Fall 1-14-2020

Optimal Boundary Detection Using Autonomous Mobile Sensors

Phillip Justin Kearns
Portland State University

Optimal Boundary Detection Using Autonomous Mobile Sensors

by

Phillip Justin Kearns

A thesis submitted in partial fulfillment of the
requirements for the degree of

Master of Science
in
Electrical and Computer Engineering

Portland State University
2019

Abstract

A fundamental challenge to modern science and engineering is the ability to rapidly and accurately determine the spatial extent of environmental phenomena. In monitoring the spread of hazardous pollution, for example, all points with pollutant concentration above or below a fixed threshold can be considered as two classes in a binary classification problem. In this instance, the goal is to accurately estimate the decision boundary as quickly as possible. To generate models and predictions, scientists must choose their sampling locations from a vast array of possibilities. This thesis develops a policy for determining the optimal sample locations for a fixed number of samples.

The motivating scenario for this work is that of determining the spatial extent of particulate matter from a wildfire with an autonomous aerial vehicle. Algorithms designed to rapidly determine a decision boundary fall within the category of active learning or adaptive sampling. These methods typically try to maximize information gain per sample, but will be accompanied by potentially dramatic drawbacks in terms of sampling costs like distance or time. Meanwhile, state-of-the-art methods that balance the above costs by sampling a certain fraction into the remaining interval at each step do not guarantee to find the optimal search procedure.

For the first situation we consider, a one-dimensional step function, we propose a finite-horizon sampling procedure that optimally balances the distance traveled during the search with the final estimation entropy. We show the resulting cost for a uniform probability distribution with noiseless measurements can be optimized in closed form, and derive the expected number of samples necessary to fall below a given final estimation error. We show our method is suitable for both distance- and time-penalized search procedures, and demonstrate that the resulting policy generalizes and improves upon existing approaches to this problem. Empirical results demonstrate that our sampling strategy outperforms existing approaches by up to 35% and agrees with our analytical predictions in terms of the resulting distance traveled and average interval size.

To extend our proposed method to two-dimensional searches, we show how a series of sequential one-dimensional transect searches can be combined to estimate a spatial boundary, assuming we have some known statistics about the function we are modeling. We demonstrate how the results from each successive search can be used to update the estimated boundary and select where to start sampling for the next search. We also illustrate the tradeoff between the number of transect searches performed and the number of samples per search when constrained by a fixed number of total samples. We find that the optimal allocation lies somewhere between the maximum number of steps and maximum number of samples.

Finally, we introduce and implement four popular methods for solving reinforcement learning problems: dynamic programming, Q-learning, deep Q networks, and rollout. Starting with a uniform distribution on the change point of a step function, we show how formalizing our search as a Markov decision process yields an optimal policy through model-based dynamic programming, which we benchmark our three model-free algorithms against. We then approach a scenario with a more complicated model, where the change points are drawn from a nonuniform distribution. In both scenarios, rollout is the fastest model-free method while a deep Q network performs best. All algorithms improve upon existing approaches, ranging from 4% to 23% improvement.

Considering future work, we propose a number of algorithmic extensions and improvements to our models, as well as a few considerations for potential further investigation. The contributions of this thesis are an optimal search policy for a distance- or time-penalized one-dimensional search, an extension of this policy to a two-dimensional boundary, and the use of reinforcement learning methods to derive optimal policies without prior knowledge of the change point's distribution.

To Madison.

Acknowledgements

I would like to start by thanking my advisor, John Lipor, for his guidance and patience over the past year. I feel truly fortunate to have had the opportunity to learn, forget, and re-learn in an environment of academic freedom and support, where showing up to every meeting with more questions than answers was met with enthusiasm and encouragement. I am grateful to be able to count John as both an impactful teacher, and, more importantly, a close friend.

I want to extend my gratitude to every professor that I have had the chance to learn from; each has taught me valuable lessons and shaped the student and person that I am. A special thanks to Karen and James Frenzel for igniting my curiosity about engineering, and to Eric Wan for cultivating my interest in signal processing. Thanks to Bruno Jedynak for showing me the power of statistics, and Ted Willke the power of deep learning. And thanks to Christof Teuscher for saving my committee.

I would also like to thank a handful of my peers for being exceptional collaborators, mentors, supporters and friends. To David Handy for teaching me patience and persistence when answers do not come easily. To Aaron Brown for his willingness to help me re-think and re-work anything and everything, and for instilling in me a desire to "actually" learn. To Annabel Li-Pershing for her generosity with both knowledge and time, helping shape the way I approach problems in school and life. And to Philippe Proctor for the motivation to stay inquisitive and the chance to learn through meaningful conversation.

Lastly, I would like to thank my family. To my parents for their unfailing support, and the incredible opportunities and experiences their hard work and love has afforded me. To my sisters, Nora and Paula, for their relentless confidence in me and for the best friendship I could ask for. And finally, to Madison, for keeping me in school, helping me to see the good, and believing in the person I can be.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The ability to provide accurate measurements across a large physical area as quickly and efficiently as possible is paramount to modern environmental science and engineering. Wildfires create an immediate safety threat and a persistent air quality hazard across the western United States [1, 2] and pollutants from heavy traffic pose health risks in urban environments [3]. Toxic algae blooms in fresh water lakes and rivers not only impair access to drinking water, but cause low oxygen levels, effectively suffocating aquatic ecosystems [4]. We focus on the first example above as a motivating problem, and our goal is to apply optimal sampling policies to determine the spatial extent of hazardous particulate matter from a wildfire (see Fig. 2.1). In this case, we establish binary classification of pollutant concentration using a fixed threshold on air quality measurements, and we aim to estimate the decision boundary. Further, we consider the case of an autonomous mobile sensor such as a rotary-winged unmanned aerial vehicle, or drone, obtaining these measurements, where there is a significant cost associated with travel.

Algorithms for efficiently determining a decision boundary can be classified as active learning or adaptive sampling [5, 6], and typically try to maximize information gain per sample. However, in the above example, there is a cost associated with both the time to take a measurement *and* the distance traveled throughout the sampling procedure. Additionally, using a search vehicle such as a drone requires a hard upper limit on search time and distance due to the finite battery life and need for recharging. Hence, standard approaches to active learning based in search space reduction [7, 8, 9] or adaptive submodularity [10] are accompanied by shortcomings in terms of total sampling cost.

Newer, bisection-style search methods [11, 12] balance the above costs by sampling a certain fraction into the remaining interval at each step, effectively trading off between number of samples and distance traveled. Though these methods provide improvements on previous approaches in terms of total sampling time, neither guarantees to find the optimal search procedure. *This thesis studies the problem of minimizing a weighted combination of estimation error and distance traveled for a fixed number of samples.*

1.2 Contributions

1.2.1 Finite Horizon Search

The finite-horizon (FH) search method presented in Chapter 2 is a novel active learning algorithm for binary classification in spatial sampling. Motivated by the case of using a drone with a fixed sampling capacity, the FH algorithm minimizes a combination of the final entropy of the estimate and the distance traveled after obtaining N measurements. We show that for a one-dimensional step function with a uniform distribution on the change point, fixing N allows the resulting cost to be optimized in closed form, eschewing the need for dynamic programming.

Generalizing our search scenario to consider the case where we seek to achieve estimation error below a certain error threshold in the least *time* possible, we provide an algorithm to derive the expected number of samples and subsequent policy for various search time parameters. Here, we show that the quantile search algorithm from [11] can be viewed as an instance of the proposed FH algorithm in the case where $N = 1$ (i.e., greedy sampling). Empirical results demonstrate that FH search outperforms existing approaches and agrees with our analytical predictions in terms of the resulting distance traveled and average interval size.

1.2.2 Two-Dimensional FH Search

To apply our algorithm to two-dimensions, we propose modeling potential spatial boundaries as instances drawn from a Gaussian process (GP). By using a periodic GP model, we are able not only to combine a series of one-dimensional searches to estimate the boundary function, but also to choose the best place to start each sequential search. We demonstrate the tradeoff in performance for a fixed number of samples between the number of transects searched and the number of samples per transect.

1.2.3 Reinforcement Learning for Adaptive Sampling

In Chapters 3 and 4, we provide background on a select number of reinforcement learning algorithms and show how they can be applied to a distance-penalized search procedure. We implement four different approximate solution methods on a discrete Markov decision process formalization of our problem. To demonstrate the effectiveness of these methods, we start with a uniform distribution on the change point's prior, and compare the learned policies against the optimal closed-form solution. We then expand the sampling scenario to a non-uniform distribution for the change point, showing the learning capacity of model-free methods in a stochastic environment.

Chapter 2

Finite-Horizon Search

2.1 Introduction

This chapter includes collaborative work with Professors John Lipor in the department of Electrical and Computer Engineering and Bruno Jedynak in the department of Mathematics and Statistics as part of a submission to the 2019 Asilomar Conference on Signals, Systems, and Computers. The extended abstract, titled "Optimal Adaptive Sampling for Boundary Estimation with Mobile Sensors", was accepted on Aug. 1, 2019, and the full paper will be published in March 2020.

Robotic systems are being increasingly utilized as data-gathering tools by scientists and engineers, bringing new perspective and a greater understanding of the environment. As the need for large-scale environmental monitoring rises due to an elevated frequency of natural and man-made disaster like floods, fires, and chemical spills [13, 14, 15], autonomous sensing vehicles are a promising solution. Robotic sampling is being implemented in locations ranging from ocean bottoms to volcanic ridges, gathering new information on algae, pollution, and climate patterns, all while lowering the associated human risk [16].

Consider our motivating scenario of using a drone to estimate the spatial extent of hazardous particulate matter emanating from a wildfire. Drones are increasingly being used to to gather environmental information; an overview of recent drone usage for forestry research and wildfire monitoring is provided in [17]. While these use cases are encouraging, drone use in forestry or agricultural work thus far has been focused largely on aerial image gathering with little consideration given to non-visual data like air quality, let alone optimal path planning for the drone taking these measurements [18]. The design of optimal algorithms for intelligently sampling the environment is the focus of this chapter.

2.2 Problem Formulation

As stated in the introduction, the full two-dimensional boundary estimation can be reduced to a series of one-dimensional search problems, where we wish to locate the change point of a step function, i.e., a function from the class

$$\mathcal{F} = \{f_\theta : [0,1] \to \mathbb{R} : f_\theta(x) = \mathbb{1}_{[0,\theta)}(x), \theta \in [0,1]\}$$

where $\mathbb{1}_S(x)$ denotes the set indicator function. These one-dimensional estimates may then be combined either in a piecewise-linear fashion [11] or using Gaussian process regression [19] as illustrated in Figs. 2.1 and 2.6.

Assume we obtain observations $\{Y_n\}_{n=1}^N \in \{0,1\}^N$ from the sample locations $\{X_n\}_{n=1}^N$ in the unit interval in a sequential fashion according to $Y_n = f_\theta(X_n)$, where θ is the actual, unknown, change point location. Under this model, each sample obtained reduces the interval in which the change point may lie. Our goal is then to estimate the change point location while minimizing the sampling cost for a fixed number of samples, a function of both the final expected interval size *and* expected distance traveled.

