, and will be explored in the next chapter.

CHAPTER 3

ON USING DEEP Q-NETWORKS TO MANAGE THE 2001 UNITED KINGDOM FOOT-AND-MOUTH DISEASE OUTBREAK

3.1 Introduction

Optimal decision-making presents computational challenges in complex decision spaces. As discussed in the previous chapter, one common method for optimal control, dynamic programming, encounters computational storage problems while updating the expected utility function in large decision spaces (Ge et al., 2010). In simple Backgammon game situations using a look-up table for expected rewards, as with dynamic programming, results in an agent to be trained perfectly. However, training an agent to the full game scenario would require a look-up table of approximately 10^{20} states. Rather than pursue methods which calculate exact expected utility, it may be easier and more feasible from a logistical standpoint to pursue methods that estimate expected utilities (Yan and Zou, 2007). Deep Q-networks (DQN) have been historically successful in complex decision spaces. Tesauro (1995) successfully trained a DQN agent to play Backgammon better than world-class human professionals. This method has also recently been explored in complex decision spaces. Mnih et al. (2015) used DQN to train an agent to play the Atari 2600 games better than a trained human professional.

Complex decision spaces also arise in disease outbreak management under resource constraints. Disease outbreak state spaces traditionally consist of discrete covariates, such as the number of infected premises at time point t, which is well-suited for look-up tables (Ge et al., 2010). However, this state space does not capture spatial correlations inherent in disease spread. A more complex state space, such as an illustrative map of the disease outbreak at time t, would be required to capture spatial relationships between locations. In addition, a policy-maker may only have the resources to provide interventions to x out of n locations. This action space construction would require $\binom{n}{x}$ columns in a look-up table. This action space coupled with a complex state space would overwhelm the storage capacity of look-up tables used in dynamic programming. Because DQN has been successful in applications with high dimensional decision spaces, e.g. - video games, I investigated this method in its ability to manage the 2001 United Kingdom foot-and-mouth disease (FMD) outbreak under resource constraints in four different scenarios.

3.2 Deep Q-Networks

The goal of DQN is the same as that of Q-learning: construct an optimal policy for a given Markov decision process (MDP). DQN combines Q-learning, to update the action value function based on experiences, with convolutional neural networks (CNN) to estimate and store network weights. As previously mentioned in the introduction chapter, CNN architecture is well-suited for image data because low-level feature information can be easily extracted and combined with other low-level features to utlimately construct more abstract concepts (Deshpande, 2016). The algorithm begins with a simplified version of the Bellman optimality equation:

$$Q^*(s,a) = E_{s'}[r + \gamma max_{a'} Q^*(s',a')|s,a] \tag{3.1}$$

By incorporating convolutional neural networks into the Q-learning algorithm, the optimal action-value function is approximated, $Q^*(s,a) \approx Q(s,a;\theta)$ where θ represents the collection of convolutional neural network weights. As with traditional Q-learning, the estimate of $Q^*(s,a)$ improves with increased agent experience. The improvement in the estimate mainly occurs through improvement of the convolutional neural network weights, θ. With each iteration of training, the squared error loss is calculated:

$$L(\theta) = E_{s,a,r}(E_{s'}(r + \gamma max_{a'} Q^*(s',a';\theta^-)) - Q(s,a;\theta))^2 \tag{3.2}$$

where θ^- refers to a previous set of weights. Following the convolutional neural network algorithm, partial derivatives of Equation (3.2) are calculated with respect to θ, and stochastic gradient descent can be performed to update the network weights.

There are two additional components to the DQN algorithm that assist with convergence: experience replay and target updating. With experience replay, an agent can experience the effects of taking "realistic" actions from given states without having to execute those actions in real time (Lin, 1993). These actions are realistic because they were previously taken from the same given state. Experience replay saves the experience (s_t, a_t, r_t, s_{t+1}) from each time step in a double ended queue, or "deque", E of pre-specified length, N. Each time the convolutional neural network is fit, a random sample of experiences from E is drawn, and is used in the weight updating process. The deque E contains only the N most recent experiences to ensure the estimates are using the most relevant data. Consecutive experiences are highly correlated, thus randomizing previous experiences reduces correlation in the estimates of $Q(s,a;\theta)$ (Mnih et al., 2015).

Target updating refers to using a separate network for the target, or the true observation, $Q(s', a'; \theta^-)$. The target network and the Q-network, for $Q(s, a, \theta)$, initially have the same weights, θ. The Q-network weights are updated with each time step, while the target network weights remains the same. After C time steps, the Q-network is cloned and equated to the target network for the next C time steps, as illustrated in Figure 3.1. This method may require more episodes of training and

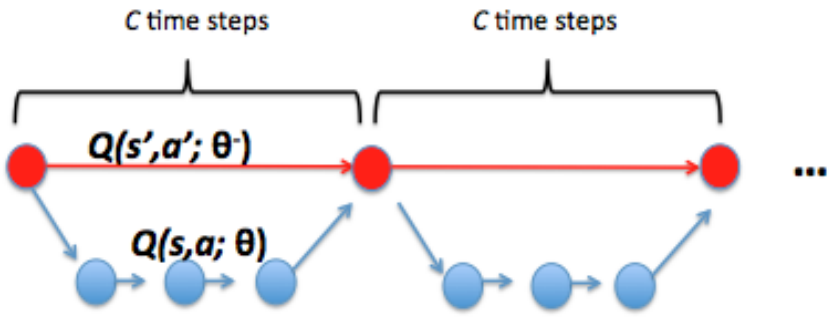

Figure 3.1: Structure of target updating in DQN

can be computationally expensive, but can greatly improve the stability of training and prevent optimal policy divergence (Lillicrap et al., 2016). The complete DQN algorithm is presented in Figure 3.2.

Initialize experience replay memory E to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with random weights $\theta^- = \theta$
For each episode
    Initialize s
    Until s is terminal
        Select action a_t randomly with probability ϵ
            otherwise select a_t using $a_t = argmax_a Q(s, a; \theta)$
        Implement a_t and observe r_t, s_{t+1}
        Store (s_t, a_t, r_t, s_{t+1}) in E
        Sample random mini batch of (s_i, a_i, r_i, s_{i+1}) from E
        Set $y_i = r_i$ if episode terminates at $i + 1$
            otherwise, $y_i = r_i + \gamma max_{a_{i+1}} \hat{Q}(s_i, a_{i+1}; \theta)$
        Perform stochastic gradient descent on $(y_i - Q(s_i, a_i; \theta))^2$
            w.r.t network parameters, θ
        Every C steps let $\hat{Q} = Q$

Figure 3.2: The DQN algorithm (Mnih et al., 2015)

3.3 Methods

3.3.1 States, Actions, Rewards

In this study, our primary goal was to simulate the management of the 2001 United Kingdom FMD outbreak using DQN in different scenarios. As with any MDP, the states, actions, and rewards were pre-specified. To better capture the spatial relationships between farm locations, I defined the state at time t using a map of the disease outbreak. Figure 3.3 illustrates the construction of the state for a given epoch of training. I first plotted the farm locations and identified the farm-level infection status on an $m \times m$ kilometer space (represented in the left panel in Figure 3.3, black represents an infected farm, and white represents a susceptible farm). I then overlaid a grid on the plot to create "pixels," i.e.- grid squares (represented by the second panel in Figure 3.3). Traditionally, a state in DQN with image data is defined by the red-green-blue (RGB) values for each pixel (Mnih et al., 2015). In this study, I used the infection status of the farm in a given pixel. I used the following as a key to construct the state: susceptible farm- 0, infected farm- 1, culled farm/no farm- 2 (represented in the right panel in Figure 3.3). I implemented this construction of state generation at every epoch of training. Each pixel has no more than one farm. This constraint helped determine potential dimensions of each grid square. In this example, on a 10 $\times$ 10 kilometer space, each grid square had dimension 1 $\times$ 1, to ensure that no more than one farm occupied a grid square. The grid square dimension can easily be smaller to better capture relative distance between farms, however this creates a larger 2-D array, causing computational challenges with stochastic gradient descent and the serial updating nature of DQN.

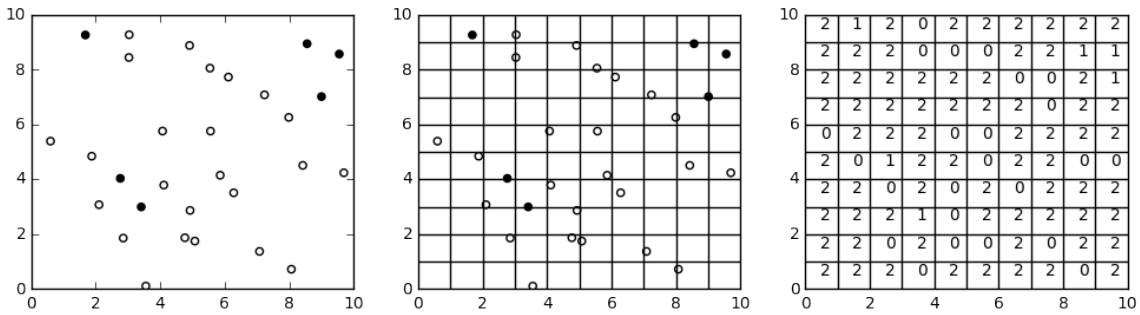

Figure 3.3: Constructing the state at a time point consists of three steps 1) plot the points of the farm locations and color code by infection status (black- infected, white- susceptible), 2) overlay a grid on the previous plot, 3) trichotomize the infection status

At each epoch of training, I tasked the DQN agent with determining which farms should be culled under resource constraints. In Figure 3.3 there are 30 farms in the

10×10 kilometer space. Suppose that there are resources to cull all of the cattle in any five farms, per day. Rather than defining the action space with $\binom{30}{5}$ possibilities at time t, I presented the agent with a smaller action space. At time step t, the DQN agent chose one farm out of the 30 to cull. In that same time step, the DQN agent chose a second farm to cull out of the remaining 29. Once the DQN agent chose five separate farms to cull, then the outbreak evolved and a new state and reward were observed. This action space construction reduced the number of possible actions in the first day of management from $\binom{30}{5}$ to 140 (30 + 29 + 28 + 27 + 26). There are other approaches to accommodate decision spaces with large action spaces, such as least squares policy iteration with multi-action (Wang and Yu, 2016). However, techniques to accommodate decision spaces with large state and action spaces are still under study.

The immediate rewards, r are closely tied to the objectives of disease management. Ideally, an outbreak should be terminated as quickly as possible with minimal costs. For this reason I included multiple components to the immediate reward structure. A -100 reward was observed for every farm culled, which encouraged conservative culling in the management process. A terminal reward of the number of cows alive at the end of an episode was also observed, which encouraged cost-effective behavior. Because FMD has a latent period of five days, it was possible for outbreaks to resurge. This would cause policy-makers to have to wait to ensure that the outbreak was truly terminated. For this reason there was a -500 reward associated with the agent assuming the outbreak was terminated, when in truth an outbreak resurgence was about to occur. I used this immediate reward structure for each of the four investigated case studies. I chose the values of the immediate rewards through prioritization of objectives; diminishing outbreak resurgence was the highest priority and was thus given a more negative reward for not being met. In a realistic setting, specification of objectives can impact the final policy. Probert et al. (2015) illustrated that final controls can differ based on the metric that defines success. For this reason, it is imperative that a decision-maker constructs clear objectives prior to management.

3.3.2 Outbreak Generation

I incorporated the 2001 United Kingdom FMD dynamics with the DQN algorithm to generate the next state after an action was implemented. The 2001 United Kingdom FMD outbreak was characterized with a susceptible-exposed-infected-recovered (SEIR) compartment model at the farm level. There was a five day latent period

before a susceptible farm became infected, represented by the exposed compartment. I simulated the outbreak using an individual-based FMD model as specified by Keeling and Rohani (2007). The rate at which a susceptible farm became infected was represented by:

$$\lambda_i = N_i s_{cow} \sum_{j \in infectious} N_j \tau_{cow} K(d_{ij}) \tag{3.3}$$

where N represents the number of cows at a farm, s represents a cow's susceptibility to the disease, τ represents a cow's transmissibility of the disease, K represents the transmission kernel, and d_{ij} represents the distance between the susceptible farm i and the infectious farm, j. I selected the following parameters, s, τ, and K, for each case study to better accommodate the characteristics of each landscape (e.g. - size of the space, positioning of farms, etc). If s or τ were too large, then the outbreak would spread too quickly, preventing any type of intervention under resource constraints to be effective. If the transmission kernel was a distribution with very large tails, then there would be a larger chance for farms that were very far away to become infected in a single time step. This may be feasible in practice due to cattle movement, movement of equipment, etc. The code to implement the simulated outbreak is presented by Keeling and Rohani (2007).

I combined the DQN algorithm from Figure 3.2 with the 2001 United Kingdom FMD outbreak dynamics, to construct a modified DQN algorithm, (Figure 3.4). The functions to implement action selection, experience replay, and target updating in Python using `keras` and `tensorflow` are presented in Appendix A.

3.4 Case Studies

I investigated four different cases studies. The first case is a proof of concept that DQN can be used to optimally terminate a disease outbreak. The succeeding two cases present small-scale examples that address particular aspects of outbreak management under resource constraints. The final case explores scaling the decision problem. For each case study, I did not change the initial state between episodes of training. In practice, a policy-maker would provide the initial state of the outbreak and request the best sequence of actions for management. Thus, this method overfits to the starting state of the outbreak. Although the starting state for each case is the same, the outbreak progression will vary due to the stochastic nature of the outbreak algorithm. This stochasticity makes the decision problem more challenging for the DQN agent.

Initialize experience replay memory E to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with random weights $\theta^- = \theta$
For each episode
    Initialize and pre-process s
    Until s is terminal
        Select action a_t randomly with probability ϵ
            otherwise select a_t using $a_t = argmax_a Q(s, a; \theta)$
        Implement a_t and observe r_t
        If the daily culling capacity has not been reached
            then s_{t+1} is determined by new farm cull and pre-processing
        If the daily culling capacity has been reached
            then evolve the outbreak and pre-process to observe s_{t+1}
        Store (s_t, a_t, r_t, s_{t+1}) in E
        Sample random mini batch of (s_i, a_i, r_i, s_{i+1}) from E
        Set $y_i = r_i$ if episode terminates at $i + 1$
            otherwise, $y_i = r_i + \gamma max_{a_{i+1}} \hat{Q}(s_i, a_{i+1}; \theta)$
        Perform stochastic gradient descent on $(y_i - Q(s_i, a_i; \theta))^2$
            w.r.t network parameters, θ
        Every C steps let $\hat{Q} = Q$

Figure 3.4: The DQN algorithm, modified to accommodate the 2001 United Kingdom FMD management environment (modifications represented in red).

3.4.1 Case 1: Which infected farm to leave out?

In the first case study I placed 30 farms randomly on a 10×10 kilometer grid. The initial pixelated state is provided in Figure 4.3. The number of cattle on each farm was uniformly distributed between 25 and 500, with a total of 6965 cows in the entire landscape. The range of cattle reflected realistic farm sizes in Cumbria, United Kingdom in 2001 (Tildesley et al., 2010). A cow's susceptibility to the disease, s, and a cow's transmissibility of the disease, τ, were both chosen to be 6.3×10^{-5}. I selected transmissibility to be slightly larger than specified by Keeling and Rohani (2007) to accommodate a more realistic spread on a smaller landscape. The transmission kernel in this case was: $K(distance) = \frac{1}{distance+400}$. This kernel has a large tail over the possible distances, meaning that farms farther away have a higher probability of becoming infected, relative to a narrower-tailed kernel. This will present some challenges for the DQN agent, because there may not be a "learnable" culling pattern for management. "Learnable" here refers to discovering rules that would guarantee the management objectives to be met. I chose six farms at random to be initially infected, represented by the black points in Figure 4.3. Because there were initially

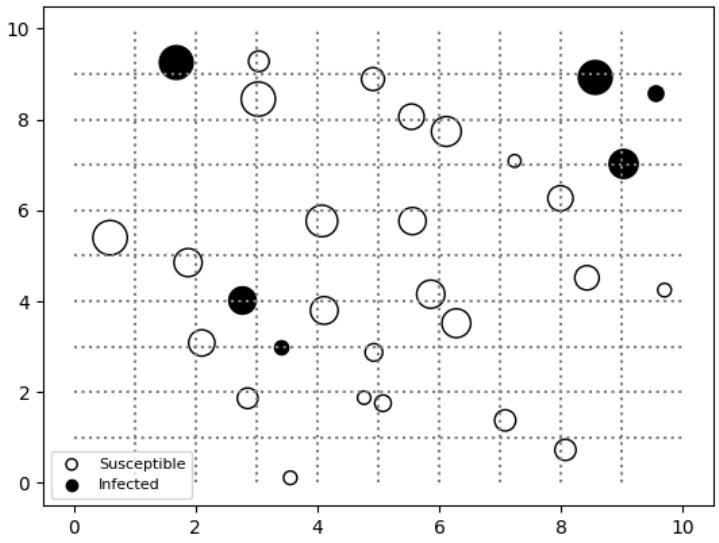

Figure 3.5: Initial state for the first case study. Black points represent locations of initial infected farms. Dot size is proportional to the number of cattle present at the farm (range: 25 - 500).

six infected farms, this case can be thought of as: which infected farm to leave out. Intuitively, the outbreak can be terminated by culling all six infected farms. However, if the first five infected farms were incorrectly chosen then the neglected infected farm could create secondary infections on the second day.

Figure 3.6 illustrates an instance of the actions suggested by the DQN agent after 1,200 episodes of training. The first five panels represent the five culls (represented by the red x's) in the first day of management. The sixth panel represents the result after the five culls and evolving the outbreak. In the seventh panel, the second day of management, the outbreak is terminated. With this sequence of culls, the outbreak never spreads. The left panel in Figure 3.7 illustrates the smoothed trajectory of total reward over the 1,200 episodes of training. It is common to present smoothed training trajectories using a moving average to better represent the progression of learning, and mitigate the uninformative oscillations, (Figure 3.7).

At the beginning of training, the total reward was about -3,000. This suggests that nearly every farm was culled at the beginning of training. With the pre-specified immediate reward structure and the DQN algorithm, the agent learns how to optimally terminate the outbreak while meeting all of the objectives. The trajectory of training begins to plateau at about 800 episodes. This suggests that the DQN model has begun to stabilize. Once the DQN agent was trained, the outbreak was simulated

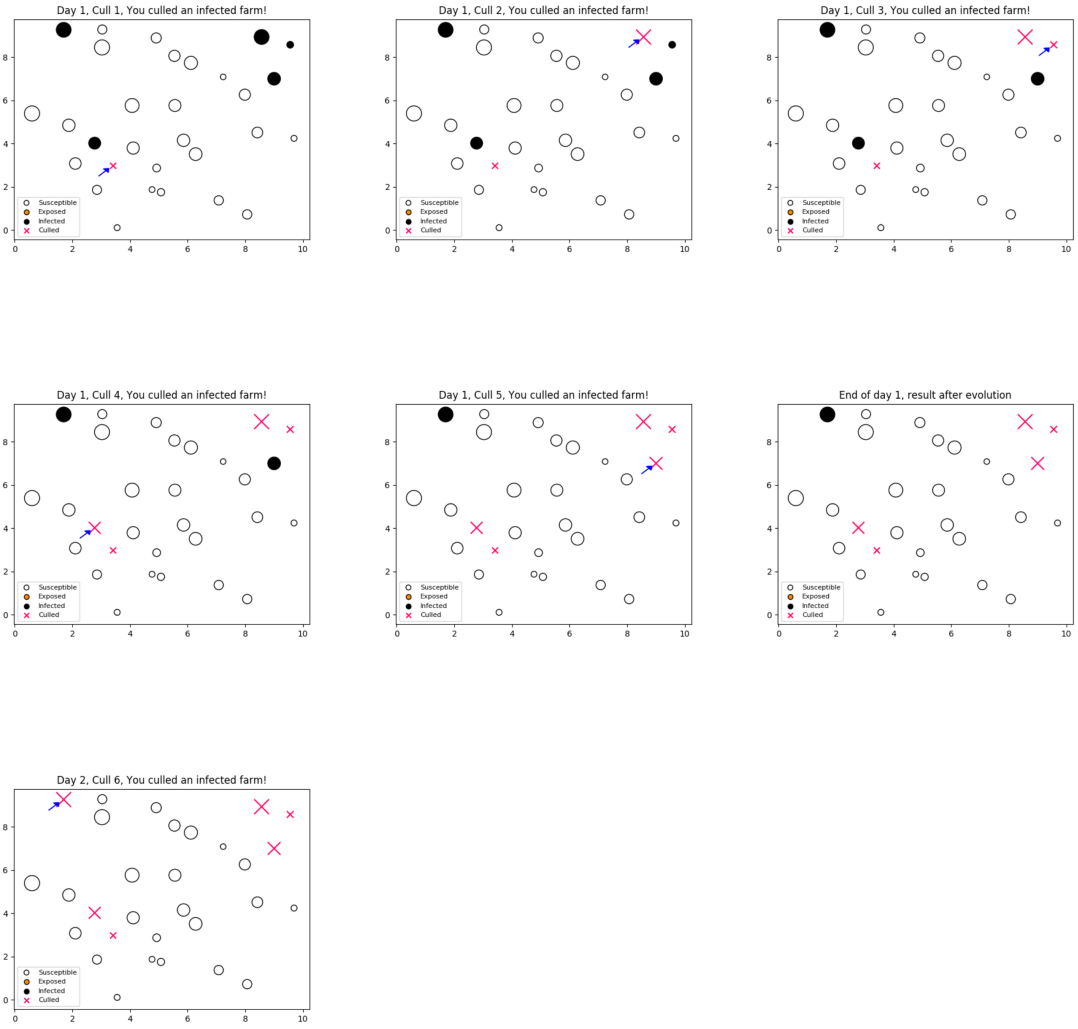

Figure 3.6: One instance of how DQN agent optimally terminates the outbreak in the first case study after model is trained. The first five panels represent an action (represented by the red x's). The sixth panel represents the result of outbreak evolution after the first five culls. The seventh panel represents the final cull on the second day.

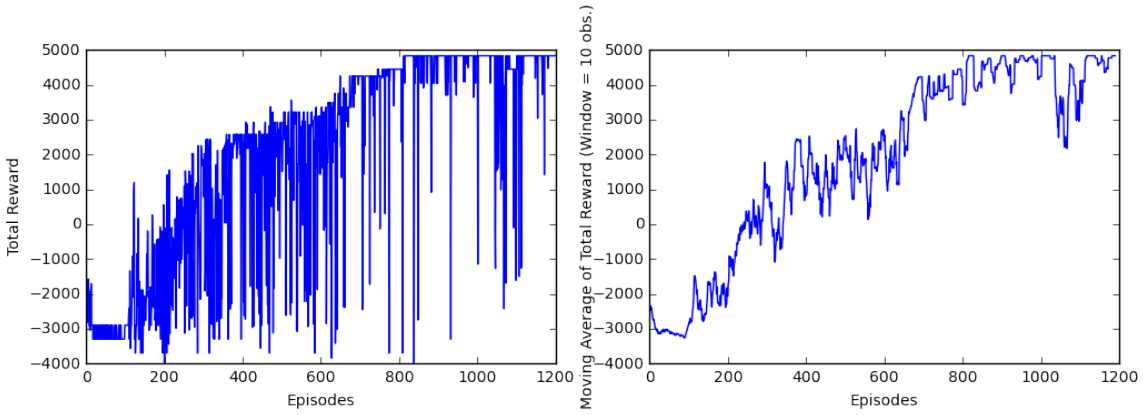

Figure 3.7: The left panel presents the raw total reward after each episode in the first case study. The right panel presents the moving average (smoothed over 10 episodes). The smoothed total reward better illustrates agent improvement, without being preoccupied in the individual oscillations.

2,000 times with management decisions informed by the policy from the DQN. Table 3.5 in Appendix B illustrates that the outbreak never spreads; there were always six farms culled. In addition, the DQN agent never assumes that the outbreak is terminated when in fact there are exposed farms (average outbreak resurgence: 0, 95%CI (0, 0)).

3.4.2 Case 2: Faster outbreak and conservative resource constraints

In the second case study, I explored a more challenging decision problem. The initial state was the same as the first case study: 30 farms, six farms initially infected, with the number of cattle per farm unchanged from the first case. However, in this second case study, I assumed that all of the cattle at any one farm could be culled per day. It was almost guaranteed that there would be at least one additional infected farm before the end of an episode. Recall FMD has a latent period of five days. If only one farm could be culled per day, then each of those five days will have a chance of at least one farm becoming exposed. In fact, in 2,000 simulations of testing the trained DQN agent, an average of 5 farms were exposed after the first cull (95%CI (3, 6)). I also slightly increased a cow's transmissibility of FMD to 8.4×10^{-5}. Both the increase in transmission and a lower daily culling capacity presented more of a challenge for the DQN agent.