2.3 Related Work

Many previous approaches to finding an unknown change point are based in search space reduction (SSR) [7, 8, 9] and do not permit the inclusion of general or dynamic

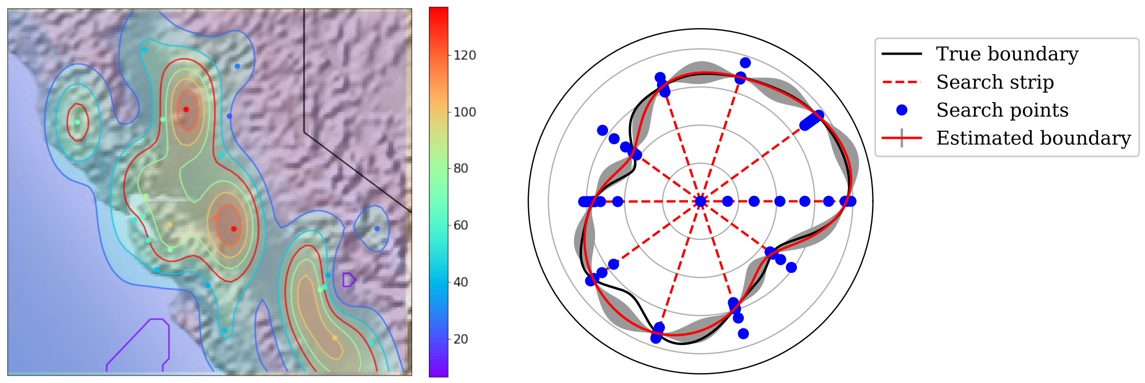

Figure 2.1: Left: map of air quality following the California Camp Fire. Points represent measurement stations and contours are generated with a Gaussian regression. The red contour represents a hazardous level as a potential decision boundary. Right: modeling a decision boundary as a GP and combining transect searches to estimate a spatial boundary.

costs. In the motivating example of a drone performing a search for a spatial boundary, it is critical to consider the time required to travel between measurement locations in addition to the time to take each measurement. Using a battery-powered search vehicle enforces a hard upper limit on search time and distance due to finite battery life and recharging requirements. Because SSR methods do not take into account these extra parameters, they tend to result in bisection-type solutions [20] that will have higher total sampling cost. Methods that seek to maximize hypothesis space reduction at each step can be classified as "greedy" search methods. Greedy methods in active learning [7, 8] lack theoretical guarantees of minimum total sampling cost, and even those that incorporate realistic costs into the algorithm formulation [21] have been shown to perform worse than the bisection-style approach in [11] when applied to distance-penalized searches.

A popular greedy approach to active learning relies on the concept of adaptive submodularity (AS) [22]. AS is a diminishing returns principle that states samples are more informative or valuable early on in the search procedure, and [10] shows that a greedy procedure is optimal up to a constant factor. However, AS is a property of set functions, and does not consider a sequential dependency among sampling locations. While [23] provides a theoretical analysis of greedy active learning with non-uniform costs, the authors only consider the case of query costs being fixed. In contrast, our scenario has non-uniform and dynamic costs, where travel time depends on the distance between points.

The authors of [24] introduce the idea of adaptive data collection for mobile path planning, or *informative path planning*, where previous samples are used to guide the motion of the sensing vehicles for further sampling. This is a prolific and evolving field of research and much of the literature so far focuses on maximizing information gain over a scalar field for an underwater autonomous vehicle. Algorithms presented in [24, 25, 26] accommodate a wide range of sampling scenarios that include varied sampling time, path constraints, and limited battery. However, these methods require a coarse sampling of the entire feature space, which is not feasible in our problem, and [26] requires mixing a network of stationary sensors with a mobile sensor. In contrast, boundary detection methods like those in [27, 28, 29] use mobile sensors to map a spatial threshold as closely as possible. These methods provide efficient and accurate mappings of a binary classification boundary, but unfortunately do not account for constraints like limited on battery life or samples on a single vehicle.

One category of particular interest is that of level set estimation (LSE) [30]. LSE focuses on the targeted estimation of measurement points relative to a threshold value, seeking to assign them into either a super- or sub-threshold level set. With the use of Gaussian process statistics to generate a posterior estimate for the distribution following each measurement, this allows for effective creation of a spatial classification boundary in 2- or 3-dimensional space. The initial algorithm in [30] was expanded in [31] and [32], providing novel approaches for selecting and grouping measurements in the fewest samples possible but neglecting to account for the distance between these points. Subsequently, a method for path-efficient LSE that seeks to reduce the distance traveled by the mobile sensor is proposed in [33], but this method assumes

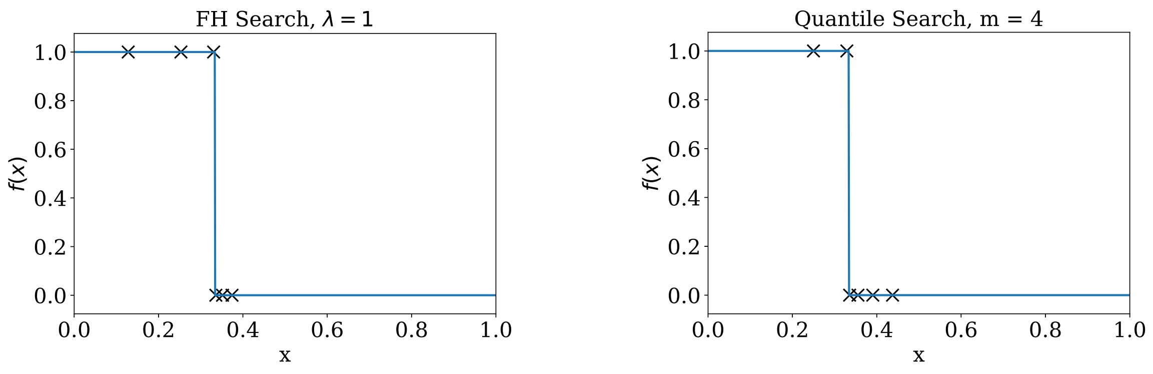

Figure 2.2: Example search for the change point on a step function performed by finite horizon search ($\lambda = 1$, left) and the corresponding quantile search ($m = 4$, right).

the vehicle can continuously acquire measurements with a negligible cost.

Of primary relevance to the work presented in this paper is the work of [11], which introduces the quantile search (QS) algorithm for determining the change point of a one-dimensional step function while balancing the above costs. QS is a generalization of binary bisection [34, 35, 20], where the idea is that by successively sampling a fixed fraction $1/m$, where $m > 2$, into the remaining hypothesis space (defined by an interval), the desired tradeoff between number of samples and distance traveled can be achieved. This work was extended in [12], introducing the uniform-to-binary (UTB) algorithm where the key observation is that QS can be improved by allowing the fraction to grow as the hypothesis space shrinks. Yet, neither algorithm provides guarantees of optimality in terms of the total sampling cost. We believe that this work is the first to provide a theoretical guarantee of optimal search procedure for an environment with non-uniform, dynamic sampling costs.

2.4 Finite-Horizon Search

It is convenient, while not restrictive, to define search strategies in terms of the *fraction of the remaining interval* to move at each step, whether forward or backward, in an analogous fashion to [11, 12]. The resulting class of policies is adaptive to the unknown location of θ and non-restrictive in the sense that any optimal policy will not sample in locations with probability zero (locations outside the remaining interval).

Begin with a uniform prior on the change point θ, and let the N fractions be $\{x_n\}_{n=1}^N$. A straightforward Bayesian update yields the posterior distribution after each sample. Let H_N be the entropy of the posterior distribution after N observations, D_N be the total distance traveled, and $\lambda > 0$ be a tuning parameter that governs the tradeoff between these costs. We define the total sampling cost after N observations as

$$J(x_1, \ldots, x_N) = \mathbb{E}_\theta \left[e^{H_N} + \lambda D_N\right]. \tag{2.1}$$

Note that for a uniform distribution on an interval of length a, $e^{H_N} = e^{\log(a)} = a$; thus, eq. (2.1) is equivalent to minimizing a weighted combination of the (expected) final interval length and expected distance traveled (proof in Appendix A.1).

2.4.1 Closed-Form Solution

We now demonstrate that the cost can be minimized in closed form. Theorem 1 admits a representation of the cost function that allows us to compute the optimal sampling fractions in linear time, resulting in an optimal policy.

Theorem 1. *Let $\lambda \in [0, 2]$ and assume the unknown change point has distribution $\theta \sim$ Unif$([0, 1])$. Further, assume the N measurements are defined via N fractions $x_1, \ldots, x_N$ denoting the proportion of the current hypothesis space to sample. Define the expected interval size at step i, ξ_i, as*

$$\xi_i = x_i^2 + (1 - x_i)^2, \quad i = 1, \ldots, N.$$

The cost function can then be written as

$$J(x_1, \ldots, x_N) = \prod_{i=0}^{N} \xi_i + \lambda \sum_{i=1}^{N} x_i \prod_{j=0}^{i-1} \xi_j, \tag{2.2}$$

where ξ_0 represents the initial interval size of 1.

Proof. A complete proof can be found in Appendix A.2. □

Thm. 1 shows that the entropy and distance components of the sampling cost can both be written in terms of the expected interval size. This allows us to minimize the cost function analytically by computing the optimal policy in Theorem 2.

Theorem 2. *Under the same conditions as Thm. 1, the optimal sampling fractions are of the form*

$$x_k^* = \frac{1}{2} - \lambda \frac{1}{4\rho_k}, \quad k = 1, \ldots, N, \tag{2.3}$$

where $\rho_N = 1$ and

$$\rho_k = \prod_{i=k+1}^{N} \xi_i + \lambda \sum_{i=k+1}^{N} x_i \prod_{j=k+1}^{i-1} \xi_j, \quad k = 1, \ldots, N-1,$$

depends only on the fractions $x_{k+1}, \ldots, x_N$.

Proof. A complete proof can be found in Appendix A.3. □

Thm. 2 shows that the optimal N-step lookahead policy may be computed in linear time, beginning with x_N and proceeding backwards. Because ρ_k is largest at $k = N$ and continues to get smaller with each step backwards, so too will the sample fractions, as can be seen in Fig. 2.3. A higher value for the distance penalty parameter λ results in a less aggressive policy, as the higher cost for potential overshoot encourages smaller steps. When $\lambda \geq 2$, the cost of travelling to obtain a measurement, λx_1, is larger than

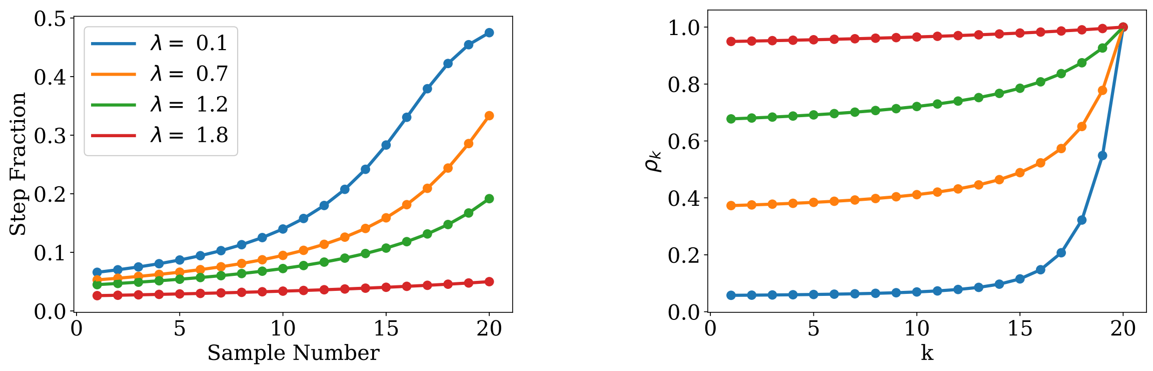

Figure 2.3: Optimal sampling behavior for a fixed-length policy. Left: fractions of the interval to cover at each step of a 20-step policy. Right: corresponding values of ρ_k for calculating the policies.

the expected reduction in entropy, $1 - \xi_1$, and the trivial sample which requires no displacement is preferred.

An alternative to N-step lookahead is to choose the *greedy* policy that minimizes the one-step lookahead for the value function without concern for future consequences. Following this protocol means calculating x_k^* in eq. (2.3) with $k = N = 1$ at each step, which results in sampling a constant fraction into the remaining interval (since the optimal action depends only on k, not interval length; see proof in Appendix A.4). This is exactly the strategy of the QS algorithm, and thus QS may be considered an instance of our proposed method with $N = 1$.

2.4.2 Samples Needed for Fixed Estimation Error

In certain instances, it is desirable to use a threshold on the final interval size rather than a fixed number of samples to terminate the search procedure. When this is the case, we use eq. (2.3) to calculate the optimal action for the final step and then proceed backwards, calculating the optimal action at each preceding step until a policy of subsequent length such that a final interval smaller than the error threshold is expected. Pseudocode for finding the optimal policy (expected number of samples and search fractions for each sample) starting with a given interval of length L and subject to a desired final estimation error and distance penalty λ is given in Algorithm 1. The

Algorithm 1 Calculating Policy for Expected Convergence

```
1: Input: interval length $L$, penalty $\lambda$, stopping error $\varepsilon$
2: Initialize: $\mathbf{x}_N \leftarrow \frac{1}{2} - \frac{\lambda}{4}$, $l \leftarrow 1$
3: while $L \prod_i \xi_i > \varepsilon$ do
4:     $\mathbf{x}_{N-l} \leftarrow \frac{1}{2} - \lambda/(4\rho_{N-l})$
5:     $l \leftarrow l + 1$
6: end while
7: $N \leftarrow l$
```

relationship between λ and the expected samples needed can be seen in Fig. 2.5.

2.4.3 Error-Threshold Search Procedure

In the case where a search terminates only after a certain estimation error has been obtained, we follow a two-phase procedure. Pseudocode is provided in Algorithm 2. Before the search begins, we use the method presented in Algorithm 1 to calculate the N steps such that the expected final interval size is less than ε. Then, in the first search stage, samples are taken according to this N-step policy. If the hypothesis space is smaller than the desired threshold before all N samples have been taken, the search terminates. Otherwise, the algorithm performs a greedy search (optimal 1-step policy, line 7) until the interval is sufficiently small.

2.5 Simulations

In this section, we verify the performance of the proposed finite-horizon sampling policy. We compare theoretical and simulated distance-penalized search costs over a range of λ values and policy lengths, and benchmark our FH search algorithm against QS and UTB in a time-penalized search scenario.

2.5.1 Cost as a Function of Entropy and Distance

To obtain a profile of performance as a function of λ, we perform 100 searches over a range of 100 uniformly-spaced values of θ in the interval $[0, 1]$ for 5 different values of

Algorithm 2 Finite Horizon Search

```
1: Input: policy $\mathbf{x}$, stopping error $\varepsilon$
2: Initialize: $X_0 \leftarrow 0$, $Y_0 \leftarrow 1$, $a \leftarrow 0$, $b \leftarrow 1$, $n \leftarrow 1$
3: while $b - a > \varepsilon$ do
4:     if $n \leq N$ then
5:         $x \leftarrow \mathbf{x}_n$
6:     else
7:         $x \leftarrow \frac{1}{2} - \frac{\lambda}{4}$
8:     end if
9:     if $Y_{n-1} = 1$ then
10:        $X_n \leftarrow X_{n-1} + x(b - a)$
11:    else
12:        $X_n \leftarrow X_{n-1} - x(b - a)$
13:    end if
14:    $Y_n \leftarrow f(X_n)$
15:    $a = \max\{X_i : Y_i = 1, i \leq n\}$
16:    $b = \min\{X_i : Y_i = 0, i \leq n\}$
17:    $\hat{\theta}_n \leftarrow \frac{a+b}{2}$
18:    $n \leftarrow n + 1$
19: end while
```

λ between 0.1 and 1.8. Fig. 2.4 shows the resulting average entropy, distance traveled, and final cost for each corresponding N-step policy. The plots demonstrate that our proposed method enacts a tradeoff between average final entropy and distance traveled via the tuning parameter λ. Further, comparing our empirical results with the expected entropy and distance calculated in Section 2.4.1, we see the values align almost exactly. It is worth noting that as the number of samples increases, the total costs tends to decrease. While the sampling cost function in eq. (2.1) trades off final entropy against distance traveled (and thus prefers a policy in which a greater number of less aggressive samples yields less error and less potential overshoot), we also consider a cost function that considers the total sampling *time*.

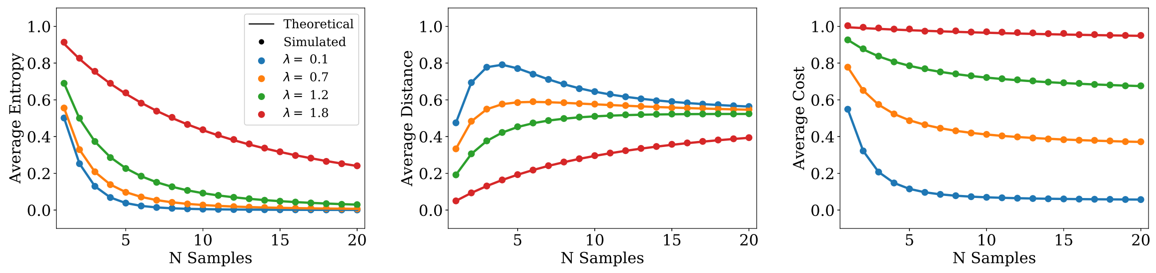

Figure 2.4: Performance of proposed FH algorithm for fixed N samples. Each data point represents an optimal N-step policy. Left-to-right: average entropy of hypothesis space, average distance traveled, and average total cost after last sample.

2.5.2 Cost as a Function of Sampling Time

If we seek to minimize the total time that a vehicle takes to complete a search, we need to consider a cost function of the form

$$J_T(x_1, \ldots, x_N) = T_s N + T_t D, \tag{2.4}$$

where T_s and T_t represent the time per sample and time per unit distance traveled, and N and D represent the number of samples and total distance. In order to minimize this cost in expectation, first calculate the number of samples, N_λ, and total distance, D_λ, expected for the optimal policy for each value of λ to reach a final interval size smaller than desired error ε using Algorithm 1. Then, selecting the value of λ that minimizes the total search time,

$$\lambda^* = \arg\min_\lambda T_s N_\lambda + T_t D_\lambda, \tag{2.5}$$

yields the optimal policy.

The left-side plot of Fig. 2.5 depicts the results from first step in the above procedure. We see that the expected number of steps to convergence increases with λ, while the expected distance decreases. Intuitively, this makes sense. A small λ corresponds to a lower distance penalty and thus entails a more aggressive search

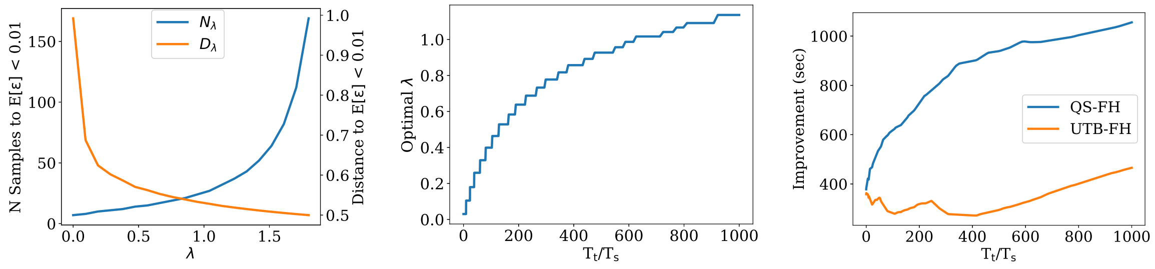

Figure 2.5: Components of the time-penalized FH search procedure. Left: N_λ and D_λ to reach an interval of 0.01 for each value of λ. Middle: values of λ^* for each ratio of T_t/T_s. Right: average improvement of FH search over QS and UTB algorithms.

policy, which takes larger steps to maximize entropy reduction and will converge in fewer samples. A large λ means a greater distance penalty and a more conservative policy with smaller steps, which requires more samples to converge. The middle plot of Fig. 2.5 shows how eq. (2.5) trades off between these values of N_λ and D_λ for various ratios of T_t/T_s. As T_t increases we prefer a higher value of λ^* that takes more samples but is less likely to overshoot the change point.

We compare the performance of the above method with the existing QS and UTB algorithms. We consider the same 100 uniformly-spaced instances of θ for 1,000 different ratios of T_t/T_s in the range of 1×10^{-4} to 1×10^{3} with $T_s = 100$ as the base sampling cost. The right-side plot in Fig. 2.5 shows the resulting improvement in sampling time obtained via the proposed finite-horizon policy. When the T_t/T_s ratio is small, we see a savings of about 380 seconds over both UTB and QS for a search that takes roughly 1,100 seconds, generating an improvement of approximately 34%. At the highest ratios of T_t/T_s, the relative improvement decreases, with savings of 470 seconds over a 55,900 second UTB search (0.8% improvement), and 1,060 seconds over a 56,500 second QS search (1.9% improvement).

2.6 Two-Dimensional Boundary Estimation

The proposed FH policy is for a single, one-dimensional search, but the goal in our motivating example is to estimate a two-dimensional spatial boundary. In [11],

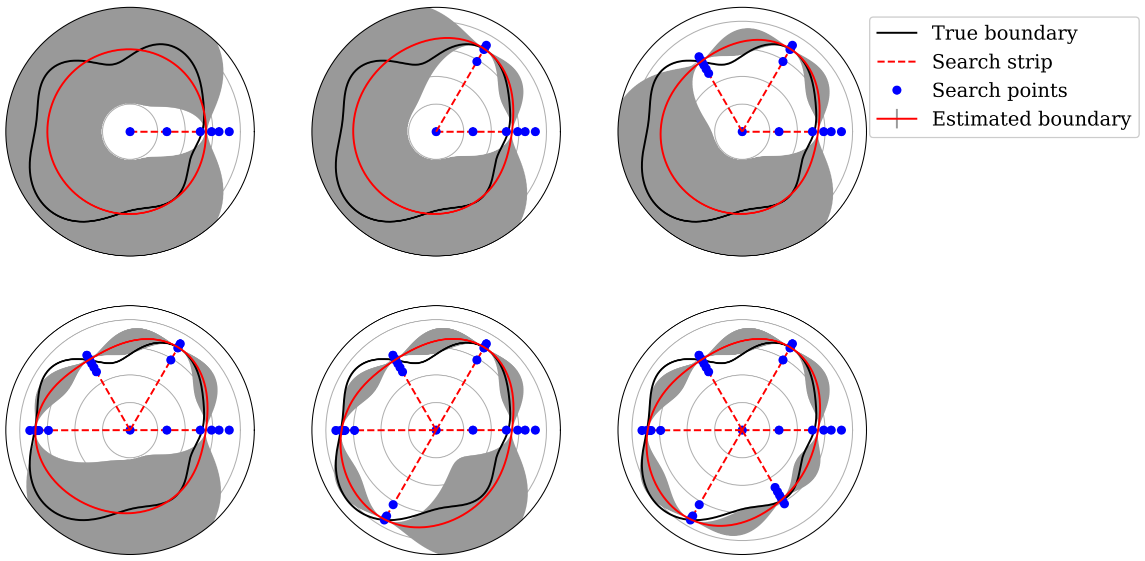

Figure 2.6: Two-dimensional boundary estimate from series of one-dimensional searches. Each search is performed using FH sampling. The black line represents the true boundary and the final estimate (solid red line) is obtained using GP regression, with the confidence bounds shown in gray.

the authors show we can combine multiple one-dimensional strip searches to find a two-dimensional boundary, and point out that a great deal of time would be lost by starting each search from the origin. To prevent this, they present two methods for intelligently initializing successive searches: using the previous estimate as a starting point or assigning a nonuniform prior based on a confidence interval around the previous estimate. In order for these approaches to work, the authors assume their boundary functions are restricted to a class of Hölder smooth functions known as Lipschitz functions. An alternative approach from spatial statistics [36, 37] that allows us to make weaker assumptions is to model our boundary as a Gaussian process.