Based on the trajectory of total reward during training, the DQN agent was able to successfully terminate the outbreak (Figure 3.8). However, there is increased variation in the trajectory of total reward, even after smoothing with a moving average,

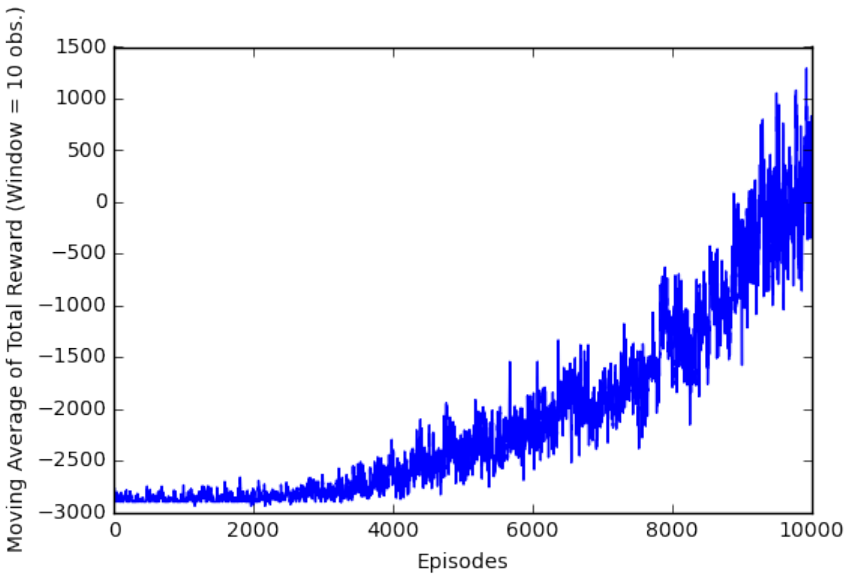

Figure 3.8: Smoothed total reward trajectory for second case study over 10,000 episodes of training.

compared to the total reward trajectory in the first case study. This is due to the higher degree of stochasticity of the outbreak caused by the increase in transmission and decrease in daily culling capacity. Even if the DQN agent implemented the same set of actions in two realizations of the disease outbreak, the outbreak progression between those two instances may be vastly different. Despite this, the trajectory of total reward continued to increase over the course of training. Also notice in Table 3.5 that there were some outbreak resurgences (mean = 1, 95%CI = (1, 2)), in comparison to the first case study. However, this phenomenon was expected due to the complexities of this decision problem compared to the first case study.

Figure 3.9 illustrates an instance of the sequence of actions suggested by the DQN agent, after 10,000 episodes of training. Each of the orange points represents an exposed farm. Recall that although we are working with an SEIR compartment model, the DQN agent only receives labels regarding if a farm is susceptible to the disease, infected, or culled/no farm, and the agent needs to determine what these labels mean. The DQN agent never received information regarding whether or not a farm was exposed, because this information was not available in the state. In a realistic setting, decision-makers will not know if cattle are exposed to the disease, they will only know if the cattle are infected. This presents an additional feature that the DQN agent had to learn: exposed farms will eventually become infected. Figure 3.9 illustrates that some exposed farms and susceptible farms were culled. In some cases the DQN agent correctly predicts which farms were exposed and culls them.

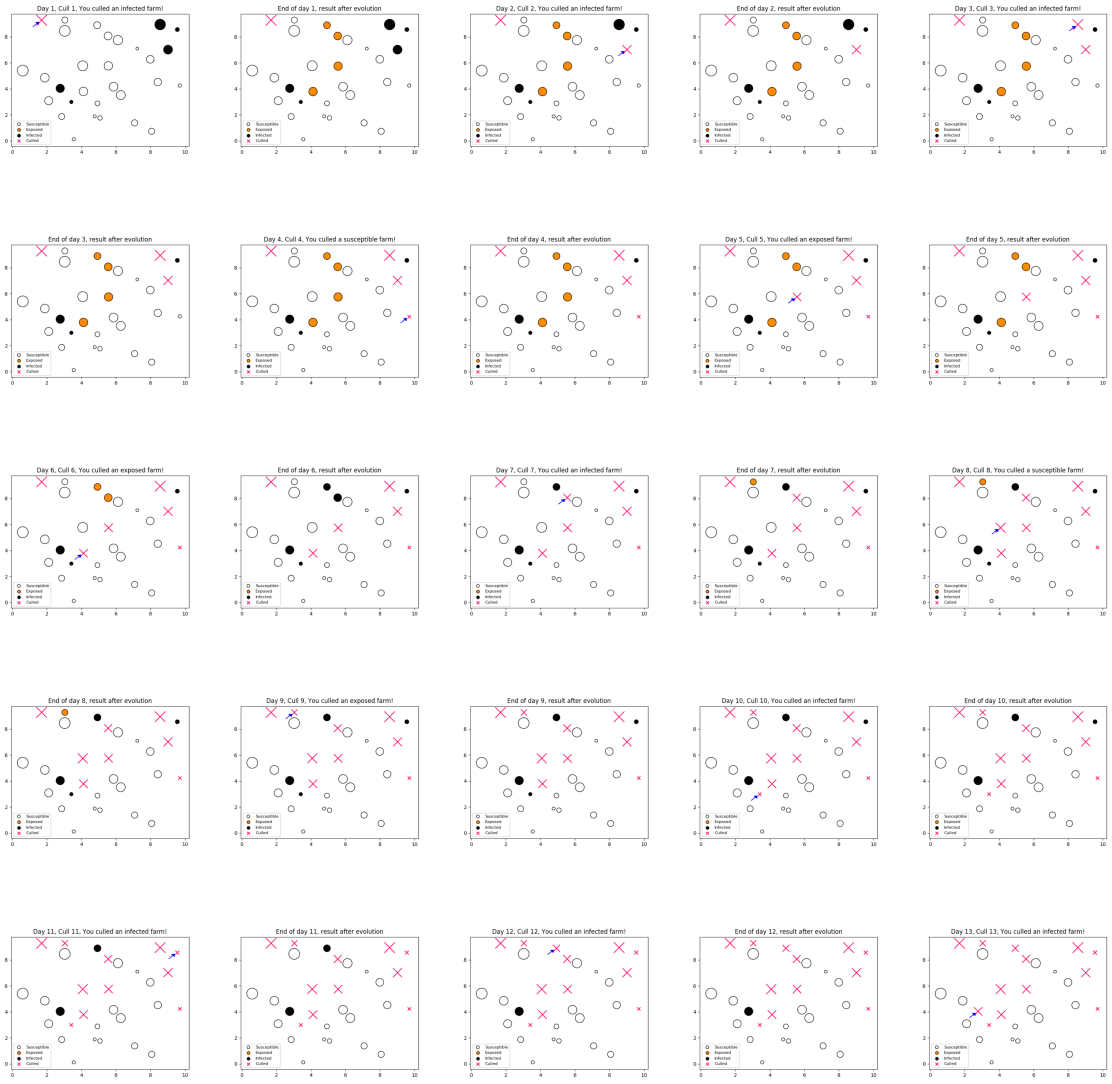

Figure 3.9: One instance of how DQN agent optimally terminates outbreak for case 2 after model is trained. Every other panel represents an action (represented by the red x's). After an action is implemented the outbreak evolves, represented by the remaining panels.

In some instances, the DQN agent calculates a higher utility in culling a susceptible farm than an infected farm (7th and 15th panels in Figure 3.9). This result could be sensitive to the amount of training. I pre-specified number of episodes for training. To investigate if management differences were based on different numbers of episodes of training, I performed a sensitivity analysis.

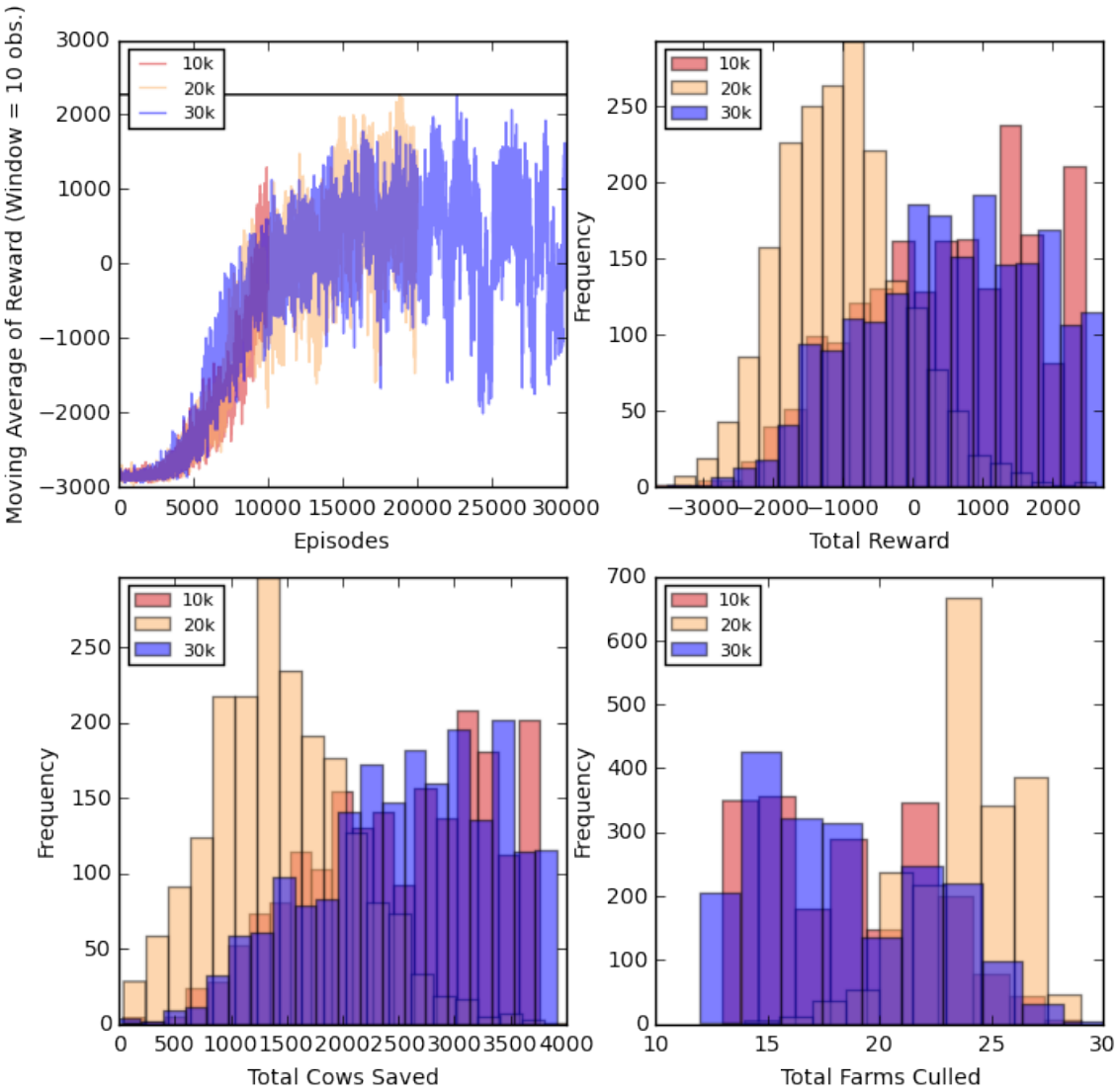

Figure 3.10: Sensitivity analysis for second case study based on different durations of training. Top left panel illustrates trajectory of training, with moving average of total reward on the y-axis. Horizontal black line represents the maximum total reward obtained from training for 20,000 episodes. Top right panel presents the total reward collected from 2,000 episodes of testing. Bottom left panel presents the number of cows saved after 2,000 simulations of testing. Bottom right panel presents the number of farms culled after 2,000 simulations of testing.

I originally trained the DQN agent in this case study for 10,000 episodes, which took about half a day to run in serial. To assess sensitivity to stopping criteria, I

trained the DQN agent separately for 20,000 and 30,000 episodes (which took a day to run in serial, and a day and a half to run in serial, respectively) with all other Q-learning and convolutional neural network parameters remaining the same. The results of this sensitivity analysis are presented in Figure 3.10. The top left panel of Figure 3.10 presents the moving averages of total reward under each stopping criteria. When the DQN agent was trained for 20,000 episodes, the trajectory begins to plateau but there is a higher degree of variation toward the end of training. When the DQN agent was trained for 30,000 episodes, the total reward was never higher than that of the total reward from 20,000 episodes of training, represented by the horizontal black line. In addition, under 30,000 episodes of training there was a high degree of instability in the last 10,000 episodes of training. The total reward oscillated between -2000 and 2000. After each DQN model was trained, 2,000 simulations of testing was implemented. Referring to top left panel of Figure 3.10, the histograms of total reward for 10,000 and 30,000 episodes of training were very similar. Decomposing the elements of total reward into: number of farms culled (bottom right panel), total cows saved (bottom left panel), and number of outbreak resurgences (average # outbreak resurgences: 1, 95%CI (0,2) for all three stopping criteria) the results from 10,000 and 30,000 episodes of training were still very similar. The proportions of susceptible farms, exposed farms and infected farms culled over the course of 2,000 simulations of testing are presented in Table 3.1. Under more episodes of training, there is a small decrease in the number of susceptible farms are culled, resulting in more infected farms culled.

Iterations of Training	Susceptible Farms	Exposed Farms	Infected Farms
10,000	21 (7, 41)	18 (9, 31)	61 (45, 73)
20,000	12 (4, 23)	16 (11, 25)	72 (59, 81)
30,000	19 (6, 38)	18 (9, 29)	64 (48, 74)

Table 3.1: Proportions of susceptible, exposed, and infected farms culled over 2,000 simulations of testing. The results are presented as average proportion (95% confidence interval).

The results from 20,000 episodes of training generated worse outcomes in comparison to the other two stopping criteria. The total reward was generally lower, and more farms were culled overall. This sensitivity analysis highlights one of the challenges of using DQN, high sensitivity to hyperparameter values. Sprague (2015) illustrates this point as well in an investigation to train a DQN agent to play the Atari 2600 games. Different values of step size and discount rate were separately used to train a DQN agent and the resulting total reward trajectories were were highly sensitive to choices of hyperparameters.

DQN successfully halted the FMD outbreak under the conditions specified for the second case study. The management objectives were successfully met with a small number of episodes of training. I performed an additional sensitivity analysis to assess the use of a "no action" management option. Decision-makers may want to have a choice to wait and see if there are additional infections, rather than cull non-infected farms.

3.4.3 Case 2: Investigating the "no action" management option

In this section I investigated the use of a "no action" management option for the second case study. The results of the second case study illustrate that, in some instances, the DQN agent found a higher utility in culling susceptible farms over exposed or infected farms. It may be possible that the DQN agent was waiting to see if more infected farms arose, by means of culling susceptible farms. Recall, there was a higher negative reward for the DQN agent incorrectly assuming the outbreak was terminated, -500 reward, versus -100 reward for culling a single farm. Thus, in this section I evaluated the utility of taking no action over culling non-infected farms. The choice to not cull a farm had the same reward as culling a farm at a given time point. It should not be assumed *a priori* that no action is preferable over culling a farm, this would be determined through training the DQN agent. Under this construction, the action space changed to $\{1, 2, 3, ..., N, N + 1\}$ to accommodate the additional action of doing nothing. The no action management option was limited to six consecutive times. This constraint accommodated the five-day latent period of the FMD outbreak. If the DQN agent chose to do nothing six consecutive times, I then forced the agent to choose another action in the next time step because new infections would arise. All other states, rewards, and hyperparameters remained the same as the second case study.

Based on the trajectory of training in Figure 3.11, using a no action management option resulted in slightly slower learning. This was expected because when the agent chose to do nothing, it still received a -100 reward. This caused the agent to reach a total reward lower than -3,000, the minimum total reward when a no-action management option was not considered. One distinct feature of Figure 3.11 is that the total reward trajectories are nearly similar toward the end of training. Once the DQN agent under the no-action scenario was trained, I performed 2,000 simulations of testing. A closer examination of the frequency chart of no action revealed the reasoning behind the similar tail-end behavior of the total reward trajectories. Table

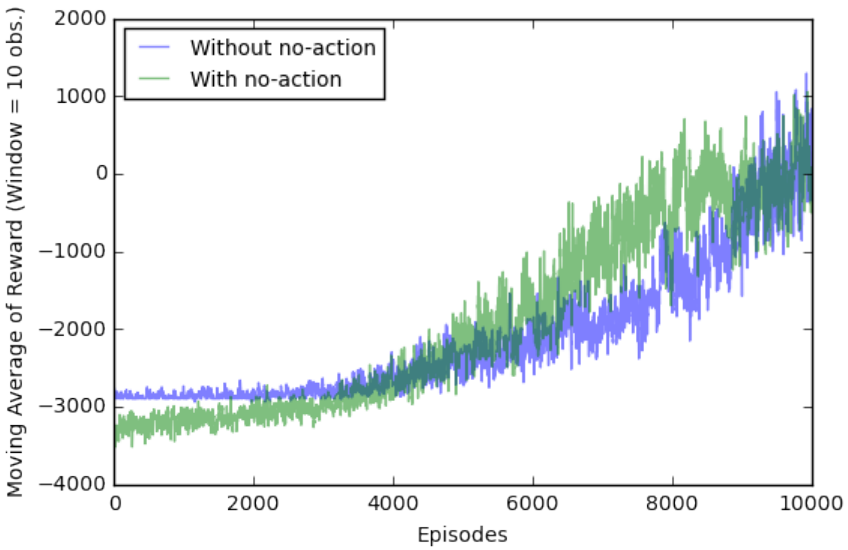

Figure 3.11: Case 2: trajectory of training represented by total reward, over 10,000 episodes with a no action management option (green) to without a no action management option (purple).

3.2 illustrates that in 2,000 simulations of testing, no-action was chosen less than 30 times across all of the epochs of 2,000 simulations of testing. These results suggest that there is very little additional value in choosing to do nothing as opposed to culling non-infected farms. For this reason, I did not implement a no-action option in the succeeding two case studies.

# Times No Action was used within a simulation	Frequency
0	1993
1	1
2	0
3	3
4	0
5	1
6	2

Table 3.2: Frequency chart of the number of times no action was chosen within a simulation of testing

3.4.4 Case 3: Identifying "bridging" farms

In the third case study, I targeted a specific feature of disease spread, network structure. Network structure refers to how individuals in a community are connected to each other. For example, suppose that a community consisted of three individuals:

A, B, and C. If individuals A and B came into frequent contact, then it could be concluded that A and B are connected. If individual C never had the opportunity to meet individuals A or B, then it could be concluded that C is not connected in the community. Salathe and Jones (2010) exploited network structure to trace disease spread using friendship data collected from Facebook. They found that targeting individuals that "bridge" communities was more effective than targeting highly connected individuals, i.e.- individuals that were in contact with many others. In this third case study, I used bridging farms to incorporate an inherent network structure. Figure 3.12 illustrates the initial state. These quantities were also uniformly distributed between 25 and 500 as in the first and second case studies with a total of 7922 cows in the entire space. The two farms in the middle of the grid space, not belonging to either cluster of farms, are the bridging farms that connect the two clusters of farms in the corners of the 10×10 kilometer space. To accommodate the bridging farms, I implemented an additional constraint to the transmission kernel. The upper cluster of farms could only become infected if either of the bridging farms became infected. This emphasized the behavior of the two farms "bridging" the two clusters of farms.

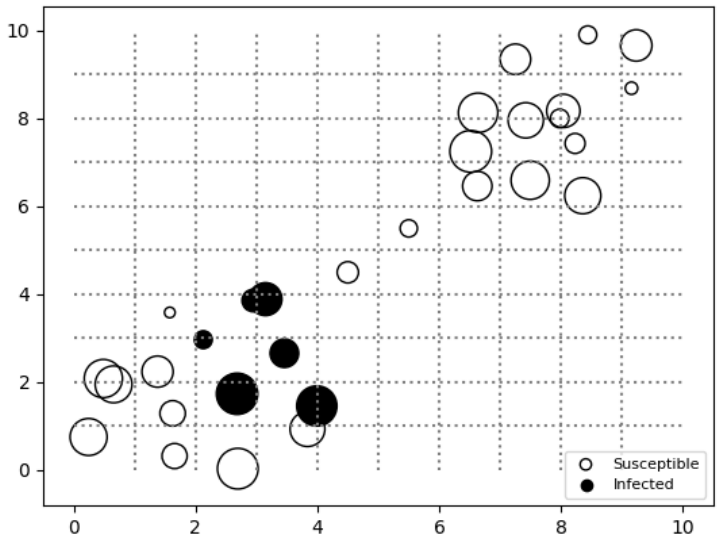

Figure 3.12: Initial state for case 3. Dot sizes reflect the number of cattle at a farm.

A cow's transmissibility of the disease and the transmission kernel were the same as in the second case study. This case study also had the same daily culling capacity constraints as the second case study: all of the cattle at any one farm could be culled per day. This presented an additional challenge to the DQN agent. Similar to the

second case study, there would be more exposed farms due to the conservative daily culling capacity. Thus, the DQN agent had to learn that exposed farms were going to become infected. In addition, the DQN agent had to learn the specific disease dynamics: if one of the bridging farms became infected, then the upper cluster of farms would begin to observe infections.

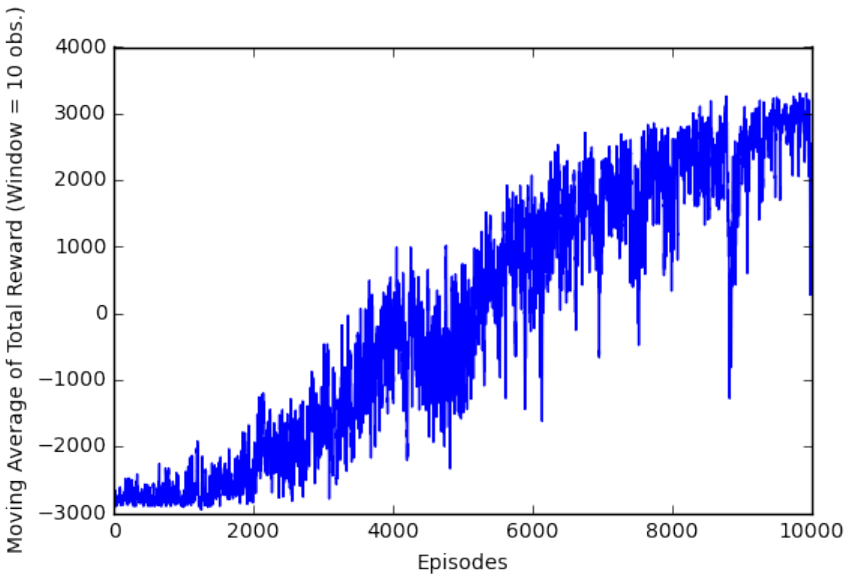

Figure 3.13: Smoothed total reward trajectory over 10,000 episodes of training, for third case study.

After 10,000 episodes of training, the DQN agent was successfully trained. Figure 3.13 shows the trajectory of total reward over the 10,000 episodes. The trajectory begins to plateau, suggesting that the training has begun to stabilize. Just as with the first and second case studies, the DQN agent began its training by culling nearly every farm, and over time the agent learned to behave in a manner that met all of the objectives.

Figure 3.14 illustrates an instance of the best sequence of actions determined by the trained DQN agent. The bridging farms were not immediately culled, the first bridging farm was culled on the third day. After the bridging farms were culled, the DQN agent focused on culling the exposed and infected farms in the lower cluster. Once the DQN agent was trained, I performed 2,000 simulations of testing. On average, two susceptible farms were culled within a simulation (95%CI (1,2)), three exposed farms (95%CI (2,3)), and ten infected farms (95%CI (8,12)). Similar to the second case study, in some instances the DQN agent calculated a higher value in culling susceptible farms over infected farms, refer to the fifth panel in Figure 3.14. In all 2,000 simulations, the outbreak never spread to the upper cluster of farms. This

Figure 3.14: One instance of how DQN agent optimally terminates the outbreak for case 3 after model is trained. Every other panel represents an action (represented by the red x's). After an action is implemented the outbreak evolves, represented by the remaining panels.

case study illustrates that the DQN agent successfully met the management objectives, represented by the stabilization of training, while learning about the network structure of the farms. This third case study could easily get more challenging. For example, rather than include two clusters of farms, one cluster of infected farms could be bridged to many other clusters of susceptible farms. In addition, the transmission kernel could be relaxed to allow farms in other susceptible clusters to observe infections without the bridging farms needing to become infected first. This third case study suggests that the DQN agent could be successfully trained to accommodate these types of future examples.

3.4.5 Case 4: Scaling

In the fourth case study, I investigated DQN under a larger FMD decision problem. In the previous three case studies, there were 30 farms under consideration. In this case study, I attempted to terminate the outbreak with 120 farms. Because more farms were considered in this case study, the size of the landscape increased to 15×15 kilometers. Each grid square still had the same dimension, 1×1 kilometers, as in the previous case studies. The number of cattle per farm was uniformly distributed between 25 and 500 as as with the previous case studies, with a total of 33,638 cows in the entire space. In addition, in this case study, I used physical landscape features to influence the positioning of farms. In the United Kingdom lakes, rivers, and large highways are some examples of physical landscape features that separate farms into clusters. Figure 3.15 provides an illustration of the initial state. Notice that the farms are separated into clusters, however, not as distinct as the clusters of farms in the third case study. The gaps between the clusters of farms, were intentionally placed to represent some of the aforementioned physical landscape features. The initial state in this case study has two clusters of infected farms. This reflects the initial state of the 2001 United Kingdom outbreak. By the time the outbreak was detected, infected livestock had already been transported to different areas of the United Kingdom, resulting in several foci of infection (Gibbens et al., 2001).

The transmission kernel denoted in Equation (3.4) was adjusted from the kernel used in the previous three case studies, to reflect a less random spread of infection, with constraints reflecting those used by Keeling and Rohani (2007). Figure 3.16 illustrates the transmission kernel behavior for this case study and the three previous case studies. There was a higher chance for closer farms to become infected under the kernel used for the fourth case study. However, the kernel used in the three

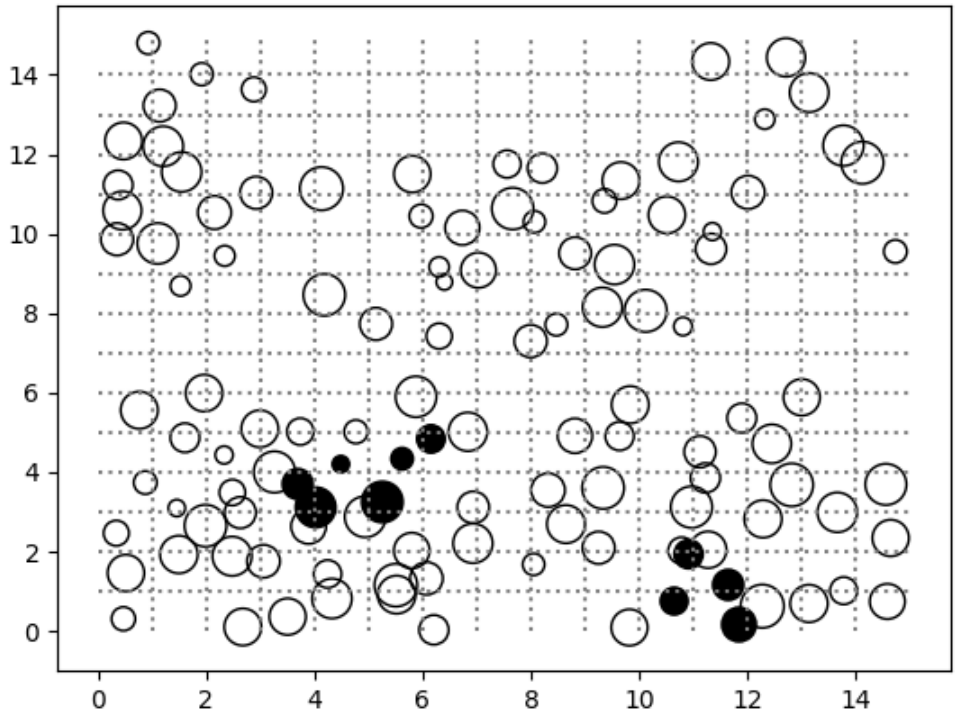

Figure 3.15: Initial state for fourth case study. Dot sizes are proportional to the number of cattle at the farm.

previous case studies assigned almost equal probability to farms becoming infected regardless of distance. Thus in the previous case studies, it was not uncommon to observe infections that were far from the index infected farm. This previous kernel created a challenging environment for the agent to learn, because the outbreak was more unpredictable. While this may still happen in the fourth case study, it is not as likely. The kernel used in the fourth case study presents a more realistic transmission kernel. A cow's susceptibility to the disease, s, and a cow's transmissibility of the disease, τ, were both decreased to 4.9×10^{-6} to allow for more gradual spread and a chance for an intervention to take place during the outbreak. The previous rates of susceptibility and transmission would have caused the outbreak to progress too quickly, rendering an intervention ineffective. Because the number of farms in this case study increased by a magnitude of four, to 120 farms, the daily culling capacity was also increased. Each day, it was assumed there were resources to cull all of the cattle at any five farms.

$$K(distance) = \begin{cases} 0.3093 & distance < 0.0138 \\ \frac{1}{\pi(1+distance)} & otherwise \end{cases} \tag{3.4}$$

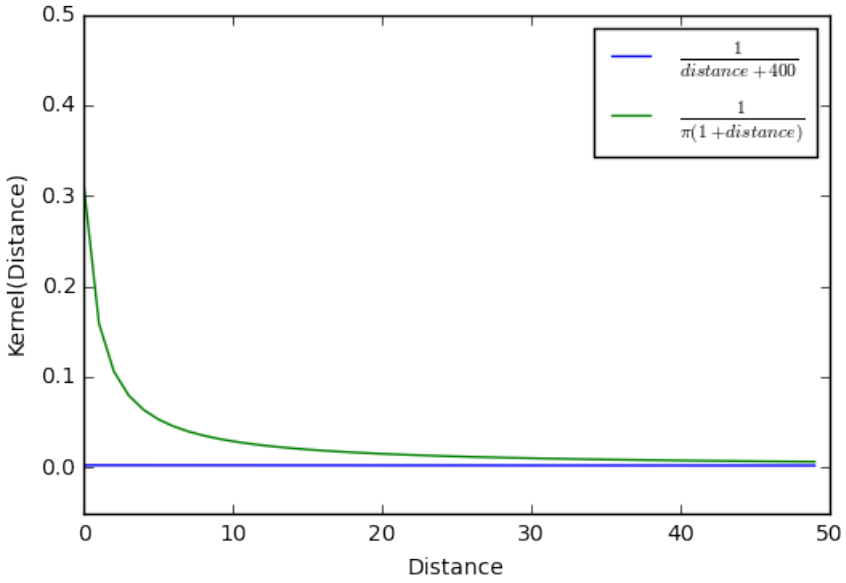

Figure 3.16: Transmission kernels used for each case. The blue line represents the transmission kernel used in the first three case studies, and the green line represents the kernel used in the fourth case study.

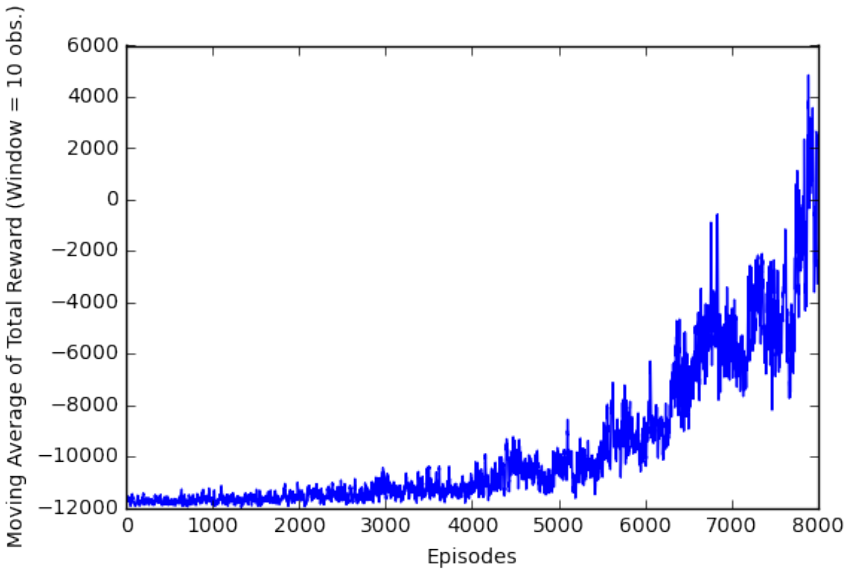

Figure 3.17: Smoothed total reward trajectory over 8,000 episodes of training, for fourth case study.

I trained the DQN agent for 8,000 episodes (Figure 3.17). Notice that the total reward continues to increase over training, but does not distinctly plateau as with the first and third case studies. This behavior is expected because of the complexity of the decision problem. This case study and the second case study were not targeting specific features of the outbreak, i.e. - the farm that could be left to cull on the second day of management (first case study), or identifying bridging farms (third case study).

Figure 3.18: One instance of how DQN agent optimally terminates the outbreak for case 4 after model is trained. Every panel represents the state after five culls. The states after evolutions are not included. The S, E, I at the top of each panel represent the number of susceptible, exposed, and infected farms culled during that day.

Figure 3.18 presents the sequence of culling actions the trained DQN agent thinks is optimal. Each panel presents the five farms to be culled, and is annotated by the

types of culls: susceptible (S), exposed (E), and infected (I). Some panels do not have five farms culled (refer to the 12th, 15th and 16th panels) because the agent assumes the outbreak is completed. There is a high degree of susceptible farms culled. In 2,000 simulations of testing, on average 48 susceptible farms were culled within a simulation (95%CI (29, 75)), 9 exposed farms (95%CI (4, 14)), and 31 infected farms (95%CI (20, 44)). Particularly noticeable from these results was the large number of susceptible farms culled compared to infected farms. This was investigated further through hyperparameter tuning.

Many of the DQN and decision environment components were tuned: immediate reward structure, daily culling capacity, experience replay size, batch size, learning rate, episodes of training. The results of the last four listed components are presented in Figure 3.19. In each panel, the value of one hyperparameter was changed with all other hyperparameters fixed at the values listed in Tables 3.6 and 3.7 in Appendix B. One of the immediate responses to improve DQN performance is more training. In the top left panel of Figure 3.19, I trained the DQN agent for 8,000, 10,000, 20,000 and 50,000 episodes (which took 2 and a half days, 3 days, 6 and a half days, and two weeks to run in serial, respectively). Interestingly, the DQN agent performed best under the least amount of training. These results could be influenced by the lack of exploration and becoming trapped in local minima. Bent et al. (n.d.) suggest that modifications to experience replay size and batch size can assist with exploration. The top right and bottom left panel present these results under 8,000 episodes of training. The experience replay size and batch size used to generate the previous results (2,500 and 36 respectively) generated the best total reward trajectories. Similar results were obtained when varying both the number of episodes and experience replay size or batch size. I also decreased the learning rate to allow the DQN agent to more gradually explore the space, and not be trapped in local minima. However, according to the trajectory in the bottom right panel, the decrease in learning rate did not improve the trajectory of training. Similar results were obtained with an increase in the number of episodes and a lower learning rate.

I also increased the daily culling capacity to seven farms per day, in an attempt to construct a policy which culled a smaller number of susceptible farms. Culling five farms per day may not have been sufficient enough to keep up with the spread of the outbreak. In an attempt to increase speed in computations, I scaled the immediate reward structure. Rather than penalize the DQN agent with 100 points for culling a farm, it received -1 points. Similarly, the agent received -5 points for incorrectly assuming an outbreak was finished as opposed to -500 points. The terminal

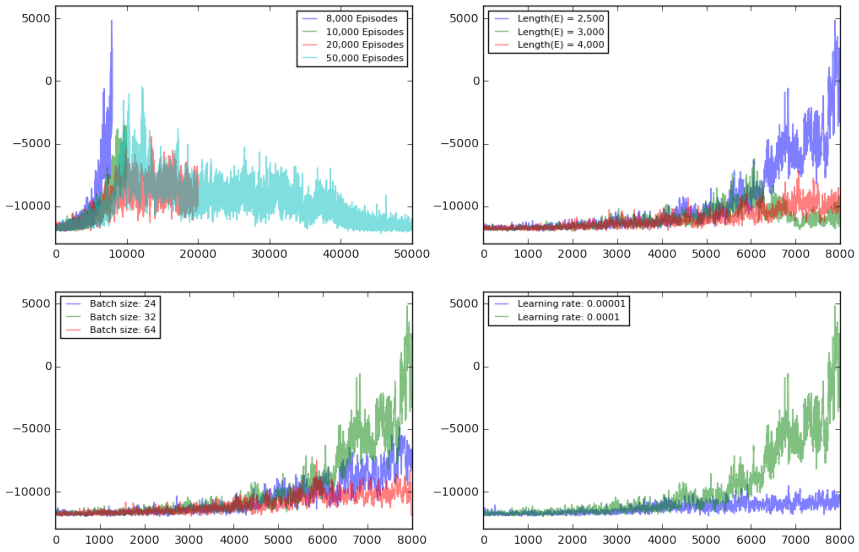

Figure 3.19: Trajectory of total reward based on hyperparameter. Top left panel changes the number of episodes of training, with all other parameters fixed. 8,000 episodes of training performed best, so that was chosen. Top right panel changes the size of the experience replay deque with all other parameters the same. 2,500 performed best, so that was chosen. Bottom left panel changes the batch size with all other parameters fixed. Batch size 32 performed best, so that was chosen. Bottom right panel changes the learning rate with all other parameters fixed. A learning rate of 0.0001 performed best, so that was chosen.

reward, of number of cows saved, was also scaled by 100. After many iterations of hyperparameter tuning, the best outcome was generated with 20,000 episodes of training, with all other hyperparameters unchanged (unfortunately, the time taken to train the agent in serial did not severely change from three days). Figure 3.20 illustrates the reward trajectory during the 20,000 episodes of training. There is a large increase in rewards during the end of training. A closer inspection of this showed that in some episodes, the agent chose to cull eleven to thirteen farms to halt the outbreak. However, there is a high degree of variation, the rewards towards the end of training range from -100 to 150. Figure 3.21 presents an instance of the suggested sequence of actions from trained DQN agent. The first two days of management show promising results; all of the infected farms are culled. However, on the seventh day of management, represented by the third panel, many susceptible farms are culled. This behavior continues, and nearly all of the farms are culled by the end of management, represented by the sixth panel. Thus increasing the daily culling capacity, even with additional hyperparameter tuning, does not improve DQN performance for the fourth case study.

This case study has illustrated another DQN challenge, management with a larger decision problem. There is a higher degree of state variation compared to the previous

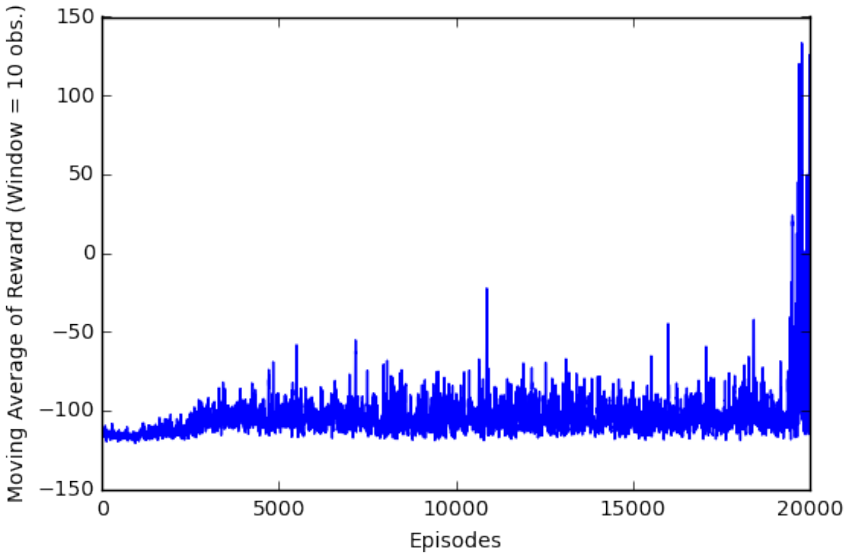

Figure 3.20: Trajectory of total reward after truncating rewards for fourth case study. There is a large increase in the trajectory towards the end of training, suggesting favorable behavior, however, the variance is very high.

case studies due to the larger number of farms. This may be addressed through more training, which would require parallel computing.

3.4.6 Case 4: Sensitivity Study

I made two modifications to the fourth case study to further investigate DQN performance in larger decision problems. The first modification was to simplify the decision problem. The first three case studies suggested that DQN could excel in managing FMD outbreaks. Thus, I used the same decision environment components and hyperparameter values in this modification to the fourth case study. The initial state is presented in Figure 3.22. There were 60 farms under consideration, as opposed to 30 in the first three case studies. There was also a more conservative daily culling capacity, all of the the cattle at any one farm could be culled, similar to the criteria of the second and third case studies. The transmission kernel was the same as the first three case studies, but a cow's susceptibility and transmissibility to the disease were decreased to 3.5×10^{-5}. Recall, if these parameters are too high, then it may be impossible to implement an intervention to terminate the outbreak. All other DQN hyperparameters were unchanged from the original fourth case study.

I trained the agent under many combinations of hyperparameters to optimize performance. The results of training under 10,000 and 20,000 episodes, with all

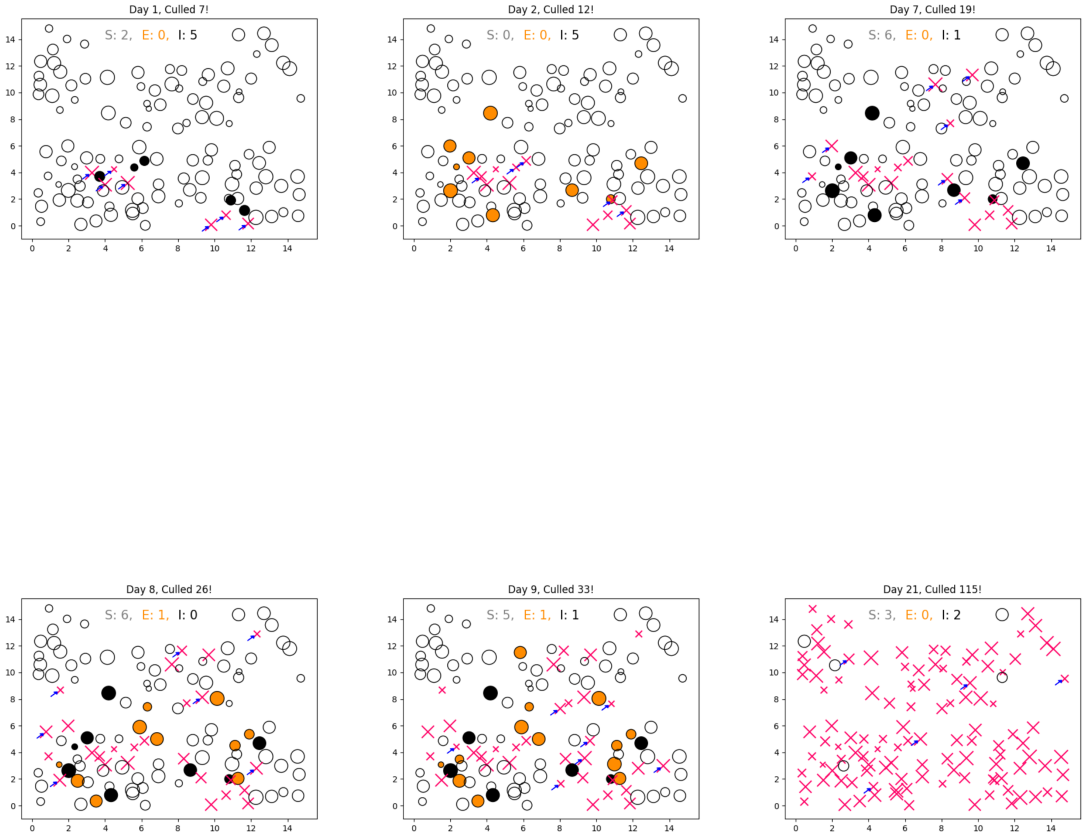

Figure 3.21: The first few actions suggested by the DQN agent when the daily culling capacity is increased. Notice that the DQN agent has found good beginning behavior, al of the initial infected farms were culled in the first two days of management. However, the DQN agent manages poorly the rest of the time, and by the end of management, nearly all farms are culled.

of hyperparameters unchanged from the original fourth case study are presented in Figure 3.23. The left panel, with 10,000 episodes of training illustrates an early plateau with negative rewards. In an effort to determine if more training is needed, 20,000 episodes of training was implemented. However, the total reward trajectory continued to be negative and plateau at a negative total reward. This suggests that the DQN agent was not able to learn effectively and meet the management objectives. Although the agent was successfully able to halt the outbreak in the second and third case studies, it was not able to do the same in a larger decision problem.

I also investigated a second modification to the fourth case study. Based on the sequence of actions illustrated in Figure 3.18, there were many susceptible farms culled that were far away from the majority of the infections. This could be attributed to the fact that, using the kernel in Equation (3.4), it was possible for farms farther away to become infected. Thus this modification to the fourth case study investigated

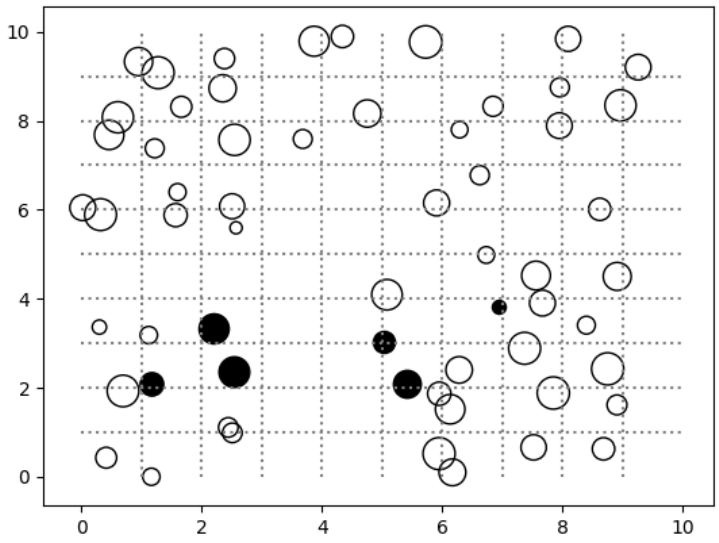

Figure 3.22: Initial state of modified version of fourth case study. 60 farms are used as opposed to 120.

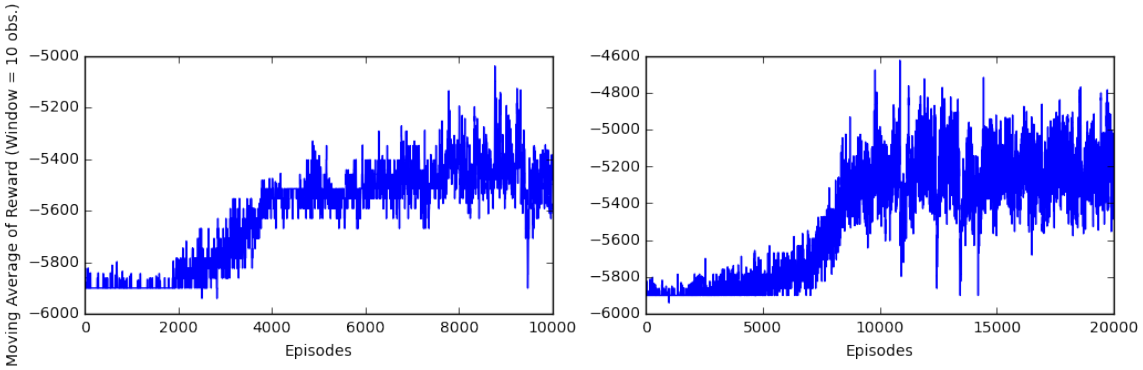

Figure 3.23: Total reward trajectory for the modified fourth case study, with 60 farms. The trajectory in the left panel is from 10,000 episodes of training, and the right panel is from 20,000 episodes of training.

DQN under a different transmission model. In the new transmission model, I scaled Equation (3.4) by 0.025, resulting in a lower risk for farms farther away to become infected. All other decision environment components (initial state, 120 farms in the space, five farms could be culled per day, etc.) and hyperparameter values were unchanged from the original fourth case study.

As with the previous modification to the fourth case study, I trained the agent under many combinations of hyperparameters to optimize performance. The results of training the agent under 20,000 and 50,000 episodes, with all of hyperparameters unchanged from the original fourth case study are presented in Figure 3.24. In the left panel with 20,000 episodes of training, the total reward trajectory gradually increases to a total reward of about 20,000 and begins to be unstable. There is a high degree of variation in the total reward for the remainder of episodes. To investigate if more training was needed, 50,000 episodes of training was implemented, and the same results were observed. This high degree of variation also resulted in a poor optimal policy. Upon testing the trained DQN agent in each instance, nearly every farm was culled. This large variation in training could be attributed to the high degree of stochasticity of disease spread in the landscape. In some instances, there were not as many additional infections. However in other instances, there were a large number of additional infections. In addition, the DQN agent still chose to cull a large number of susceptible farms that were farther away from the primary congregation of infections. This was further investigated by adjusting the immediate reward structure.

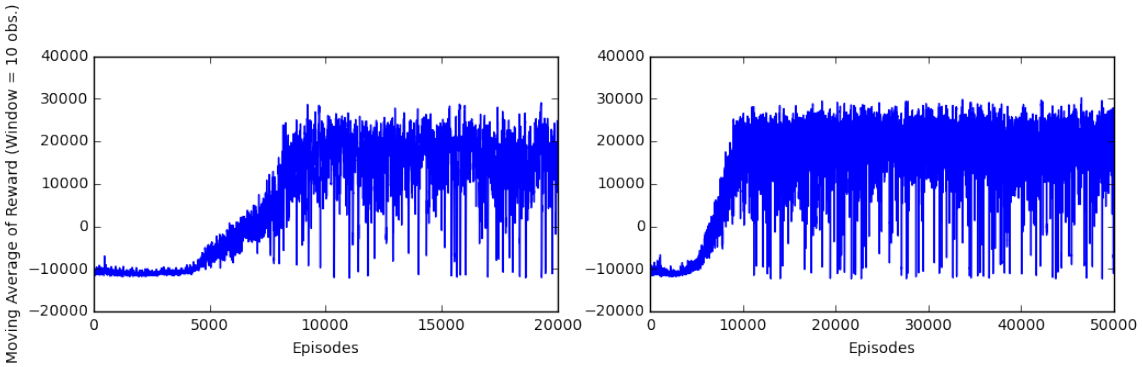

Figure 3.24: Total reward trajectory for the modified fourth case study, with a less sporadic disease spread. The trajectory in the left panel is from 20,000 episodes of training, and the right panel is from 50,000 episodes of training.

A -1000 reward was added for an agent culling a farm that was at least five kilometers away from any infected farm. This negative reward would ideally encourage the DQN agent to cull farms that were closer to the primary congregation of infections.