2.6.1 Introduction to Gaussian Processes

As discussed in [38, 19], a Gaussian process (GP) is a nonparametric generalization of linear regression that allows for the representation of uncertainty about predictions made over the sensed field. To learn the parameters of a certain GP model, we can use data from a pilot study or use previous expert knowledge about the environment.

The learned GP can then be used to predict the boundary with greater accuracy and increase the efficiency of our search procedure. Figure 2.6 illustrates this point: once the change point has been estimated on the first transect, we can use this to generate an estimate and confidence interval for where the boundary is likely to be on the next transect, and start sampling there. This process repeats at each strip until our search procedure terminates.

The crucial component of a GP predictor is the *covariance function*, which encodes our assumptions about the function we want to learn by defining the similarity between data points. The covariance function is a function of two arguments, mapping input pairs $x \in \mathcal{X}$ and $x' \in \mathcal{X}$ into $\mathbb{R}$; the general name for this type of function is a kernel. A kernel is *stationary* if it is a function of $(x - x')$, and falls into the category of radial basis functions if it is an *isotropic* function only of $r = |x - x'|$. Further, a kernel is *symmetric* if $k(x, x') = k(x', x)$; all covariance functions are symmetric by definition.

2.6.2 Estimating Spatial Boundaries with Gaussian Processes

To model our spatial boundary of interest, we use a smooth periodic covariance function. The non-linear mapping of the one-dimensional input x to the two-dimensional $u(x) = (\cos(x), \sin(x))$ provided in [39] creates a periodic random function of x. Using the squared exponential kernel in u-space then gives

$$k(x, x') = \exp\left(- \frac{2\sin^2(\frac{x-x'}{2})}{l^2} \right), \tag{2.6}$$

where l is the characteristic length scale and x is constrained to the interval $[0, 2\pi]$. The effect of varying l on samples drawn from this prior can be seen in Figure 2.7. Clearly, a smaller length scale causes functions to vary more rapidly while a larger length scale produces slower variations and a smoother function.

With prior knowledge (or an initial measurement) that the average distance of the

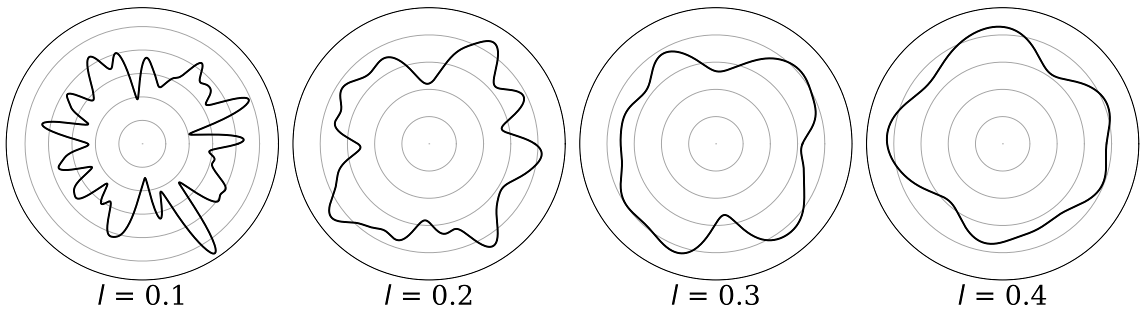

Figure 2.7: Random boundaries drawn from a periodic covariance kernel with varying length scales. The length scale determines the smoothness of the boundary.

change point θ from the origin is μ_θ, and assuming we know the characteristic length scale l for the covariance kernel, it is possible to estimate a new boundary function after each strip. Assume we start sampling at an angle of $\varphi = 0$ and distance μ_θ from the origin. Depending on this result, we then proceed to sample either back towards or away from the origin according to the N-step FH policy. To calculate where to start sampling along the next strip, we estimate the boundary function using the GP equations for a multivariate normal distribution given in [19] eq. (2.18 - 2.24).

If we have performed searches along n strips at angles in the $n \times 1$ vector ϕ from the origin and wish to estimate the boundary at n' potential locations of angle ϕ', then $K_{\phi\phi'}$ denotes the $n \times n'$ matrix of the covariances evaluated at all pairs of observed and future points. The same notation holds for $K_{\phi\phi}$ and $K_{\phi'\phi'}$. Assuming we have some measurement noise in our model with variance ϵ^2, we incorporate this as $K_{\phi\phi} = K_{\phi\phi} + \epsilon^2 I_n$. Each new search gives an estimate of the change point, $\hat{\theta}$, which can be appended to the observation vector Θ. Noting that $K_{\phi'\phi} = K_{\phi\phi'}^T$, the boundary estimate and standard deviation for establishing a confidence interval are then

$$\theta_{pred} = \mu_\theta + K_{\phi'\phi} K_{\phi\phi}^{-1} (\Theta - \mu_\Theta)$$

$$\sigma_{pred} = K_{\phi'\phi'} - K_{\phi'\phi} K_{\phi\phi}^{-1} K_{\phi'\phi}^T.$$

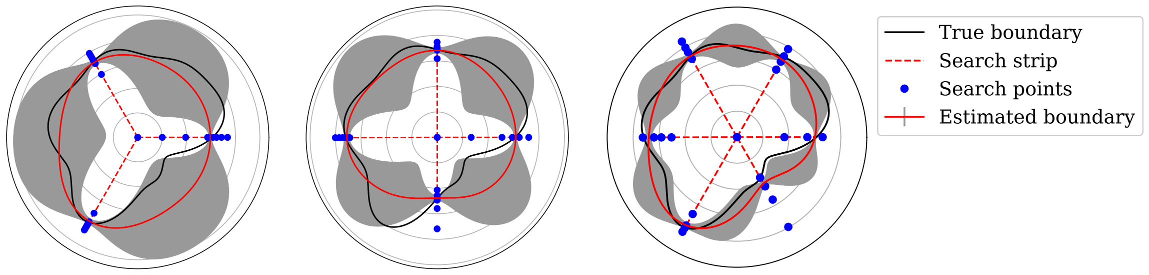

Figure 2.8: Boundary estimates with fixed total samples but varying number of transects. Left-to-right: 3 strips of 8 samples, 4 strips of 6 samples, 6 strips of 4 samples.

The search strategy shown in Fig. 2.6 relies on using a fixed, pre-determined number of evenly-spaced transects. If our search procedure is constrained by a finite number of total samples, then minimizing search cost is a tradeoff between the number of strips and samples per strip, balancing local and global estimation error and distance traveled. Intuitively, fewer strips with more samples per strip means more accurate estimates at the selected locations and less total distance, but higher total estimation error due to less accurate predictions between transects. Searching more transects will give less accurate local estimates but better overall estimation, though it requires greater distance. Figure 2.8 demonstrates this concept, splitting a fixed number of samples $N_{samp} = 24$ between 3, 4, or 6 transects.

2.6.3 Simulations

To quantify this tradeoff, we limited total samples to a maximum of $N_{samp} = 100$ and compared the distance, estimation error, and cost for every number of transects, N_{trans}, from 3 to 20. The number of samples per transect, N_{st}, was determined using the floor of the total samples divided by number of transects, $N_{st} = \lfloor N_{samp}/N_{trans} \rfloor$. For each N_{trans}, we performed searches over the same 100 boundaries randomly generated by the covariance kernel in eq. (2.6) with $\mu_\theta = 3$, $l = 0.4$, and $\epsilon = 0.1$. We compare the average performance of the FH policy for $\lambda = 0.4$ and corresponding QS policy. We define our error as the total area between actual and estimated boundaries.

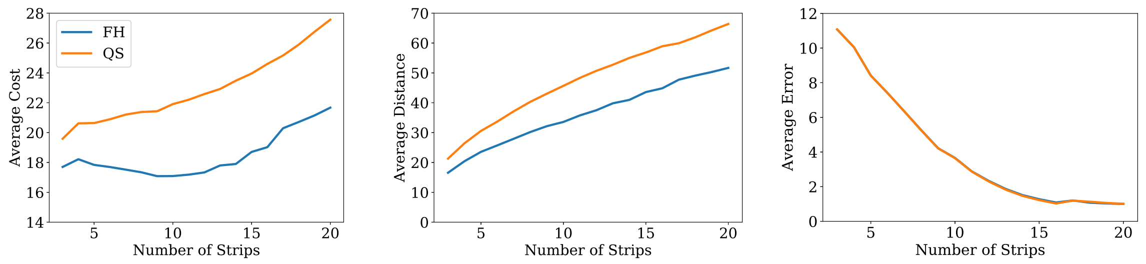

Figure 2.9: Search performance for a fixed number of total samples as a function of number of transects. Left-to-right: total search cost, distance traveled, and final estimation error.

Figure 2.9 shows the results of our simulation. As expected, total distance traveled increases with the number of transects while total estimation error decreases. Initially, the increase in accuracy outweighs the increase in λ-penalized distance, but we see diminishing returns as N_{trans} continues to grow, achieving our minimum cost at 9 transects. However, examining the differences in costs reveals less than a 2% spread between $N_{trans} = 8$ and 12.

Confirming the results from Section 2.5, we see that FH search outperforms QS once again, achieving similar estimation accuracy with substantially less travel. Over the range of our simulations, the improvement in cost is anywhere from 10% to 24%. It is worth noting that the performance numbers reported here depend on all of the parameters of the GP covariance kernel, $\mu_\theta = 3$, $l = 0.4$, and $\epsilon = 0.1$. A different mean boundary distance, shorter length scale or more uncertainty on the model would lead to a different conclusion in terms of the optimal number of transects and reported estimation error, as would a different value of our search parameter, λ.

Chapter 3

Reinforcement Learning For Adaptive Sampling

3.1 Introduction

Reinforcement learning (RL) refers to learning how to act (mapping situations to actions) so as to maximize a reward. Because action selection determines future situations and rewards in addition to the immediate reward, the key components of reinforcement learning are interactive search and delayed reward. Most RL methods fall into one of two categories: *model-free* methods (trial-and-error) that rely on exploration and learning as the primary component, or *model-based* methods (finding optimal control sequences using value functions and dynamic programming) that rely principally on planning. However, the computation of value functions is at the heart of both methods; they look ahead to future events, compute a backed-up value, and use this as an update target for approximating a value function.

In this chapter we introduce model-based and model-free methods for optimizing search procedures in scenarios where the change point is drawn from either a uniform or nonuniform distribution.

3.2 Background

3.2.1 Markov Decision Processes

Markov decision processes (MDPs), developed in [40], are a formalization of sequential decision making where actions influence not just immediate rewards but also subsequent states and future rewards. MDPs thus involve the tradeoff between immediate and delayed reward. In MDPs we estimate the optimal value $q_*(s, a)$ of each action a in each state s, or the optimal value $v_*(s)$ of each state assuming optimal action selections. Classical MDP framing involves a learner and decision maker called the *agent* interacting with an *environment*. At each time step, $t = 0, 1, 2, \dots$ the agent receives a representation of the environment's *state*, $S_t \in \mathcal{S}$, and selects an *action*, $A_t \in \mathcal{A}(s)$. The environment then responds, and at the next time step the agent receives a *reward*, $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$, and is in a new state, S_{t+1}.