Recall that because the transmission kernel was scaled in this second modification to the fourth case study, I would not expect farms farther away to become infected. The total reward trajectory under 20,000 episodes of training is presented in Figure 3.25. There is a gradual increase in rewards during training, as with Figure 3.24, and a large degree of variation after about 5,000 episodes of training. While there are times that the agent achieves high rewards during training, there are also many times where the agent achieves very negative rewards during training.

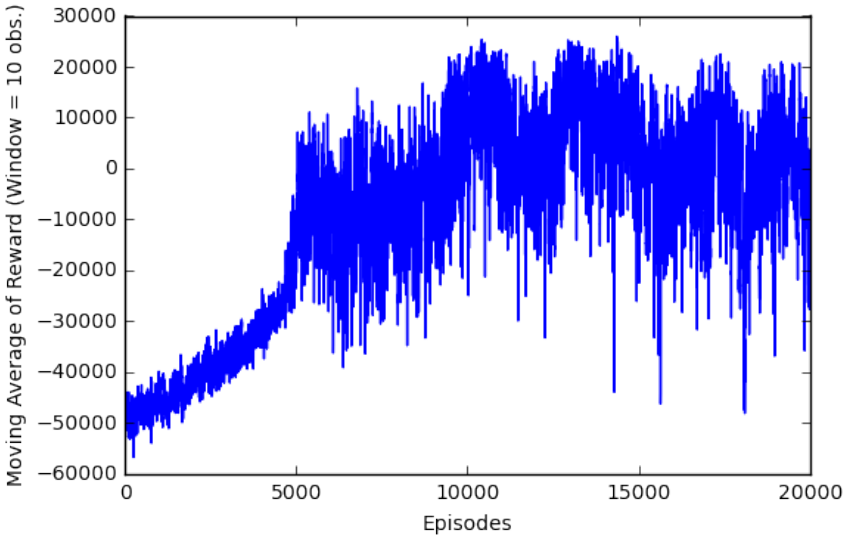

Figure 3.25: Total reward trajectory for the modified fourth case study, with a less sporadic disease spread and an additional negative reward for culling farms at least five kilometers away from any infected farm.

Figure 3.26 presents an instance of the first ten days of the suggested sequence of actions from the trained DQN agent. The sequence of actions suggested in the beginning of management suggests that the DQN agent understood this new objective. Many of the culled farms were close to the index infected farms. However, the DQN agent is unable to identify exposed farms and cull them, and instead culls susceptible farms that are close to the index infected farms. This sensitivity analysis to the fourth case study illustrates the challenges of using DQN to manage a larger decision space. Even with more training, and hyperparameter tuning the agent is unable to halt the outbreak while meeting the objectives. However, the agent was able to successfully meet a very specific objective of culling farms that were close to infected farms.

Figure 3.26: An instance of the suggested actions from the DQN agent for the 2nd modified fourth case study with the additional negative reward for culling farms at least five kilometers away from any infected farm. The last panel presents the final state after management. Notice the promising behavior at the beginning of management, farms close to the index infected farms are culled. However, the agent is not able to keep up with the outbreak with its policy.

3.5 Challenges of Deep Q-Networks

3.5.1 DQN Parameter Selection

There are two main components to DQN: Q-learning and CNN. Both components require choices to be made *a priori* about the value and/or functional form of hyperparameters. Some of the primary hyperparameters involved in Q-learning and CNN are provided in Tables 3.3, and Table 3.4 respectively. Selection of hyperparameters may affect DQN performance. For example, it is strongly recommended to implement target network updating to improve stability and prevent divergence (Mnih et al., 2015). This was not implemented in the first case study due to the simplicity of the decision problem. However, the other three case studies implemented target updating during training. Figure 3.27 illustrates the differences in the trajectory of total reward over 10,000 episodes of training with and without target updating, in the second case study. When target updating is not used, the total reward trajectory plateaus at a lower value earlier on in training. This one binary decision, with all other hyperparameters unchanged, can result in a sub-optimal final policy.

Hyperparameter	Description
Immediate reward function	Immediate reward for taking action a from state s at time t
Initial and final ϵ	Initial and final value of ϵ in ϵ-greedy algorithm
Epsilon decay function	Function describing how epsilon decays with each time step
Experience replay start size	#times policy is run before experience replay begins
Experience replay memory size	#sample stored with each experience replay
Target network update frequency	#times target network is matched with current network
Discount rate	Degree of important of long/short term rewards
Stopping criteria	Criteria to stop training

Table 3.3: Parameters to Tune in Q-learning/ DQN

One of the most important decisions with DQN is input specification. Input features are typically chosen *a priori* with guidance from domain knowledge. States in a decision problem represent a decision-maker's belief of which inputs are most important for the agent to learn about. However, DQN will only achieve good performance if it is provided with informative features to train upon (Boone, 1997). In this investigation, I constructed the state using an image pixelated by farm-level infection statuses, inspired by the RGB values typically used in CNN. However, this choice of just using farm-level infection status may have limited the DQN agent's ability to learn more about the decision space. One extension to this state space could be to have each value within each pixel on a continuous spectrum. Each pixel could contain a risk value of infection determined by other farm level parameters, such as area of a farm, or number of cattle at a farm. The DQN agent may need more episodes

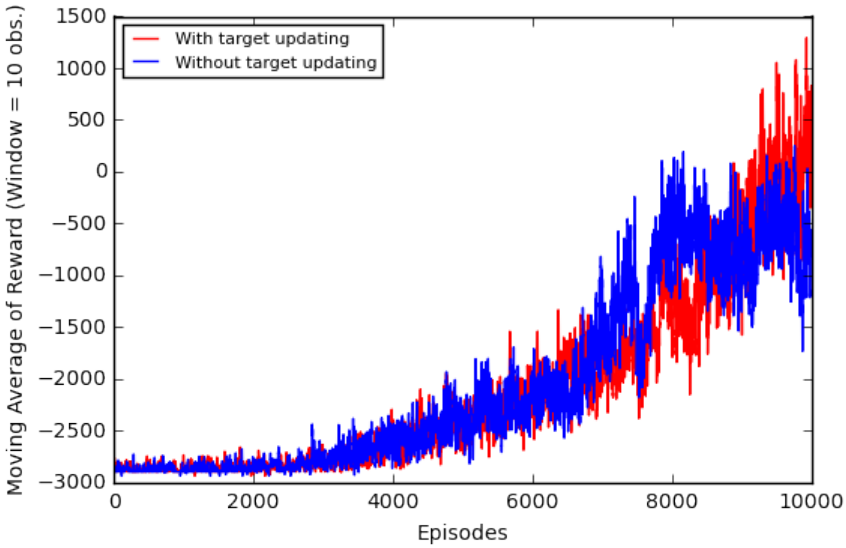

Figure 3.27: Trajectory of total reward over 10,000 episodes of training for the second case study. The trajectory without target updating is in blue, with target updating is in red. Notice the early plateau without target updating.

Hyperparameter	Description
Inputs	Features to describe state space
# hidden layers	# times nonlinear transformations occur on input
#nodes in dense layers	#variables in linear combination
#filters in convolutional layers	#features to be extracted
Patch size	Dimensions of window in a convolutional layer
Activation function	Function (linear/nonlinear) to transform previous layer of data
Mini batch size	#samples for model fit
Learning rate	Rate of abandoning old beliefs
Optimizer	Process used to search parameter space
Epochs	#times sample is forward and backwards propagated

Table 3.4: Parameters to Tune in CNN

of training to explore the wide range of state values, but may ultimately generate different and potentially better policies due to the increase in information.

Parameter selection in deep reinforcement learning methods is still understudied. Mnih et al. (2015) mentioned that, in their study, the hyperparameters were selected by performing an informal search, and a systematic grid search would result in high computational costs. Referring to Tables 3.3, and 3.4, there are many hyperparameters to tune, and this excludes searching for a proper functional form of continuous hyperparameters. This investigation used an informal hand-tuning method for the first case study, and similar hyperparameters as Mnih et al. (2015) for the remaining case studies. The choices of hyperparameters are presented in Tables 3.6 and 3.7 in

Appendix B. Manual parameter tuning is highly subjective. It is almost guaranteed that if two individuals were given the same initial data, the final tuned models would be very different due to the large number and functional forms of hyperparameters. The alternative to manual tuning and grid search methods are formal optimization methods. There have been some advances in optimizing neural network hyperparameters, such as using genetic algorithms or expected improvement in Gaussian Processes (Fridrich, 2017; Bergstra et al., 2011). However, these approaches have not been implemented in a DQN environment.

3.5.2 Real Outbreaks

One distinguishing characteristic of this study was the use of a grid to create "pixels" of a given image of the outbreak. This study never used more than a 15×15 grid, or 225 kilometers^2. Recall, that the number of pixels to use was determined by ensuring there was no more than one farm per pixel, or grid square. This allowed for the agent to gain spatially explicit information from each pixel of the image, by means of farm-level infection statuses. While I have shown that the agent was able to meet all of the objectives given this input, overlaying a grid on a realistic landscape may not be as straightforward. For example, Cumbria, a county in the United Kingdom, was severely affected during the 2001 United Kingdom FMD outbreak. It also has a land area of about 6,767 kilometer^2. This area is significantly larger that those investigated in this study, and would thus require more grid squares. However, an increase in grid squares results in a larger matrix to be interpreted by the DQN agent. In a previous study by Chavez et al. (2015) it was determined that the frame width of an image had a quadratic relationship with run time in CNN calculations. Thus larger matrices may result in computational complexities. This could be addressed using parallelization approaches, discussed in another section.

Due to the size of the United Kingdom, and the computational limitations of the number of pixels that can be used in this method, each pixel in the overlaid grid may not be able to capture at most one farm. DEFRA (2011) conducted a census in 2000 regarding the number of cattle in the UK. Even after using 5×5 kilometer grid squares to represent density of cattle, each grid square captured information from more than one farm. This is one of the tradeoffs of using this method on a real landscape. Rather than use spatially explicit information on a farm-level, spatially explicit information may need to be obtained on a land area level. The process would still begin by dividing the area of interest into grid squares, to the extent

of computational resources available. Each grid square would represent the number of cattle that were infected in that particular land area. Or, using the extension mentioned in the previous section, each grid square could represent the risk of that particular land area observing an infection.

One approach to potentially decrease computational complexity could be to coarsen the grid to allow for more than one farm per pixel. The action space, and decision environment would then need to be modified. Rather than calculate the expected utility for culling a single farm, the expected utility would need to be calculated for culling all of the farms in a given grid square. A decision-maker would need to determine *a priori* if regional-level interventions would be economically and practically sufficient to terminate the outbreak.

3.5.3 Generalizability

The goal of this investigation was to implement DQN to halt a FMD outbreak, by specifically using the dynamics of the 2001 United Kingdom FMD outbreak. However, this method can be generalized to any type of disease or disease dynamics. The key element to implement this method to a different historical FMD outbreak or to a new disease entirely, is the disease dynamics. Disease dynamics can be decomposed into two main components: the disease transmission model, and the compartment model. The disease transmission model generates a new state after an action(s) has been implemented. The more accurate the transmission model, the more reliable the final policy will be to the decision-maker. A rigorous transmission model for the 2001 UK FMD outbreak provided by Keeling and Rohani (2007), was used for this study.

The state space is also affected by the disease dynamics. In this investigation, the disease progression followed an SEIR compartment model and the pixels represented the observable statuses of the farms: susceptible to the disease, infected, or culled. A culled farm was considered "recovered" in this study. It was assumed that a culled farm would not be able to observe another infection. However, a different investigation may consider implementing vaccinations, which would create another compartment in the model. If a vaccination scheme was implemented in an FMD context, the new compartment model would be SEITR, where T represents a "treated" compartment (Brauer et al., 2008). The state would thus need to be modified to accommodate the new compartment. In reality, vaccinations do not guarantee that an individual will become recovered. Vaccinations reduce the risk of an individual becoming infected, but it is still possible to observe infections. If a study were to use an SEITR model,

the decision environment would need to incorporate the possibility that treated individuals could later become infected. Constructing a computational package to implement this method to any disease may pose some challenges, due to the fact that different disease dynamics can affect a decision environment very differently. It is recommended to use the functions provided in Appendix A. to implement DQN, and manually construct the decision environment based on the disease dynamics.

This method may present some challenges in generalizing policies to any initial state, given the disease dynamics. Recall that in this investigation, I fixed the initial state and overfit the final policy to that initial state. This was intentionally done because in practice, a decision-maker would provide the current status of a disease outbreak and request the best sequence of actions. However, in the instance where disease monitoring can occur and updates in state information is received, it may be more useful to have a policy that does not depend on the initial state. As previously mentioned, DQN may require many iterations of hyperparameter tuning for optimal performance. Having to do this every time new data is collected would be impractical. However, generalizing this method to any initial state may require more episodes of training and potentially, a more complex CNN. Figure 3.28 illustrates the total reward trajectory for the first case study when the initial state is fixed, and when the initial state changes from episode to episode, and all other hyperparameters are unchanged. The total reward trajectory is nearly constant when the initial state changes from episode to episode, compared to the fixed initial state scenario. Neural networks traditionally require a large amount of data for optimal performance. Le (2013) constructed a neural network to recognize faces using image data from YouTube videos. 10 million images were required to train this neural network on a completely different test set. For this reason, if the initial state changes with every episode of training, more data and thus more computational resources may be required to train the DQN agent.

3.5.4 Scaling

The fourth case study revealed another limitation of using DQN to halt disease outbreaks: the larger the decision problem, the less likely the agent will be able to successfully meet management objectives. Increasing the number of farms by a magnitude of four, resulted in poor performance. In an attempt to make the decision problem easier for the agent, by reducing the number of farms, and changing the outbreak spread to be less sporadic, the agent was still unable to learn. An

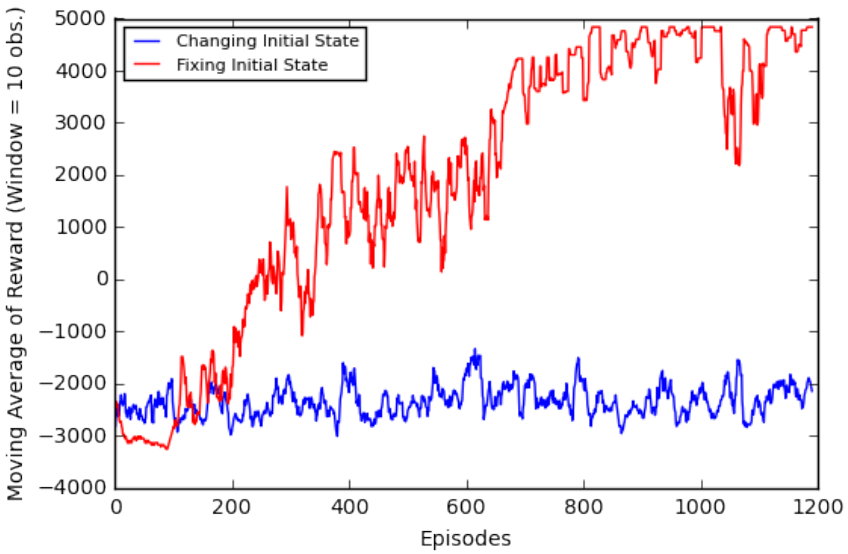

Figure 3.28: Trajectory of total reward over 1,200 episodes of training for the first case study. The trajectory when the initial state is changed at the beginning of every episode is in blue, and the trajectory when the initial state is fixed at the beginning of every episode is in red. All other parameters are exactly the same. Notice the lack of improvement in the trajectory when the initial state changes with every episode.

intuitive solution to this problem would be to train the agent for a larger number of episodes. Recall that DQN involves a Q-update, i.e. updating the expected reward based on: the immediate reward obtained from visiting the most recent state, and an estimate of the expected reward for the next state using the current model. Due to this architecture, DQN is serial in nature. Thus serially training an agent for a larger number of episodes would present some challenges on run time. Chavez et al. (2015) explored a parallelization alternative for DQN, involving Downpour stochastic gradient descent. This process involves an omniscient parameter server that stores a global copy of the DQN model. It also has "worker nodes" to perform the following:

1. Get the most updated model from the parameter server
2. Use model to generate an action
3. Update local model using stochastic gradient descent
4. Deliver the updated local model to the omniscient parameter server

Chavez et al. (2015) used this parallelization method with two worker nodes to train a DQN agent to play the Snake game in Python. The DQN agent was able to meet objectives much faster than a serial approach. However, if this parallelization

method was used in disease outbreak management with many farms, then more worker nodes would be required. This could potentially create bottlenecks in communication between the worker nodes and the parameter server. Furthermore, worker nodes may be using outdated models due to those bottlenecks in communication, i.e. gradient staleness.

Another contributor to longer run time in a larger decision problem is model complexity. Chavez et al. (2015) suggests that complexity is $O(d^2Fk^2N^2L_C+H^2L_B+H_d^2N)$ where d is frame width, F is frame count, k is patch size, N is patch count, L_C is convolutional layer count, H is node count, and L_B is hidden layer count. Frame width, d, in particular will be affected in a disease management problem with a larger number of farms. GPU computation, or sparse matrix computation, could be incorporated to speed up matrix computation for scenarios with larger frame widths, i.e. - larger matrices.

3.5.5 High Variance

DQN also tends to have a high degree of variance during training, and thus the final policy. Figure 3.29 illustrates the total reward trajectory of the first case study under three different instances of training with all hyperparameters unchanged. Even when all of the hyperparameters are the same, the trajectories can be very different. The blue line represents the trajectory of the instance that was used in this analysis. For each case study the training was implemented three separate times, and the best trajectory was used in the analysis. Anschel et al. (2017) suggests that one of the contributors to high variance is target approximation error, which refers to the difference in the estimated expected reward and the true target. Some hypotheses as to why target approximation error occurs is: estimation of CNN weights due to inexact minimization, limited representation of a state via a CNN, and the finite size of the experience replay (Anschel et al., 2017). For this reason, the results of DQN should not be over-interpreted. The DQN agent may have a successful final policy due to chance. Some methods have recently been investigated to reduce the high degree of variance, such as averaged-DQN and double DQN (DDQN).

While DQN itself is known to have a high level of variance, there are also other sources of randomness that may contribute to policy deviations. Firstly, the DQN network weights are randomly initialized at the beginning of training. Secondly, this algorithm is ϵ-greedy, meaning that actions are randomly selected an ϵ proportion of the time. These choices of action will also change between instances of training.

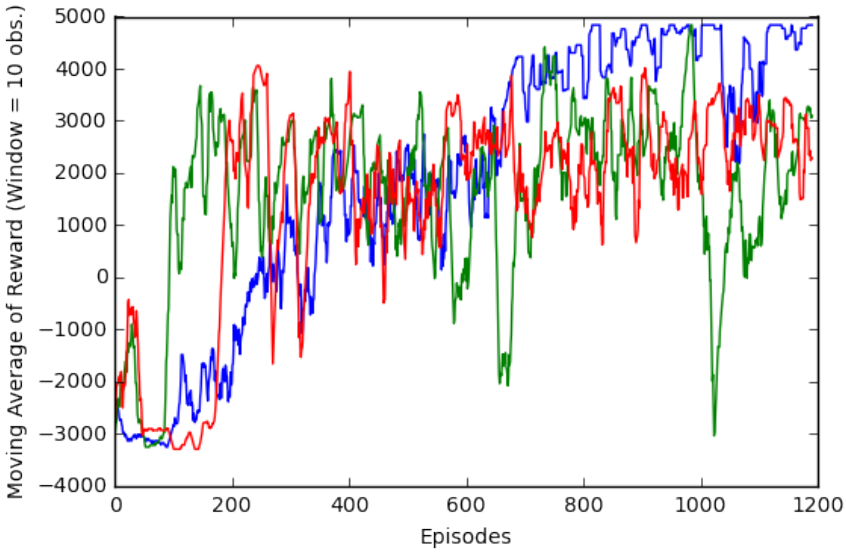

Figure 3.29: Trajectory of total reward over 1,200 episodes of training for the first case study. Three separate instances of training was performed with all hyperparameters unchanged. The blue line represents the total reward trajectory for the instance that was used in this study's analysis. Notice the other two instances of training resulted in very different trajectories.

Thirdly, recall that a random sample of N state, action, reward, new state observations are chosen from the experience replay deque to update the network weights. Finally, the outbreak itself is stochastic. All of these components in addition to the DQN's inherent high variance behavior may require many instances of training to estimate an optimal policy.

3.6 Discussion

Deep Q-networks have recently been used with high dimensional input data in applications such as video games and robotics. This method has never been explored in a disease management context, but I adapted the algorithm to do so in our study. Traditionally, deep Q-networks are tasked with winning a game, as is the case in video games. This idea of winning a game was evident in our study, where "winning" corresponded to the agent successfully meeting all of the management objectives. I have illustrated that deep Q-networks could be used in simple cases of disease management. However, there are some areas that would need further development before implementation. As previously stated, optimal hyperparameter tuning, modifications to accommodate real landscapes, generalizability, scaling, and high variance have all illustrated some limitations to DQN.

Based on our findings, there are also many opportunities for further research.

For example, parameter selection may be overwhelming for someone new to DQN. There are many blogs regarding neural network and reinforcement learning parameter selection. However, a synthesis of this information, and perhaps a flowchart of how to make decisions on parameter tuning, may help individuals construct more informative DQNs. In addition, there are computational issues associated with allocating a single farm to a "pixel." Under this arrangement, a very large number of pixels would be required for a real outbreak. Some approaches to this would be to coarsen the grid, allowing for more farms to occupy a pixel, and possibly use a GPU to assist with stochastic gradient descent computations on a larger matrix. A larger landscape would also require more data to train the DQN. This presents an opportunity to investigate parallelization in DQN training. Another area for potential future research is exploration of a generalizable DQN to any disease outbreak. As previously stated, our study aimed to generate an optimal policy for a given initial state. A more flexible policy could be created, to prevent iterations of retraining as additional data is collected. Because DQN is associated with many challenges, one might question whether alternative methods such as epidemiological methods, or simpler models can be just as effective. This question is investigated further in the next chapter.

```
def chooseAction(currentState,epsilon,original_farm_inds):
  ##########################
  #currentState- flattened 2-D array of infection statuses
    #currentState = 0, Susceptible
    #currentState = 1, Infected
    #currentState = 2, Culled/ No Farm
  #epsilon- value of ϵ at given epoch
  #original_farm_inds- indicies of non-culled farms
  ##########################
  farm_mat = np.c_[list(range(N)),original_|farm_inds,
    currentState[original_farm_inds]]
  #Subset the farms that are not culled
  sub_farm_mat = farm_mat[farm_mat[:,2] != 2]
  #Choose your action
  if np.random.rand() <= epsilon:
    currentAction = np.random.choice(sub_farm_mat[:,0])
    return(currentAction)
  else:
    # Size- number of grid square in 1-D
    mat = currentState.reshape(Size,Size,1)
    prediction = model.predict(np.array([mat]))
    currentAction_ind = np.argmax(prediction[0][sub_farm_mat[:,0]])
    #Make sure you cull a non-culled farm
    currentAction = sub_farm_mat[:,0][currentAction_ind]
    return(currentAction)

def train_replay(batch_size, discount_factor,state_size):
  ##########################
  # batch_size- # samples to take from E
  # discount_factor- degree of impact of future rewards
  # state_size- Size**2
  ##########################
  #No replay until E (memory) has at least train_start samples
  if len(memory) < train_start:
    return
```

```
  batch_size = min(batch_size, len(memory))
  mini_batch = random.sample(list(memory), batch_size)
  #Create lists of s,a,r,s', episode done in batch
  update_input, update_target,action, reward, done =
    [], [], [], [], []
  for i in range(batch_size):
    update_input.append(mini_batch[i][0])
    action.append(mini_batch[i][1])
    reward.append(mini_batch[i][2])
    done.append(mini_batch[i][3])
    if len(mini_batch[i]) == 5:
       update_target.append(mini_batch[i][4])
  #Convert s, s' from batch to numpy arrays
  update_input = np.stack(update_input,axis=0)
  update_target = np.stack(update_target,axis=0)
  target = model.predict(update_input)
  target_val = target_model.predict(update_target)
  #Keep track of indices in lists, length s and s' differ
  else_counter2 = 0
  for i in range(batch_size):
    if done[i] == 1:
       target[i][action[i]] = reward[i]
    else:
       #Perform the Q-update
       target[i][action[i]] = reward[i] + discount_factor * np.amax
          (target_val[else_counter2][inds_to_cull[else_counter2]])
       else_counter2 += 1
  model.fit(update_input, target, batch_size=batch_size, epochs=1)

def update_target_model(modelweights):
  ########################
  #modelweights- result of model.get_weights()
  ########################
  target_model.set_weights(modelweights)
```

3.8 Appendix B. Additional results

	Total Reward	#Cows Saved	#Farms Culled	#Outbreak Resurgences
Case 1	4737 (4737, 4737)	5336 (5336, 5336)	6 (6, 6)	0 (0, 0)
Case 2	502 (-1973, 2427)	2522 (878, 3727)	19 (13, 26)	1 (1, 2)
Case 3	3138 (2173, 3605)	4590 (3994, 4905)	14 (13, 16)	0 (0, 1)
Case 4	-97 (-9586, 10342)	9319 (2210, 16834)	86 (59, 110)	2 (0, 4)

Table 3.5: Results of 2000 simulations with each cell containing the average and the 95% confidence interval.

	Case 1	**Case 2**	**Case 3**	**Case 4**
Initial epsilon	1	1	1	1
Final epsilon	0.01	0.01	0.01	0.01
Epsilon decay function	Exponential	Linear	Linear	Linear
Experience replay start size	100	100	100	100
Experience replay memory size	500	2000	2000	2500
Target network update frequency	N/A	After every episode	After every episode	After every episode
Discount rate	0.99	0.99	0.99	0.99
#Episodes for training (Stopping criteria)	1200	10000	10000	8000

Table 3.6: Q-learning parameter values for each case as specified by Table 3.3.

	Case 1	**Case 2**	**Case 3**	**Case 4**
#Hidden layers	4	4	4	4
#Nodes in dense layers	55	155	155	155
#Filters in convolutional layers	15, 15, 30	32, 64, 64	32, 64, 64	32, 64, 64
Patch size	3x3	3x3	3x3	3x3
Activation function	relu, linear	relu, linear	relu, linear	relu, linear
Mini batch size	64	32	32	32
Learning rate	0.001	0.0001	0.0001	0.0001
Optimizer	RMSprop	RMSprop	RMSprop	RMSProp
Epochs	1	1	1	1

Table 3.7: CNN parameter values for each case as specified by Table 3.4

CHAPTER 4

MANAGING THE 2001 UNITED KINGDOM FOOT-AND-MOUTH DISEASE OUTBREAK: A COMPARISON OF METHODS

4.1 Introduction

Deep reinforcement learning (RL) methods have recently illustrated strong performance in decision spaces with high dimensional input. In the field of robotics, these methods have been successfully used to teach robots to perform simple tasks such as walking or picking up objects (Schulman et al., 2016; Levine et al., 2016). One particular deep RL method, deep Q-networks (DQN) is well-suited for image data due to its use of convolutional neural networks (CNN). This method successfully trained DQN agents to play video games, such as Atari 2600 or Snake, using the red, green and blue (RGB) values from screenshots of the game (Mnih et al., 2015; Chavez et al., 2015; Sprague, 2015). While DQN has shown promise with high dimensional inputs, there have been several challenges with this method. For example, DQN is composed of many hyperparameters, related to CNN and Q-learning, and all need to be tuned to ensure optimal performance. In training a DQN agent to play the Atari 2600 games, Mnih et al. (2015) mentioned that the hyperparameters were selected by performing an informal search, and a systematic grid search would result in high computational costs. Furthermore, choice of hyperparameter values can affect the final policy. Sprague (2015) illustrated this point in a separate investigation to train a DQN agent to play the Atari 2600 games. Different values of the step size and discount rate hyperparameters were separately used to train a DQN agent, and the resulting learning trajectories were highly sensitive to choices of hyperparameters. In addition, DQN tends to have a high degree of variance during training and thus the final policy. Anschel et al. (2017) suggested that one of the contributors to high variance is target approximation error, which refers to the difference in the estimated expected reward and the true target. Some hypotheses as to why target approximation error occurs are: estimation of CNN weights due to inexact minimization, limited representation of a state via a CNN, and the finite size of experience replay.

These are only some of the challenges using DQN. Thus, one must question if a simpler approach may yield similar or adequate results. In this investigation, I extended the use of DQN to the management of disease outbreaks. I specifically investigated management of the foot-and-mouth disease (FMD), which is a highly transmittable disease that affects cloven-hoofed animals. The disease dynamics I

used reflected those of the 2001 United Kingdom FMD outbreak, where over six million cattle were culled in order to manage the epidemic. I compared the results of management using DQN to simpler alternatives: culling farms at random, culling only infected farms, and using a risk model to choose which non-infected farms to cull in four separate case studies.

4.2 Overview of Deep Q-Networks

As with any reinforcement learning problem, DQN assumes that the decision problem of interest follows a Markov decision process (MDP). An MDP assumes that the decision only depends on the current environment and not on the path the decision process had to take to arrive at the current environment. All reinforcement learning problems require the objectives, states, actions, and utilities (also expressed as rewards or losses) to be defined *a priori*. These components are combined in the Bellman optimality equation which calculates the expected reward of being in state s and executing action a.

$$Q^*(s,a) = E_{s'}[r + \gamma max_{a'} Q^*(s',a')|s,a] \tag{4.1}$$

DQN combines Q-learning, as a means to explore the decision space and update the expected reward, with CNN. As discussed in Chapter 2, methods to construct optimal policies such as dynamic programming experience computational storage limitations in large decision spaces. CNN serves as a function approximator for the expected reward, where the expected reward is stored in a function as opposed to a large look-up table. In Q-learning, an agent uses Q^* to choose an action from the current state, resulting in a new state and the corresponding reward to be be observed following the state dynamics that result from the decision. This information is used in a Q-update, presented in Equation (4.2), to update the expected reward function with the new information.

$$Q(s,a) = Q(s,a) + \alpha[r + \gamma max_{a'} Q(s',a') - Q(s,a)] \tag{4.2}$$

When the Q-update equation is combined with CNN, a new expected reward function is expressed using, $Q(s,a;\theta)$ where θ represents the collection of convolutional neural network weights. With each iteration of training, the squared error loss is calculated:

$$L(\theta) = E_{s,a,r}(E_{s'}(r + \gamma max_{a'} Q^*(s',a';\theta^-)) - Q(s,a;\theta))^2 \tag{4.3}$$

where θ^- refers to the previous set of weights. Following the convolutional neural network algorithm, partial derivatives of Equation (4.3) are calculated with respect to θ, and stochastic gradient descent can be performed to update the network weights.

There are two additional components to the DQN algorithm that help with convergence: experience replay and target updating. With experience replay, an agent can experience the effects of taking actions with which the system has had experience. (Lin, 1993). This is accomplished by saving experiences (s_t, a_t, r_t, s_{t+1}) from each time step in a list, E of pre-specified length, N. Each time the convolutional neural network is fit, a random sample of experiences from E is drawn, and is used in the weight updating process. The list E contains only the N most recent experiences to ensure the estimates are using the most relevant data. Consecutive experiences are highly correlated, thus randomizing previous experiences reduces correlation in the estimates of $Q(s, a; \theta)$ (Mnih et al., 2015). Target updating refers to using a separate network for the target, $Q(s', a'; \theta^-)$. As previously described in Chapter 3, the target network and the Q-network, for $Q(s, a, \theta)$, initially have the same weights, θ. The Q-network weights are updated with each time step, while the target network weights remain the same. After C time steps, the Q-network is cloned and equated to the target network for the next C time steps. This method may require more episodes of training, but can greatly improve the stability of training and prevent optimal policy divergence (Lillicrap et al., 2016).

4.3 Methods

The 2001 United Kingdom FMD outbreak was managed using a combination of culling infected premises and pre-emptive culling (Tildesley et al., 2009). However, under limited resources, the decision problem requires an additional layer of complexity, intervention prioritization. The FMD contingency plan published by the Department for Environment, Food and Rural Affairs (DEFRA) currently addresses a need to prioritize identification of the source of an outbreak, but there are currently no guidelines specifying which premises should be prioritized to receive an intervention (DEFRA, 2004). Previous studies have examined prioritization in pre-emptive strategies, i.e. - reactive vaccination programs which identify contiguous premises and direct contacts, or premises that would contribute most to future transmission (Tildesley et al., 2006; Keeling et al., 2003). I explored four different methods to construct a prioritization scheme regarding how farms should receive an intervention: random culls, cull infected premises, a risk model using logistic regression, and DQN.

All four methods required a six-day monitoring period to ensure that the outbreak was terminated to account for the latent behavior of the disease. According to Keeling and Rohani (2007), farms can experience a latent period of five days, where the livestock may not exhibit disease symptoms but will ultimately become infected. Thus on the sixth day, it may be possible for farms to observe infections. If this event happened in any of our simulates outbreaks, I resumed management of the outbreak.

4.3.1 Random Culls

Random culling is the simplest of the four investigated methods to simulate. For each day of management I implemented the following process: if the daily culling capacity is x farms, choose x non-culled farms at random to cull. This approach does not consider any farm-level attributes such as the number of cattle at a farm, distance to an infected farm, or even a farm's infection status in the decision-making process. Keeling et al. (2003) used random farm selection to simulate management of the 2001 United Kingdom FMD outbreak, and compared the results to other methods, e.g. - intervene at the largest farms, intervene at infected premises, intervene at farms with direct contact (DC) of infected premises, and intervene at farms near infected farms, contiguous premises (CP). Of the explored interventions, random action yielded the longest epidemics and the largest total number of infected farms. Motivated by these results, I let random culling serve as a baseline to compare the performance of the three other methods.

4.3.2 Cull Infected Premises

Previous research has examined prioritization of pre-emptive outbreak strategies, however less is known about prioritizing interventions to infected premises. Keeling and Rohani (2007) constructed an algorithm to simulate the 2001 United Kingdom FMD outbreak, and there were two main contributors to a farm becoming infected: the number of cattle at a farm, and the distance to an infected farm. Intuitively, the more cattle there are at a farm, the more likely the farm is to become infected. Using this intuition, I implemented a "cull-infected-premises" approach, where farms were ranked based on their respective number of cattle. Every day in the management process, I implemented the following steps:

1. Identify all of the infected farms

2. Rank the farms based on the number of cattle, regardless of when an infection

was reported

3. If daily culling capacity is x, choose the x top-ranked farms to cull

4. Repeat steps (1) - (3) until there are no more infected farms for six consecutive days

McLaws and Ribble (2007) conducted a survey of 24 FMD epidemics that occurred between 1992 and 2003, and investigated the relationship between time to detection and epidemic size. Time to detection was defined as the difference between the estimated date of infection and the date the infection was reported. It was concluded that there was no direct relationship between the time to detection and the epidemic size, i.e. - the total number of infected premises or livestock culled. I assumed there was no lag time between when the virus was contracted and when the infection was reported in our simulated outbreaks. Furthermore, the findings by McLaws and Ribble (2007) motivated the decision to rank farms based on number of cattle regardless of when the infection occurred.

This method of ranking infected farms based on number of cattle has previously illustrated better outcomes in comparison to a random action approach (Keeling et al., 2003). However, this method may not be adequate to manage epidemics where the rate of transmission is high (Woolhouse, 2003). Keeling et al. (2001) illustrated that if only infected premises were culled, the 2001 United Kingdom FMD outbreak would have been much larger. Pre-emptive measures of intervening at non-infected farms may need to be combined with infected premises interventions to better manage the outbreak.

4.3.3 Logistic Regression

I constructed a logistic regression model to determine which non-infected farms were most at risk to become infected in the future. As previously discussed in Chapter 2, regression is often paired with disease surveillance in the literature as a means of pre-emptive disease outbreak planning. However, regression can also be used in a decision analysis framework. We can easily calculate the immediate utility of a control strategy suggested by the regression model. Logistic regression provides more opportunity for inference and exploration of the covariate-outcome relationship in comparison to a random culling approach or a cull infected premises strategy. In addition, logistic regression provides a simpler modeling alternative to our last method, DQN.

To account for the stochasticity of the disease spread, the data set used in the logistic regression was generated through many simulations of the FMD outbreak. I executed the following steps 1,000 times:

1. Begin in the initial state of the outbreak
2. Collect all of the predictor information of non-infected farms at baseline
3. Simulate the disease and monitor for six days
4. Collect the infection statuses of all farms on the sixth day

Thus, if there were x non-infected farms under consideration, the data set contained $1,000x$ observations. The following predictors were considered in the logistic regression: distance to the nearest infected farm, number of cattle at the nearest infected farm, and number of cattle at the current farm. The final model was selected using Akaike information criterion (AIC). Equation (4.4) presents the logistic regression model used to predict the risk, p, of a susceptible farm becoming infected in six days.

$$\begin{aligned} log(\frac{p}{1-p}) = \beta_0 &+ \beta_1(\text{Distance to nearest infected farm}) \\ &+ \beta_2(\#\text{Cows at nearest infected farm}) \\ &+ \beta_3(\#\text{Cows at current farm}) \\ &+ \beta_4(\text{Distance to nearest infected farm}) * (\#\text{Cows at nearest infected farm}) \end{aligned} \tag{4.4}$$

Once the regression coefficients from Equation (4.4) were estimated, the non-infected farms at baseline could then be ranked. Infected farms were ranked as well, based on the number of cattle. Infected and non-infected farms were culled according to rank order every day until the daily culling capacity was met. More resources were given to non-infected farms to preemptively terminate the outbreak. For example, if there were resources available to cull all of the cattle at any five farms per day, three of the culled farms would not be infected, and two of the culled farms would be infected.

Continuing to use the initial risk quantities calculated for the farms at baseline as the disease progressed, would result in stale estimates. Thus, using the same logistic regression coefficients calculated at baseline, I updated the risk quantities in real time using the most recent data. Figure 4.1 illustrates the complete sequence of events used to implement the logistic regression management method.

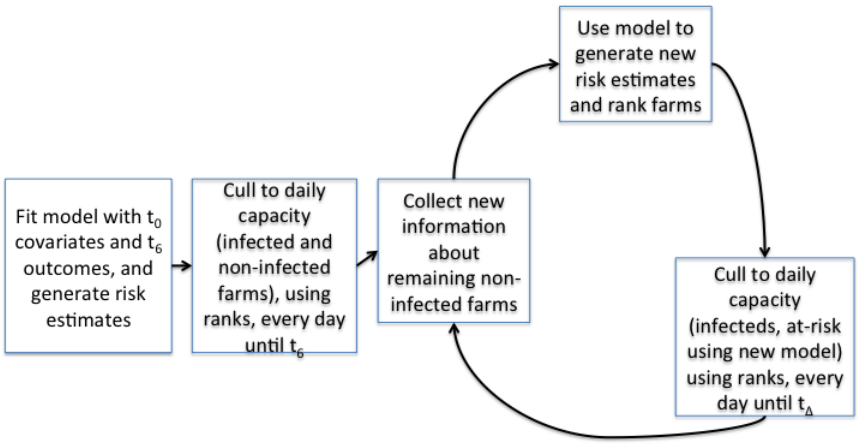

Figure 4.1: Sequence of steps needed to implement FMD management using a combination of logistic regression and culling infected farms based on number of cattle. Notice that the model it fit only once, using the predictor information collected at baseline. The risk estimates, and the culling order of the infected farms are updated every δ days, using the most recent information.

4.3.4 Deep Q-Networks

The DQN method does not choose which farms to cull based on infection status, as with the cull infected premises or the logistic regression methods. Rather, DQN chooses which farms to cull based on a trained convolutional neural network. As with any MDP, the states, actions, and rewards were pre-specified. To better capture the spatial relationships between farm locations, the state at time t was defined by a map of the disease outbreak. Figure 4.2 illustrates the construction of the state for a given epoch of training. The first step consisted of plotting the farm locations and identifying the farm-level infection status on an $m \times m$ kilometer space (represented in the left panel in Figure 4.2, black represents an infected farm, and white represents a susceptible farm). A grid was then overlaid on the plot to create "pixels," i.e.- grid squares (represented by the second panel in Figure 4.2). Traditionally, a state in DQN with image data is defined by the RGB values for each pixel (Mnih et al., 2015). In this study, the infection status of a farm in a pixel was used. The following was used as a key to construct the state: susceptible farm- 0, infected farm- 1, culled farm/no farm- 2 (represented in the right panel in Figure 4.2). This construction of the state was implemented at every epoch of training. In this example, on a 10×10 kilometer space, each grid square had dimension 1×1, to ensure that no more than

one farm occupied a grid square. The grid square dimension can easily be smaller to better capture relative distance between farms, however this creates a larger 2-D array, causing computational challenges with stochastic gradient descent and the serial updating nature of DQN.

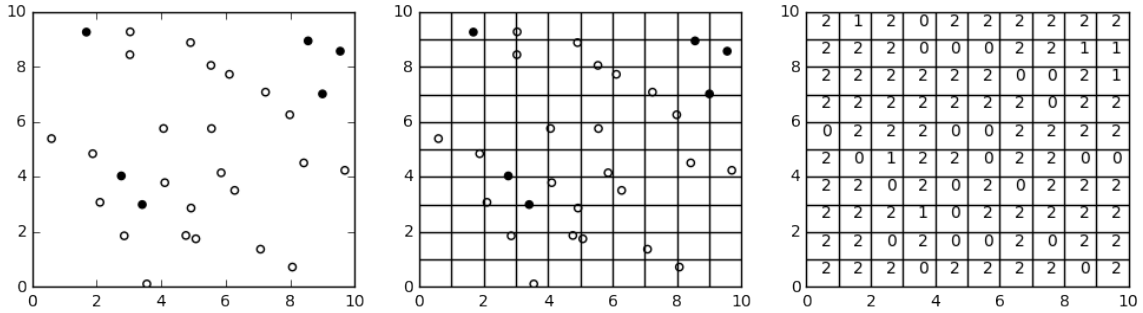

Figure 4.2: Constructing the state at a time point consists of three steps 1) plot the points of the farm locations and color code by infection status (black- infected, white- susceptible), 2) overlay a grid on the previous plot, 3) trichotomize the infection status

At each epoch of training, the DQN agent is tasked with determining which farms to cull under resource constraints. In Figure 4.2 there are 30 farms in the 10×10 kilometer space. Suppose that there are resources to cull all of the cattle in any five farms, per day. Rather than defining the action space with $\binom{30}{5}$ possibilities at time t, the agent is presented with a smaller action space. At time step t, the DQN agent chooses one farm out of the 30 to cull. In that same time step, the DQN agent chooses a second farm to cull out of the remaining 29. Once the DQN agent chooses five separate farms to cull, then the outbreak evolves and a new state and reward are observed. This action space construction reduces the number of possible actions in the first day of management from $\binom{30}{5}$ to 140 (30 + 29 + 28 + 27 + 26). There are other approaches to accommodate decision spaces with large action spaces, such as least squares policy iteration with multi-action (Wang and Yu, 2016). However, techniques to accommodate decision spaces with large state and action spaces are still under study.

The immediate rewards, r are closely tied to the objectives of disease management. Ideally, an outbreak should be managed as quickly as possible with minimal costs. For this reason there were multiple components to the immediate reward structure. A -100 reward was observed for every farm culled, which encouraged conservative culling in the management process. A terminal reward of the number of cows alive at the end of an episode was also applied, which encouraged cost-effective behavior. Because FMD has a latent period of five days, it was possible for outbreaks to resurge. This would

cause policy-makers to have to wait to ensure that the outbreak was truly terminated. For this reason there was a -500 reward associated with the agent assuming the outbreak was terminated, when in truth an outbreak resurgence was about to occur. This immediate reward structure was used for each of the four scenarios investigated in this study. Selection of immediate rewards was made through prioritization of objectives; diminishing outbreak resurgence was the highest priority and was thus given a higher negative reward for not being met. In a realistic setting, specification of objectives can impact the final policy. Probert et al. (2015) illustrated that final controls can differ based on the metric that defines success. For this reason, it is imperative that a decision-maker explicitly considers what they want to accomplish with outbreak management.

DQN requires some degree of hyperparameter tuning. In this study, an informal search method was used to choose the best combination of hyperparameters. The starting set of hyperparameters were nearly identical to that of Mnih et al. (2015). DQN is also known to have a high degree of variance during training, and thus with the final policy. For this reason, each combination of hyperparameters were trained three separate times and the model with the best reward trajectory, defined by being the most stable, and achieving the highest rewards, was used for the final policy.

4.4 Outbreak Generation

A model of the 2001 United Kingdom FMD disease dynamics was used to generate the next state after an action(s) was implemented. The 2001 United Kingdom FMD outbreak was characterized by a susceptible-exposed-infected-recovered (SEIR) compartment model at the farm level. Following exposure to the virus, there was a five-day latent period. The outbreak in this study was simulated using an individual-based FMD model as specified by Keeling and Rohani (2007). The rate at which a susceptible farm became infected was represented by:

$$\lambda_i = N_i s_{cow} \sum_{j \in infectious} N_j \tau_{cow} K(d_{ij}) \tag{4.5}$$

where N represents the number of cows at a farm, s represents a cow's susceptibility to the disease, τ represents a cow's transmissibility of the disease, K represents the transmission kernel, and d_{ij} represents the distance between the susceptible farm i and the infectious farm, j. The following parameters, s, τ, and K were individually selected for each case study in this investigation to better accommodate the characteristics of each landscape (e.g. - size of the space, positioning of farms, etc). If s or

τ were too large, then the outbreak would spread too quickly, preventing any type of intervention under resource constraints to be effective. If the transmission kernel was a distribution with very large tails, then there would be a larger chance for farms that were very far away to become infected in a single time step. This may be feasible in practice due to cattle movement, movement of equipment, etc.

4.5 Case Studies

There were four cases investigated in this study. The first case represents a proof of concept that DQN can be used to manage a disease outbreak. The succeeding two cases present small-scale examples that address particular aspects of outbreak management under resource constraints. The final case explores scaling the decision problem. For each case, the initial state did not change between episodes of training. In practice, a policy-maker would provide the initial state of the outbreak and request the best sequence of actions for management. Thus, this method overfits to the starting state of the outbreak. Although the starting state for each case is the same, the outbreak progression will vary due to the stochastic nature of the outbreak algorithm. This stochasticity makes the decision problem more challenging for the DQN agent. For each of the four case studies, I trained the DQN agent in three separate instances and chose the DQN agent that generated the best stability in training. As previously mentioned, DQN is known to have a high degree of variance during training, thus I used the most stable DQN agent to test in each of the four case studies. If more instances of training instances were implemented, it could be possible to observe agents with better performance, refer to Chapter 3.

4.5.1 Case 1: Which infected farm to leave out?

In the first case study I placed 30 farms randomly on a 10×10 kilometer grid. The initial pixelated state is provided in Figure 4.3. The size of each point in Figure 4.3 represents the number of cattle at the corresponding farm. The number of cattle was uniformly distributed between 25 and 500, with a total of 6965 cows in the entire landscape. The range of cattle reflected realistic farm sizes in Cumbria, United Kingdom in 2001 (Tildesley et al., 2010). A cow's susceptibility to the disease, s, and a cow's transmissibility of the disease, τ, were both chosen to be 6.3×10^{-5}. Transmissibility was slightly larger than specified by Keeling and Rohani (2007) to accommodate a more realistic spread on a smaller landscape. The transmission kernel in this case was: $K(distance) = \frac{1}{distance+400}$. This kernel has a large tail over the possible distances,

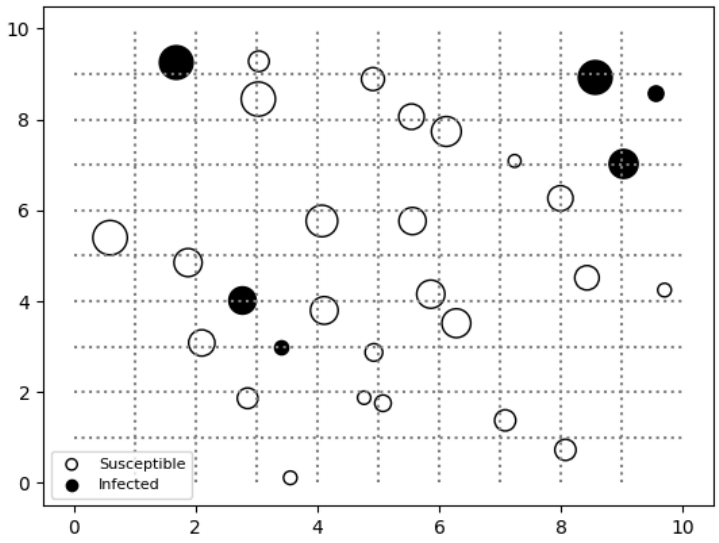

Figure 4.3: Initial state for case 1.

meaning that farms farther away have a high probability of becoming infected, relative to a narrower-tailed kernel. This will present some challenges for the DQN agent, because there may not be a "learnable" culling pattern for management. "Learnable" here refers to discovering rules that would guarantee the management objectives to be met. I selected six farms at random to be initially infected, represented by the black points in Figure 4.3. In this case study, I assumed that each day there were resources to cull all of the cattle at any five farms. Because there were initially six infected farms, this case can be thought of as: which infected farm should not be included in the first day of culling. Intuitively, the outbreak can be terminated by culling all six infected farms. However, if the first five infected farms were incorrectly chosen then the neglected infected farm could create secondary infections on the second day of management.

For the logistic regression method, recall that more resources were provided to cull non-infected farms as a pre-emptive management measure. For this method, if there were at least two infected farms, I culled the two highest ranked infected farms, according to number of cattle, first. Then I culled the three highest ranked non-infected farms, according to the logistic regression. Once five farms were culled, the outbreak evolved and the next state was observed. If there were less than two infected farms, those infected farms were culled and a six day monitoring period occurred to ensure the outbreak was terminated. Table 4.1 presents the logistic regression

Variable	Estimate of β	Standard Error	Exp(β)	95% CI (Odds Scale)
Distance to nearest infected farm	-3.1607	0.046	0.0423	(0.0388, 0.0464)
#Cows at nearest infected farm	-0.0704	0.001	0.9320	(0.9302, 0.9338)
#Cows at current farm	0.0110	0.000	1.0110	(1.0106, 1.0115)
(Distance to nearest infected farm)* (#Cows at nearest infected farm)	0.0263	0.000	1.0266	(1.0260, 1.0273)
Constant	2.8360	0.081		
			LR χ^2	20427
			AIC	21914

Table 4.1: Case 1: Regression coefficients used for the logistic regression model. Model selection was performed using AIC and previous information. Notice that distance to the nearest infected farm is highly associated to the outcome of a farm becoming infected.

coefficients used to rank the non-infected farms. These regression estimates were fixed over the course of management, however the risk estimates were updated every six days as new data were collected. The magnitude of the regression coefficient for the distance to the nearest infected farm predictor, suggests that this predictor is highly associated with a farm becoming infected in six days. The regression coefficients for all four case studies were used primarily for predictive purposes and not inferential purposes. However, these regression coefficients suggest that under this transmission kernel, distance may be highly associated with the outcome.