In this instance, the random variables for reward R_t and state S_t have a well-defined probability distribution that depends on the previous state and action. The dynamics of a finite stochastic MDP are completely characterized by the probability distribution

$$p(s', r|s, a) \doteq Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}.$$

The goal of the agent is to maximize the return, G_t, or sum of rewards it receives from time t until the end of the procedure,

$$G_t \doteq R_{t+1} + R_{t+2} + \cdots + R_T,$$

where T is the final time step, and corresponds to reaching the terminal state. It is common to introduce a discounting factor γ for future rewards, but we have omitted it since we do not use discounting in the cost function of our scenarios of interest.

3.2.2 Policies and Value Functions

Value functions estimate how good it is for an agent to be in a given state or perform a given action based on the expected return. Because this return depends on the action the agent takes, value functions are defined with respect to certain ways of acting, known as policies. A policy maps states to probabilities of selecting each action: $\pi(a|s)$ represents the probability that an agent following policy π selects $A_t = a$ if $S_t = s$. The *value* of state s under policy π is the expected return starting in s and following policy π until termination, written $v_\pi(s)$. For an MDP, we define the *state-value function*, v_π, as

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s].$$

Similarly, the value of taking action a in state s and thereafter following policy π, known as the *action-value function*, $q_\pi(s, a)$, is defined as

$$q_\pi(s, a) \doteq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a].$$

The value functions satisfy recursive relationships between the value of a state and the values of its successor states, wherein the value of the start state must equal the value of the expected next state plus the expected reward upon transition. These are known as the **Bellman equations** for v_π and q_π,

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + v_\pi(s')\right] \tag{3.1}$$

$$q_\pi(s, a) = \sum_{s',r} p(s', r|s, a) \left[r + \sum_{a'} \pi(a'|s') q_\pi(s', a')\right]. \tag{3.2}$$

The Bellman equation forms the basis for updates or *backup* operations that are the core of reinforcement learning methods: transferring value information back to a state from its successors in order to inform decisions.

Solving a reinforcement learning task means finding a policy that leads to a high expected return [41]. For finite MDPs (where the state, action, and reward sets are finite), there is always one policy, known as the *optimal policy*, π_*, that is equal to or better than all other policies. The optimal state-value function, $v_*(s)$, satisfies the

Bellman optimality equation and expresses the fact that the value of any state under an optimal policy must equal the expected return for the best action from that state,

$$\begin{aligned} v_*(s) &\doteq \max_\pi v_\pi(s) \\ &= \max_a q_{\pi_*}(s,a) \\ &= \max_a \mathbb{E}_{\pi_*}[G_t \mid S_t = s, A_t = a] \\ &= \max_a \mathbb{E}[R_{t+1} + v_*(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \max_a \sum_{s',r} p(s', r|s,a) \left[r + v_*(s') \right], \end{aligned}$$

for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$.

The same holds true for the optimal action-value function, $q_*(s,a)$,

$$\begin{aligned} q_*(s,a) &\doteq \max_\pi q_\pi(s) \\ &= \mathbb{E}[R_{t+1} + v_*(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \mathbb{E}[R_{t+1} + \max_{a'} q_*(S_{t+1}, a') \mid S_t = s, A_t = a] \\ &= \sum_{s',r} p(s', r|s,a) \left[r + \max_{a'} q_*(s', a') \right], \end{aligned}$$

for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$.

Once one has v_* or q_*, it is relatively easy to determine the optimal policy. For any state, the agent can simply select the action that maximizes $q_*(s,a)$. However, explicitly solving the Bellman optimality equation is equivalent to an exhaustive search, looking ahead at all possibilities and computing the probability of occurrence and expected reward. This solution requires an accurate model of the environment and a large amount of computation to complete and store the corresponding q-table to be used for the solution.

3.3 Model-Based Methods

If we have a perfect model of the environment, the optimal search procedure problem becomes one of optimal control. Using a system's state and value functions to define a Bellman equation, we can cast this problem as a stochastic MDP to be solved via dynamic programming.

3.3.1 Dynamic Programming

Dynamic programming (DP) [42] refers to a collection of algorithms that, given a perfect model of the environment as an MDP, can be used to compute optimal policies. DP can be used to compute the value functions defined in Section 3.2.2, and through an iterative process of evaluation and improvement devised in [43], allows us to obtain the optimal policies. A more in-depth description of this process can be found in [41], but using Bellman equations as updating rules for improved function approximation essentially consists of three steps:

Policy evaluation computes the state-value function, v_π, for an arbitrary policy π. If environment dynamics are completely known, then the Bellman equation is a system of $|\mathcal{S}|$ linear equations in $|\mathcal{S}|$ unknowns, and solving it is a straightforward computation. Starting with an initial value function approximation v_0, each successive approximation is calculated using the Bellman equation as an update rule,

$$v_{k+1}(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + v_k(s')\right],$$

in an algorithm known as *iterative policy evaluation*. Each iteration updates the value of every state once to produce the new approximate value function v_{k+1}. This update is called an *expected update* because it is based on an expectation over all possible next states rather than a sample next state.

Policy improvement compares the value of all possible actions $a \in \mathcal{A}(s)$ for a given state and selects the option with the highest expected return under policy π for all future states. For a given state s, this *greedy* policy is given by

$$\begin{aligned}\pi'(s) &\doteq \arg\max_a q_\pi(s,a) \\ &= \arg\max_a \sum_{s',r} p(s',r|s,a)\left[r + v_\pi(s')\right].\end{aligned}$$

Applying the same action selection criteria at each subsequent state yields the improved policy, π'.

Policy iteration is the process of repeatedly alternating between policy evaluation and improvement to yield a series of monotonically improving policies and value functions. Ultimately, because a finite MDP has a set number of policies, this process must converge to an optimal policy and optimal value function after a finite number of iterations.

In order to reduce the iterative computation associated with policy evaluation at each step, the method of **value iteration** can be used instead. Value iteration combines the policy improvement step with a shortened policy evaluation, stopping after just one sweep and requiring the maximum to be taken over all actions,

$$v_{k+1}(s) = \max_a \sum_{s',r} p(s',r|s,a)\left[r + v_k(s')\right].$$

As demonstrated by [41], truncating the policy iteration step maintains the convergence guarantees of policy iteration while significantly reducing computational time.

3.4 Model-Free Methods

If we do not have perfect knowledge of the environment, we cannot rely on straightforward dynamic programming to derive an optimal policy. Instead, we must learn good policies by estimation and iteration. Many of the same ideas from model-based methods still apply, but the agent relies on learning from experience rather than a model. Methods that require only experience to estimate value functions and determine optimal policies are known as Monte Carlo methods. Experience can be in the form of real or simulated sequences of interaction with the environment, and solutions are reached by sampling and averaging returns from each state-action pair.

3.4.1 Q-Learning

Q-learning (QL) is a widely-adopted algorithm that learns a policy by estimating the optimal action value function in the case of an unknown model. For a non-deterministic reward function, the QL algorithm consists of two main steps: sampling an action a in state s and updating the policy values according to the equation

$$q(s,a) \leftarrow \alpha q(s,a) + (1-\alpha)[r + \max_{a'} q(s',a')], \tag{3.3}$$

where α is the learning rate of the algorithm. QL can be viewed as a stochastic formulation of the value iteration in Section 3.3.1. Within each state, an action is selected according to a policy π derived from q. A theoretical guarantee that the QL algorithm will converge to the optimal action-value function $q_*(s,a)$ regardless of the policy, so long as it ensures every (s,a) pair is visited infinitely many times, is provided in [44].

A standard choice of policy is the ϵ-greedy policy determined by the action-value function at time t. ϵ-greedy selects the greedy action, $a = \arg\max_a q_t(s,a)$, with probability (1-ϵ), and selects a random action with probability ϵ for some $\epsilon \in [0, 1]$. It is common practice to use a policy which starts with a high value of ϵ and decays over time, so the QL algorithm will *explore* the environment early on, effectively building a model of the environment, and *exploit* its knowledge of the environment later.

3.4.2 Deep Q-Learning

Q-learning, while popular for many reinforcement learning agents, is limited in applicability to domains in which useful features can be handcrafted, or domains with fully observed, low-dimensional states spaces [45]. Q-learning also struggles to generalize past experiences to new situations, needing to explore every action from a given state before it is capable of making a good policy.

The work of [45] proposes a novel reinforcement learning agent, called a Deep Q Network (DQN), that learns good policies using end-to-end reinforcement learning. To approximate the optimal action-value function the authors use a deep neural network to generate an approximate action-value function, $q(s, a; \theta_i)$, where θ_i represents the parameters (weights) of the network at iteration i. DQN utilizes *experience replay* [46], wherein the agent's experiences $e_t = (s_t, a_t, r_t, s_{t+1})$ are stored in a data set, D, at each time step. During learning, updates are applied based on samples of experience, $(s, a, r, s') \sim U(D)$, drawn uniformly at random from the pool of stored samples to remove correlations in the observation sequence. The *target values* for the updates, $r + \max_{a'} q(s', a')$, are updated only periodically to reduce correlations between the action-values and target. The Q-learning update at iteration i thus uses the loss function,

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

where θ_i are the parameters of the network at iteration i and θ_i^- are the parameters used to compute the target at iteration i. The target network parameters θ_i^- are only updated with the Q-network parameters every C steps.

Recently, the work of [47] and [48] has shown that function approximation with QL and DQN can lead to overoptimism and subsequent under-performance, issues that are addressed in [48, 49] by proposed Double Q-Learning and Double-DQN solutions. However, for the complexity of our motivating scenario, which has a relatively low-dimensional state space and can be discretized effectively into finite state and action spaces, these more advanced methods are beyond the scope of our work.

3.4.3 Rollout

Rollout algorithms [50, 51] are function-approximation and policy improvement algorithms based on Monte Carlo simulation. Starting in a given state, rollout estimates the values of each possible action by averaging the returns of many simulated trajectories that start with that action and then follow a *heuristic policy* from the next state. When action-values estimates are accurate enough, the action with the highest estimated value is executed, then the process repeats from the next state. This is also known as *limited lookahead minimization*, and the aim of rollout is to improve upon the default policy. Naturally, performance of the improved policy depends on performance of the heuristic policy and accuracy of the Monte Carlo estimates. Rollout is similar to the policy improvement method discussed in Section 3.2.2, but relies on simulated trajectories instead of a perfect model of the environment for policy evaluation. Additionally, rollout produces estimates of action values only for the current state under the selected policy, and does not store the estimates after selecting an action.

Consider a stochastic MDP with finite controls and given initial state, s. For each possible action, rollout generates simulations of the next states and uses the chosen base heuristic H to estimate their value. Rollout then selects the optimal action, a_*, given by the maximization

$$a_* = \arg\max_a \tilde{q}(s, a),$$

where $\tilde{q}(s, a)$ is the approximate Q-factor defined by

$$\tilde{q}(s, a) = r + v_H(s'),$$

and $v_H(s')$ represents the value of the subsequent state, s', following heuristic H until termination.

A single pass through this method generates an improved but still sub-optimal policy known as the rollout policy, $\tilde{\pi}$. Proofs for the guarantee of sequential improvement on this algorithm can be found in [52]. As with policy iteration, we can then repeat this estimation and improvement procedure, now using use the rollout policy as a base heuristic.

Chapter 4

Performance of Reinforcement Learning Algorithms

To assess the performance of both model-based and model-free value-approximation methods, we compare learned policies against the closed-form optimal policy for a uniform probability distribution on the change point. We then examine the case of a nonuniform distribution, where we do not have a closed-form solution. In both cases, we compare the policies, computation time, and performance of each algorithm over a large number of simulations.