After fitting the logistic regression to the data, and training the DQN agent for 1,200 episodes, I performed 2,000 simulations of testing. The best trajectory begins to plateau after about 800 episodes suggesting model stability. DQN yields the highest total reward of the four methods, due to the fact that DQN never culled more than six farms. The cull infected premises approach resulted in an average of eight farms to be culled. The lower reward from the cull infected premises approach occurred because after the first day of management, and a period of monitoring, at least one exposed farm appeared. This suggests that it matters which farm is neglected by the end of the first day of management. In the DQN approach, the farm with 453 cattle was always saved for the second day of management. However, in the cull infected premises approach, this farm was always culled in the first day of management.

The logistic regression approach had a lower total reward overall compared to the cull infected premises approach. This suggests that ranking non-infected farms according to the fitted model provides no additional support for management. As expected, the random culling intervention had the lowest total reward compared to the other three methods. This first case study suggests that DQN could successfully be used to manage a simulated disease outbreak and outperform other methods.

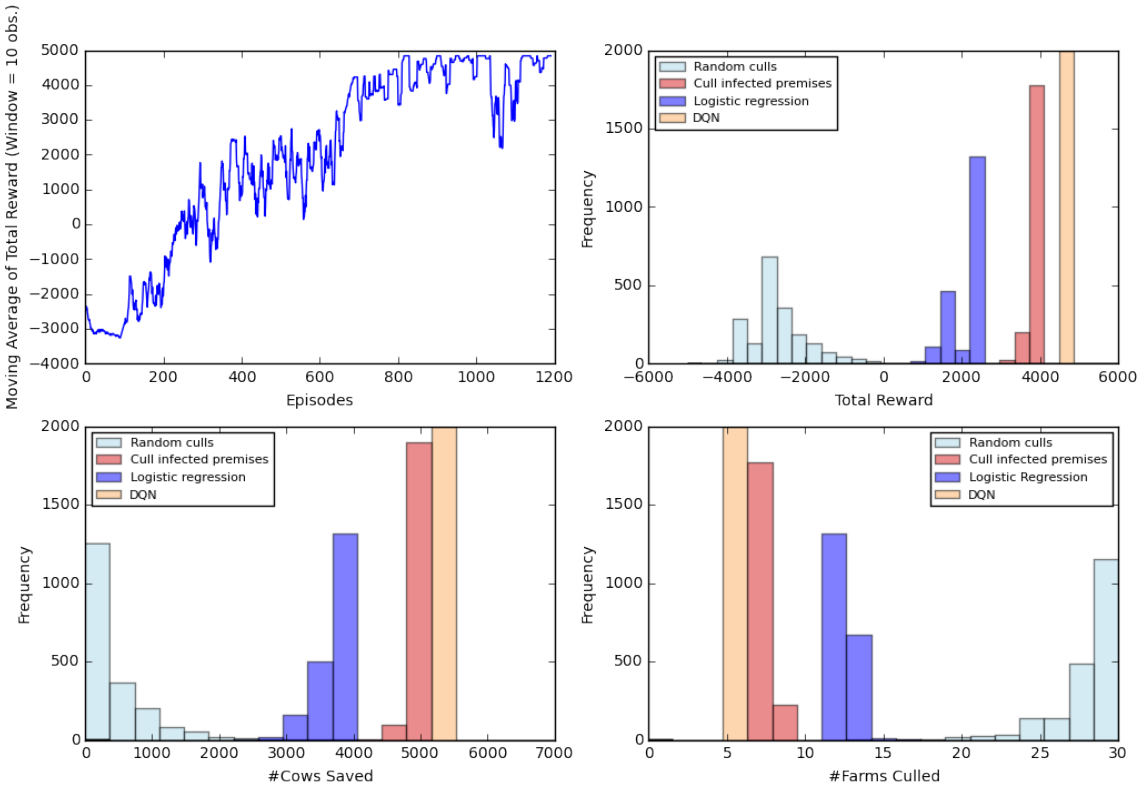

Figure 4.4: Comparison of results for first case study across 2,000 simulations of testing. Top left panel presents the trajectory of total reward during the 1,200 episodes of training. Top right panel presents histograms of total reward across the four different methods from 2,000 simulations of testing. Bottom left panel presents histograms of number of cattle saved across the four different methods from 2,000 simulations of testing. Bottom left panel presents histograms of total number of farms culled across the four different methods from 2,000 simulations of testing. Notice that the DQN method performs the best of the four methods.

	Total Reward	#Cows Saved	#Farms Culled	#Outbreak Resurgences
Random Culls	-2643 (-3500, -671)	393 (0, 1769)	29 (22, 30)	1 (0, 2)
Cull Infected Premises	3802 (3393, 3841)	5014 (4733, 5041)	8 (7, 8)	1 (1,1)
Logistic Regression	3584 (3143, 3688)	2173 (1234, 2488)	13 (12, 14)	1 (0, 1)
DQN	4737 (4737, 4734)	5336 (5336, 5336)	6 (6, 6)	0 (0, 0)

Table 4.2: Statistics for each of the four methods for the first case study: averages (95% CI). Notice that the DQN method outperforms the other three methods.

4.5.2 Case 2: Faster outbreak and conservative resource constraints

In the second case study, I explored a more challenging decision problem. The initial state was the same as the first case study: 30 farms, six farms initially infected, with the number of cattle per farm unchanged. However, in this case study, I assumed all of the cattle at any one farm could be culled per day. A cow's transmissibility was also slightly increased to 8.4×10^{-5}. Both the increase in transmission and the lower daily culling capacity presented more of a challenge for the DQN agent.

Notice that even though the transmission constant was increased from the first case study, distance to the nearest infected farm is highly associated with a farm becoming infected, Table 4.3. In this second case study, the logistic regression management method still prioritized resources to non-infected farms as a pre-emptive management measure. However, because only one farm could be culled per day, I used a cyclic culling sequence. The culling cycle began with culling an infected farm on the first day, followed by culling another infected farm on the second day. I then culled three non-infected farms over the next three days, according to the rankings from the logistic regression model. Then the cycle started over, two infected farm was culled over the next two days.

Variable	**Estimate of β**	**Standard Error**	**Exp(β)**	**95% CI (Odds Scale)**
Distance to nearest infected farm	-0.7056	0.018	0.4938	(0.4767, 0.5115)
#Cows at nearest infected farm	-0.0117	0.000	0.9884	(0.9879, 0.9888)
#Cows at current farm	0.0024	0.000	1.0020	(1.0023, 1.0027)
(Distance to nearest infected farm)* (#Cows at nearest infected farm)	0.0049	0.000	1.0050	(1.0047, 1.0050)
Constant	0.2953	0.046		
			LR χ^2	5194
			AIC	53259

Table 4.3: Case 2: Regression coefficients used for the logistic regression model. Model selection was performed using AIC and previous information. Notice that distance to the nearest infected farm is highly associated to the outcome of a farm becoming infected.

The total reward trajectory continues to increase over the 10,000 episodes of training, Figure 4.5. There is an increased variation in the trajectory compared to that of the first case study, even after smoothing with a moving average. This is due to the higher degree of stochasticity of the outbreak caused by the increase in transmission and decrease in daily culling capacity. Even if the DQN agent implemented the same set of actions in two realizations of the disease outbreak, the outbreak progression between those two instances may be vastly different. Despite this, the trajectory of total reward continued to increase over the course of training. There were some outbreak resurgences (mean = 1, 95%CI = (1, 2)), in comparison to the first case study,

Table 4.4. However, this phenomenon was expected due to the complexities of this decision problem.

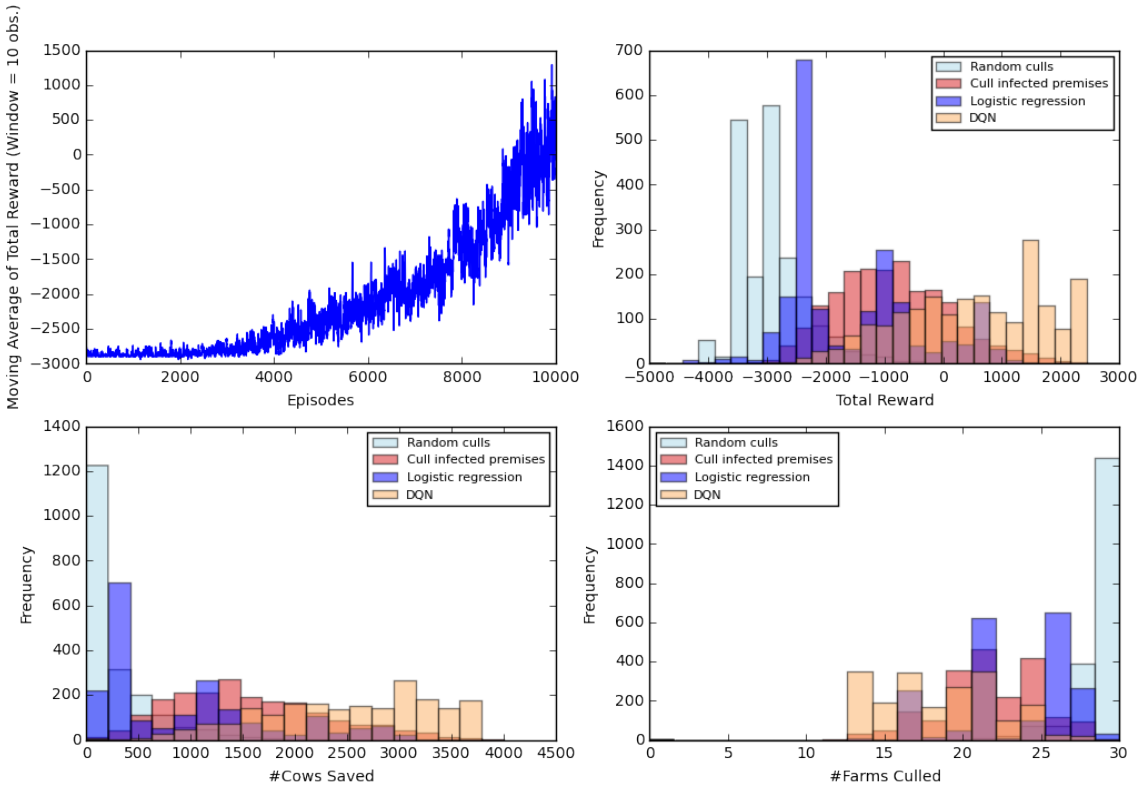

Figure 4.5: Comparison of results for second case study across 2,000 simulations of testing. Notice that the DQN method performs the best of the four methods the majority of the time. In some scenarios of testing, the other three methods perform just as well or better than DQN.

These results illustrate that DQN generated the highest total reward of the four compared methods, Figure 4.5. However the distributions of total reward were not as distinct as in the first case study; there was some overlap between the DQN and cull infected premises distributions of total reward. This was due to DQN occasionally culling more farms than the cull infected premises approach. In some instances, the DQN agent would calculate a higher utility in culling a susceptible farm as opposed to culling an infected farm. This resulted in occasionally more farms being culled compared to the cull infected premises or logistic regression approaches. These results reflect the management objectives specified. If there were a higher negative reward associated with each cull, then the final policy may be slightly different.

As with the first case study, the logistic regression approach generated a lower average total reward compared to the cull infected premises approach. Also as expected, all three methods generated a higher average total reward than the baseline approach, random culling.

	Total Reward	#Cows Saved	#Farms Culled	#Outbreak Resurgences
Random Culls	-2939 (-4000, -1363)	247 (0, 1254)	29 (24, 30)	1 (0, 2)
Cull Infected Premises	-874 (-2561, 1293)	1554 (412, 3156)	22 (15, 27)	1 (0, 2)
Logistic Regression	-1568 (-3025, 814)	937 (87, 2875)	24 (16, 28)	1 (0, 3)
DQN	502 (-1973, 2427)	2522 (878, 3727)	19 (13, 26)	1 (1, 2)

Table 4.4: Statistics for each of the four methods for the second case study: averages (95% CI). Notice that the DQN method outperforms the other three methods.

4.5.3 Case 3: Identifying "bridging" farms

In the third case study, I targeted a specific feature of disease spread, network structure. Network structure refers to how individuals in a community are connected to each other. For example, Salathe and Jones (2010) exploited network structure to trace disease spread using friendship data collected from Facebook. They found that targeting individuals that "bridge" communities was more effective than targeting highly connected individuals, i.e.- individuals that were in contact with many others. In the third case study, I used bridging farms to incorporate an inherent network structure. Figure 4.6 illustrates the initial state. The number of cattle per farm was uniformly distributed between 25 and 500, as in the first and second case studies, with a total of 7922 cows in the entire space. The two farms in the middle of the grid space, with 129 and 85 cattle, are the bridging farms that connect the two clusters of farms in the corners of the 10×10 kilometer space. To accommodate the bridging farms, there was an additional constraint to the transmission kernel. The upper cluster of farms could only become infected if either of the bridging farms became infected. This emphasized the behavior of the two farms "bridging" the two clusters of farms.

A cow's transmissibility of the disease and the transmission kernel were the same as in the second case study. This case study also had the same daily culling capacity constraints as the second case study: all of the cattle at any one farm could be culled per day. This presented even more of a challenge to the DQN agent. Similar to the second case study, there would be more exposed farms due to the conservative daily culling capacity. Thus, the DQN agent had to learn to manage the outbreak with states that were not observable. Moreover, the DQN agent would need to also learn specific dynamics of the outbreak: if one of the bridging farms became infected, then

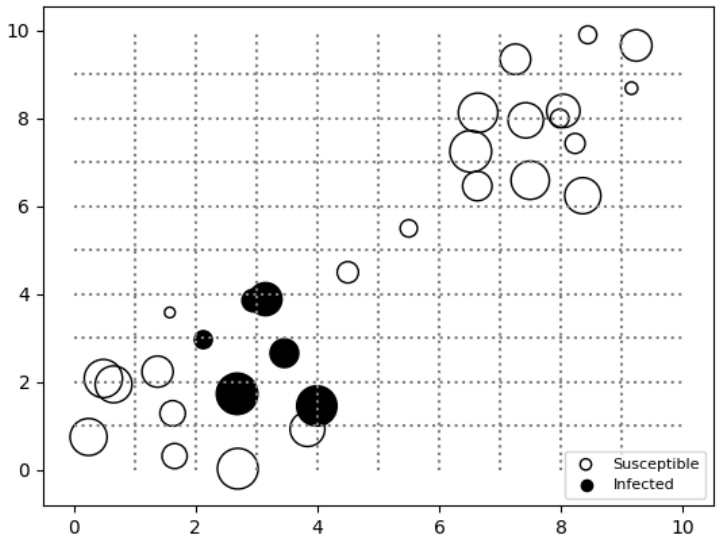

Figure 4.6: Initial state for case 3, dot size is proportional to the number of cattle at the farm.

the upper cluster of farms would begin to observe infections.

Variable	**Estimate of β**	**Standard Error**	**Exp(β)**	**95% CI (Odds Scale)**
Distance to nearest infected farm	-6.8845	0.260	0.0010	(0.0006, 0.0017)
#Cows at nearest infected farm	-0.0331	0.001	0.9674	(0.9653, 0.9695)
#Cows at current farm	-0.0165	0.000	0.9836	(0.9827, 0.9845)
(Distance to nearest infected farm)* (#Cows at nearest infected farm)	0.0094	0.001	1.0094	(1.0081, 1.0107)
Constant	21.4058	0.554		
			LR χ^2	36503
			AIC	14246

Table 4.5: Case 3: Regression coefficients used for the logistic regression model. Model selection was performed using AIC and previous information. Notice that distance to the nearest infected farm is highly associated to the outcome of a farm becoming infected.

I trained the DQN agent for 10,000 episodes, Figure 4.7. Notice that the total reward trajectory begins to plateau at about 8,000 episodes, suggesting that the training has begun to stabilize. Just as with the first and second case studies, the DQN agent began its training by culling nearly every farm, and over time the agent learned to behave in a manner that met all of the objectives. The DQN policy suggests that the bridging farms should be culled early in the management process, however these farms should not be the first to be culled. DQN had the best performance of the four methods after 2,000 episodes of testing, Figure 4.7. While DQN was successfully

able to identify and cull the bridging farms, the cull infected premises approach was not. Because the bridging farms were smaller, they were ranked lower on the culling order. This resulted in the outbreak spreading to the upper cluster of farms.

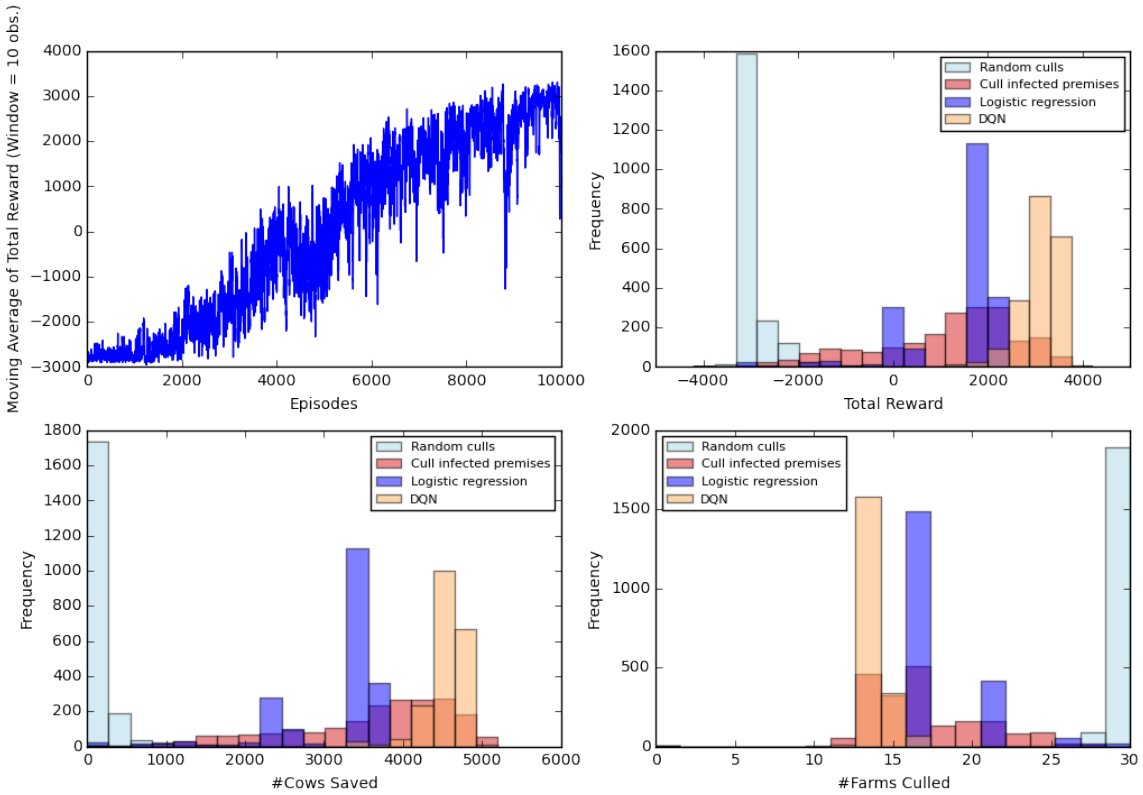

Figure 4.7: Comparison of results for third case study across 2,000 simulations of testing. Notice that the DQN method performs the best of the four methods the majority of the time. Notice that DQN almost always has the best outcomes, and sometimes the cull infected only method is comparable.

Because only one farm could be culled per day, culling was performed using the same cyclic sequence of culls as in the second case study. The logistic regression approach sometimes outperformed the cull infected premises approach, in contrast to the previous case studies. In some instances of testing, the logistic regression approach identified that the non-infected bridging farms were at high risk of becoming infected in the next six days. In turn, under this method there were many instances of testing with shorter outbreak durations and a smaller number of farms culled. However, in some instances, the logistic regression failed to cull the bridging farms early enough resulting in a larger outbreak, represented by the bi-modality in the total reward distribution.

	Total Reward	#Cows Saved	#Farms Culled	#Outbreak Resurgences
Random Culls	-2886 (-3000, -2016)	89 (0, 784)	30 (27, 30)	0 (0, 0)
Cull Infected Premises	1115 (-2243, 3404)	3623 (1359, 5104)	17 (12, 25)	2 (0, 3)
Logistic Regression	1319 (-1922, 2010)	3179 (1109, 3610)	19 (16, 26)	1 (0, 1)
DQN	3138 (2173, 3605)	4590 (3994, 4905)	14 (13, 16)	0 (0, 1)

Table 4.6: Statistics for each of the four methods for the third case study: averages (95% CI). Notice that the DQN method outperforms the other three methods.

4.5.4 Case 4: Scaling

In the fourth case study, I investigated a larger FMD decision problem. In the previous three case studies, I considered a landscape with 30 farms. In this fourth case study, I attempted to manage a simulated FMD outbreak with 120 farms. Because more farms were considered in this case study, the size of the landscape increased to 15×15 kilometers. Each grid square still had the same dimension, 1×1 kilometers, as in the previous case studies. The number of cattle per farm was uniformly distributed between 25 and 500 as as with the previous case studies, with a total of 33,638 cows in the entire space. In addition, this case study used physical landscape features to influence the positioning of farms. In the United Kingdom lakes, rivers, and large highways are some examples of physical landscape features that separate farms into clusters. The farms are separated into clusters, however, not as distinct as the clusters of farms in the third case study (Figure 4.8). Also notice that the initial state has two clusters of infected farms. This reflects the initial state of the 2001 United Kingdom outbreak. By the time the outbreak was detected, infected livestock had already been transported to different areas of the United Kingdom, resulting in several foci of infection (Gibbens et al., 2001).

The transmission kernel denoted in Equation (4.6) was adjusted from the kernel used in the previous three case studies, to reflect a less random spread of infection, with constraints reflecting those used by Keeling and Rohani (2007). The kernel used in this case study assigns a higher probability for closer farms to become infected. A cow's susceptibility to the disease, s, and a cow's transmissibility of the disease, τ, were both decreased to 4.9×10^{-6} to allow for more gradual spread and a chance for an intervention to take place during the outbreak. The previous rates of susceptibility

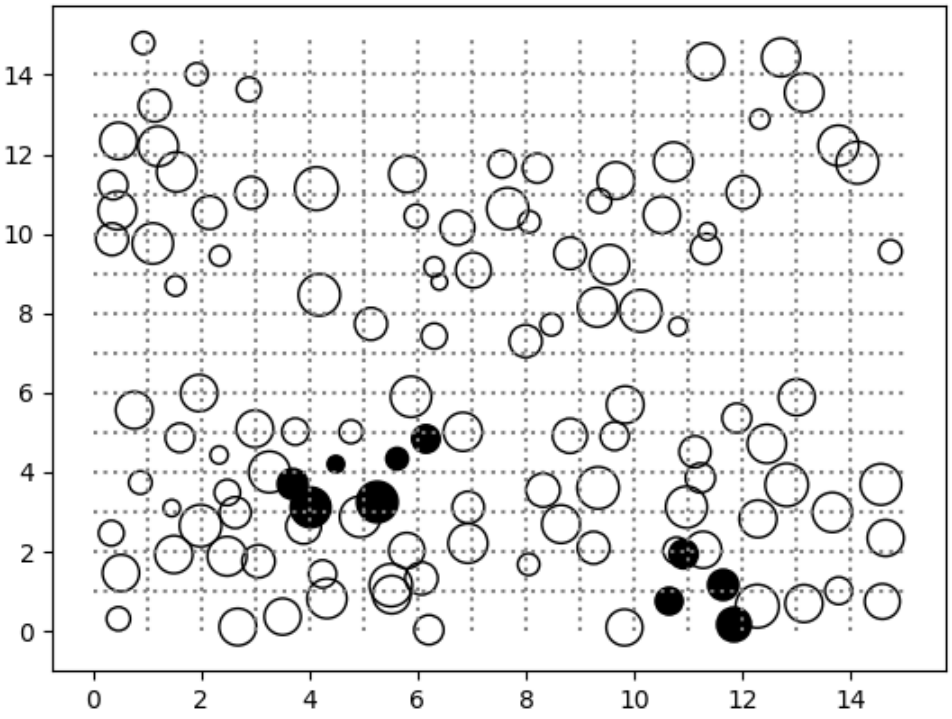

Figure 4.8: Initial state for fourth case study.

and transmission would have caused the outbreak to progress too quickly, rendering an intervention ineffective. Because the number of farms in this case study increased by a magnitude of four, to 120 farms, I also increased the daily culling capacity. Each day, I assumed there were resources to cull all of the cattle at any five farms.

$$K(distance) = \begin{cases} 0.3093 & distance < 0.0138 \\ \frac{1}{\pi(1+distance)} & otherwise \end{cases} \tag{4.6}$$

The top left panel Figure 4.9 presents the trajectory of training the DQN agent for 8,000 episodes. The total reward continues to increase over training, but does not distinctly plateau as with the first and third case studies. This behavior is expected because of the complexity of the decision problem. This fourth case study and the second case study did not target specific features of the outbreak, such as the farm that could be left to cull on the second day of management (first case study), or identifying bridging farms (third case study). This instability was reflected in testing

Variable	Estimate of β	Standard Error	Exp(β)	95% CI (Odds Scale)
Distance to nearest infected farm	-0.7554	0.011	0.4670	(0.4598, 0.4801)
#Cows at nearest infected farm	-0.0068	0.000	0.9932	(0.9929, 0.9935)
#Cows at current farm	0.0011	0.000	1.0011	(1.0010, 1.0013)
(Distance to nearest infected farm)* (#Cows at nearest infected farm)	0.0008	0.000	1.0008	(1.0007, 1.0009)
Constant	0.8131	0.032		
			LR χ^2	27592
			AIC	116506

Table 4.7: Case 4: Regression coefficients used for the logistic regression model. Model selection was performed using AIC and previous information. Notice that distance to the nearest infected farm is highly associated to the outcome of a farm becoming infected.

the DQN model, as the cull infected premises approach performed the best of the four proposed methods, Figure 4.9. The cull infected premises approach yielded the highest total reward, culled the lowest number of farms, and saved the most cattle.