4.1 Uniform Change Point Distribution

4.1.1 Methodology

To model a search over an assumed uniform distribution as an MDP, we represent the state of an N-step procedure at step n as $s_n = (L_n, n)$, where L_n is length of our hypothesis space. Here, we make use of the concept of a *sufficient statistic* [53] to summarize the essential content of the state information available to the controller and reduce the size of the state space. Because our actions are the fractions into the remaining hypothesis space to travel (not fixed locations) and a uniform distribution guarantees equal probability over the hypothesis space, the end points of the search interval at step n are irrelevant to choosing an action and only the length L matters. In order to generate a finite q-table, we discretize the initial hypothesis space $[0, 1]$ into 501 possible lengths, $\mathcal{L} = [0, 0.002, \ldots, 1]$, and the action space into 101 possible fractions $\mathcal{A} = [0.0, 0.005, \ldots, 0.5]$. Because action a_n may choose a point that causes the subsequent state L_{n+1} to be a value not in $\mathcal{L}$, L_{n+1} is selected as the closest value in $\mathcal{L}$.

In all instances, we are evaluating a 5-step search procedure. All tests will be performed using the same 1000 values of θ drawn from the unit interval (set with random seed of 0), with a distance penalty of $\lambda = 0.4$. All simulated training values of θ are generated starting with a random seed of 1.

Dynamic Programming

To provide an initial benchmark for comparing value-approximation-based solution methods against the closed-form policy, we start with tabular dynamic programming in the form of value iteration. The value of each terminal state is the total reduction in hypothesis space

$$v(s_N) = 1 - L_N.$$

The value of all preceding state-action pairs will be determined by backing up the probabilistic sum of terminal state values plus rewards using the Bellman equation given in eq. (3.2). We use known transition probabilities to build up a q-table, which the agent can then use to select the action that maximizes $q(s, a)$ at each step.

Rollout

Unlike dynamic programming, rollout does not rely on a perfect model of the environment to create a tabular solution. Instead, rollout selects the best action at each step by averaging returns for each action and subsequent heuristic search over simulated values of θ drawn from the remaining hypothesis space. Initially, our heuristic policy is a greedy constant fraction. To implement policy improvement in the form of policy iteration, we keep track of the actions taken at each of the 5 steps for every search, and average the selected actions for each step after every 100 searches to generate an updated heuristic.

At every step, the rollout algorithm sweeps all actions with 10 simulated values of θ. For 1,000 instances of a 5-step search, with 101 possible actions at each step and 10 simulations per action, this yields 5,050,000 total θ simulations.

Q-Learning

In an attempt to establish a fair comparison, we restricted the learning process for Q-learning to 5,050,000 simulations of θ as well. The QL algorithm was trained for 50,500 epochs, drawing 100 random values of θ from a uniform distribution every epoch. For each value of θ, the QL algorithm performed a 5-step search procedure, and at each step selected action a_n according to an ϵ-greedy policy, starting with $\epsilon = 1$ and using a decay rate of $\epsilon_{k+1} = 0.99995 \cdot \epsilon_k$ per epoch until reaching a minimum value of $\epsilon = 0.1$. After each step, the Q-learning update was performed according to eq. (3.3) with a learning rate of $\alpha = 0.001$ and a reward of $r_n = -\lambda \cdot a_n \cdot L_n$. As in the DP q-table, the value of each terminal state is the total reduction in hypothesis space, $1 - L_N$.

DQN

Due to the greater computational time associated with DQN updates, the DQN agent was trained with only 10% of the simulations of QL, for 5,050 epochs with 100 random values of θ each. The network consisted of two hidden layers with 16 neurons each, ReLU activation, and a learning rate of $\alpha = 0.001$ was used for both the Q-network and the target network. The target was updated after every 10 searches, and the replay buffer held 100,000 instances of (s_t, a_t, r_t, s_{t+1}) experience tuples. DQN selected actions using an ϵ-greedy policy, starting with $\epsilon = 1$ and using a decay rate of $\epsilon_{k+1} = 0.9995 \cdot \epsilon_k$ per epoch until reaching a minimum value of $\epsilon = 0.1$.

Computation

All simulations were run on an AMD Ryzen 7 1800x CPU using parallel processing over 8 cores and 16 threads. The search costs provided are calculated by averaging the final interval size and λ-penalized distance travelled over 1,000 change points drawn with a set random seed from either the uniform or nonuniform distributions. The policies shown are the average actions taken at each step across the 1,000 searches, except for the rollout policy, where the most updated heuristic is shown.

4.1.2 Results & Discussion

To establish the effectiveness of dynamic programming, we first compare the policy generated by the DP algorithm to the policy generated by the closed-form solution in Section 2.4.1. The left-side plot of Fig. 4.1 shows that DP learns the exact policy of the closed-form solution, with the only deviations coming from the discretization of actions in the DP method. The performance of DP was also verified to match the performance of the optimal policy, and serves as a benchmark for comparing the other reinforcement learning algorithms. A summary of the computational times and performances for each method can be found in Table 4.1.

In the case of a uniform distribution, where the state depends only on length and sample number, the q-table has only $5 \times 501 \times 101 \approx 250,000$ entries and DP is the fastest approximation method. Rollout is the second-fastest method, more than twice as fast as Q-learning, and roughly four times as fast as DQN. As expected, dynamic programming performs the best. Despite limited training data, DQN performs second best, ahead of both rollout in third and Q-learning in fourth. All approximation methods beat a standard QS search procedure.

Examining the policies shown in the right-side plot of Fig. 4.1, we see that rollout with policy improvement learns a policy similar to DP. While this policy would

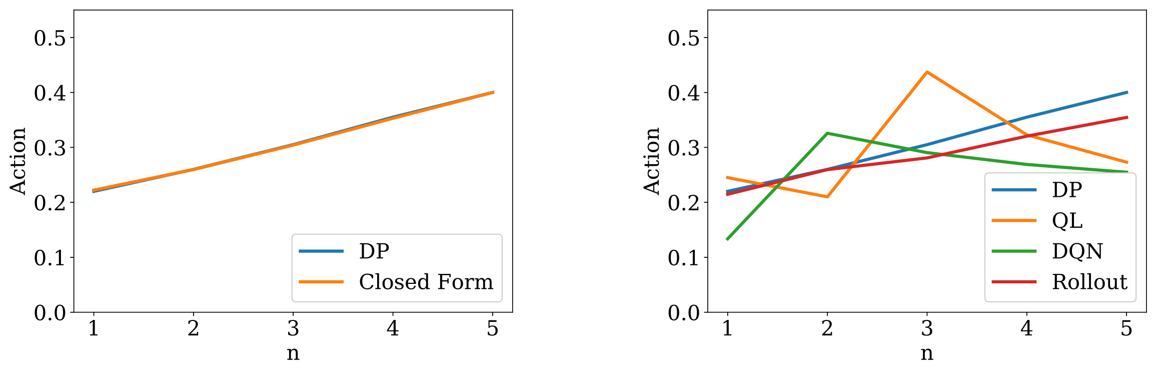

Figure 4.1: Verification of DP performance and comparison of learned policies. Left: DP matches the closed form solution almost perfectly. Right: learned policies for a uniform distribution.

Method	Time	Cost	Improvement
DP	**0:00:03**	**0.3231**	**9.70%**
Rollout	0:06:24	0.3432	4.08%
Q-Learning	0:16:33	0.3486	3.83%
DQN	0:27:33	0.3388	5.29%
QS	-	0.3578	-

Table 4.1: Total training time (h:mm:ss) and average performance for each method over 1,000 change points drawn from a uniform distribution. Fastest computation and best performance are shown in bold, with improvement scores relative to Quantile Search.

seemingly produce better results than given in Table 4.1, this is because early searches rely on a greedy heuristic, and performance suffers accordingly. For example, rollout with no policy improvement performs roughly 3% worse over the last 100 searches than rollout with improvement. Both DQN and Q-learning generate policies that differ significantly from DP, taking a larger step in the middle of the search but smaller steps at the end.

It is worth noting that the performances listed in Table 4.1 and the policies seen in Fig. 4.1 are dependent on both the training and test values of θ, as well as the model parameters; some random seeds and values of ϵ for training yield better test results for certain models. However, the overall rankings (DP $>$ DQN $>$ Rollout $>$ QL) seem constant across multiple seeds, and we have chosen model parameters that generally perform well. The optimization of hyperparameters is a topic for future work.

4.2 Nonuniform Change Point Distribution

4.2.1 Methodology

In the instance of a nonuniform prior for the change point, we model the state of an N-step procedure at step n as $s_n = (Xc_n, Xo_n, n)$, where Xc_n is the *current* location of our sampling vehicle and Xo_n the *opposite* end of the hypothesis space. Similar to the uniform case, we discretize the unit interval into 501 possible search locations $\mathcal{X} = [0.0, 0.002, \ldots, 1.0]$ and the action space into 101 possible fractions $\mathcal{A} = [0.0, 0.005, \ldots, 0.5]$. Because the action a_n may choose a location not in $\mathcal{X}$, the next sampling location is decided by the closest location in $\mathcal{X}$. We use a truncated normal distribution on the interval $[0, 1]$ with $\mu = 0.5$ and $\sigma = 0.1$. All tests will be performed using the same 1,000 values of θ drawn from this distribution (again with a random seed of 0), with a distance penalty of $\lambda = 0.4$. All simulated training values of θ are generated starting with a random seed of 1.

Algorithm Implementation and Computation

The framework for all of the RL algorithms remains the same for the nonuniform case, except that the state depends on the ends of the hypothesis space instead of just the length, which causes the value tables to be of higher dimension. The computational resources remain the same.

4.2.2 Results & Discussion

Though dynamic programming was by far the fastest method for a uniform prior on the change point, this is not the case for a nonuniform distribution. As the q-table has grown to $5 \times 501 \times 501 \times 101 \approx 125M$ entries, DP becomes the slowest approximation method to compute. A summary of results can be found in Table 4.2.

Rollout and DQN barely change in terms of total computation time, becoming the

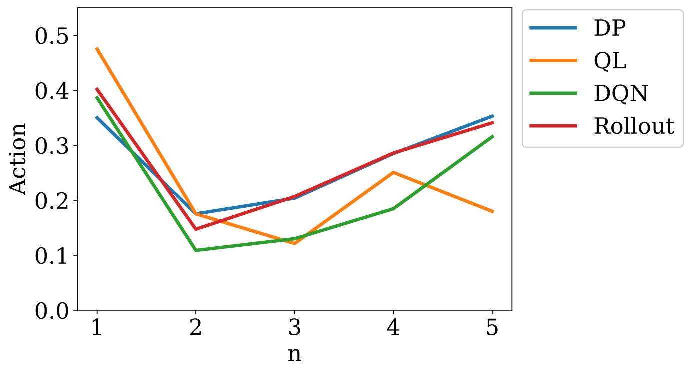

Figure 4.2: Learned policies for each of the RL algorithms when sampling from a nonuniform distribution. DP serves as the reference for the optimal policy.

Method	Time	Cost	Improvement
DP	1:08:48	**0.2719**	**23.15%**
Rollout	**0:06:23**	0.2837	19.82%
Q-Learning	0:23:25	0.3154	10.85%
DQN	0:27:30	0.2742	22.49%
QS	-	0.3538	-

Table 4.2: Total training time and average performance for each method over 1,000 change points drawn from a nonuniform distribution with $\mu = 0.5$ and $\sigma = 0.1$. Fastest computation and best performance are shown in bold, with improvement scores relative to Quantile Search.

first and third fastest methods, while Q-learning increases by about 50% to move into second. Dynamic programming is the slowest by a significant margin, nearly three times slower than Q-learning. The order of performance remains the same as last time: DP, DQN, Rollout, Q-learning. This time, however, rollout performs nearly as well as DQN, and Q-learning displays improved performance relative to the competition as well. Additionally, all approximation methods show much greater improvement over a standard QS search procedure.

Examining the policies shown in the right plot of Fig. 4.1, all methods learn to take aggressive steps initially due to the low probability of overshoot when θ is drawn from a distribution that is tightly concentrated around $X = 0.5$. The policy generated by QL again takes a larger first step than other methods but mimics the DP policy more closely in the nonuniform case than the uniform, as does DQN.