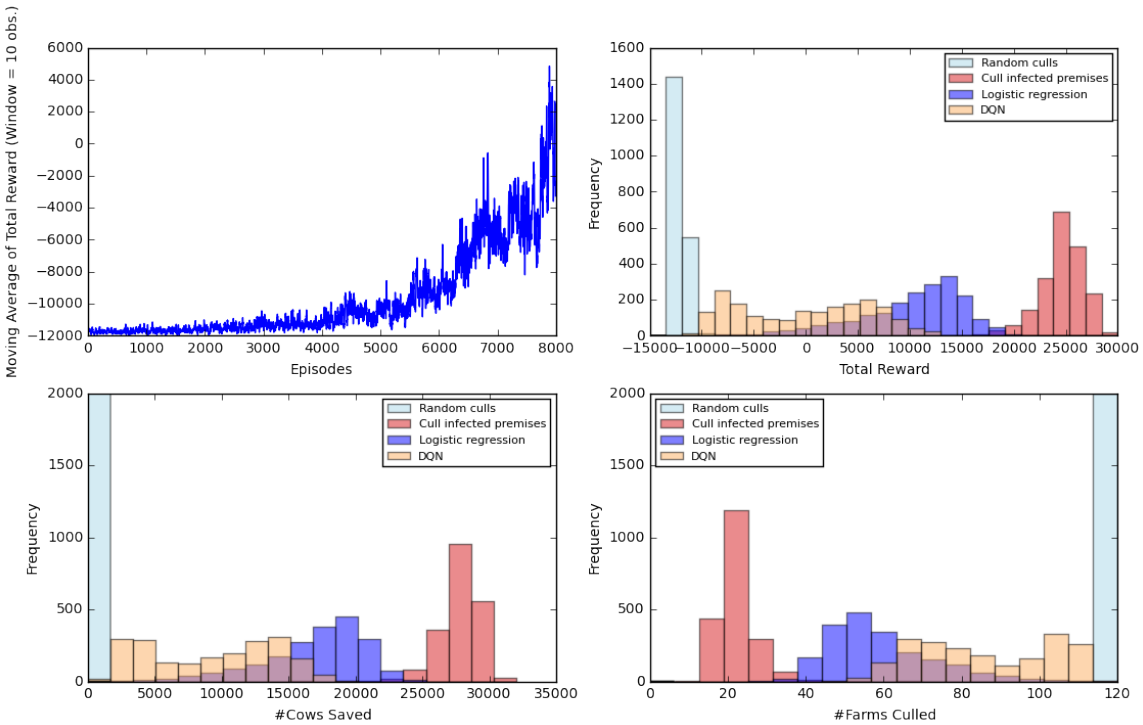

Figure 4.9: Comparison of results for fourth case study across 2,000 simulations of testing. Notice that the culling-infected only method performs the best of the four methods the majority of the time. DQN rarely does better than the logistic regression approach.

DQN, in contrast, performed worse than the logistic regression and the cull infected premises approach. Even after extensive manual hyperparameter tuning, and adjusting the total reward structure, DQN was not able to outperform logistic regres-

sion or the cull infected premises approach. These results suggest that DQN may need more modifications before it can be implemented in a larger decision problem. In addition a simpler model, such as logistic regression, seems to yield better performance and less tuning in comparison to DQN.

	Total Reward	#Cows Saved	#Farms Culled	#Outbreak Resurgences
Random Culls	-11855 (-12500, -10899)	124 (0, 884)	120 (117, 120)	1 (0, 1)
Cull Infected Premises	24600 (19630, 28046)	27777 (24333, 30065)	23 (15, 33)	2 (1, 3)
Logistic Regression	10161 (-2646, 17916)	16972 (7558, 22536)	59 (39, 91)	2 (1, 3)
DQN	-97 (-9586, 10342)	9319 (2210, 16834)	86 (59, 110)	2 (0, 4)

Table 4.8: Statistics for each of the four methods for the fourth case study: averages (95% CI). Notice that the DQN performs poorly compared to logistic regression and the cull infected premises approaches.

4.6 Discussion

In this study, I investigated two primary concepts: use DQN to manage a disease outbreak online, and compare the performance of DQN to other simpler approaches. I found that DQN could be used to manage disease outbreaks with a small number of farms. The first three case studies investigated management with 30 farms. In each of these case studies, DQN generated high total reward during testing and illustrated stability in training. However, DQN did not perform as well in a larger management setting. This is likely due to two main factors: the need for more training, and hyperparameter optimization. DQN is serial in nature, due to stochastic gradient descent and experience replay. A larger number of episodes of training would require a more sophisticated form of parallelization. Chavez et al. (2015) has investigated an alternate form of parallelization with DQN involving Downpour stochastic gradient descent. This process involves an omniscient parameter server that stores a global copy of the DQN model. It also uses "worker nodes" to generate local DQN models, while sending and receiving the most up-to-date parameter weights from the parameter server. However, this approach would require further study, because it has been previously shown to experience bottlenecks in communication between the parameter server and worker nodes. As previously mentioned, hyperparameter tuning has

illustrated limitations to DQN. There have been some advances in optimizing neural network hyperparameters, such as genetic algorithms or expected improvement in Gaussian Processes (Fridrich, 2017; Bergstra et al., 2011). However, these approaches have not been implemented in a DQN environment.

DQN generated overall higher total reward in simulation of testing, compared to the other proposed methods in three out of the four case studies. It may be possible for DQN to improve in performance with increased training or a more sophisticated hyperparameter tuning algorithm. However, one would need to consider the additional time and resources this would require. The first case study illustrated the greatest disparity between the four proposed methods. DQN had a total reward about twice that of logistic regression. A decision-maker would need to consider if this difference was worth the time and effort involved with optimizing DQN. For example in the fourth case study, the DQN agent was trained for 8,000 episodes, or serially for about one week. Optimal decisions, in reality, may need to be made faster than a week's time. For this reason, while DQN illustrates strong outbreak management potential in simulated outbreaks, one would need to consider the trade-off of implemented a simpler model.

CHAPTER 5

CONCLUSION

This dissertation consisted of three primary aims related to disease outbreak management: review current computational methods used to terminate disease outbreaks, explore deep Q-networks (DQN) in the context of disease outbreak management, and investigate if simpler alternatives can achieve similar performance to DQN.

Chapter 2 investigated the first aim. The following computational methods were reviewed: regression models, neural networks, dynamic programming, Monte Carlo methods, genetic algorithms, and network-based approaches. Regression models and neural networks were both identified as decision support tools for disease surveillance. These methods have been used in previous studies to determine if a disease outbreak would occur in the future, given pre-specified covariates. This chapter discussed how regression models were better suited if a primary goal of management was to learn about the covariate-outcome relationship. It was found that in many cases, neural networks had similar or slightly improved predictive performance compared to the standard regression modeling approach. This chapter also discussed network-based approaches as a decision support tool alternative that did not assume homogenous mixing. Network-based approaches have been previously used to investigate targeted interventions, i.e. - determine which individuals should be prioritized to receive interventions to prevent an epidemic. However, this chapter concluded that full contact networks are generally unknown, and decision support tools in general do not generate, or even estimate, optimal decisions for disease management.

Chapter 2 also discussed computational approaches that have been previously investigated to optimally manage disease outbreaks. Dynamic programming was the clear approach to construct optimal policies in smaller decision spaces. This method is exponentially faster than manual search methods, but is constrained by the curse of dimensionality. This means that the number of states has a polynomial relationship with the run time needed to determine the optimal policy. Heuristics such as genetic algorithms or Monte Carlo methods have previously been investigated to estimate an optimal policy. It was shown in a previous study that genetic algorithms and classical optimization methods yielded nearly the same number of infected individuals and costs (Yan and Zou, 2007). This brings into question if optimal approaches are truly necessary, when heuristics are easier implemented and have competitive performance. Chapter 2 ends with a segue of the current computational methods being limited

in complex decision spaces, thus requiring another approach to be used in these instances.

Chapter 3 investigated the second aim. A machine learning alternative, previously shown to excel in complex decision spaces, was explored in a disease management context under resource constraints. Deep Q-networks (DQN) were investigated in four different case studies in their ability to terminate a foot-and-mouth disease outbreak with dynamics similar to that of the 2001 epidemic that occurred in the United Kingdom. This chapter illustrated that DQN excelled in scenarios with a clear way to "win the game." In all of the case studies, "winning the game" meant terminating the outbreak, while meeting management objectives. Just as with video games, where DQN was traditionally used, it is not enough just to finish the game. Another goal is to get a high score. The "score" in DQN is influenced by the extent to which the management objectives are met. In all four case studies, the objective was to conservatively cull while terminating the outbreak as quickly as possible. Two of the four case studies had inherent goals that the agent could learn, and thus "win the game" faster. For example, in the first case study the inherent goal was to determine which infected farm could be left for the second day of management. In the third case study the inherent goal was to identify bridging farms. In both of these case studies, the learning agent was successfully able to identify these inherent goals and determine the best way to achieve them, justified by the stability in training. The second and fourth case studies did not have an inherent goal, but the learning agent was still required to terminate the outbreak while meeting the overall objectives. The learning agent was able to manage the outbreak in the second case study, but failed to effectively do so in a scaled landscape. It was suspected that more training and a different combination of hyper parameters may be required to achieve improved performance in a larger management scenario. While DQN illustrated an ability to terminate disease outbreaks in complex decision spaces, it also presented many limitations. Perhaps the most widely investigated limitations are parameter selection and high variance. DQN, in regards to disease management, presents some limitations in generalizability, and implementation in a real, larger outbreak. Chapter 3 concludes with a discussion of simpler alternatives that could be used to the same effect as DQN.

Chapter 4 investigated the third aim. Because Chapter 3 presented many challenges with DQN, simpler alternatives were investigated in their ability to manage disease outbreaks under resource constraints. The following approaches were compared: random culling, cull infected premises, logistic regression, and DQN. The four approaches were implemented on the same four cases studies presented in Chapter

3. As expected, based on the results from Chapter 3, DQN was the clear optimal choice for the first and third case studies. In the second case study, DQN generated an overall higher total reward compared to the other three methods. However, the degree of disparity between the four methods was not as distinct as with the first and third case studies. Also as expected, DQN performed poorly compared to the cull infected premises approach and logistic regression approach in the fourth case study, which investigated scaling. These results suggest that DQN is a superior method when there are inherent goals to complete an episode, and when the complex decision space has a relatively small number of farms. This chapter concluded that the decision-maker is ultimately responsible for determining which objectives were the highest priority. While DQN may achieve optimal performance in some scenarios, tuning and training may require a lot of time and resources. If these components are important to the decision-maker, it may be prudent to explore other methods such as cull infected premises or logistic regression for more timely management.

This dissertation generated many areas of future research for disease outbreak management under resource constraints. Chapter 3 illustrated a scaling limitation of using DQN for disease outbreak management. Previous research investigated using a distributed learning approach for DQN. The Downpour method was used, which allowed for the training to be parallelized between two worker nodes. This has not yet been investigated in a disease management context, and may require more nodes for a larger management problem. GPU computation and sparse matrix operations could also assist in large scale matrix computations that would be present in stochastic gradient descent. The logistic regression method used in Chapter 4 to rank non-infected farms illustrated an opportunity to incorporate sequential updating. In Chapter 4, once new data was collected, risk estimates were recalculated using the initial logistic regression coefficient estimates. Under sequential updating, Bayesian approaches could be implemented to update the regression weights (or DQN weights) as new data is collected in real time. One aspect of analysis not covered in this dissertation is parameter estimate uncertainty. Adaptive management could be combined with logistic regression, or DQN, to account for different forms of uncertainty.

Ajelli, M., Goncalves, B., Balcan, D., Colizza, V., Hu, H., Ramasco, J. J., Merler, S. and Vespignani, A. (2010), Comparing large-scale computational approaches to epidemic modeling: Agent-based versus structured metapopulation models, *BMC Infectious Diseases* .
URL: *https://doi.org/10.1186/1471-2334-10-190*

Akil, L. and Ahmad, H. A. (2016), Salmonella infections modelling in mississippi using neural network and geographical information system (gis), *BMJ Open* **6**(3).
URL: *http://bmjopen.bmj.com/content/bmjopen/6/3/e009255.full.pdf*

Althouse, B. M., Ng, Y. Y. and Cummings, D. A. T. (2011), Prediction of dengue incidence using search query surveillance, *PLOS Biology* .
URL: *https://doi.org/10.1371/journal.pntd.0001258*

Anschel, O., Baram, N. and Shimkin, N. (2017), Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning, *arxiv* .
URL: *https://arxiv.org/pdf/1611.01929*

Baluja, S. and Caruana, R. (1995), Removing the genetics from the standard genetic algorithm, Technical Report CMU-CS-95-141, Pittsburgh, PA.

Barrett, S. and Hoel, M. (2007), Optimal disease eradication, *Environment and Development Economics* .

Baxter, P., Wilcox, C., McCarthy, M. and Possingham, H. (2007), Optimal management of an annual weed: A stochastic dynamic programming approach, *in* 'International Congress on Modelling and Simulation "Land, Water and Environmental Management: Integrated Systems for Sustainability"', Modelling and Simulation Society of Australia and New Zealand.

Bellman, R. (1954), The theory of dynamic programming, *The Rand Corporation* .
URL: *https://www.rand.org/content/dam/rand/pubs/papers/2008/P550.pdf*

Bent, O., Rashid, T. and Whiteson, S. (n.d.), Improving exploration in deep reinforcement learning, *AIMS* .
URL: *http://aims.robots.ox.ac.uk/wp-content/uploads/2015/07/Oliver_Bent_Report_CDT_MP1.pdf*

Bergstra, J., Bardenet, R., Bengio, Y. and Kegl, B. (2011), Algorithms for hyper-parameter optimization, *NIPS* .
URL: *http://papers.nips.cc/paper/4443-algorithms-for-hyper-parameter-optimization.pdf*

Best, N., Richardson, S. and Thomson, A. (2005), A comparison of bayesian spatial models for disease mapping, *Statistical methods in medical research* .
URL: *https://www.ncbi.nlm.nih.gov/pubmed/15690999*

Blackburn, J. K., McNyset, K. M., Curtis, A. and Hugh-Jones, M. E. (2007), Modeling the geographic distribution of bacillus anthracis, the causative agent of anthrax disease, for the contiguous united states using predictive ecological niche modeling, *American Journal of Tropical Medicine and Hygiene* .
URL: *https://www.ncbi.nlm.nih.gov/pubmed/18165531*

Blayneh, K., Cao, Y. and Kwon, H.-D. (2009), Optimal control of vector-borne diseases: Treatment and prevention, *American Institute of Mathematical Sciences* .
URL: *http://aimsciences.org/article/doi/10.3934/dcdsb.2009.11.587*

Boone, G. (1997), Efficient reinforcement learning: model-based acrobat control, *Institute of Electrical and Electronics Engineers* .
URL: *http://ieeexplore.ieee.org/abstract/document/620043/*

Boëlle, P. Y., Bernillon, P. and Desenclos, J. C. (2009), A preliminary estimation of the reproduction ratio for new influenza a(h1n1) from the outbreak in mexico, march-april 2009, *Eurosurveillance* .
URL: *https://www.ncbi.nlm.nih.gov/pubmed/19442402*

Brauer, F., Driessche, P. v. d. and Wu, J. (2008), *Mathematical epidemiology*, Springer.

Brooker, S. and Michael, E. (2000), The potential of geographical information systems and remote sensing in the epidemiology and control of human helminth infections, *Advances in parasitology* .
URL: *https://www.ncbi.nlm.nih.gov/pubmed/10997209*

Caetano, M. A. L. and Yoneyama, T. (2001), Optimal and sub optimal control in dengue epidemics, *Optimal Control Applications and Methods* .
URL: *https://onlinelibrary.wiley.com/doi/pdf/10.1002/oca.683*

Castilho, C. (2006), 'Optimal control of an epidemic through educational campaigns'.
URL: *ftp://ftp.gwdg.de/pub/EMIS/journals/EJDE/Volumes/2006/125/castilho.pdf*

Cauchemez, S., Bhattarai, A., Marchbanks, T. L., Fagan, R. P., Ostroff, S., Ferguson, N. M. and Swerdlow, D. (2011), Role of social networks in shaping disease transmission during a community outbreak of 2009 h1n1 pandemic influenza, *Proceedings of the National Academy of Sciences* **108**.
URL: *https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3041067*

Chansang, C. and Kittayapong, P. (2007), Application of mosquito sampling count and geospatial methods to improve dengue vector surveillance, *The American journal of tropical medicine and hygiene* .
URL: *https://www.ncbi.nlm.nih.gov/pubmed/17984350*

Chavez, K., Ong, H. Y. and Hong, A. (2015), Distributed deep q-learning, *arxiv* .
URL: *https://arxiv.org/abs/1508.04186*

Cheek, R. B. (2010), 'Playing god with hiv'.
URL: *https://www.tandfonline.com/doi/abs/10.1080/10246029.2001.9627949*

Chin, H. H. and Jafari, A. A. (1998), Genetic algorithm methods for solving the best stationary policy of finite markov decision processes - ieee conference publication, Institute of Electrical and Electronic Engineers.
URL: *http://ieeexplore.ieee.org/document/660132/*

Clancy, D. and Green, N. (2007), Optimal intervention for an epidemic model under parameter uncertainty, *Mathematical Biosciences* .
URL: *https://www.ncbi.nlm.nih.gov/pubmed/17070866*

Clements, A. C. A., Lwambo, N. J. S., Blair, L., Nyandindi, U., Kaatano, G., Kinung'hi, S., Webster, J. P., Fenwick, A. and Brooker, S. (2006), Bayesian spatial analysis and disease mapping: tools to enhance planning and implementation of a schistosomiasis control programme in tanzania, *Tropical Medicine & International Health* **11**(4), 490–503.
URL: *https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1365-3156.2006.01594.x*

Cooper, D. L., Smith, G., Baker, M., Chinemana, F., Verlander, N., Gerard, E., Hollyoak, V. and Griffiths, R. (2003), National symptom surveillance using calls to a telephone health advice service, *Centers for Disease Control and Prevention* .
URL: *https://www.cdc.gov/mmwr/preview/mmwrhtml/su5301a33.htm*

Costagliola, D., Flahault, A., Galinec, D., Garnerin, P., Menares, J. and Valleron, A. J. (1991), A routine tool for detection and assessment of epidemics of influenza-like syndromes in france, *American Journal of Public Health* .
URL: *https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1404927/*

Craft, M. E. (2015), Infectious disease transmission and contact networks in wildlife and livestock, *Philosophical Transactions of the Royal Society B: Biological Sciences* .
URL: *https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4410373/*

Curran, P. J., Atkinson, P. M., Foody, G. M. and Milton, E. J. (2000), Linking remote sensing, land cover and disease, *Advances in Parasitology* .
URL: *https://doi.org/10.1016/S0065-308X(00)47006-5*

Dasaklis, T. K., Pappis, C. P. and Rachaniotis, N. P. (2012), Epidemics control and logistics operations: A review, *International Journal of Production Economics* .
URL: *https://www.sciencedirect.com/science/article/pii/S0925527312002150*

Dayan, P. and Niv, Y. (2008), Reinforcement learning: the good, the bad, and the ugly, *Current opinion in neurobiology* .
URL: *https://www.princeton.edu/ yael/Publications/DayanNiv2008.pdf*

de Souza, F. and T., Y. (1993), Control of dengue using genetic algorithms for optimizing the investment in educational campaigns, *in* 'Proceedings of ECC 93'.

DEFRA (2004), 'Foot and mouth disease contingency plan'.
URL: *http://www.defra.gov.uk/footandmouth/contingency/index.html*

DEFRA (2011), 'Maps of livestock populations in 2000 and 2010 across england'. [Online; accessed 2-April-2018].
URL: *https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/183109/defra-stats-foodfarm-landuselivestock-june-detailedresults-livestockmaps111125.pdf*

Deng, Y., Shen, S. and Vorobeychik, Y. (2013), Optimization methods for decision making in disease prevention and epidemic control, *Mathematical Biosciences* .
URL: *https://www.ncbi.nlm.nih.gov/pubmed/24121040*

Deshpande, A. (2016), 'A beginner's guide to understanding convolutional neural networks'.

URL: *https://adeshpande3.github.io/A-Beginner27s-Guide-To-Understanding-Convolutional-Neural-Networks/*

Driessche, P. v. d. and Watmough, J. (2001), Reproduction numbers and subthreshold endemic equilibria for compartmental models of disease transmission, *PLOS Biology* .
URL: *https://doi.org/10.1016/S0025-5564(02)00108-6*

Duczmal, L., Cancado, A. L., Takahashi, R. H. and Bessegato, L. F. (2007), A genetic algorithm for irregularly shaped spatial scan statistics, *Computational Statistics and Data Analysis* .
URL: *https://www.sciencedirect.com/science/article/pii/S0167947307000199*

Eales, R., Thomas, P., Bostock, D., Lingard, S., Derbyshire, I., Burmiston, A. and Kitson, H. (2002), 'The 2001 outbreak of foot and mouth disease'.
URL: *https://www.nao.org.uk/wp-content/uploads/2002/06/0102939.pdf*

Eames, K. T. (2008), Modelling disease spread through random and regular contacts in clustered populations, *Theoretical Population Biology* .
URL: *https://www.ncbi.nlm.nih.gov/pubmed/18006032*

Eames, K. T. D. and Keeling, M. J. (2003), Contact tracing and disease control, *Proceedings of the Royal Society of London B: Biological Sciences* **270**(1533), 2565–2571.
URL: *http://rspb.royalsocietypublishing.org/content/270/1533/2565*

Eckhardt, R. (1987), Stan ulam, john von neumann, monte carlo method, *Los Alamos Science* 131–141.
URL: *http://permalink.lanl.gov/object/tr?what=info:lanl-repo/lareport/LA-UR-88-9068*

Edmunds, W. J., O'callaghan, C. J. and Nokes, D. J. (1997), Who mixes with whom? a method to determine the contact patterns of adults that may lead to the spread of airborne infections, *Proceedings of the Royal Society of London B: Biological Sciences* **264**(1384), 949–957.
URL: *http://rspb.royalsocietypublishing.org/content/264/1384/949*

Eisen, L. and Lozano-Fuentes, S. (2009), Use of mapping and spatial and space-time modeling approaches in operational control of aedes aegypti and dengue, **3**, 1–7.
URL: *https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2668799/pdf/pntd.0000411.pdf*

Eisen, R. J. and Eisen, L. (2008), Spatial modeling of human risk of exposure to vector-borne pathogens based on epidemiological versus arthropod vector data, *Journal of medical entomology* .
URL: *https://www.ncbi.nlm.nih.gov/pubmed/18402133*

Elnaiem, D. E., Schorscher, J., Bendall, A., Obsomer, V., Osman, M. E., Mekkawi, A. M., Connor, S. J., Ashford, R. W. and Thomson, M. C. (2003), Risk mapping of visceral leishmaniasis: the role of local variation in rainfall and altitude on the presence and incidence of kala-azar in eastern sudan, *The American journal of tropical medicine and hygiene* .
URL: *https://www.ncbi.nlm.nih.gov/pubmed/12556141*

Eubank, S., Guclu, H., Kumar, V. S. A., Marathe, M. V., Srinivasan, A., Toroczkai, Z. and Wang, N. (2004), Modelling disease outbreaks in realistic urban social networks, *Nature News* .
URL: *https://www.nature.com/articles/nature02541*

Eysenbach, G. (1970), Infodemiology: tracking flu-related searches on the web for syndromic surveillance., *Health communication* .
URL: *http://europepmc.org/articles/PMC1839505*

Farrington, C. P. (1996), A statistical algorithm for the early detection of outbreaks of infectious disease, *Journal of the Royal Statistical Society. Series A (Statistics in Society)* .
URL: *http://www.jstor.org/stable/2983331*

Ferguson, N. M., Cummings, D. A., Cauchemez, S., Fraser, C., Riley, S., Meeyai, A., Iamsirithaworn, S. and Burke, D. S. (2005), Strategies for containing an emerging influenza pandemic in southeast asia., *Nature* .
URL: *https://www.ncbi.nlm.nih.gov/pubmed/16079797*

Ferguson, N. M., Donnelly, C. A. and Anderson, R. M. (2001), The foot and mouth epidemic in great britain: Pattern of spread and impact of interventions, *Science* **292**, 1155–1160.
URL: *http://www.sciencemag.org/content/292/5519/1155*

Fraser, C., Riley, S., Anderson, R. M. and Ferguson, N. M. (2004), Factors that make an infectious disease outbreak controllable, *Proceedings of the National Academy of Sciences* **101**(16), 6146–6151.
URL: *http://www.pnas.org/content/101/16/6146*

Fridrich, M. (2017), Hyperparameter optimization of artificial neural network in customer churn prediction using genetic algorithm, *Trends of Economy and Management* .
URL: *https://trends.fbm.vutbr.cz/index.php/trends/article/download/385/323*

Ge, L., Mourits, M. C., Kristensen, A. R. and Huirne, R. B. (2010), A modelling approach to support dynamic decision-making in the control of fmd epidemics, *Preventive Veterinary Medicine* **95**, 167–174.
URL: *http://www.sciencedirect.com/science/article/pii/S0167587710001042*

Gibbens, J. C., Wilesmith, J. W., Sharpe, C. E., Mansley, L. M., Michalopoulou, E., Ryan, J. B. M. and Hudson, M. (2001), Descriptive epidemiology of the 2001 foot and mouth disease epidemic in great britain: the first five months, *Veterinary Record* .
URL: *http://veterinaryrecord.bmj.com/content/149/24/729*

Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S. and Lew, M. S. (2015), Deep learning for visual understanding: A review, *Neurocomputing* .
URL: *https://www.sciencedirect.com/science/article/pii/S0925231215017634*

Hastie, T., Tibshirani, R. and Friedman, J. (2005), *The elements of statistical learning*, Springer.