Comparing the results from Tables 4.1 and 4.2, we can see that *all* methods perform better on average over a nonuniform distribution for the change point. Intuitively, this makes sense: tightly grouped change points means a smaller effective window to search, thus causing fewer instances of overshoot and ultimately smaller final intervals. Once again, the exact performance is subject to the seeding of the training data and the choice of parameters, but the general performance trends seem to hold across multiple training sets.

Chapter 5

Conclusion & Future Work

This thesis has presented a number of methods for solving spatial sampling problems with dynamic costs. We have presented a novel active learning algorithm for a one-dimensional search and used Gaussian methods to generalize to two dimensions. We have considered the case of both distance- and time-penalized search procedures, and shown how our method improves upon the current state-of-the-art. We have implemented a select few reinforcement learning methods and compared their relative performance on both simple and more complex search scenarios. We now summarize the key contributions of this thesis, and propose future work for each.

5.1 Finite-Horizon Search

We have presented a novel active learning algorithm for spatial sampling that optimally balances the final estimation error and the distance traveled for a fixed number of samples. We have derived the closed-form solution and to the best of our knowledge, believe that this work is the first to provide a theoretical guarantee of an optimal search procedure for an environment with non-uniform, dynamic sampling costs. We have also shown how our solution generalizes existing approaches to this problem, and empirical results indicate the performance benefits of finite-horizon search over existing methods in the literature.

Though we have considered only the case of noiseless measurements, extending to noisy measurements as done in [11] is an important next step. While the search parameter λ allows for various search costs (sampling time, travel time, recharging, etc.) to be approximated in a compact state space notation, we have only provided a formula for converting travel and sample time into our notation. Further work is needed to fully understand how to represent other search considerations into this single parameter. Finally, while the FH algorithm calculates one policy at the beginning of the search and then follows that policy until termination regardless of the subsequent hypothesis space sizes, it is possible to implement an *adaptive* FH algorithm that re-calculates the optimal policy after each step.

5.2 GP-Based FH Search

Following the ideas in [11, 34] and implementing knowledge of Gaussian processes from [19, 39], we have generalized a one-dimensional search to a two-dimensional problem. Using a periodic GP model enabled us to to combine a series of one-dimensional transect searches to estimate the boundary function, and calculate updated priors to efficiently choose the start for each sequential search. Under the constraint of a limited total number of samples, we demonstrated how to determine the optimal number of transects and samples per transect, and showed the performance improvement of the FH algorithm relative to QS in this setting.

While we base our two-dimensional search procedure around sampling along predefined transects, one potential option for improvement is to select sequential search locations by setting a threshold on the uncertainty of the updated prior. The sampling vehicle can then selectively perform searches further apart when the estimates are of high quality and the boundary appears smooth, and closer together when measurements indicate greater variability. Though our initial results are promising, this thesis relies on being able to use known or assumed statistics about a boundary to combine one-dimensional searches into a two-dimensional procedure. Because we do not focus on boundaries of arbitrary shape, optimal methods for two- or even three-dimensional searches continue to be an interesting problem.

5.3 Reinforcement Learning

Chapter 3 provided a summary of several relevant RL algorithms, applied to our problems of interest in Chapter 4. After demonstrating that model-based dynamic programming learns the optimal policy for a uniform prior on the change point, we compared the chosen methods over both uniform and nonuniform distributions for the change point, analyzing computational expense and total search cost for each.

While rollout and DQN were both highly promising methods, rollout requires the learning agent to simulate trajectories from the environment, something that is not always possible without a well-defined model. Thus, enhanced implementation of DQN or one of its many successors (e.g., [48, 49]) is of primary interest for future investigation. Large-scale neural networks show incredible promise in their ability to make sense of high-dimensional state and action spaces, and do not need a model of the environment to learn, relying instead on gathering real-world experience or equivalent simulation. However, deep neural networks require vast amounts of data to learn effective policies and are computationally power hungry. Thus, the use case of autonomous sensing, where gathering real-world experience necessitates hours of battery charging and monitoring search procedures and which sometimes relies on limited on-board computational resources, merits continued effort to develop an algorithm capable of learning on less data and operating on a low-power system.

Appendix

A.1 Differential Entropy of a Uniform Distribution

Given a continuous random variable X with probability density function $f_X(x)$, the differential entropy $h(X)$ is defined as

$$h(X) = -\int_{-\infty}^{\infty} f_X(x) \ln f_X(x) dx.$$

Consider a random variable distributed uniformly over the interval $[0, a]$, so that its density is $1/a$ over this interval and 0 elsewhere. The differential entropy is then

$$\begin{aligned} h(X) &= -\int_0^a \frac{1}{a} \ln \frac{1}{a} dx \\ &= -\ln \frac{1}{a}, \end{aligned}$$

and the exponential of the entropy is thus

$$e^{h(X)} = e^{-\ln \frac{1}{a}} = a.$$

A.2 Proof of Theorem 1 (Value Function)

Thm. 1 states we can represent our cost function after N measurements as

$$\begin{aligned} J(x_1, \ldots, x_N) &= \mathbb{E}\left[e^{H_N} + \lambda D_N\right] \\ &= \prod_{i=0}^{N} \xi_i + \lambda \sum_{i=1}^{N} x_i \prod_{j=0}^{i-1} \xi_j, \end{aligned} \tag{A.1}$$

where $\xi_0 = 1$ and

$$\xi_i = x_i^2 + (1 - x_i)^2, \quad i = 1, \ldots, N$$

represents the expected interval length of the hypothesis space at step i.

Proof. By Lemma 1 below, we have that

$$\mathbb{E}\left[e^{H_N}\right] = \prod_{i=0}^{N} \xi_i.$$

Let D_N be the distance traveled after N samples. Note that

$$D_N = \sum_{i=1}^{N} x_i e^{H_{i-1}}.$$

Therefore,

$$\begin{aligned}\mathbb{E}\left[D_N\right] &= \mathbb{E}\left[\sum_{i=1}^{N} x_i e^{H_{i-1}}\right] \\ &= \sum_{i=1}^{N} x_i \mathbb{E}\left[e^{H_{i-1}}\right].\end{aligned}$$

Applying Lemma 1 then yields

$$\mathbb{E}\left[D_N\right] = \sum_{i=1}^{N} x_i \prod_{j=0}^{i-1} \xi_j.$$

The proof is completed by applying linearity of expectation. □

Lemma 1. *Let H_N be the entropy of the hypothesis space after N measurements. Assuming the unknown change point has a distribution $\theta \sim$ Unif$([0,1])$ and our actions are defined via N fractions $x_1, \ldots, x_N$ denoting the proportion of the current hypothesis space to sample, we have*

$$\mathbb{E}\left[e^{H_N}\right] = \prod_{i=0}^{N} \xi_i. \tag{A.2}$$

Proof. First note that the exponentiated differential entropy of a uniform distribution is the length of the hypothesis space after N samples. The proof will proceed by induction on N. Consider the base case, $N = 1$, for which it is trivial to show that, since $\xi_0 = 1$,

$$\mathbb{E}\left[e^{H_1}\right] = x_1^2 + (1 - x_1)^2 = \xi_1.$$

Now assume that (A.2) holds for some $N \in \mathbb{N}$. Sampling some fraction x_{N+1} into the remaining hypothesis space e^{H_N} results in two potential entropies with corresponding probabilities

$$e^{H_{N+1}} = \begin{cases} x_{N+1} e^{H_N} & \text{w/ probability} \quad x_{N+1} \\ (1 - x_{N+1}) e^{H_N} & \text{w/ probability} \quad 1 - x_{N+1}. \end{cases}$$

Therefore,

$$\begin{aligned}
\mathbb{E}[e^{H_{N+1}}] &= x_{N+1}^2 \mathbb{E}[e^{H_N}] + (1 - x_{N+1})^2 \mathbb{E}[e^{H_N}] \\
&= \left(x_{N+1}^2 + (1 - x_{N+1})^2\right) \mathbb{E}[e^{H_N}] \\
&= \prod_{i=0}^{N+1} \left(x_i^2 + (1 - x_i)^2\right) \\
&= \prod_{i=0}^{N+1} \xi_i.
\end{aligned}$$

□

A.3 Proof of Theorem 2 (Optimal Policy)

Thm. 2 states the optimal sampling fractions are of the form

$$x_k^* = \frac{1}{2} - \lambda \frac{1}{4\rho_k}, \quad k = 1, \ldots, N, \tag{A.3}$$

where $\rho_N = 1$ and

$$\rho_k = \prod_{i=k+1}^{N} \xi_i + \lambda \sum_{i=k+1}^{N} x_i \prod_{j=k+1}^{i-1} \xi_j, \quad k = 1, \ldots, N-1,$$

depends only on the fractions $x_{k+1}, \ldots, x_N$.

Proof. The cost function given by eq. (A.1) is differentiable, and taking the gradient yields

$$\frac{\partial J}{\partial x_k} = \left(\prod_{i=0}^{k-1} \xi_i\right) \left((4x_k - 2)\,\rho_k + \lambda\right),$$

which, when set to 0, yields a critical point (minimum) at

$$x_k = \frac{1}{2} - \lambda \frac{1}{4\rho_k}.$$

To prove that this is at least a local minimum, we find the Hessian of the cost function,

$$\frac{\partial^2 J}{\partial x_k \partial x_l} = \begin{cases} \frac{4x_k - 2}{\xi_k} \left(\prod_{i=0}^{l-1} \xi_i \right) \left[(4x_l - 2)\rho_l + \lambda \right], & k < l \\ 4\rho_l \left(\prod_{i=0}^{l-1} \xi_i \right), & k = l \\ \frac{4x_l - 2}{\xi_l} \left(\prod_{i=0}^{k-1} \xi_i \right) \left[(4x_k - 2)\rho_k + \lambda \right], & k > l. \end{cases}$$

At the critical point, the off-diagonal entries evaluate to 0, which gives

$$\frac{\partial^2 J}{\partial x_k \partial x_l} = \begin{cases} 0, & k < l \\ 4\rho_l \left(\prod_{i=0}^{l-1} \xi_i \right), & k = l \\ 0, & k > l. \end{cases}$$

The Hessian at the critical point is thus a diagonal matrix with positive entries on the diagonal, constituting a *positive definite* matrix, and proving the critical point is at least a local minimum.

Showing that this critical point is not simply a local minimum but rather the global minimum over the domain $\lambda \in [0, 2]$ and $x \in [0, 0.5]$ requires verifying that the global minimum does not occur in the boundary of $[0, 0.5]^N$. Checking this condition remains as future work. □

A.4 Proof of Policy Invariance to Interval Length

In Section 2.4 we claim that, under the assumption of a uniform prior on the change point, the optimal action at any step is independent of the length of the hypothesis space. Thms. 1 and 2, which give us the cost function and the subsequent optimal search policy, assume we start with a uniform distribution over the unit interval, $\theta \sim \text{Unif}([0, 1])$. Assume instead we start with an interval of length L.

Theorem 3. *Assume the unknown change point has a distribution $\theta \sim$ Unif$([0, L])$ and our actions are defined via N fractions $x_1, \ldots, x_N$ denoting the fraction into the current hypothesis space to sample from our current position. The cost function after N measurements is*

$$J(x_1, \ldots, x_N) = L \left(\prod_{i=0}^{N} \xi_i + \lambda \sum_{i=1}^{N} x_i \prod_{j=0}^{i-1} \xi_j \right), \tag{A.4}$$

and the optimal fractions are

$$x_k^* = \frac{1}{2} - \lambda \frac{1}{4\rho_k}, \quad k = 1, \dots, N. \tag{A.5}$$

Proof. By Lemma 2 below, we have that

$$\mathbb{E}\left[e^{H_N}\right] = L \prod_{i=0}^{N} \xi_i.$$

Let D_N be the distance traveled after N samples. As in Section A.2,

$$\mathbb{E}\left[D_N\right] = \sum_{i=1}^{N} x_i \mathbb{E}\left[e^{H_{i-1}}\right],$$

and applying Lemma 2 yields

$$\mathbb{E}\left[D_N\right] = \sum_{i=1}^{N} x_i L \prod_{j=0}^{i-1} \xi_j.$$

We thus have the cost function,

$$\begin{aligned} J(x_1, \dots, x_N) &= L \prod_{i=0}^{N} \xi_i + \lambda \sum_{i=1}^{N} x_i L \prod_{j=0}^{i-1} \xi_j \\ &= L \left(\prod_{i=0}^{N} \xi_i + \lambda \sum_{i=1}^{N} x_i \prod_{j=0}^{i-1} \xi_j \right), \end{aligned}$$

As in Section A.3, the cost function given by eq. (A.4) is differentiable, and because L is a constant, taking the gradient and setting to 0 yields a critical point (minimum) at

$$x_k = \frac{1}{2} - \lambda \frac{1}{4\rho_k}.$$

This critical point does not depend on L, and thus, the optimal action does not depend on interval size, only step number. □

Lemma 2. *Let H_N be the entropy of the hypothesis space after N measurements. Under the same conditions as Thm. 3, we have*

$$\mathbb{E}\left[e^{H_N}\right] = L \prod_{i=0}^{N} \xi_i. \tag{A.6}$$

Proof. First note that the exponentiated differential entropy of a uniform distribution is the length of the hypothesis space after N samples. The proof will proceed by induction on N. Consider the base case, $N = 1$, for which it is trivial to show that

$$\mathbb{E}\left[e^{H_1}\right] = x_1^2 L + (1 - x_1)^2 L = \xi_1 L.$$

Now assume that (A.6) holds for some $N \in \mathbb{N}$. Sampling some fraction x_{N+1} into the remaining hypothesis space e^{H_N} results in two potential entropies with corresponding probabilities

$$e^{H_{N+1}} = \begin{cases} x_{N+1} e^{H_N} & \text{w/ probability} \quad x_{N+1} \\ (1 - x_{N+1}) e^{H_N} & \text{w/ probability} \quad 1 - x_{N+1}. \end{cases}$$

Therefore,

$$\begin{aligned} \mathbb{E}[e^{H_{N+1}}] &= \left(x_{N+1}^2 + (1 - x_{N+1})^2\right) \mathbb{E}[e^{H_N}] \\ &= L \prod_{i=0}^{N+1} \xi_i. \end{aligned}$$

$\square$

Bibliography

[1] Q. Sun, X. Hong, and L. E. Wold, "Cardiovascular effects of ambient particulate air pollution exposure," *Circulation*, vol. 121, p. 2755–2765, june 2010.

[2] State of Oregon. (2018, Aug.) Fire information and statistics. [Online]. Available: https://www.oregon.gov/ODF/Fire/pages/FireStats.aspx

[3] E. T. Gall, L. A. George, R. B. Cal, and A. Laguerre, "Indoor and outdoor air quality at harriet tubman middle school and the design of mitigation measures: Phase i report," Portland State University, Tech. Rep., 2018.

[4] Oregon Health Authority. (2018, Jun.) Harmful algae blooms, environmental public health. [Online]. Available: https://www.oregon.gov/OHA/PH/HealthyEnvironments/Recreation/HarmfulAlgaeBlooms/pages/index.aspx

[5] B. Settles, *Active Learning.* Morgan & Claypool, 2012.

[6] R. Castro and R. Nowak, "Active learning and sampling," in *Foundations and Applications of Sensor Management*, 1st ed. New York, NY: Springer, 2008, ch. 8.

[7] R. Nowak, "Generalized binary search," in *Proc. Allerton Conference on Communication, Control, and Computing*, 2008.

[8] S. Dasgupta, "Analysis of a greedy active learning strategy," in *Proc. Advances in Neural Information Processing Systems*, 2005.

[9] R. Willett, R. Nowak, and R. M. Castro, "Faster rates in regression via active learning," in *Advances in Neural Information Processing Systems*, 2006, pp. 179–186.

[10] D. Golovin and A. Krause, "Adaptive submodularity: Theory and applications in active learning and stochastic optimization," *Journal of Artificial Intelligence Research*, vol. 42, pp. 427–486, 2011.

[11] J. Lipor, B. P. Wong, D. Scavia, B. Kerkez, and L. Balzano, "Distance-penalized active learning using quantile search," *IEEE Transactions on Signal Processing*, vol. 65, no. 20, pp. 5453–5465, 2017.

[12] J. Lipor and G. Dasarathy, “Quantile search with time-varying search parameter,” in *2018 52nd Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2018, pp. 1016–1018.

[13] M. M. Keen, M. P. K. Freeman, and M. M. Mani, *Dealing with increased risk of natural disasters: challenges and options*. International Monetary Fund, 2003, no. 3-197.

[14] L. Coleman, “Frequency of man-made disasters in the 20th century,” *Journal of Contingencies and Crisis Management*, vol. 14, no. 1, pp. 3–11, 2006.

[15] C. for Research on the Epidemiology of Disasters, “Em-dat: the international disaster database.”

[16] M. Dunbabin and L. Marques, “Robots for environmental monitoring: Significant advancements and applications,” *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 24–39, 2012.

[17] L. Tang and G. Shao, “Drone remote sensing for forestry research and practices,” *Journal of Forestry Research*, vol. 26, no. 4, pp. 791–797, 2015.

[18] A. H. Goktogan, S. Sukkarieh, M. Bryson, J. Randle, T. Lupton, and C. Hung, “A rotary-wing unmanned air vehicle for aquatic weed surveillance and management,” *Journal of Intelligent and Robotic Systems*, vol. 57, no. 1-4, p. 467, 2010.

[19] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT Press Cambridge, MA, 2006, vol. 2, no. 3.

[20] M. V. Burnashev and K. S. Zigangirov, “An interval estimation problem for controlled observations,” *Problems in Information Transmission*, vol. 10:223-231, 1974, translated from Problemy Peredachi Informatsii, 10(3):51-61, July-September, 1974.

[21] P. Donmez and J. G. Carbonell, “Proactive learning: cost-sensitive active learning with multiple imperfect oracles,” in *Proceedings of the 17th ACM conference on Information and knowledge management*. ACM, 2008, pp. 619–628.

[22] D. Golovin and A. Krause, “Adaptive submodularity: A new approach to active learning and stochastic optimization.” in *COLT*, 2010, pp. 333–345.

[23] A. Guillory and J. Blimes, “Average-case active learning with costs,” in *Proc. Algorithmic Learning Theory*, 2009.

[24] A. Singh, R. Nowak, and P. Ramanathan, “Active learning for adaptive mobile sensing networks,” in *Proc. Information Processing in Sensor Networks*, 2006.

[25] J. Binney, A. Krause, and G. S. Sukhatme, “Informative path planning for an autonomous underwater vehicle,” in *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 4791–4796.

[26] B. Zhang and G. S. Sukhatme, "Adaptive sampling for estimating a scalar field using a robotic boat and a sensor network," in *Proc. IEEE International Conference on Robotics and Automation*, 2007.

[27] D. Marthaler and A. L. Bertozzi, "Tracking environmental level sets with autonomous vehicles," in *Recent developments in cooperative control and optimization.* Springer, 2004, pp. 317–332.

[28] C. J. Cannell and D. J. Stilwell, "A comparison of two approaches for adaptive sampling of environmental processes using autonomous underwater vehicles," in *Proceedings of OCEANS 2005 MTS/IEEE.* IEEE, 2005, pp. 1514–1521.

[29] Z. Jin and A. L. Bertozzi, "Environmental boundary tracking and estimation using multiple autonomous vehicles," in *2007 46th IEEE Conference on Decision and Control.* IEEE, 2007, pp. 4918–4923.

[30] A. Gotovos, N. Casati, G. Hitz, and A. Krause, "Active learning for level set estimation," in *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[31] G. Hitz, A. Gotovos, M.-É. Garneau, C. Pradalier, A. Krause, R. Y. Siegwart *et al.*, "Fully autonomous focused exploration for robotic environmental monitoring," in *2014 IEEE International Conference on Robotics and Automation (ICRA).* IEEE, 2014, pp. 2658–2664.

[32] I. Bogunovic, J. Scarlett, A. Krause, and V. Cevher, "Truncated variance reduction: A unified approach to bayesian optimization and level-set estimation," in *Advances in neural information processing systems*, 2016, pp. 1507–1515.

[33] L. Bottarelli, J. Blum, M. Bicego, and A. Farinelli, "Path efficient level set estimation for mobile sensors," in *Proceedings of the Symposium on Applied Computing.* ACM, 2017, pp. 262–267.

[34] R. Castro and R. Nowak, "Minimax bounds for active learning," *IEEE Trans. Inf. Theory*, vol. 54, pp. 2339–2353, May 2008.

[35] M. Horstein, "Sequential decoding using noiseless feedback," *IEEE Trans. Inf. Theory*, vol. 9, 1963.

[36] W. Caselton and J. Zidek, "Optimal network monitoring design, star," *Prob. Lett*, vol. 2, pp. 223–227, 1984.

[37] N. Cressie, "Statistics for spatial data," *Terra Nova*, vol. 4, no. 5, pp. 613–617, 1992.

[38] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies," *Journal of Machine Learning Research*, vol. 9, no. Feb, pp. 235–284, 2008.

[39] D. J. MacKay, “Introduction to gaussian processes,” *NATO ASI Series F Computer and Systems Sciences*, vol. 168, pp. 133–166, 1998.

[40] R. Bellman, “A markovian decision process,” *Journal of mathematics and mechanics*, pp. 679–684, 1957.

[41] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.

[42] R. Bellman, *Dynamic Programming*, 1st ed. Princeton, NJ, USA: Princeton University Press, 1957.

[43] R. A. Howard, *Dynamic programming and markov processes.* John Wiley, 1960.

[44] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning.* The MIT Press, 2012.

[45] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[46] J. L. McClelland, B. L. McNaughton, and R. C. O’Reilly, “Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory.” *Psychological review*, vol. 102, no. 3, p. 419, 1995.

[47] S. Thrun and A. Schwartz, “Issues in using function approximation for reinforcement learning,” in *Proceedings of the 1993 Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum*, 1993.

[48] H. v. Hasselt, “Double q-learning,” in *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS’10. USA: Curran Associates Inc., 2010, pp. 2613–2621. [Online]. Available: http://dl.acm.org/citation.cfm?id=2997046.2997187

[49] H. v. Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double q-learning,” in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, ser. AAAI’16. AAAI Press, 2016, pp. 2094–2100. [Online]. Available: http://dl.acm.org/citation.cfm?id=3016100.3016191

[50] G. Tesauro and G. R. Galperin, “On-line policy improvement using monte-carlo search,” in *Advances in Neural Information Processing Systems*, 1997, pp. 1068–1074.

[51] D. P. Bertsekas, J. N. Tsitsiklis, and C. Wu, “Rollout algorithms for combinatorial optimization,” *Journal of Heuristics*, vol. 3, no. 3, pp. 245–262, 1997.

[52] D. P. Bertsekas, *Reinforcement Learning and Optimal Control*, 1st ed. Nashua, NH, USA: Athena Scientific, 2019.

[53] E. J. G. Pitman, "Sufficient statistics and intrinsic accuracy," in *Mathematical Proceedings of the cambridge Philosophical society*, vol. 32, no. 4. Cambridge University Press, 1936, pp. 567–579.