Haupt, R. L. (1995), 'An introduction to genetic algorithms for electromagnetics - ieee journals and magazine'.
URL: *http://ieeexplore.ieee.org/abstract/document/382334/*

Haupt, R. L. and Haupt, S. E. (1998), *Practical Genetic Algorithms*, John Wiley & Sons, Inc., New York, NY, USA.

Hogan, W. R., Tsui, F.-C., Ivanov, O., Gesteland, P. H., Grannis, S., Overhage, M., Robinson, M. and Wagner, M. M. (2003), Detection of pediatric respiratory and diarrheal outbreaks from sales of over-the-counter electrolyte products, *Journal of American Medical Informatics Association* .
URL: *https://www.ncbi.nlm.nih.gov/pmc/articles/PMC264433/*

Hubel, D. H. and Wiesel, T. N. (1968), Receptive fields and functional architecture of monkey striate cortex., *The Journal of physiology.* .
URL: *https://www.ncbi.nlm.nih.gov/pubmed/4966457*

Husin, N. A., Mustapha, N. and Sulaiman, N. (2012), A hybrid model using genetic algorithm and neural network for predicting dengue outbreak, Data Mining and Optimization.
URL: *http://ieeexplore.ieee.org/document/6329793/*

Husin, N. A., Salim, N. and Ahmad, A. R. (2008), Modeling of dengue outbreak prediction in malaysia: A comparison of neural network and nonlinear regression model, *Wiley-IEEE Press* .
URL: *https://ieeexplore.ieee.org/abstract/document/4632022/*

J. Newman, M. E. (2006), The structure and function of complex networks, *SIAM* **45**.
URL: *http://www-personal.umich.edu/ mejn/courses/2004/cscs535/review.pdf*

Jaakkola, T., Jordan, M. I. and Singh, S. P. (1994), Convergence of stochastic iterative dynamic programming algorithms, *NIPS* .
URL: *https://papers.nips.cc/paper/764-convergence-of-stochastic-iterative-dynamic-programming-algorithms.pdf*

Jackson, M. L., Baer, A., Painter, I. and Duchin, J. (2007), A simulation study comparing aberration detection algorithms for syndromic surveillance, *BMC medical informatics and decision making* .
URL: *https://www.ncbi.nlm.nih.gov/pubmed/17331250*

Keeling, M. J., J. Woolhouse, M. E., May, R. M., Davies, G. and Grenfell, B. T. (2003), Modelling vaccination strategies against foot and mouth disease, *Nature* **421**, 136–142.
URL: *https://www.nature.com/articles/nature01343.pdf*

Keeling, M. J., J. Woolhouse, M. E., Shaw, D., Matthews, L., Chase-Topping, M., Haydon, D. T., Cornell, S. J., Kappey, J., Wilesmith, J., Grenfell, B. T. and et al. (2001), Dynamics of the 2001 uk foot and mouth epidemic: Stochastic dispersal in a heterogeneous landscape, *Science* **294**, 813–817.
URL: *http://science.sciencemag.org/content/294/5543/813.long*

Keeling, M. J. and Rohani, P. (2007), *Modeling Infectious Diseases in Humans and Animals*, 1 edn, Princeton University Press, Princeton, N.J.

Khouzani, M. R., Sarkar, S. and Altman, E. (2011), Optimal control of epidemic

evolution, *in* 'Institute of Electrical and Electronics Engineers', IEEE Xplore.
URL: *http://ieeexplore.ieee.org/abstract/document/5934963/*

Kiang, R., Adimi, F., Soika, V., Nigro, J., Singhasivanon, P., Sirichaisinthop, J., Leemingsawat, S., Apiwathnasorn, C. and Looareesuwan, S. (2006), Meteorological, environmental remote sensing and neural network analysis of the epidemiology of malaria transmission in thailand, *Geospatial Health* .
URL: *https://doi.org/10.4081/gh.2006.282*

Kiss, I. Z., Green, D. M. and Kao, R. R. (2006), Infectious disease control using contact tracing in random and scale-free networks, *Journal of The Royal Society Interface* **3**(6), 55–62.
URL: *http://rsif.royalsocietypublishing.org/content/3/6/55*

Kretzschmar, M., van Duynhoven, Y. T. H. P. and Severijnen, A. J. (1996), Modeling prevention strategies for gonorrhea and chlamydia using stochastic network simulations, *American Journal of Epidemiology* **144**(3), 306–317.
URL: *http://dx.doi.org/10.1093/oxfordjournals.aje.a008926*

Kroese, D. P., Brereton, T., Traimre, T. and Botev, Z. I. (2014), Why the monte carlo method is so important today, *WIREs Computational Statistics* **6**, 386–392.
URL: *http://wires.wiley.com/WileyCDA/WiresArticle/wisId-WICS1314.html*

Kumar, S. (2009), *Basics of remote sensing and GIS*, Laxmi Publications (P) Ltd.

Le, Q. V. (2013), Building high-level features using large scale unsupervised learning, *Google* .
URL: *https://research.google.com/pubs/pub38115.html*

Lefevre, C. (1981), Optimal control of a birth and death epidemic process., *Operations Research* .
URL: *https://www.ncbi.nlm.nih.gov/pubmed/10253251*

Leung, P. and Tran, L. T. (1999), Predicting shrimp disease occurrence: artificial neural networks vs. logistic regression, *Aquaculture* **187**, 35–49.
URL: *http://www.sciencedirect.com/science/article/pii/S0044848600003008*

Levine, R. S., Peterson, A. T. and Benedict, M. Q. (2004), Geographic and ecologic distributions of the anopheles gambiae complex predicted using a genetic algorithm, *American Journal of Tropical Medicine and Hygiene* .
URL: *https://www.ncbi.nlm.nih.gov/pubmed/14993618*

Levine, S., Finn, C., Darrell, T. and Abbeel, P. (2016), End-to-end training of deep visuomotor policies, *arxiv* .
URL: *https://arxiv.org/abs/1504.00702*

Liao, Y., Xu, B., Wang, J. and Liu, X. (2017), A new method for assessing the risk of infectious disease outbreak, *Scientific Reports* **7**, 1–12.
URL: *https://www.nature.com/articles/srep40084.pdf*

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D. and Wierstra, D. (2016), Continuous control with deep reinforcement learning, *International Conference on Learning Representations* .
URL: *https://arxiv.org/pdf/1509.02971.pdf*

Lin, L.-J. (1993), Reinforcement learning for robots using neural networks, PhD thesis.

Lipsitch, M., Cohen, T., Cooper, B., Robins, J. M., Ma, S., James, L., Gopalakrishna, G., Chew, S. K., Tan, C. C., Samore, M. H. and et al. (2003), 'Transmission dynamics and control of severe acute respiratory syndrome'.
URL: *http://science.sciencemag.org/content/300/5627/1966*

Liu, M. and Zhao, L. (2009), Optimization of the emergency materials distribution network with time windows in anti-bioterrorism system, *International Journal of Innovative Computing, Information and Control* .
URL: *http://www.ijicic.org/isii08-064-1.pdf*

Lloyd-Smith, J. O., Schreiber, S. J., Kopp, P. E. and Getz, W. M. (2005), Superspreading and the effect of individual variation on disease emergence, *Nature News* .
URL: *doi:10.1038/nature04153*

Ludkovski, M. and Niemi, J. (2013), Optimal dynamic policies for influenza management, *Statistical Communications in Infectious Diseases* **2**.
URL: *https://doi.org/10.2202/1948-4690.1020*

Luma, A., Anwar, A. H. and S., R. R. (2014), Effects of climate change on salmonella infections, *Foodborne Pathogens and Disease* .
URL: *https://doi.org/10.1089/fpd.2014.1802*

Magruder, S. F. (2003), Evaluation of over-the-counter pharmaceutical sales as a possible early warning indicator of human disease, *Johns Hopkins APL Technical*

Digest .
URL: *http://www.jhuapl.edu/techdigest/TD/td2404/Magruder.pdf*

Marescot, L., Chapron, G., Chades, I., Fackler, P. L., Duchamp, C., Marboutin, E. and Giminez, O. (2013), 'Complex decisions made simple: a primer on stochastic dynamic programming'.
URL: *https://doi.org/10.1111/2041-210X.12082*

MathWorks (2018), 'Simple neural network'. [Online; accessed 9-November-2017].
URL: *https://www.mathworks.com/matlabcentral/fileexchange/64247-simple-neural-network*

McLaws, M. and Ribble, C. (2007), Description of recent foot and mouth disease outbreaks in nonendemic areas: exploring the relationship between early detection and epidemic size., *The Canadian Veterinary Journal* .
URL: *http://europepmc.org/articles/PMC1978293*

Merl, D., Johnson, L. R., Gramacy, R. B. and Mangel, M. (2009), A statistical framework for the adaptive management of epidemiological interventions, *PLOS ONE* .
URL: *https://doi.org/10.1371/journal.pone.0005807*

Meyers, L. A., Pourbohloul, B., J. Newman, M. E., Skowronski, D. M. and Brunham, R. C. (2004), Network theory and sars: predicting outbreak diversity, *Journal of Theoretical Biology* **232**, 71–81.
URL: *https://www.sciencedirect.com/science/article/pii/S0022519304003510*

Miller Neilan, R. L., Schaefer, E., Gaff, H., Fister, K. R. and Lenhart, S. (2010), Modeling optimal intervention strategies for cholera, *SpringerLink* **72**, 2004–2018.
URL: *https://link.springer.com/article/10.1007/s11538-010-9521-8*

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G. and et al. (2015), Human-level control through deep reinforcement learning, *Nature* **518**, 529–533.
URL: *https://www.nature.com/articles/nature14236.pdf*

Moloney, J. M., Skelly, C., Weinstein, P., Maguire, M. and Ritchie, S. (1998), Domestic aedes aegypti breeding site surveillance: limitations of remote sensing as a predictive surveillance tool, *The American journal of tropical medicine and hygiene*

.
URL: *https://www.ncbi.nlm.nih.gov/pubmed/9715943*

Mondini, A. and Chiaravalloti-Neto, F. (2008), Spatial correlation of incidence of dengue with socioeconomic, demographic and environmental variables in a brazilian city, *The Science of the total environment* .
URL: *https://www.ncbi.nlm.nih.gov/pubmed/18262225*

Morris, R. S., Wilesmith, J. W., Stern, M. W., Sanson, R. L. and Stevenson, M. A. (2001), Predictive spatial modelling of alternative control strategies for the foot and mouth disease epidemic in great britain, 2001, *Veterinary Record* .
URL: *https://www.ncbi.nlm.nih.gov/pubmed/11517981*

Muharam, F. M., Ruslan, S. A., Zulkafli, S. L., Mazlan, N., Adam, N. A. and Husin, N. A. (2017), Remote sensing derivation of land surface temperature for insect pest monitoring, *Asian Journal of Plant Sciences* .
URL: *https://scialert.net/abstract/?doi=ajps.2017.160.171*

Okosun, K. O., Ouifki, R. and Marcus, N. (2011), Optimal control analysis of a malaria disease transmission model that includes treatment and vaccination with waning immunity, *Biosystems* **106**, 136–145.
URL: *https://doi.org/10.1016/j.biosystems.2011.07.006*

Patel, R., Longini, I. M. and Halloran, E. (2005), Finding optimal vaccination strategies for pandemic influenza using genetic algorithms, *Journal of Theoretical Biology* **234**, 201–212.
URL: *https://www.ncbi.nlm.nih.gov/pubmed/15757679*

Perez, L. and Dragicevic, S. (2009), 'An agent-based approach for modeling dynamics of contagious disease spread'.
URL: *https://doi.org/10.1186/1476-072X-8-50*

Peterman, R. M. and Anderson, J. L. (2012), Decision analysis: Taking uncertainties into account in risk-based decision making, *Human and ecological risk assessment* .
URL: *https://www.tandfonline.com/doi/abs/10.1080/10807039991289383*

Peterson, A. T., Bauer, J. T. and Mills, J. N. (2004), Ecologic and geographic distribution of filovirus disease, *Emerging Infectious Disease* .
URL: *https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3322747/*

Peterson, A. T., Martínez-Campos, C., Nakazawa, Y. and Martínez-Meyer, E. (2005), Time-specific ecological niche modeling predicts spatial dynamics of vector insects and human dengue cases, *Transactions of The Royal Society of Tropical Medicine and Hygiene* **99**(9), 647–655.
URL: *http://dx.doi.org/10.1016/j.trstmh.2005.02.004*

Polgreen, P. M., Chen, Y., Pennock, D. M. and Nelson, F. D. (2008), Using internet searches for influenza surveillance, *OUP Academic* .
URL: *https://academic.oup.com/cid/article/47/11/1443/282247*

Preciado, V. M., Zargham, M., Enyioha, C., Jadbabaie, A. and Pappas, G. (2013), Optimal vaccine allocation to control epidemic outbreaks in arbitrary networks.
URL: *https://arxiv.org/abs/1303.3984*

Probert, W. J., Shea, K., Fonnesbeck, C. J., Runge, M. C., Carpenter, T. E., Durr, S., Garner, M. G., Harvey, N., Stevenson, M. A., Webb, C. T. and et al. (2015), Decision making for foot and mouth disease control: Objectives matter, *Epidemics* .
URL: *https://doi.org/10.1016/j.epidem.2015.11.002*

Rahmandad, H. and Sterman, J. (2008), Heterogeneity and network structure in the dynamics of diffusion: Comparing agent-based and differential equation models, *Management Science* **54**(5), 998–1014.

Raso, G., Matthys, B., N'Goran, E. K., Tanner, M., Vounatsou, P. and Utzinger, J. (2005), Spatial risk prediction and mapping of schistosoma mansoni infections among schoolchildren living in western cote d'ivoire, *Cambridge Core* .
URL: *https://www.cambridge.org/core/services/aop-cambridge-core/content/view/S0031182005007432*

Rogers, D. J. and Randolph, S. E. (2003), Studying the global distribution of infectious diseases using gis and rs, *Nature reviews. Microbiology.* .
URL: *https://www.ncbi.nlm.nih.gov/pubmed/15035027*

Salathe, M. and Jones, J. H. (2010), Dynamics and control of diseases in networks with community structure, *PLOS Computational Biology* **6**.
URL: *https://doi.org/10.1371/journal.pcbi.1000736*

Schulman, J., Mortiz, P., Levine, S., Jordan, M. I. and Abbeel, P. (2016), High-dimensional continuous control using generalized advantage estimation, *arxiv* .
URL: *https://arxiv.org/abs/1506.02438*

Serfling, R. E. (1963), Methods for current statistical analysis of excess pneumonia-influenza deaths, *Public Health Reports* .
URL: *https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1915276/*

Shea, K. (1998), Management of populations in conservation, harvesting and control, *Trends in Ecology and Evolution* .
URL: *https://doi.org/10.1016/S0169-5347(98)01381-0*

Shiflet, A. B. and Shiflet, G. W. (2014), An introduction to agent-based modeling for undergraduates, *Procedia Computer Science* .
URL: *https://www.sciencedirect.com/science/article/pii/S1877050914003032*

Shoemaker, C. A. (1981), Applications of dynamic programming and other optimization methods in pest management, *Wiley-IEEE Press* .
URL: *https://ieeexplore.ieee.org/document/1102782/*

Simonsen, L., Clarke, M. J., Stroup, D. F., Williamson, G. D., Arden, N. H. and Cox, N. J. (1997), A method for timely assessment of influenza-associated mortality in the united states, *Epidemiology* .
URL: *https://www.ncbi.nlm.nih.gov/pubmed/9209852*

Situngkir, H. (2004), Epidemiology through cellular automata: Case of study avian influenza in indonesia.
URL: *https://arxiv.org/abs/nlin/0403035*

Sprague, N. (2015), Parameter selection for the deep q-learning algorithm, *Semantic Scholar* .
URL: *https://pdfs.semanticscholar.org/c101/208ae84b83235a12367797d4ecdb7d87fd26.pdf*

Stroup, D. F., Williamson, G. D., Herndon, J. L. and Karon, J. M. (1989), Detection of aberrations in the occurrence of notifiable diseases surveillance data, *Statistics in Medicine* .
URL: *https://onlinelibrary.wiley.com/doi/pdf/10.1002/sim.4780080312*

Sutton, R. S. and Barto, A. G. (1998), *Introduction to Reinforcement Learning*, 1st edn, MIT Press, Cambridge, MA, USA.

Suyama, J., Sztajnkrycer, M., Lindsell, C., Otten, E. J., Daniels, J. M. and Kressel, A. B. (2003), Surveillance of infectious disease occurrences in the community: an analysis of symptom presentation in the emergency department, *Advances in pediatrics* .
URL: *https://www.ncbi.nlm.nih.gov/pubmed/12837650*

Taylor, B. and Sit, V. (1998), *Statistical methods for adaptive management studies*, British Columbia Ministry of Forests, Research Branch.

Tesauro, G. (1995), Temporal difference learning and td-gammon, *Communications of the ACM* **38**, 58–68.
URL: *https://dl.acm.org/citation.cfm?id=203343*

Thomson, M. C., Connor, S. J., D'Alessandro, U., Rowlingson, B., Diggle, P., Cresswell, M. and Greenwood, B. (1999), Predicting malaria infection in gambian children from satellite data and bed net use surveys: the importance of spatial correlation in the interpretation of results, *The American journal of tropical medicine and hygiene* .
URL: *https://www.ncbi.nlm.nih.gov/pubmed/10432046*

Tildesley, M. J., Bessell, P. R., Keeling, M. J. and Woolhouse, M. E. J. (2009), The role of pre-emptive culling in the control of foot and mouth disease, *Proceedings of the Royal Society of London B: Biological Sciences* .
URL: *http://rspb.royalsocietypublishing.org/content/276/1671/3239*

Tildesley, M. J., House, T. A., Bruhn, M. C., Curry, R. J., O'Neil, M., Allpress, J. L., Smith, G. and Keeling, M. J. (2010), 'Impact of spatial clustering on disease transmission and optimal control'.
URL: *doi: 10.1073/pnas.0909047107*

Tildesley, M. J., Savill, N. J., Shaw, D. J., Deardon, R., Brooks, S. P., J. Woolhouse, M. E., Grenfell, B. T. and Keeling, M. J. (2006), Optimal reactive vaccination strategies for a foot and mouth outbreak in the uk, *Nature* 83–86.
URL: *doi:10.1038/nature04324*

Toubiana, L. and Flahault, A. (1998), A space-time criterion for early detection of epidemics of influenza-like-illness., *Advances in pediatrics.* .
URL: *https://www.ncbi.nlm.nih.gov/pubmed/9744678*

Tsui, F.-C., Wagner, M. M., Dato, V. and Ho Chang, C.-C. (2002), Value of icd-9-coded chief complaints for detection of epidemics, *OUP Academic* .
URL: *https://www.ncbi.nlm.nih.gov/pubmed/11825278*

Tu, J. V. (1999), Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes, *Journal of Clinical Epidemiology* **49**, 1225–1231.
URL: *https://www.sciencedirect.com/science/article/pii/S0895435696000029*

Unkel, S., Farrington, C. P., Garthwaite, P. H., Robertson, C. and Andrews, N. (2011), Statistical methods for the prospective detection of infectious disease outbreaks: a review, *Journal of the Royal Statistical Society: Series A (Statistics in Society)* **175**, 49–82.
URL: *http://doi.wiley.com/10.1111/j.1467-985X.2011.00714.x*

Valetta, N. (2007), Incentive systems under ex post moral hazard to control outbreaks of classical swine fever in the netherlands, *in* 'American Agricultural Economics Association Annual Meeting'.

Wagner, M. M., Tsui, F. C., Espino, J. U., Dato, V. M., Sittig, D. F., Caruana, R. A., McGinnis, L. F., Deerfield, D. W., Druzdzel, M. J., Fridsma, D. B. and et al. (2001), The emerging science of very early detection of disease outbreaks., *Journal of public health management and practice* .
URL: *https://www.ncbi.nlm.nih.gov/pubmed/11710168*

Wang, H., Wang, X. and Zeng, A. Z. (2009), Optimal material distribution decisions based on epidemic diffusion rule and stochastic latent period for emergency rescue, *Inderscience Publishers* .
URL: *https://doi.org/10.1504/IJMOR.2009.022876*

Wang, H. and Yu, Y. (2016), Exploring multi-action relationship in reinforcement learning.
URL: *https://www.researchgate.net/publication/303487241*

Wang, J. and Deng, Z. (2016), Modeling and prediction of oyster norovirus outbreaks along gulf of mexico coast, *Environmental Health Perspectives* .
URL: *https://www.ncbi.nlm.nih.gov/pubmed/26528621*

Wang, J., Jia, P., Cuadros, D. F., Xu, M., Wang, X., Guo, W., Portnov, B. A., Bao, Y., Chang, Y., Song, G. and et al. (2017), A remote sensing data based

artificial neural network approach for predicting climate-sensitive infectious disease outbreaks: A case study of human brucellosis, *MDPI* **9**, 1–17.
URL: *http://mdpi.com/2072-4292/9/10/1018*

Wang, X., Zhang, M., Zhu, J. and Geng, S. (2008), Spectral prediction of phytophthora infestans infection on tomatoes using artificial neural network (ann), *International Journal of Remote Sensing* **29**(6).
URL: *https://doi.org/10.1080/01431160701281007*

White, L. F. and Pagano, M. (2008), A likelihood-based method for real-time estimation of the serial interval and reproductive number of an epidemic, *Statistics in Medicine* .
URL: *https://www.ncbi.nlm.nih.gov/pubmed/18058829*

White, L. F., Wallinga, J., Finelli, L., Reed, C., Riley, S., Lipsitch, M. and Pagano, M. (2009), 'Estimation of the reproductive number and the serial interval in early phase of the 2009 influenza a/h1n1 pandemic in the usa.'.
URL: *https://www.ncbi.nlm.nih.gov/pubmed/19903209*

Wilson, D. P. and Blower, S. M. (2005), Designing equitable antiretroviral allocation strategies in resource-constrained countries, *PLOS Biology* .
URL: *https://doi.org/10.1371/journal.pmed.0020050*

Woolhouse, M. (2003), Foot and mouth disease in the uk: What should we do next time?, *Journal of Applied Microbiology* .
URL: *http://onlinelibrary.wiley.com/doi/10.1046/j.1365-2672.94.s1.15.x/full*

Yaesoubi, R. and Cohen, T. (2011*a*), Dynamic health policies for controlling spread of emerging infections: Influenza as an example, *PLOS ONE* **6**.
URL: *http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0024043*

Yaesoubi, R. and Cohen, T. (2011*b*), Generalized markov models of infectious disease spread: A novel framework for developing dynamic health policies, *European Journal of Operational Research* .
URL: *https://www.sciencedirect.com/science/article/pii/S0377221711006187*

Yan, X. and Zou, Y. (2007), Optimal and sub optimal quarantine and isolation control in sars epidemics, *Mathematical and Computer Modelling* **247**, 235–245.
URL: *http://www.sciencedirect.com/science/article/pii/S0895717707001628*

Yan, X. and Zou, Y. (2008), Optimal internet worm treatment strategy based on the two factor model, *ETRI Journal* .
URL: *https://onlinelibrary.wiley.com/doi/abs/10.4218/etrij.08.0107.0050*

Yu, L., Zhou, L., Tan, L., Jiang, H., Wang, Y., Wei, S. and Nie, S. (2014), Application of a new hybrid model with seasonal auto-regressive integrated moving average (arima) and nonlinear auto-regressive neural network (narnn) in forecasting incidence cases of hfmd in shenzhen, china, *PLOS Biology* .
URL: *https://doi.org/10.1371/journal.pone.0098241*

Zhang, Z., Deng, Z. and Rusch, K. A. (2012), Development of predictive models for determining enterococci levels at gulf coast beaches, *Water research* .
URL: *https://www.ncbi.nlm.nih.gov/pubmed/22130001*

Zhang, Z., Deng, Z. and Rusch, K. A. (2014), Modeling fecal coliform bacteria levels at gulf coast beaches, *SpringerLink* .
URL: *https://link.springer.com/article/10.1007/s12403-014-0145-3*

Zhu, J. M., Wang, L. and Liu, J. B. (2016), Eradication of ebola based on dynamic programming, *Computational and Mathematical Methods in Medicine* .
URL: *https://www.ncbi.nlm.nih.gov/pubmed/27313655*